Programming with Python Questions

Fragen zum Programmieren mit Python

Thomas Weise (汤卫思)

November 4, 2026



Abstract

Here you find questions for practice for the course *Programming with Python*. The course book and all teaching material are provided at `https://thomasweise.github.io/programmingWithPython`. The questions are provided in both English and German language.

Hier finden Sie Fragen zum Üben für den Kurs *Programming with Python*. Das Kursbuch und alles Lehrmaterial wird auf `https://thomasweise.github.io/programmingWithPython` zur Verfügung gestellt. Die Fragen sind in Englisch und Deutsch bereitgestellt.

Contents

Contents i

1 Basics / Grundlagen 1

2 Information 2

3 Integer Arithmetic / Rechnen mit Ganzzahlen 3

4 Floating Point Arithmetic / Fließkommaarithmetik 6

5 Boolean Expressions / Boolesche Ausrücke 10

6 Text 12

7 Nothing / Nichts 16

8 Built-Ins / Eingebaute Funktionen 17

9 Variable 21

10 Tools / Werkzeuge 23

11 Equality and Identity / Gleichheit und Identität 26

12 Lists / Listen 27

13 Tuple / Tuple 28

14 Sets / Mengen 30

15 Dictionaries 32

16 Alternatives / Alternativen 36

17 For-Loop / For-Schleife 39

18 While-Loop / While-Schleife 44

19 Functions / Funktionen 48

20 Unit Tests 53

21 Exceptions / Ausnahmen 59

22 Comprehension 65

23 Classes / Klassen 67

24 Underhanded / Hinterhältig 72

Backmatter 76

Glossary **77**

Python Commands **93**

Bibliography **94**

Preface

In this document, we provide questions to support your understanding and self-studying for the subject of Python programming. These questions accompany the book *Programming with Python* [331], which is freely available at https://thomasweise.github.io/programmingWithPython. The questions are provided both in English and German.

In diesem Dokument stellen wir Fragen zur Verfügung, um Ihr Verstehen und Selbst-Lernen der Python-Programmierung zu unterstützen. Die Fragen begleiten das Buch *Programming with Python* [331], welches unter https://thomasweise.github.io/programmingWithPython frei zur Verfügung steht. Die Fragen stehen in Englisch und Deutsch bereit.

Visit the course website / Besuchen Sie die Kurswebseite:



Chapter 1

Basics / Grundlagen

1.1 Python Versions (Q1)

EN Which are the two major versions of the programming language Python that are currently in use?

DE Was sind die beiden Hauptversionen der Programmiersprache Python, die aktuell in Gebrauch sind?

1.2 IDE (Q2)

EN Name one Integrated Development Environment (IDE) that is suitable for Python software development.

DE Nennen Sie eine integrierte Entwicklungsumgebung (IDE) die für Softwareentwicklung mit Python geeignet ist.

1.3 Execution of Programs / Programme ausführen (Q3)

EN Name two different ways to execute the Python program `my_program.py`.

DE Nennen Sie zwei verschiedene Möglichkeiten, das Python-Programm `my_program.py` auszuführen.

Chapter 2

Information

2.1 Information Sources / Informationsquellen (Q4)

EN Name three different types of information sources that you can use to learn more about `Python` programming.

DE Nennen Sie drei verschiedene Arten von Informationsquellen, mit denen Sie mehr über das Programmieren mit `Python` lernen können.

2.2 Trustworthyness / Verlässlichkeit (Q5)

EN What is the most trustworthy and reliable source of information for the programming with `Python`?

DE Was ist ist die verlässlichste bzw. vertrauenswürdigste Informationsquelle zum Programmieren mit `Python`?

Chapter 3

Integer Arithmetic / Rechnen mit Ganzzahlen

3.1 Data type(s) / Datentyp(en) (Q6)

EN Which data type(s) does `Python` 3 provide for whole numbers, i.e., integer numbers?

DE Welchen Datentyp bzw. welche Datentypen stellt `Python` 3 für Ganzzahlen zur Verfügung?

3.2 Range / Wertebereich (Q7)

EN What are the theoretical *and* practical limits for the largest integer number that can be represented with `Python` 3?

DE Was sind die theoretischen *und* praktischen Grenzen für die größte Ganzzahl, die mit `Python` 3 dargestellt werden kann?

3.3 Division (Q8)

EN Explain the difference between the `//` and `/` operators in `Python` 3.

DE Was ist der Unterschied zwischen den Operatoren `//` und `/` in `Python` 3?

3.4 Remainder / Rest (Q9)

EN How can we compute the remainder of the division of two integer variables `a` and `b` in `Python`?

DE Wie können wir den Rest der Division zweier Ganzzahl-Variablen `a` und `b` in `Python` berechnen?

3.5 **-Operator (Q10)

EN What is are the results of `3 ** 2` and `2 ** 4` in Python?

DE Was sind die Ergebnisse von `3 ** 2` und `2 ** 4` in Python?

3.6 Program Understanding / Programm Verstehen (Q11)

Listing 3.1: The program `int_02.py` (src)

```
1 """Fun with Arithmetics and Divisions."""
2
3 print(7 * 3)     # Line 3: What is the output of this line?
4 print(21 // 7)   # Line 4: What is the output of this line?
5 print(21 / 7)    # Line 5: What is the output of this line?
6 print(23 % 7)    # Line 6: What is the output of this line?
```

EN Which output will the program `int_02.py` in Listing 3.1 produce?

DE Welche Ausgabe produziert das Programm `int_02.py` in Listing 3.1?

3.7 Program Understanding / Programm Verstehen (Q12)

Listing 3.2: The program `int_01.py` (src)

```
1  """Compute how many potatoes I can buy."""
2
3  potatoes_in_store: int      = 1000  # total number of potatoes
4  cost_for_all_potatoes: int = 50     # cost if we would buy all potatoes
5  my_money: int               = 10    # the money I have
6
7  # Compute the cost of one single potato.
8  cost_per_potato = cost_for_all_potatoes // potatoes_in_store  # Line 8
9  print(f"One potato costs {cost_per_potato} RMB.")            # Line 9
10
11 # Compute the number of potatoes that I can buy with my money.
12 i_can_buy: int = my_money / cost_per_potato                  # Line 12
13 print(f"I can buy {i_can_buy} potatoes.")                    # Line 13
```

EN
1. Explain what the program `int_01.py` in Listing 3.2 is *supposed* to do.
2. Does it achieve this? If not, why?
3. What will happen if we execute this program?
4. Find at least two errors in this program.

DE
1. Erklären Sie, was das Programm `int_01.py` in Listing 3.2 machen *soll*.
2. Macht es das auch? Wenn nicht, warum?
3. Was wird passieren, wenn wir dieses Programm ausführen?
4. Finden Sie mindestens zwei Fehler in diesem Programm.

3.8 Calculation / Berechnung (Q13)

EN Write a `Python` 3 program computing the integer value $Z = 3 + 9^7 - \frac{8}{2}$ using *only* integer arithmetics. Notice that all the operations must actually be performed by the program. You cannot just compute the result on paper and just print that result...

DE Schreiben Sie ein `Python` 3-Programm, das den Ganzzahlwert $Z = 3 + 9^7 - \frac{8}{2}$ berechnet und dabei *nur* Ganzzahlarithmetik verwendet. Beachten Sie, dass alle Rechenoperationen wirklich vom Programm ausgeführt werden müssen. Sie dürfen die Formel nicht einfach auf dem Papier ausrechnen und nur das Ergebnis ausgeben...

Chapter 4

Floating Point Arithmetic / Fließkommaarithmetik

4.1 Data type(s) / Datentyp(en) (Q14)

EN Which data type(s) does `Python` 3 provide for fractional numbers like 3.4, -0.001, and 1.934?

DE Welchen Datentyp bzw. welche Datentypen stellt `Python` 3 für reelle Zahlen wie 3.4, -0.001, und 1.934 zur Verfügung?

4.2 Range / Wertebereich (Q15)

EN Give the rough limits for the largest and smallest representable positive (greater than zero!) floating point number in `Python` 3.

DE Geben sie grobe Grenzen für die größte und kleinste darstellbare positive (größer als 0!) Fließkommazahl in `Python` 3 an.

4.3 Area of Circle / Fläche eines Kreises (Q16)

$$A = \pi * r^2 \tag{4.1}$$

EN The equation above is used to compute the area A of a circle with radius r. Write a `Python` program computing the area of a circle with radius $r = 5$. The program should print the computed area to the console.

DE Die Gleichung oben wird benutzt, um die Fläche A eines Kreises mit Radius r zu berechnen. Schreiben Sie ein `Python`-Programm, das die Fläche eines Kreises mit Radius $r = 5$ berechnet. Das Programm soll die berechnete Fläche auf der Konsole ausgeben.

4.4 Special Values / Spezialwerte 1 (Q17)

EN What is the meaning of the floating point value `inf` in Python?

DE Was bedeutet der Fließkommawert `inf` in Python?

4.5 Special Values / Spezialwerte 2 (Q18)

EN What is the meaning of the floating point value `nan` in Python?

DE Was bedeutet der Fließkommawert `nan` in Python?

4.6 Understanding / Verstehen (Q19)

Listing 4.1: Program `float_01.py` (stored in file `float_01.py`; output in Listing 4.2)

```
1 """ Print the result of `0.1 + 0.2 - 0.3`. """
2 print(0.1 + 0.2 - 0.3)
```

↓ `python3 float_01.py` ↓

Listing 4.2: The standard output stream (`stdout`) of the program `float_01.py` given in Listing 4.1.

```
1 5.551115123125783e-17
```

EN Explain the output of program `float_01.py` in Listing 4.2.

DE Erklären Sie die Ausgabe des Programms `float_01.py` in Listing 4.2.

4.7 Programming / Programmieren (Q20)

$$\texttt{C} = \frac{5}{9} * (\texttt{F} - 32) \tag{4.2}$$

EN A temperature given as `F` degrees Fahrenheit can be translated to `C` degrees centigrade by the above equation. Write Python code that stores, in a variable `C`, the degrees centigrade corresponding to the degrees Fahrenheit given in a variable `F`.

DE Eine Temperature gegeben als `F` Grad Fahrenheit kann in `C` Grad Celsius mit der obigen Gleichung übersetzt werden. Schreiben Sie Python-Kode der die Grade Celsius die den Grad Fahrenheit gegeben in Variable `F` in einer Variable `C` speichert.

4.8 Pitfalls / Fallen (Q21)

Listing 4.3: The program `float_02.py` (src)

```
1  """ Print several results of rounding. """
2  print("==== -1.6 .. -1.4 ==== ")
3  print(round(-1.6))  # Line 3
4  print(round(-1.5))  # Line 4
5  print(round(-1.4))  # Line 5
6
7  print("==== -0.6 .. -0.4 ==== ")
8  print(round(-0.6))  # Line 8
9  print(round(-0.5))  # Line 9
10 print(round(-0.4))  # Line 10
11
12 print("==== 0.4 .. 0.6 ==== ")
13 print(round(0.4))  # Line 13
14 print(round(0.5))  # Line 14
15 print(round(0.6))  # Line 15
16
17 print("==== 1.4 .. 1.6 ==== ")
18 print(round(1.4))  # Line 18
19 print(round(1.5))  # Line 19
20 print(round(1.6))  # Line 20
```

EN
- What will the output of `float_02.py` given in Listing 4.3 be (line by line)?
- Why?
- Explain why this is a dangerous pitfall. Which error can one easily make?

DE
- Welche Ausgabe wird das Programm `float_02.py` in Listing 4.3 produzieren (Zeile für Zeile)?
- Warum?
- Erklären Sie, warum das eine gefährliche Falle beim Programmieren ist. Welchen Fehler kann man leicht machen?

4.9 Functions (Q22)

EN Explain the banker's rounding method used by Python's `round` function.

DE Erklären Sie das Bankier's Runden (banker's rounding), das von der Python-Funktion `round` verwendet wird.

4.10 Understanding / Verstehen (Q23)

Listing 4.4: Program `float_03.py` (stored in file `float_03.py`; output in Listing 4.5)

```
""" Limitations of Floating Point Numerics """

from math import pi, sin

print( 3 - sin(pi) )      # This prints 3.0.
print( 1 / sin(pi) )      # Then why is this "8165619676597685.0"?

print( 1e100 / 1e100 )    # This is 1.0

print( 1e400 / 1e400 )    # Then why is this "nan"?
```

↓ `python3 float_03.py` ↓

Listing 4.5: The stdout of the program `float_03.py` given in Listing 4.4.

```
3.0
8165619676597685.0
1.0
nan
```

EN Explain the output of program `float_03.py` in Listing 4.5.

DE Erklären Sie die Ausgabe des Programms `float_03.py` in Listing 4.5.

Chapter 5

Boolean Expressions / Boolesche Ausrücke

5.1 Data type(s) / Datentyp(en) (Q24)

EN Which data type(s) does `Python` 3 provide truth for (yes/no) values?

DE Welchen Datentyp bzw. welche Datentypen stellt `Python` 3 für Wahrheitswerte (Ja/Nein) zur Verfügung?

5.2 Values / Werte (Q25)

EN What is the meaning of `True` and `False`? To which datatype do they belong?

DE Was ist die Bedeutung von `True` und `False`? Zu welchem Datentyp gehören sie?

5.3 Program Understanding / Programm Verstehen (Q26)

Listing 5.1: The program `bool_02.py` (src)

```
1  """Working with Boolean expressions."""
2
3  x1 = True
4  x2 = False
5
6  print(x1 and x2)                        # Line 6: What does it print?
7
8  print(not x1 or x2)                     # Line 8: What does it print?
9
10 print(not x1 and not x2)                # Line 10: What does it print?
11
12 print((x1 or x2) and (not x1 and x1))   # Line 12: What does it print?
```

EN Which output will the program `bool_02.py` in Listing 5.1 produce?

DE Welche Ausgabe produziert das Programm `bool_02.py` in Listing 5.1?

5.4 Program Understanding / Programm Verstehen (Q27)

Listing 5.2: The program bool_01.py (src)

```
"""Working with Boolean expressions."""

a: bool = 1 < 5 <= 5 < 6   # Line 3
print(a)                   # Line 4: What does this print?

b: str  = "Hello World!"   # Line 6
c: bool = b is not b       # Line 7
print(c)                   # Line 8: What does this print?

print(a or c)              # Line 10: What does this print?
print(a and c)             # Line 11: What does this print?
print(c or b)              # Line 12 (BONUS): What does this print?
```

EN

1. Which output will the program bool_01.py in Listing 5.2 produce?
2. The last line of the program is a bonus...

DE

1. Welche Ausgabe produziert das Programm bool_01.py in Listing 5.2?
2. Die letzte Zeile ist ein Bonus...

Chapter 6

Text

6.1 Data type(s) / Datentyp(en) (Q28)

EN Which data type(s) does `Python` 3 provide for text?

DE Welchen Datentyp bzw. welche Datentypen stellt `Python` 3 für Text bzw. Zeichenketten zur Verfügung?

6.2 Delimiters / Begrenzung (Q29)

EN How can we mark the beginning and ending of a text in `Python` 3. Give *three* choices.

DE Wie wird der Anfang und das Ende von Zeichenketten in `Python` 3 markiert? Geben Sie *drei* Möglichkeiten an.

6.3 General Knowledge / Generelles Verstehen (Q30)

EN Assume a text string is stored in variable `text`. Give *three* possible methods to check whether a string stored in variable `find` is contained somewhere in `text`. You get a bonus point if you can name a fourth method.

DE Nehmen Sie an, dass ein Text-String in der Variable `text` gespeichert ist. Nennen Sie *drei* Möglichkeiten, herauszufinden ob ein String gespeichert in Variable `find` irgendwo in `text` enthalten ist. Sie bekommen einen Bonus-Punkt, wenn Sie eine vierte Methode nennen.

6.4 Program Understanding / Programm Verstehen (Q31)

Listing 6.1: The program str_01.py (src)

```
"""Printing some strings."""

print("Hello World!")            # Line 3: What output does it produce?
print("Hello " + "World!")       # Line 4: What output does it produce?
print("Hello\nWorld!")           # Line 5: What output does it produce?
print("Hello " * 3 + "World!")   # Line 6: What output does it produce?
print(f"{8 / 2}")                # Line 7: What output does it produce?
print(f"{8 / 2 = }")             # Line 8: What output does it produce?
```

EN Which output will the program str_01.py in Listing 6.1 produce?

DE Welche Ausgabe produziert das Programm str_01.py in Listing 6.1?

6.5 Program Understanding / Programm Verstehen (Q32)

Listing 6.2: The program str_02.py (src)

```
"""Printing some strings."""

print("Hello World!")           # Line 3: What output does it produce?
print("Hello World!"[3])        # Line 4: What output does it produce?
print("Hello World!"[2:5])      # Line 5: What output does it produce?
print("Hello World!"[2:-2])     # Line 6: What output does it produce?
print("Hello World!"[2:-2:3])   # Line 7: What output does it produce?
```

EN Which output will the program str_02.py in Listing 6.2 produce?

DE Welche Ausgabe produziert das Programm str_02.py in Listing 6.2?

6.6 Program Understanding / Programm Verstehen (Q33)

Listing 6.3: The program str_03.py (src)

```
"""Printing some strings."""

my_int: int = 12
print(f"my_int is {my_int}")  # Line 4: Which output does this produce?

print("Hello\nWorld!")        # Line 6: Which output does this produce?
```

EN Which output will the program str_03.py in Listing 6.3 produce?

DE Welche Ausgabe produziert das Programm str_03.py in Listing 6.3?

6.7 f-Strings (Q34)

EN Explain what an f-string in Python 3 is.

DE Erklären Sie, was ein f-String in Python 3 ist.

6.8 Programming / Programmieren (Q35)

EN Assume that the name of a person is stored in the variable `name`. Write a piece of Python-code which prints the following text: `Hello XXX! Nice to meet you!`, where `XXX` is to be replaced with the name of the person.

DE Nehmen Sie an, dass der Name einer Person in der Variable `name` gespeichert ist. Schreiben Sie ein Stück Python-code, der den folgenden Text ausgibt `Hello XXX! Nice to meet you!`, wobei `XXX` mit dem Name der Person zu ersetzen ist.

6.9 Pitfalls / Fallen (Q36)

Listing 6.4: The program `str_04.py` (src)

```
1  """Conversion of strings to bools."""
2
3  print(str(True))                  # Line 3
4  print(bool("True"))               # Line 4
5
6  print(str(False))                 # Line 6
7  print(bool("False"))              # Line 7
8
9  print(bool(str(True))  == True)   # Line 9
10 print(bool(str(False)) == False)  # Line 10
11
12 print(bool(""))                   # Line 11
```

EN
- What will the output of `str_04.py` given in Listing 6.4 be (line by line)?
- Why?
- Explain why this is a dangerous pitfall. Which error can one easily make?

DE
- Welche Ausgabe wird das Programm `str_04.py` in Listing 6.4 produzieren (Zeile für Zeile)?
- Warum?
- Erklären Sie, warum das eine gefährliche Falle beim Programmieren ist. Welchen Fehler kann man leicht machen?

6.10 Pitfalls & Understanding / Fallen & Verstehen (Q37)

Listing 6.5: Program `str_05.py` (stored in file `str_05.py`; output in Listing 6.6)

```
"""Using the Result of 'find' as Index."""

text: str = "Hello World!"  # Store "Hello World!" in variable 'text'.

print(text.find("World"))          # Line 5

print(text[text.find("World")])    # Line 7
print(text[text.find("World"):])   # Line 8

print(text[text.find("world")])    # Line 10
print(text[text.find("world"):])   # Line 11
```

↓ `python3 str_05.py` ↓

Listing 6.6: The stdout of the program `str_05.py` given in Listing 6.5.

```
6
W
World!
!
!
```

EN
- Explain the output of program `float_01.py` in Listing 6.6.
- Why is using the result of `find` as index into a string dangerous?

DE
- Erklären Sie die Ausgabe des Programms `float_01.py` in Listing 6.6.
- Warum ist es gefährlich, das Ergebnis von `find` als index in einen String zu verwenden?

6.11 Program Understanding / Programm Verstehen (Q38)

Listing 6.7: The program `str_06.py` (src)

```
"""Printing a multi-line string strings."""

print(
"""
Hello world!
This is a multi-line string.
It has multiple lines.
""")

print("How\nmany lines\ndoes this print?")
```

EN Which output will the program `str_06.py` in Listing 6.7 produce?

DE Welche Ausgabe produziert das Programm `str_06.py` in Listing 6.7?

Chapter 7

Nothing / Nichts

7.1 Data type(s) / Datentyp(en) (Q39)

EN Which data type(s) or value(s) does Python 3 provide to signify that something has no value?

DE Mit welche(n) Datentyp(en) / Wert(e) kann Python 3 ausgedrückt werden, das etwas keinen Wert hat?

7.2 Program Understanding / Programm Verstehen (Q40)

Listing 7.1: The program none_01.py (src)

```
1 """Program output."""
2
3 print(print("Hello World!"))
```

EN Which output will the program none_01.py in Listing 7.1 produce?

DE Welche Ausgabe produziert das Programm none_01.py in Listing 7.1?

Chapter 8

Built-Ins / Eingebaute Funktionen

8.1 Ausgeben (Q41)

EN What does the function `print`?

DE Was macht die Funktion `print`?

8.2 Rounding / Runden (Q42)

EN Explain what the four functions `round`, `int`, `floor`, and `ceil` are doing.

DE Erklären Sie, was die vier Funktionen `round`, `int`, `floor` und `ceil` machen.

8.3 Basic Functions / Grundfunktionen (Q43)

EN Explain what the four functions `sqrt`, `log`, `exp`, and `sin` are doing.

DE Erklären Sie, was die vier Funktionen `sqrt`, `log`, `exp` und `sin` machen.

8.4 Program Understanding / Programm Verstehen (Q44)

Listing 8.1: The program builtins_01.py (stored in file builtins_01.py; output in Listing 8.2)

```
"""Example of the built-in functions str, ceil, floor, and round."""

from math import ceil, floor

a = 0.99999999999999999999999999999999
b = 1.5
print("==== " + str(a) + " and " + str(b) + " ====")
print(round(a) - round(b))    # prints -1
print( ceil(a) - ceil(b))     # prints -1
print(floor(a) - floor(b))    # prints 0

c = 1.99999999999999999999999999999999
d = 2.5
print("==== " + str(c) + " and " + str(d) + " ====")
print(round(c) - round(d))    # prints 0
print( ceil(c) - ceil(d))     # prints -1
print(floor(c) - floor(d))    # prints 0
```

↓ `python3 builtins_01.py` ↓

Listing 8.2: The stdout of the program builtins_01.py given in Listing 8.1.

```
==== 1.0 and 1.5 ====
-1
-1
0
==== 2.0 and 2.5 ====
0
-1
0
```

EN Explain the output of the program builtins_01.py above.

DE Erklären Sie die Ausgabe des Programms builtins_01.py oben.

8.5 Program Understanding / Programm Verstehen (Q45)

Listing 8.3: The program builtins_02.py (stored in file builtins_02.py; output in Listing 8.4)

```
"""Example of the built-in functions exp, log and the constant e."""

from math import exp, log, e

print(exp(0) - 1)            # prints 0.0
print(exp(1) - e)            # prints 0.0
print(exp(2) - e ** 2)       # prints 8.881784197001252e-16
print(exp(3) - e ** 3)       # prints 3.552713678800501e-15
print(exp(4) - e ** 4)       # prints 7.105427357601002e-15

print(log(1))                # prints 0.0
print(log(e))                # prints 1.0
print(log(e ** 2))           # prints 2.0
print(log(e ** 3))           # prints 3.0
print(log(e ** 4))           # prints 4.0

print(log(exp(10))  - 10)    # prints 0.0
print(log(exp(0.1)) - 0.1)   # prints 6.938893903907228e-17
```

↓ `python3 builtins_02.py` ↓

Listing 8.4: The stdout of the program builtins_02.py given in Listing 8.3.

```
0.0
0.0
8.881784197001252e-16
3.552713678800501e-15
7.105427357601002e-15
0.0
1.0
2.0
3.0
4.0
0.0
6.938893903907228e-17
```

EN Explain the output of the program builtins_02.py above.

DE Erklären Sie die Ausgabe des Programms builtins_02.py oben.

8.6 Program Understanding / Programm Verstehen (Q46)

Listing 8.5: The program builtins_03.py (stored in file builtins_03.py; output in Listing 8.6)

```
"""Example of the built-ins sin, asin, cos, acos, and pi."""

from math import sin, cos, asin, acos, pi

print(sin(0))                    # prints 0.0
print(sin(0.5 * pi))             # prints 1.0
print(asin(1))                   # prints 1.5707963267948966
print(sin(pi / 4))               # prints 0.7071067811865475
print((1 / sin(pi / 4)) ** 2)    # prints 2.0000000000000004

print(cos(0))                    # prints 1.0
print(cos(pi / 3))               # prints 0.5000000000000001
print(acos(0.5))                 # prints 1.0471975511965979
```

↓ `python3 builtins_03.py` ↓

Listing 8.6: The stdout of the program builtins_03.py given in Listing 8.5.

```
0.0
1.0
1.5707963267948966
0.7071067811865475
2.0000000000000004
1.0
0.5000000000000001
1.0471975511965979
```

EN Explain the output of the program builtins_03.py above.

DE Erklären Sie die Ausgabe des Programms builtins_03.py oben.

Chapter 9

Variable

9.1 What is it? / Was ist das? (Q47)

EN What is a variable in Python?

DE Was ist eine Variable in Python?

9.2 Program Understanding / Programm Verstehen (Q48)

Listing 9.1: The program variable_01.py (src)

```
"""Variable Assignment."""

a: int = 12         # What is the meaning of "a", ": int", "=", and "12"?
b: int = 6          # What is the meaning of "b", ": int", "=", and "6"?
c: float = a / b    # What is the meaning of "c", ": float", "=", "a / b"?

a, b = b, a         # What does this do?
d: float = a / b    # What does this do?
print(c - d)        # What is the result of this?
```

EN

1. In variable_01.py in Listing 9.1, what is the meaning of
   a) `a`, `b`, `c`, and `d`,
   b) `: int`,
   c) `: float`, and
   d) `a / b`?
2. What is the output of the program variable_01.py ?

DE

1. In variable_01.py in Listing 9.1, was ist die Bedeutung von
   a) `a`, `b`, `c`, and `d`,
   b) `: int`,
   c) `: float` und
   d) `a / b`?
2. Welche Ausgabe produziert das Programm variable_01.py?

9.3 Type Hints (Q49)

EN
1. Explain what type hints in Python 3 are.
2. Provide at least two reasons why type hints should be used.

DE
1. Erklären Sie, was Type-Hints in Python 3 sind.
2. Geben Sie mindestens zwei Gründe an, warum Type-Hints verwendet werden sollten.

9.4 Program Understanding / Programm Verstehen (Q50)

Listing 9.2: The program variable_02.py (src)

```
1 """Variable Assignments and f-strings."""
2
3 a: int = 1
4 print(f"{a = }")        # Line 4: What does this print?
5
6 b: float = 3.14
7 print(f"{ a * b }")     # Line 7: What does this print?
8
9 print(f"{a == b = }")   # Line 9: What does this print?
```

EN What output does program variable_02.py in Listing 9.2 produce?

DE Was gibt das Programm variable_02.py in Listing 9.2 aus?

Chapter 10

Tools / Werkzeuge

10.1 MyPy (Q51)

EN What is Mypy?

DE Was ist Mypy?

10.2 MyPy (Q52)

Listing 10.1: The program mypy_01.py (src)

```
"""An example of using MyPy."""

x: int = 56
y: int = 4
x = x
z: int = x / y
```

Listing 10.2: The result of Mypy applied to Listing 10.1.

```
$ mypy mypy_01.py --no-strict-optional --check-untyped-defs
mypy_01.py:6: error: Incompatible types in assignment (expression has type
    ↪ "float", variable has type "int")  [assignment]
Found 1 error in 1 file (checked 1 source file)
# mypy 2.4.1 failed with exit code 1.
```

EN Listing 10.2 shows the output of Mypy applied to program mypy_01.py given in Listing 10.1. Explain this output.

DE Listing 10.2 zeigt die Ausgabe von Mypy angewandt auf Programm mypy_01.py in Listing 10.1. Erklären Sie diese Ausgabe.

10.3 Linter (Q53)

EN What is a linter?

DE Was ist ein Linter?

10.4 Ruff (Q54)

EN What is Ruff?

DE Was ist Ruff?

10.5 Ruff (Q55)

Listing 10.3: The program ruff_01.py (src)

```
"""An example of using Ruff."""

x: int = 56
y: int = 4
x = x
z: int = x / y
```

Listing 10.4: The result of Ruff applied to Listing 10.3.

```
$ ruff check --target-version py312 --select=A,AIR,ANN,ASYNC,B,BLE,C,C4,COM
    ↪ ,D,DJ,DTZ,E,ERA,EXE,F,FA,FIX,FLY,FURB,G,I,ICN,INP,ISC,INT,LOG,N,NPY,
    ↪ PERF,PIE,PLC,PLE,PLR,PLW,PT,PYI,Q,RET,RSE,RUF,S,SIM,T,T10,TD,TID,TRY,
    ↪ UP,W,YTT --ignore=A005,ANN001,ANN002,ANN003,ANN204,ANN401,B008,B009,
    ↪ B010,C901,D203,D208,D212,D401,D407,D413,INP001,N801,PLC2801,PLR0904,
    ↪ PLR0911,PLR0912,PLR0913,PLR0914,PLR0915,PLR0916,PLR0917,PLR1702,
    ↪ PLR2004,PLR6301,PT011,PT012,PT013,PYI041,RUF100,S,T201,TRY003,UP035,W
    ↪  --line-length 79 ruff_01.py
PLW0127 Self-assignment of variable `x`
 --> ruff_01.py:5:1
3 | x: int = 56
4 | y: int = 4
5 | x = x
  | ^
6 | z: int = x / y
Found 1 error.
# ruff 0.17.3 failed with exit code 1.
```

EN Listing 10.4 shows the output of Ruff applied to program ruff_01.py given in Listing 10.3. Explain this output.

DE Listing 10.4 zeigt die Ausgabe von Ruff angewandt auf Programm ruff_01.py in Listing 10.3. Erklären Sie diese Ausgabe.

10.6 Pylint (Q56)

EN What is Pylint?

DE Was ist Pylint?

10.7 Pylint (Q57)

Listing 10.5: The program pylint_01.py (src)

```
"""An example of using Pylint."""

x: int = 56
y: int = 4
x = x
z: int = x / y
```

Listing 10.6: The result of Pylint applied to Listing 10.5.

```
$ pylint pylint_01.py --disable=C0103,C0302,C0325,R0801,R0901,R0902,R0903,
    ↪ R0911,R0912,R0913,R0914,R0915,R1702,R1728,W0212,W0238,W0703
************* Module pylint_01
pylint_01.py:5:0: W0127: Assigning the same variable 'x' to itself (self-
    ↪ assigning-variable)
-----------------------------------
Your code has been rated at 7.50/10
# pylint 4.1.3 failed with exit code 4.
```

EN Listing 10.6 shows the output of Pylint applied to program pylint_01.py given in Listing 10.5. Explain this output.

DE Listing 10.6 zeigt die Ausgabe von Pylint angewandt auf Programm pylint_01.py in Listing 10.5. Erklären Sie diese Ausgabe.

Chapter 11

Equality and Identity / Gleichheit und Identität

11.1 Understanding / Verstehen (Q58)

Listing 11.1: Program `identity_01.py` (stored in file `identity_01.py`; output in Listing 11.2)

```
"""Identity and Equality."""

from math import e

print(e)

same_as_e: float = 2.718281828459045
print(same_as_e)

print(same_as_e == e)
print(same_as_e is e)
```

↓ `python3 identity_01.py` ↓

Listing 11.2: The stdout of the program `identity_01.py` given in Listing 11.1.

```
2.718281828459045
2.718281828459045
True
False
```

EN Explain the output of program `identity_01.py` in Listing 11.2.

DE Erklären Sie die Ausgabe des Programms `identity_01.py` in Listing 11.2.

Chapter 12

Lists / Listen

12.1 What is it? / Was ist das? (Q59)

EN What is a `list` in Python? Name four of its properties or things that can be done with it in Python.

DE Was ist eine Liste (`list`) in Python? Nennen Sie vier Eigenschaften bzw. Dinge die man mit Listen in Python machen kann.

12.2 Program Understanding / Programm Verstehen (Q60)

Listing 12.1: The program `list_01.py` (src)

```
"""Lists: What output is produced by this program?"""

a: list[int] = [1, 2, 3, 4]
a.append(5)
a.remove(3)
del a[2]
a.extend([5, 2])
a.sort()

print(a)
```

EN Which output will the program `list_01.py` in Listing 12.1 produce?

DE Welche Ausgabe produziert das Programm `list_01.py` in Listing 12.1?

Chapter 13

Tuple / Tuple

13.1 What is it? / Was ist das? (Q61)

EN What is a `tuple` in Python? Name four of its properties or things that can be done with it in Python.

DE Was ist eine Tupel (`tuple`) in Python? Nennen Sie vier Eigenschaften bzw. Dinge die man mit Tupeln in Python machen kann.

13.2 What is it? / Was ist das? (Q62)

EN What is the difference between a `tuple` and a `list` in Python?

DE Was ist der Unterschied zwischen einem Tupel (`tuple`) und einer Liste (`list`) in Python?

13.3 Program Understanding / Programm Verstehen (Q63)

Listing 13.1: The program `tuple_01.py` (src)

```
"""Tuples: What output is produced by this program?"""

a: tuple[int, ...] = (4, 1, 2, 3)

b, c, d, e = a
print(b)                    # Line 6: What does it print?
print(e)                    # Line 7: What does it print?

a = (c, e, d, b)
print(a)                    # Line 10: What does it print?

f: tuple[tuple[int, int], int, tuple[int, int]] = (b, b), c, (d, d)
print(f.index((2, 2)))      # Line 13: What does it print?
```

EN Which output will the program `tuple_01.py` in Listing 13.1 produce?

DE Welche Ausgabe produziert das Programm `tuple_01.py` in Listing 13.1?

13.4 Programming / Programmieren (Q64)

EN The variable `numbers` holds a tuple of integer numbers. Write some Python code that prints `Yes` if the first and the last number in the tuple are the same and `No` otherwise. For example, if `numbers = (3, 6, 3)`, it would print `Yes` but for `numbers = (7, 6, 4, 7, 3)`, it would print `No`.

DE In der Variable `numbers` ist Tupel von Ganzzahlen gespeichert. Schreiben Sie Python-Kode der `Yes` ausgibt, wenn die erste und letzte Zahl im Tupel gleich sind und sonst `No`. Zum Beispiel, wenn `numbers = (3, 6, 3)` würde er `Yes` ausgeben, aber für `numbers = (7, 6, 4, 7, 3)` würde er `No` ausgeben.

13.5 Program Understanding / Programm Verstehen (Q65)

Listing 13.2: The program `tuple_02.py` (src)

```
1 """Tuples: What output is produced by this program?"""
2
3 names: tuple[str, ...] = ("Bibbo", "Bebbo", "Bebba", "Bobbo")
4
5 print(f"The fourth person is {names[4]}.")
```

EN Why does this code not work?

DE Warum funktioniert dieser Kode nicht?

Chapter 14

Sets / Mengen

14.1 What is it? / Was ist das? (Q66)

EN What is a `set` in Python? Name four of its properties or things that can be done with it in Python.

DE Was ist eine Menge (`set`) in Python? Nennen Sie vier Eigenschaften bzw. Dinge die man mit Mengen in Python machen kann.

14.2 Program Understanding / Programm Verstehen (Q67)

Listing 14.1: The program set_01.py (src)

```
1 """Sets in Python."""
2
3 st1: set[str] = {"a", "b", "c", "e", "e", "f", "g"}
4 print("a" in st1)            # Line 4: What does it print?
5 print("x" in st1)            # Line 5: What does it print?
6
7 st2: set[str] = {"f", "h", "x", "c", "f"}
8 print(st1.union(st2))        # Line 8: What could it print?
9 print(st1.difference(st2))   # Line 9: What could it print?
```

EN Which output will the program set_01.py in Listing 14.1 produce?

DE Welche Ausgabe produziert das Programm set_01.py in Listing 14.1?

14.3 Program Understanding / Programm Verstehen (Q68)

Listing 14.2: The program set_02.py (src)

```
"""Sets in Python."""

s: set[str] = {"a", "b", "c", "e", "e", "f", "g", "h", "i", "j", "k"}
print(s)  # How does this behave?
```

Listing 14.3: One independent execution of set_02.py.

```
{'g', 'f', 'a', 'i', 'h', 'b', 'j', 'k', 'e', 'c'}
```

Listing 14.4: Another independent execution of set_02.py.

```
{'c', 'j', 'f', 'i', 'k', 'b', 'a', 'e', 'h', 'g'}
```

EN Listing 14.3 and Listing 14.4 show the outputs of two independent executions of set_02.py in Listing 14.2. Why are they different?

DE Listing 14.3 und Listing 14.4 zeigen die Ausgaben von zwei unabhängigen Ausführungen von Programm set_02.py in Listing 14.2. Warum sind sie verschieden?

14.4 Programming / Programmieren (Q69)

EN Assume that a list of integer numbers is stored in variable `numbers`. Print each number contained in `numbers` exactly once. In other words, if `numbers = [1, 2, 2, 3, 5, 7, 7, 7]`, you could print `1, 2, 3, 5, 7`.

DE Nehmen Sie an, dass eine Liste von Ganzzahlen in der Variable `numbers` gespeichert ist. Geben Sie jede Zahl in `numbers` genau einmal aus. Zum Beispiel, wenn `numbers = [1, 2, 2, 3, 5, 7, 7, 7]`, dann könnten Sie `1, 2, 3, 5, 7` ausgeben.

Chapter 15

Dictionaries

15.1 What is it? / Was ist das? (Q70)

EN What is a dictionary (`dict`) in Python? Name four of its properties or things that can be done with it in Python.

DE Was ist ein Dictionary (`dict`) in Python? Nennen Sie vier Eigenschaften bzw. Dinge die man mit Dictionaries in Python machen kann.

15.2 Program Understanding / Programm Verstehen (Q71)

Listing 15.1: The program dict_01.py (src)

```
1  """Dictionaries: What output is produced by this program?"""
2
3  from math import sqrt
4
5  roots: dict[int, float] = {2: sqrt(2), 3: sqrt(3), 4: sqrt(4)}
6
7  print(3 in roots)            # Line 7: What does it print?
8  print(5 in roots)            # Line 8: What does it print?
9
10 print(roots[4])              # Line 10: What does it print?
11
12 roots[9] = sqrt(9)
13 del roots[2]
14 print(max(roots.values()))  # Line 14: What does it print?
```

EN Which output will the program dict_01.py in Listing 15.1 produce?

DE Welche Ausgabe produziert das Programm dict_01.py in Listing 15.1?

15.3 Programming / Programmieren (Q72)

Listing 15.2: The program `dict_02.py` (stored in file `dict_02.py`; output in Listing 15.3)

```
"""Dictionaries: How could `create_dictionary` be implemented?"""

from dict_02_module import create_dictionary

print(create_dictionary())
```

↓ `python3 dict_02.py` ↓

Listing 15.3: The stdout of the program `dict_02.py` given in Listing 15.2.

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', -1: 'minus one', -2:
    ↪ 'minus two', -3: 'minus three', -4: 'minus four', -5: 'minus five'}
```

EN The output of program `dict_02.py` is given in Listing 15.3 above. Write a function `create_dictionary` in Python that **returns a dictionary** (`dict`) which would print exactly as shown above.

DE Die Ausgabe von Programm `dict_02.py` ist in Listing 15.3 oben gegeben. Schreiben Sie eine Funktion `my_function` in Python die **ein Dictionary** (`dict`) zurückliefert, das genau die obige Ausgabe produzieren würde.

15.4 Programming / Programmieren (Q73)

Listing 15.4: The beginning of program dict_03.py (src)

```
"""Processing a dictionary of name-(nationality, age) pairs."""

# A dictionary where the names of people are keys and the values are
# tuples of their nationality and age.
people: dict[str, tuple[str, int]] = {
    "Bibbo": ("Chinese", 25), "Bebbo": ("French", 33),
    "Bibba": ("Italian", 12), "Thomas": ("German", 44),
    "Zhou": ("Chinese", 21), "Frank": ("English", 86),
    "Wang": ("Chinese", 34), "Bodderich": ("Polish", 31)
}
```

EN In file dict_03.py above, a dictionary `people` that has the names of people as keys and tuples of their nationality and age as values. Continue/extend this program with code that...

1. finds the names of all Chinese people and prints them,
2. finds the oldest person and prints their name and age,
3. finds the youngest Chinese person and prints their name.

Your code must work with the dictionary `people` and cannot just print values that you found by hand.

DE In Datei dict_03.py oben wird ein Dictionary `people` angelegt, dass die Namen der Leute als Schlüssel und Tupel von ihrer Nationalität und ihrem Alter als Werte speichert. Erweitern Sie das Programm mit Kode der...

1. die Namen aller chinesischen (EN: *Chinese*) Leute findet und ausgibt,
2. den Name der ältesten Person findet und ausgibt,
3. den Namen der jüngsten chinesischen Person findet und ausgibt.

Ihr Kode muss mit dem Dictionary `people` arbeiten und kann nicht einfach Werte ausgeben, die Sie selber per Hand gefunden haben.

15.5 Programming / Programmieren (Q74)

Listing 15.5: The program dict_05.py (src)

```
"""Fix the code by replacing `???` with something."""

dct: dict[int, str] = {1: "One", 2: "Two", 3: "Three"}

for ??? in dct.items():
    print(f"key: {key}")
    print(f"value: {value}")
```

EN Repair dict_05.py by replacing the (and only the) `???`.

DE Reparieren Sie dict_05.py, in dem Sie das (und nur das) `???` ersetzen.

15.6 Programming / Programmieren (Q75)

Listing 15.6: The program dict_04.py (src)

```
"""A dictionary with explanations and marks."""

# A dictionary with explanations as keys and marks as values.
marks: dict[str, str] = {
    "excellent": "A+", "very good": "A", "good": "B", "satisfying": "C",
    "pass": "D", "inadequate": "E", "insufficient": "F",
}
```

EN You are given the dictionary `marks` with the content given above. Write a function that returns the explanation (e.g., *good*) for a given mark (e.g., *B*). Notice that the dictionary is "inverted": It has the explanations as keys and the marks as values. You thus first want to create an dictionary where the marks are the keys and the explanations are the values. You want to use that dictionary in your function.

DE Sie bekommen das Dictionary `marks` mit dem Inhalt oben. Schreiben Sie eine Funktion, die die Erklärung (zum Beispiel *good*) fpür eine Zensur (zum Beispiel *B*) liefert. Beachten Sie, dass das Dictionary "falsch herum ist": Es hat die Erklärungen als Schlüssel und die Zensuren als Werte. Sie wollen daher zuerst ein neues Dictionary erstellen, wo die Zensuren die Schlüssel und die Erklärungen die Werte sind. Sie wollen dann dieses Dictionary in Ihrer Funktion verwenden.

Chapter 16

Alternatives / Alternativen

16.1 What is it? / Was ist das? (Q76)

EN Explain the syntax *and* purpose/use of an `if` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `if`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

16.2 What is it? / Was ist das? (Q77)

EN Explain the syntax *and* purpose/use of an `else` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `else`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

16.3 What is it? / Was ist das? (Q78)

EN Explain the syntax *and* purpose/use of an `elif` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `elif`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

16.4 Program Understanding / Programm Verstehen (Q79)

Listing 16.1: The program `if_01.py` (src)

```
""" If-then-else Examples. """

from math import isqrt

number = 121
if (number % 2) == 0:
    print(f"Number {number} is even.")

if number > 12:
    print(f"Number {number} is bigger than 12.")

if number == isqrt(number) ** 2:
    print("Interesting.")
```

EN Which output will the program `if_01.py` in Listing 16.1 produce?

DE Welche Ausgabe produziert das Programm `if_01.py` in Listing 16.1?

16.5 Programming / Programmieren (Q80)

EN Write code that checks whether a number stored in variable `k` is in 5..10. The interval borders are part of the interval.

DE Schreiben Sie Kode der prüft ob eine Zahl – gespeichert in Variable `k` – aus dem Intervall 5..10 ist. Die Intervallgrenzen sind Teil des Intervalls.

16.6 Program Understanding / Programm Verstehen (Q81)

Listing 16.2: The program `if_02.py` (src)

```
""" If-then-else Examples. """

price_of_shoes: int = 100

if price_of_shoes <= 40:
    print("The shoes are cheap.")
elif price_of_shoes <= 150:
    print("The shoes are reasonably priced.")
elif price_of_shoes <= 250:
    print("The shoes are a bit expensive.")
else:
    print("Not going to happen.")
```

EN Explain program `if_02.py` in Listing 16.2 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `if_02.py` in Listing 16.2 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

16.7 Programming / Programmieren (Q82)

EN Assume that the variable `v` stores the volume of a cube (all side-lengths are the same). Write a `Python` program that checks whether the cube can be composed of smaller cubes whose side-lengths are all 1. The program should output `Yes` if that is possible and otherwise `No`.

DE Nehmen Sie an, dass die Variable `v` das Volumen eines Würfels (dessen Seiten alle gleichlang sind) beinhaltet. Schreiben Sie ein `Python`-Programm, das prüft ob der Würfel aus Würfeln der Seitenlänge 1 zusammengesetzt werden kann. Das Programm soll `Yes` ausgeben, wenn das geht, und `No`, wenn nicht.

16.8 Programming / Programmieren (Q83)

Income/Einkommen E	Tax/Steuer
$E \leq 10\,000$	0%
$10\,000 < E \leq 50\,000$	10%
$50\,000 < E \leq 250\,000$	20%
$250\,000 < E$	30%

EN In the table above, the tax rates for different income brackets are given. Assume that the income of a person is stored in variable `income`. Write `Python` code that prints the amount of tax that they have to pay.

DE In der Tabelle oben sind die Steuerraten für verschiedene Einkommensklassen angegeben. Nehmen Sie an, dass das Einkommen einer Person in einer Variable `income` gespeichert ist. Schreiben Sie `Python`-Kode der ausgibt, wie viel Steuern die Person zahlen muss.

16.9 Programming / Programmieren (Q84)

Score/Punkte P	Mark / Zensuren/Note M
$P = 100$	A+
$95 \leq P < 100$	A
$90 \leq P < 95$	A-
$88 \leq P < 90$	B+
$82 \leq P < 88$	B
$80 \leq P < 82$	B-
$70 \leq P < 80$	C
$60 \leq P < 70$	D
$P < 60$	F

EN In the table above, the score brackets and corresponding marks are given for an exam. Assume that the exam score of a person is stored in variable `P`. Write `Python` code that prints the mark that they received.

DE In der Tabelle oben sind die Punkte für verschiedene Zensuren für eine Prüfung angegeben. Nehmen Sie an, dass das Prüfungspunkte einer Person in einer Variable `P` gespeichert sind. Schreiben Sie `Python`-Kode der ausgibt, welche Zensur die Person erhalten hat.

Chapter 17

For-Loop / For-Schleife

17.1 What is it? / Was ist das? (Q85)

EN Explain the syntax *and* purpose/use of a `for` statement in Python. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `for`-Statements in Python. Sie können ein Beispiel in Ihrer Erklärung verwenden.

17.2 Program Understanding / Programm Verstehen (Q86)

Listing 17.1: The program `for_01.py` (src)

```
""" For-Loop Example. """

for i in range(5):
    for j in range(5):
        if i < j:
            print(f"({i}, {j})")
```

EN Explain program `for_01.py` in Listing 17.1 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `for_01.py` in Listing 17.1 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

17.3 Program Understanding / Programm Verstehen (Q87)

Listing 17.2: The program `for_02.py` (src)

```
""" For-Loop Example. """

for i in range(5):
    for j in range(i, 2 * i):
        print(j - i)
```

EN Explain program `for_02.py` in Listing 17.2 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `for_02.py` in Listing 17.2 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

17.4 Programming / Programmieren (Q88)

EN The variable `lists` holds a list of lists of integers, e.g. something like `[[1, 2], [3], [-1, 2, 3]]`. Write some Python code that creates a new list `flat`, which is a single list of integers with the same elements as `lists`, e.g. `[1, 2, 3, -1, 2, 3]`. In other words, your code must take an arbitrary list-of-lists-of-integers `lists` and creates a list `flat` containing all their integer elements.

DE Die Variable `lists` speichert eine Liste von Listen von Ganzzahlen, zum Beispiel sowas wie `[[1, 2], [3], [-1, 2, 3]]`. Schreiben Sie Python-Kode der eine neue Liste `flat` erstellt, die eine Liste von Ganzzahlen ist und die selben Elemente wie `lists`, einhaltet, zum Beispiel `[1, 2, 3, -1, 2, 3]`. Mit anderen Worten, Ihr Kode muss eine beliebige Liste-von-Listen-von-Ganzzahlen `lists` nehmen nd eine Liste `flat` mit den selben Ganzzahl-Elementen erstellen.

17.5 Programming / Programmieren (Q89)

$$\text{mean}(A) = \frac{1}{n}\sum_{i=0}^{n-1} a_i \quad \text{for a data sample} \quad A = (a_0, a_1, \ldots, a_{n-1}) \tag{17.1}$$

EN The *arithmetic mean* $\text{mean}(A)$ is an estimate of the expected value of a distribution from which a data sample A was sampled. It is computed as shown in Equation 17.1.

- Given is a `list` or `tuple` variable `A` containing integer values.
- Write `Python` code that computes and `prints` the arithmetic mean of the values in `A`.
- Do not use any built-in `Python` functions for summation or for computing the arithmetic mean.
- Handle the special case of an empty `list` or `tuple` in a reasonable way.
- Do not use any of the statistical functions or packages available in `Python`.

DE Der *arithmetische Mittelwert* $\text{mean}(A)$ ist eine Näherung für den Erwartungswert einer stochastische Verteilung von der ein Datensatz A gesampled wurde. Er wird nach Equation 17.1 berechnet.

- Gegeben sei eine Variable `A` vom Typ `list` oder `tuple` mit Ganzzahlen.
- Schreiben Sie `Python`-Kode, der den arithmetischen Mittelwert der Werte in `A` berechnet und mit `print` ausgibt.
- Verwenden Sie keine built-in `Python`-Funktionen zum Berechnen der Summe oder des Mittelwerts.
- Beachten und behandeln Sie den Sonderfall einer leeren Liste bzw. eines leeren Tupels auf sinnvolle Art.
- Nutzen Sie keine der statistischen Funktionen oder Pakete, die es in `Python` gibt.

17.6 Programming / Programmieren (Q90)

EN The variable `numbers` holds a list of integer numbers. Write some `Python` code that prints all the numbers in `numbers` that are not divisible by any of the other numbers in `numbers`. For example, if `numbers = [3, 6, 12, 3, 7, 5, 8, 2]`, it would print the numbers 7, 5, and 2.

DE In der Variable `numbers` ist eine Liste von Ganzzahlen gespeichert. Schreiben Sie `Python`-Kode der alle Zahlen in `numbers` ausdruckt, die nicht durch irgendeine andere Zahl in `numbers` teilbar sind. Zum Beispiel für `numbers = [3, 6, 12, 3, 7, 5, 8, 2]` würde Ihr Kode die Zahlen 7, 5, und 2 ausgeben.

17.7 Programming / Programmieren (Q91)

$$^\circ F = ^\circ C * 1.8 + 32,0 \tag{17.2}$$
$$K = ^\circ C + 273.15 \tag{17.3}$$

EN Write a program that prints a conversation table from degree centigrade (i.e., Celsius, $^\circ C$) to Fahrenheit ($^\circ F$) and Kelvin (K) [119, 249, 283]. The table should go from $-15^\circ C$ to $100^\circ C$ in steps of $5^\circ C$. Use a `for` loop over a `range` with start, end, and step value.

DE Schreiben Sie ein Programm, das eine Umrechnungstabelle von Grad Celsius ($^\circ C$) nach Fahrenheit ($^\circ F$) sowie Kelvin (K) ausgibt [119, 249, 283]. Die Tabelle soll dabei von $-15^\circ C$ bis $100^\circ C$ in Schritten von $5^\circ C$ berechnet werden. Benutzen Sie eine `for`-Schleife über eine `range` mit Start- und End-Wert sowie Schrittweite.

17.8 Programming / Programmieren (Q92)

Listing 17.3: The output of a program `for_03.py`

```
1: *
2: **
3: ***
4: ****
5: *****
6: ******
7: *******
8: ********
9: *********
```

EN Write a Python program that produces exactly the output given in Listing 17.3. This includes the numbers at the beginning of the lines, e.g., `4: ****`, not `****`.

DE Schreiben Sie ein Python-Programm, das genau die Ausgabe gezeigt in Listing 17.3 produziert. Das beinhaltet die Zahlen am Anfang der Linien, also wir wollen wirklich `4: ****`, nicht `****`.

17.9 Programming / Programmieren (Q93)

Listing 17.4: The output of a program `for_04.py`

```
*****
*****
*****
```

EN Write a `Python` program that produces a rectangle of stars based on the values of two variables. The number of stars per row are stored in a variable `n` (in Listing 17.4, that would be 5). The number of rows would be stored in a variable `m` (in Listing 17.4, that would be 3).

DE Schreiben Sie ein `Python`-Program das ein Rechteck aus Sternen produziert, basierend auf den Werten von zwei Variablen. Die Anzahl Sterne pro Zeile sei gegeben in Variable `n` (in Listing 17.4 ist das 5). Die Anzahl von Zeilen sei gegeben in Variable `m` (in Listing 17.4 ist das 3).

17.10 Programming / Programmieren (Q94)

Listing 17.5: The output of a program `for_05.py`

```
XOOOOOX
OXOOOXO
OOXOXOO
OOOXOOO
OOXOXOO
OXOOOXO
XOOOOOX
```

EN Write a `Python` program that produces exactly the output given in Listing 17.5.

DE Schreiben Sie ein `Python`-Programm, das genau die Ausgabe gezeigt in Listing 17.5 produziert.

Chapter 18

While-Loop / While-Schleife

18.1 What is it? / Was ist das? (Q95)

EN Explain the syntax *and* purpose/use of a `while` statement in Python. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `while`-Statements in Python. Sie können ein Beispiel in Ihrer Erklärung verwenden.

18.2 Program Understanding / Programm Verstehen (Q96)

Listing 18.1: The program `while_01.py` (src)

```
1  """ While-Loop Example. """
2
3  a, b, c, d, e = 5, 3, 2, 1, 0
4  counter = 0
5
6  while not (a < b < c < d < e):
7      counter += 1
8      if a > b:
9          a, b = b, a
10     if b > c:
11         b, c = c, b
12     if c > d:
13         c, d = d, c
14     if d > e:
15         d, e = e, d
16
17 print(f"Loop finished after {counter} iterations.")
```

EN Explain program `while_01.py` in Listing 18.1 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `while_01.py` in Listing 18.1 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

18.3 Programming / Programmieren (Q97)

EN The variable `text` holds text string. Write some `Python` code that iterates over the characters in `text` until it either reaches the character `"Q"` or the end of the string and that sums up all digits and prints the result. For example, if `text = "ksd934.df4Qff4"`, it would print `20` and for `text = "f5q6GH"`, it would print `11`.

DE Die Variable `text` speichert einen Text-String. Schreiben Sie `Python`-Kode der über die Zeichen in `text` iteriert bis er entweder das Zeichen `"Q"` oder das Ende des Strings erreicht und alle Ziffern, die er findet, aufaddiert und das Ergebnis ausgibt. Zum Beispiel für `text = "ksd934.df4Qff4"` würde er `20` ausgeben und für `text = "f5q6GH"` würde er `11` ausgeben.

18.4 Programming / Programmieren (Q98)

EN The variable `my_num` holds a positive `int` number. Write the code to print `my_num` as *binary* number, i.e., as sequence of 0s and 1s. Do not use any text formatting or conversion routines. Use a `while` loop to do the necessary computation.

DE Die variable `my_num` speicher eine positive `int`-Zahl. Schreiben Sie den Kode, um `my_num` als *Binärzahl* auszugeben, also als Sequence von 0en und 1en. Benutzen Sie keine Textformatierungsfunktionen oder Konvertierungsfunktionen. Benutzen Sie eine `while`-Schleife für die notwendigen Berechnungen.

18.5 Programming / Programmieren (Q99)

$$a_n = \begin{cases} \frac{1}{2}a_{n-1} & \text{if } a_{n-1} \text{ even} \quad \text{wenn } a_{n-1} \text{ gerade} \\ 3a_{n-1}+1 & \text{if } a_{n-1} \text{ odd} \quad \text{wenn } a_{n-1} \text{ ungerade} \end{cases} \tag{18.1}$$

EN The sequence defined by the equation above is called Collatz Problem [185]. It is believed that, for any positive integer $a_0 \in \mathbb{N}_1$ with $a_0 > 0$, there will eventually be an $a_n = 1$. Expect that a_0 be an integer greater than zero stored in a variable `a`. Write `Python` code that prints all the values of the corresponding Collatz Problem sequence until (and including) `1`.

DE Die Sequence die die Gleichung oben definiert wird Collatz Problem genannt [185]. Es wird angenommen, dass für jede positive Ganzzahl $a_0 \in \mathbb{N}_1$ mit $a_0 > 0$ irgendwann ein $a_n = 1$ erreicht wird. Nehmen Sie an, dass a_0 eine Ganzzahl größer als Null gespeichert in der Variable `a` ist. Schreiben Sie `Python`-Kode der alle Werte der entsprechenden Collatz-Problem-Sequence bis einschließlich `1` ausgibt.

18.6 Programming / Programmieren (Q100)

Listing 18.2: The program `while_03.py` (src)

```
""" Convert a `while`-loop to a `for`-loop. """

index: int = 0
while index < 5:
    print(f"The index is {index}.")
    index += 1
```

EN Change `while_03.py` by converting the `while` loop into a `for` loop.

DE Ändern Sie `while_03.py`, in dem Sie die `while`-Schleife in eine `for`-Schleife umwandeln.

18.7 Programming / Programmieren (Q101)

EN Your bank account at the *Bank of Python* holds `money = 20_000` RMB. Every year, you get `interest`% of interest, i.e., after one year, you have $\lfloor (1 + \texttt{interest}/100)\texttt{money} \rfloor$ RMB. Notice that $\lfloor M \rfloor$ means rounding down to the nearest integer smaller or equal to M. Write `Python` code that counts the number of years you have to wait until there are 100 0000 (= 100万) RMB in your account (if you do not deposit or withdraw any money).

DE Ihr Bankkonto bei der *Bank of Python* hat `money = 20_000` RMB. Jedes Jahr beommen Sie `interest`% Zinsen. Also nach einem Jahr haben Sie $\lfloor (1 + \texttt{interest}/100)\texttt{money} \rfloor$ RMB. Beachten Sie dass $\lfloor M \rfloor$ bedeutet, das zur nächsten Ganzzahl kleiner oder gleich M (ab)gerundet wird. Schreiben Sie `Python`-Kode der die Anzahl von Jahren zählt, die Sie warten müssen, bis 100 0000 (= 100万) RMB in Ihrem Konto sind (wenn Sie weder Geld einzahlen noch abheben).

18.8 Programming / Programmieren (Q102)

Listing 18.3: The program `while_04.py` (src)

```
""" Convert a `while`-loop to a `for`-loop. """

data: list[str] = ["Hello", "World", "I", "am", "here"]

index: int = 0
while index < len(data):
    print(f"The element at index {index} is {data[index]}.")
    index += 1
```

EN Change `while_03.py` by converting the `while` loop into a `for` loop using `enumerate`.

DE Ändern Sie `while_03.py`, in dem Sie die `while`-Schleife in eine `for`-Schleife, die `enumerate` benutzt, umwandeln.

18.9 Programming / Programmieren (Q103)

EN Assume that a positive natural number is stored in Variable `x`. Write `Python` code which finds and prints the largest power of two which is less or equal to `x`. In other words, find value `v` $= 2^i$ with $i \in \mathbb{N}_0$, and `v` $\leq$ `x` and that no larger number exists that fulfills this inequality. For example, for `x=15`, you would print 8, for `x=16`, you would print 16, and for `x=17`, you would also print 16.

DE Nehmen Sie an, dass eine positive natürliche Zahl in der Variable `x` gespeichert ist. Schreiben Sie `Python`-Kode, welcher die größte Zweierpotenz findet, die kleiner oder gleich `x` ist. Mit anderen Worten, finden Sie den Wert `v` $= 2^i$ mit $i \in \mathbb{N}_0$ und `v` $\leq$ `x` so das es keine größere Zahl gibt, die diese Ungleichung erfüllt. Zum Beispiel für `x=15` würden Sie 8 ausdrucken, für `x=16` 16 und für `x=17` würden Sie ebenfalls 16 ausgeben.

18.10 Programming / Programmieren (Q104)

Listing 18.4: The output of a program `while_02.py`

```
start: 20 beetles, V/beetle: 1cm^3, reproduction_rate: 0.8, V/box: 1000000
week 1: 36 beetles, V=36cm^3
week 2: 64 beetles, V=64cm^3
week 3: 115 beetles, V=115cm^3
week 4: 207 beetles, V=207cm^3
week 5: 372 beetles, V=372cm^3
week 6: 669 beetles, V=669cm^3
week 7: 1204 beetles, V=1204cm^3
week 8: 2167 beetles, V=2167cm^3
week 9: 3900 beetles, V=3900cm^3
week 10: 7020 beetles, V=7020cm^3
week 11: 12636 beetles, V=12636cm^3
week 12: 22744 beetles, V=22744cm^3
week 13: 40939 beetles, V=40939cm^3
week 14: 73690 beetles, V=73690cm^3
week 15: 132642 beetles, V=132642cm^3
week 16: 238755 beetles, V=238755cm^3
week 17: 429759 beetles, V=429759cm^3
week 18: 773566 beetles, V=773566cm^3
week 19: 1392418 beetles, V=1392418cm^3
```

EN You buy `n` beetles and put them into a box of volume `box`cm^3. Each beatle has a volume of `b`cm3. The beetles reproduce at a rate of `r` per week, i.e., after one week, you have $\lfloor$`n`$(1+$`r`$)\rfloor$ beetles, and so on. Write a `Python` program that computes how long it takes until the complete box is filled, based on the values of the variables `box`, `n`, `b`, and `r`. For each week, the program should print the number of beetles and their volume. In Listing 18.4 above, you can see an example output for an example setup.

DE Sie kaufem `n` Käfer und tun sie in eine Box mit Volumen `box`cm^3. Jeder Käfer hat ein Volumen von `b`cm3. Die Käfer vermehren sich mit einer Rate `r` pro Woche, also nach einer Woche haben Sie $\lfloor$`n`$(1+$`r`$)\rfloor$ Käfer, und so weiter. Schreiben Sie ein `Python`-Program, das berechnet wie lange es dauert bis die ganze Box mit Käfern gefüllt ist, basierend auf den Variablen `box`, `n`, `b`, und `r` Für jede Woche soll das Programm die Anzahl Käfer und ihr Volumen ausgebene. Listing 18.4 oben ist eine Beispielausgabe, wie sie Ihr Programm liefern könnte.

Chapter 19

Functions / Funktionen

19.1 What is it? / Was ist das? (Q105)

EN Explain the syntax *and* purpose/use of a `def` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `def`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

19.2 Programming / Programmieren (Q106)

EN Implement a function named `power`, which accepts two integer parameters `base` and `exponent` and that returns $\texttt{base}^{\texttt{exponent}}$. In other words, if `base` $= b$ and `exponent` $= y$, then the function computers b^y. The default value for `base` be `2` and the default value for `exponent` be `1`. If `base=0` and a negative number for `power` are supplied, the function should raise a `ValueError`. Make sure to use proper type hints and a docstring.

DE Implementieren Sie eine Funktion mit dem Namen `power`, die die beiden Integer-Parameter `base` und `exponent` akzeptiert und die $\texttt{base}^{\texttt{exponent}}$ zurückliefert. In andren Worten, wenn `base` $= b$ und `exponent` $= y$, dann berechnet die Funktion b^y. Der Default-Wert für `base` ist `2` und der Default-Wert für `exponent` ist `1`. Wenn `base=0` und eine negative Zahl für `power` in die Funktion gegeben werden, soll ein `ValueError` ausgelöst werden. Annotieren Sie die Funktion mit Type Hints und einem DocString.

19.3 Program Understanding / Programm Verstehen (Q107)

Listing 19.1: The program function_01.py (src)

```
""" An example of a function. """

def compute(lst: list[int]) -> list[int]:
    """
    Perform a computation with the list `lst`.

    :param lst: the input list
    :return: the output list
    """
    result: list[int] = []
    copy: list[int] = list(lst)

    while len(copy) > 0:
        x: int = min(copy)
        result.append(x)
        copy.remove(x)

    return result


print(compute([5, 4, 1, 3, 2]))       # Line 21: What does it print?
print(compute([25, -34, 6, -5, 9]))   # Line 22: What does it print?
```

EN Carefully read the program function_01.py in Listing 19.1.

1. What does the function `compute` do?
2. What output does the program produce?
3. Is the program correct, i.e., produces the output that it should? (Justify your answer.)
4. Is the program efficient, i.e., does it do its job without wasting time? (Justify your answer.)

DE Lesen Sie das Programm function_01.py in Listing 19.1 genau.

1. Was macht die Funktion `compute`?
2. Welche Ausgabe produziert das Programm?
3. Ist das Programm korrekt, also produziert es wirklich den gewünschten Output (Rechtfertigen Sie Ihre Antwort.)
4. Ist das Programm effizient, also erfüllt es seine Aufgabe ohne Zeit zu verschwenden? (Rechtfertigen Sie ihre Antwort.)

19.4 Programming / Programmieren (Q108)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{19.1}$$

EN The equation above shows how the binomial coefficient $\binom{n}{k}$ can be computed. Here, $i! = 1 * 2 * 3 * \cdots * i$ is the factorial of a number i. Implement a function `binomial` taking two parameters `n` and `k` and returning $\binom{\texttt{n}}{\texttt{k}}$.

DE Die Gleichung oben zeigt, wie der Binomialkoeffizient $\binom{n}{k}$ berechnet werden kann. Dabei ist $i! = 1 * 2 * 3 * \cdots * i$ die Fakultät der Zahl i. Implementieren Sie eine Funktion `binomial` mit den beiden Parametern `n` und `k`, die $\binom{\texttt{n}}{\texttt{k}}$ zurückliefert.

19.5 Program Understanding / Programm Verstehen (Q109)

Listing 19.2: The program `function_02.py` (src)

```
""" An example of a function. """

def compute(number: int) -> int:
    """
    Perform a computation with the number `number`.

    :param number: the input number
    :return: the output number
    """
    result: int = 0
    while result / number != number:
        result += number

    return result


print(compute(4))  # Line 17: What does it print?
print(compute(7))  # Line 18: What does it print?
print(compute(0))  # Line 19: What does it print?
```

EN Carefully read the program `function_02.py` in Listing 19.2.

1. What does the function `compute` do?
2. What output does the program produce?
3. Is the program correct, i.e., produces the output that it should? (Justify your answer.)
4. Is the program efficient, i.e., does it do its job without wasting time? (Justify your answer.)

DE Lesen Sie das Programm `function_02.py` in Listing 19.2 genau.

1. Was macht die Funktion `compute`?
2. Welche Ausgabe produziert das Programm?
3. Ist das Programm korrekt, also produziert es wirklich den gewünschten Output (Rechtfertigen Sie Ihre Antwort.)
4. Ist das Programm effizient, also erfüllt es seine Aufgabe ohne Zeit zu verschwenden? (Rechtfertigen Sie ihre Antwort.)

19.6 Programming / Programmieren (Q110)

EN Implement a Python function named `sort`, which accepts a list as parameter and sorts this list. **Do not use** any of Python's predefined sorting functions, like `list.sort` or `sorted`. This question is not about efficiency – as long as the list gets sorted, it is OK. Make sure to use proper type hints and a docstring.

DE Implementieren Sie eine Python Funktion names `sort`, die eine List als Parameter akzeptiert und diese Liste sortiert. Benutzen Sie **keine** von Python's vordefinierten Sortierungsfunktionen wie `list.sort` oder `sorted`. Es geht nicht um Effizienz/Schnelligkeit – nur darum, dass die Liste sortiert wird. Annotieren Sie die Funktion mit Type Hints und einem DocString.

19.7 Programming / Programmieren (Q111)

EN Write a Python function `digit_range` that accepts an integer number as parameter and returns the smallest and largest digit of that number in a tuple. For the number `72234`, it would return the tuple `(2, 7)`. Make sure to use proper type hints and a docstring.

DE Schreiben Sie eine Python Funktion `digit_range` die eine Ganzzahl als Parameter akzeptiert und die kleinste und größte Ziffer der Zahl in einem Tupel zurückliefert. Für die Zahl `72234` würde sie das Tupel `(2, 7)` zurückliefern. Annotieren Sie die Funktion mit Type Hints und einem DocString.

19.8 Program Understanding / Programm Verstehen (Q112)

Listing 19.3: The program function_03.py (stored in file function_03.py; output in Listing 19.4)

```
""" Guess the definition of `my_function`. """

from function_03_module import my_function

print(f"{my_function(1, 2, 3) = }")
print(f"{my_function(a=7, b=5, c=6) = }")
print(f"{my_function(9, 4) = }")
print(f"{my_function(b=11, c=-3) = }")
print(f"{my_function(a=-5, b=8) = }")
print(f"{my_function(a=20, c=10) = }")
```

↓ `python3 function_03.py` ↓

Listing 19.4: The stdout of the program function_03.py given in Listing 19.3.

```
my_function(1, 2, 3) = 6
my_function(a=7, b=5, c=6) = 18
my_function(9, 4) = 16
my_function(b=11, c=-3) = 18
my_function(a=-5, b=8) = 6
my_function(a=20, c=10) = 36
```

EN The output of program function_03.py is given in Listing 19.4 above. Write a function `my_function` in Python that would produce exactly that output.

DE Die Ausgabe von Programm function_03.py ist in Listing 19.4 oben gegeben. Schreiben Sie eine Funktion `my_function` in Python die genau diesen Output produzieren würde.

19.9 Programming / Programmieren (Q113)

$$\text{geom}(A) = \sqrt[n]{\prod_{i=0}^{n-1} a_i} = (a_0 * a_1 * \cdots * a_{n-1})^{\frac{1}{n}} \quad \text{for data} \quad A = (a_0, a_1, \ldots, a_{n-1}) \tag{19.2}$$

EN The *geometric mean* $\text{geom}(A)$ is the n^{th} root of the product of n *positive* values in a dataset A. It is computed as shown in Equation 19.2.

- Implement a function that takes a `list` of `float` values as parameter and that returns their geometric mean.
- Handle the special case of an empty `list` in a reasonable way.
- Do not use any of the statistical functions or packages available in Python.

DE Der *geometrische Mittelwert* $\text{geom}(A)$ ist die n-te Wurzel des Products von n positiven Werten in einem Datensatz A. Er wird nach Equation 19.2 berechnet.

- Schreiben Sie eine sei eine Funktion, die eine Liste mit Fließkommawerten (`float`) als Parameter erhält und die deren geometrischen Mittelwert als Ergebnis zurückliefert.
- Beachten und behandeln Sie den Sonderfall einer leeren Liste auf sinnvolle Art.
- Nutzen Sie keine der statistischen Funktionen oder Pakete, die es in Python gibt.

Chapter 20

Unit Tests

20.1 Unit Tests (Q114)

EN What is the goal of unit tests? How can unit tests in `Python` be done?

DE Was ist das Ziel von Unit Tests? Wie kann man Unit Tests in `Python` realisieren?

20.2 Program Understanding / Programm Verstehen (Q115)

Listing 20.1: The program tests_01.py (src)

```
"""Working with tests."""

def my_isqrt(a: int) -> int:
    """
    An implementation of the integer square root of numbers.

    :param a: an integer number
    :return: the integer square root of that number
    """
    result: int = 0
    while result * result < a:
        result += 1

    return result
```

Listing 20.2: The unit test program tests_01_test.py (src)

```
"""Test our function."""

from tests_01 import my_isqrt

def test_my_sqrt() -> None:
    """Test `my_sqrt`."""
    assert my_isqrt(0) == 0
    assert my_isqrt(1) == 1
    assert my_isqrt(3) == 1
    assert my_isqrt(4) == 2
    assert my_isqrt(8) == 2
    assert my_isqrt(9) == 3
    assert my_isqrt(16) == 4
```

Listing 20.3: The output of pytest when using tests_01_test.py

```
$ pytest --timeout=10 --no-header --tb=short tests_01_test.py
========================== test session starts ==========================
collected 1 item
tests_01_test.py F                                                  [100%]
============================== FAILURES ==============================
_____________________________ test_my_sqrt _____________________________
tests_01_test.py:9: in test_my_sqrt
    assert my_isqrt(3) == 1
E   assert 2 == 1
E    +  where 2 = my_isqrt(3)
======================== short test summary info ========================
FAILED tests_01_test.py::test_my_sqrt - assert 2 == 1
 +  where 2 = my_isqrt(3)
=========================== 1 failed in 0.02s ===========================
# pytest 9.2.0 with pytest-timeout 2.4.0 failed with exit code 1.
```

EN

1. Explain function `my_isqrt` in module tests_01.py *line-by-line*.
2. Explain function `test_my_sqrt` in module tests_01_test.py. What is it for?
3. Explain the output of pytest executing tests_01_test.py given in Listing 20.3.

DE

1. Erklären Sie die Funktion `my_isqrt` in Modul tests_01.py Zeile für Zeile.
2. Erklären Sie die Funktion `test_my_sqrt` in Modul tests_01_test.py. Wofür ist sie da?
3. Erklären Sie die Ausgabe von pytest, was tests_01_test.py ausgeführt hat, die in Listing 20.3 gelistet ist.

20.3 Timeouts / Zeitlimits (Q116)

EN Why is it always a good idea to define a timeout for unit tests?

DE Warum ist es immer eine gute Idee, ein Zeitlimit für Unit Tests zu definieren?

20.4 Timeouts / Zeitlimits (Q117)

EN What are doctests? Explain two benefits of annotating functions with doctests.

DE Was sind Doc-Tests? Erklären Sie zwei Vorteile, die das Annotieren von Funktionen mit Doc-Tests bringt.

20.5 Programming / Programmieren (Q118)

EN The function `def check_div_3(number: int |float)-> bool` checks whether a number `number` is divisible by `3`, i.e., if `number %3 == 0`. If yes, it is supposed to return `True`, otherwise `False`. Write unit tests to check the function. For each test case that you define, explain why you choose it.

DE Die Funktion `def check_div_3(number: int |float)-> bool` prüft ob eine Zahl `number` durch `3` teilbar ist, also ob `number %3 == 0`. Wenn ja, dann soll sie `True` zurückliefern, andernfalls `False`. Schreiben Sie Unit Tests um diese Funktion zu prüfen. Für jeden Testfall, den Sie definieren, erklären Sie, warum Sie ihn ausgewählt haben.

20.6 Programming / Programmieren (Q119)

$$\log x \approx 3(x^2 - 1)/((x + 1)^2 + 2x) \tag{20.1}$$

EN The equation above was given by Kellogg in [173] as an approximation for the natural logarithm of x, i.e., $\log x$, for values of x close to 1. This means that it does *not* return the values of $\log x$ exactly, but it computes values very close to them. And this only works if $x \approx 1$. But it has the advantage that it is fast.

1. Implement the right-hand-side of the equation as a function.
2. Annotate the function with type hints and a docstring.
3. How would you test this function? Remember that it only is an approximation. Write some doctests in the docstring.

DE Die Gleichung oben wurde von Kellogg in [173] als eine Annäherung des natürlichen Logarithmus von x, also $\log x$, für Werte von x nahe 1. Das bedeutet, dass die Formel *nicht* die exakten Werte von $\log x$ berechnet, sondern nur Werte *nahe* an ihnen. Das funktioniert auch nur für $x \approx 1$. Aber es ist schnell.

1. Implementieren Sie die rechte Seite der Gleichung als eine Funktion.
2. Annotieren Sie die Funktion mit Type-Hints und einem DocString.
3. Wie würden Sie diese Funktion testen? Bedenken Sie, dass sie nur eine Näherung ist. Schreiben Sie ein paar DocTests in den DocString.

20.7 Program Understanding / Programm Verstehen (Q120)

Listing 20.4: The program doctest_01.py (src)

```
""" An example of a function. """

from math import sqrt

def solve(a: float, b: float, c: float) -> tuple[float, float] | float:
    """
    Solve the quadratic equation `0=a*x²+b*x+c`.

    We use the formula `x = (-b +/- sqrt(b² - 4ac))/(2a)`.
    Due to the `+/-`, there are two solutions if `sqrt(b² - 4ac) != 0`.

    :param a: the coefficient `a`: We have `a*x²`.
    :param b: the coefficient `b`: We have `+b*x`.
    :param c: the coefficient `c`: We have `+c`.
    :return: either a tuple of two solutions or a single solution.

    >>> solve(2.0, -4.0, 2.0)
    1.0
    >>> solve(2.0, 0.0, -8.0)
    (-2.0, 2.0)
    >>> solve(2.0, -8.0, 6.0)
    (1.0, 3.0)
    """
    term = sqrt(b ** 2 - 4 * a * c)
    if term == 0.0:          # Check if there is only 1 solution.
        return -b / (2 * a)  # There is only 1 solution.
    return (-b - term) /(2*a), (-b * term) /(2*a)  # Return 2 solutions.
```

Listing 20.5: The output of pytest when running the doctests of doctest_01.py

```
$ pytest --timeout=10 --no-header --tb=short --doctest-modules doctest_01.
    ↪ py
============================ test session starts ============================
collected 1 item
doctest_01.py F                                                        [100%]
================================ FAILURES =================================
______________________ [doctest] doctest_01.solve _______________________
010 Due to the `+/-`, there are two solutions if `sqrt(b² - 4ac) != 0`.
011
012 :param a: the coefficient `a`: We have `a*x²`.
013 :param b: the coefficient `b`: We have `+b*x`.
014 :param c: the coefficient `c`: We have `+c`.
015 :return: either a tuple of two solutions or a single solution.
016
017 >>> solve(2.0, -4.0, 2.0)
018 1.0
019 >>> solve(2.0, 0.0, -8.0)
Expected:
    (-2.0, 2.0)
Got:
    (-2.0, -0.0)
/home/runner/work/programmingWithPythonQuestions/
    ↪ programmingWithPythonQuestions/__git__/realms/git/
    ↪ gh_thomasWeise_programmingWithPythonCode/questions/doctest_01.py:19:
    ↪ DocTestFailure
======================== short test summary info ========================
FAILED doctest_01.py::doctest_01.solve
============================ 1 failed in 0.12s ============================
# pytest 9.2.0 with pytest-timeout 2.4.0 failed with exit code 1.
```

20.7 Program Understanding / Programm Verstehen (Q120) (continued/fortgesetzt)

EN In `doctest_01.py`, we implemented code to solve the quadratic equation $0 = ax^2 + bx + c$ by using $x_{0/1} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. We also implemented several doctests to check our implementation. Listing 20.5 shows the output that `pytest` produces when we run these doctests.

1. Explain the output of `pytest`.
2. If there is an error in `doctest_01.py`, then find it and propose how to fix it.
3. Propose how we could handle parameter values such as
   - `a` $= 0$ and
   - `b`$^2 - 4$`a c` < 0.

DE In `doctest_01.py` haben wir Kode implementiert um die quadratische Gleichung $0 = ax^2 + bx + c$ zu lösen, in dem wir $x_{0/1} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ verwenden. Wir haben auch mehrere DocTests bereitgestellt, um unsere Implementierung zu testen. Listing 20.5 zeigt die Ausgaben, die `pytest` produziert, wenn wir diese doctests ausführen.

1. Erklären Sie die Ausgaben von `pytest`.
2. Wenn ein Fehler in `doctest_01.py` ist, dann finden Sie ihn und schlagen Sie vor, wie er korrigiert werden könnte.
3. Schlagen Sie vor, wie wir Parameterwerte wie
   - `a` $= 0$ und
   - `b`$^2 - 4$`a c` < 0 behandeln könnten.

Chapter 21

Exceptions / Ausnahmen

21.1 What is it? / Was ist das? (Q121)

EN What are `Exceptions` in `Python`?

DE Was sind `Exceptions`, also Ausnahmen, in `Python`?

21.2 What is it? / Was ist das? (Q122)

EN Explain the syntax *and* purpose/use of a `try-except` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `try-except`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

21.3 What is it? / Was ist das? (Q123)

EN Explain the syntax *and* purpose/use of a `try-finally` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `try-finally`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

21.4 What is it? / Was ist das? (Q124)

EN Explain the syntax *and* purpose/use of a `with` statement in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck eines `with`-Statements in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

21.5 General Understanding / Allgmeines Verständnis (Q125)

EN Why is Garbage In–Garbage Out (GIGO) a bad design choice?

DE Warum ist Garbage In–Garbage Out (GIGO), also "falsche Eingabedaten führen zu falschen Ausgabedaten", eine schlechte Designentscheidung?

21.6 General Understanding / Allgmeines Verständnis (Q126)

EN Why is it bad to sanitize incorrect input data into valid data a bad design choice?

DE Warum ist es eine schlechte Designentscheidung, fehlerhafte Eingabedaten zu reparieren, also in richtige Daten umzuwandeln?

21.7 Program Understanding and Programming / Programm Verstehen und Programmieren (Q127)

Listing 21.1: The program except_02.py and its output. (stored in file except_02.py; output in Listing 21.2)

```
# This program has an error.

def is_prime(value: int) -> bool:
    for v in range(value):
        if value % v == 0:
            return False
    return True

print(is_prime(7))
print(is_prime(10))
```

↓ `python3 except_02.py` ↓

Listing 21.2: The stdout, standard error stream (stderr), and exit code of the program except_02.py given in Listing 21.1.

```
Traceback (most recent call last):
  File "{...}/questions/except_02.py", line 9, in <module>
    print(is_prime(7))
          ~~~~~~~~~~^
  File "{...}/questions/except_02.py", line 5, in is_prime
    if value % v == 0:
       ~~~~~~~~^~~
ZeroDivisionError: division by zero
# 'python3 except_02.py' failed with exit code 1.
```

EN The function in the program at the top is supposed to check whether a number is a prime number. However, if we run the program, it produces the given output.

1. What does this output mean?
2. What is a `ZeroDivisionError`?
3. Which line and command causes this output? Why does it do this?
4. Which output would we have expected instead?
5. How can we change the function in order to produce this expected output?

DE Die Funktion im Programm oben soll prüfen, ob eine Zahl eine Primzahl ist. Wenn wir das Programm jedoch ausführen, dann bekommen wir die Ausgabe oben.

1. Was bedeutet diese Ausgabe?
2. Was ist ein `ZeroDivisionError`?
3. Welche Zeile und Anweisung löst diese Ausgabe aus? Warum tut sie das?
4. Welche Ausgabe hätte man eigentlich erwartet?
5. Wie können wir die Funktion verändern, so dass diese Ausgabe auch kommt?

21.8 Programming / Programmieren (Q128)

EN Implement a function that takes a string parameter as input and raises a `ValueError` if its argument is empty or longer than 1000 characters. Otherwise, it should return the `True`. Make sure to use proper type hints and a docstring.

DE Implementieren Sie eine Funktion die einen String/Zeichenketten-Parameter als Eingabe akzeptiert und einen `ValueError` auslöst wenn ihr Argument leer oder länger als 1000 Zeichen ist. Andernfalls soll sie `True` zurückliefern. Annotieren Sie die Funktion mit Type Hints und einem DocString.

21.9 Program Understanding and Programming / Programm Verstehen und Programmieren (Q129)

Listing 21.3: The program except_01.py and its output. (stored in file except_01.py; output in Listing 21.4)

```
"""Safe factorial."""

def factorial(number: int) -> int:
    """
    Compute the factorial of a number. DO NOT CHANGE THIS FUNCTION.

    :param number: the number to compute the factorial of
    :return: `number!` for `number in 0...12`.
    :raises ValueError: if `number < 0` or `number > 12`.
    """
    if (number < 0) or (number > 12):
        raise ValueError(
            f"Cannot reasonably compute factorial of {number}.")
    for i in range(2, number):
        number *= i
    return number

# Modify this loop by using exception handling in order to prevent the
# program for crashing. Don't change the range of values `num` takes on!
for num in range(2, 20):
    print(f"{num}! = {factorial(num)}")
```

↓ `python3 except_01.py` ↓

Listing 21.4: The stdout, stderr, and exit code of the program except_01.py given in Listing 21.3.

```
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
Traceback (most recent call last):
  File "{...}/questions/except_01.py", line 21, in <module>
    print(f"{num}! = {factorial(num)}")
                      ~~~~~~~~~~~~~^^
  File "{...}/questions/except_01.py", line 12, in factorial
    raise ValueError(
        f"Cannot reasonably compute factorial of {number}.")
ValueError: Cannot reasonably compute factorial of 13.
# 'python3 except_01.py' failed with exit code 1.
```

EN Modify the `for`-loop in Line 20 of program except_01.py by using exception handling such that the program does no longer crash. In case of a `ValueError`, a proper error message should be printed. Do **not** change the function `factorial` and also **do not** change any loop condition!

DE Verändern Sie die `for`-Schleife in Zeile 20 des Programs except_01.py in dem Sie Ausnahmebehandlung verwenden, so dass das Program nicht mehr abstürzt. Im Falle eines `ValueError` soll eine vernünftige Fehlermeldung ausgegeben werden. Verändern Sie **nicht** die Funktion `factorial` und verändern Sie **auch nicht** die Schleifenbedingungen!

21.10 Program Understanding and Programming / Programm Verstehen und Programmieren (Q130)

Listing 21.5: The program except_03.py and its output. (stored in file except_03.py; output in Listing 21.6)

```
# This program has an error.

def largest_divisor(value: int)->int:
    largest: int = -1
    for v in range(value):
        if value % v == 0:
            largest = v
    return largest

print(largest_divisor(10))
print(largest_divisor(16))
```

↓ `python3 except_03.py` ↓

Listing 21.6: The stdout, stderr, and exit code of the program except_03.py given in Listing 21.5.

```
Traceback (most recent call last):
  File "{...}/questions/except_03.py", line 10, in <module>
    print(largest_divisor(10))
          ~~~~~~~~~~~~~~~~~~^^^^
  File "{...}/questions/except_03.py", line 6, in largest_divisor
    if value % v == 0:
       ~~~~~~~~^~~~~~
ZeroDivisionError: division by zero
# 'python3 except_03.py' failed with exit code 1.
```

EN The function in the program at the top is supposed to check whether a number is a prime number. However, if we run the program, it produces the given output.

1. What does this output mean?
2. What is a `ZeroDivisionError`?
3. Which line and command causes this output? Why does it do this?
4. Which output would we have expected instead?
5. How can we change the function in order to produce this expected output?

DE Die Funktion im Programm oben soll prüfen, ob eine Zahl eine Primzahl ist. Wenn wir das Programm jedoch ausführen, dann bekommen wir die Ausgabe oben.

1. Was bedeutet diese Ausgabe?
2. Was ist ein `ZeroDivisionError`?
3. Welche Zeile und Anweisung löst diese Ausgabe aus? Warum tut sie das?
4. Welche Ausgabe hätte man eigentlich erwartet?
5. Wie können wir die Funktion verändern, so dass diese Ausgabe auch kommt?

Chapter 22

Comprehension

22.1 What is it? / Was ist das? (Q131)

EN Explain the syntax *and* purpose/use of list comprehension in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck von List-Comprehension in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

22.2 What is it? / Was ist das? (Q132)

EN Explain the syntax *and* purpose/use of set comprehension in `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck von set/Mengen-Comprehension in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

22.3 What is it? / Was ist das? (Q133)

EN Explain the syntax *and* purpose/use of dictionary comprehension `Python`. You can use an example for your explanation.

DE Erklären Sie die Syntax *und* den Verwendungszweck von Dictionary-Comprehension in `Python`. Sie können ein Beispiel in Ihrer Erklärung verwenden.

22.4 Programming / Programmieren (Q134)

EN Construct a list containing all tuples of the form (i, j) for the values of $i \in 1..3$ and $j \in 0..4$ using list comprehension.

DE Konstruieren Sie eine Liste mit allen Tupeln der Form (i, j) für die Werte von $i \in 1..3$ und $j \in 0..4$ mit Hilfe von List-Comprehension.

22.5 Programming / Programmieren (Q135)

EN Use dictionary comprehension to create a dictionary with the numbers q ranging from 11 to 90 as keys and the values $120 - q$.

DE Verwenden Sie Dictionary Comprehension, um ein Dictionary zu erstellen, dessen Schlüssel die Zahlen q von 11 bis 90 sind und die Werte jeweils $120 - q$.

22.6 Programming / Programmieren (Q136)

```
[[0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2]]
```

EN Write a list comprehension expression that produces the list above.

DE Schreiben Sie einen List-Comprehension Ausdruck, der die Liste oben produziert.

22.7 Programming / Programmieren (Q137)

EN Use dictionary comprehension to create a dictionary with the numbers t ranging from 2 to 912 as keys and the values t^3.

DE Verwenden Sie Dictionary Comprehension, um ein Dictionary zu erstellen, dessen Schlüssel die Zahlen t von 2 bis 912 sind und die Werte jeweils t^3.

22.8 Programming / Programmieren (Q138)

```
[(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

EN Write a list comprehension expression that produces the list above.

DE Schreiben Sie einen List-Comprehension Ausdruck, der die Liste oben produziert.

Chapter 23

Classes / Klassen

23.1 What is it? / Was ist das? (Q139)

EN Show the syntax of classes in Python. Include and additionally explain the following elements

1. the class name,
2. the initializer method,
3. the attributes as well as how and where they declared/initialized,
4. methods of a class,
5. the use of `self`.

DE Zeigen Sie die Syntax von Klassen in Python. Beinhalten und erklären Sie die folgenden Elemente

1. den Name der Klasse,
2. die Initialisierer-Methode,
3. die Attribute sowohl wie und wo sie deklariert/initialisiert werden,
4. Methoden,
5. die Verwendung von `self`.

23.2 Use Case? / Nutzen? (Q140)

EN Name and explain two general use cases of classes.

DE Nennen und erklären Sie zwei generelle Szenarien, in denen man Klassen verwenden kann/sollte.

23.3 What is it? / Was ist das? (Q141)

EN Explain what *immutable* class (instances) are:

1. When is a class instance immutable?
2. What do we need to do in `Python` so that a class instance is annotated to be immutable?
3. Give two advantages of making classes immutable.

DE Erklären Sie, was *unveränderliche* Klassen(-Instanzen) sind:

1. Wann ist eine Klassen(-Instanz) unveränderlich?
2. Was What do we need to do in `Python` so dass eine Klasse-Instanz als unveränderlich annotiert ist?
3. Nennen Sie zwei Vorteile von unveränderlichen Klassen.

23.4 Syntax (Q142)

EN In which situation would we start the name of an attribute of a class with two underscore characters, e.g., `__counter`?

DE In welcher Situation würden wir den Name eines Attributs einer Klasse mit zwei Unterstrichen beginnen, zum Beispiel `__counter`?

23.5 Programming / Programmieren (Q143)

Listing 23.1: The program `class_01.py` (stored in file `class_01.py`; output in Listing 23.2)

```
"""Using the class `Person` class."""

from class_01_module import Person

p1 = Person("Voltaire", "1694-11-21")
p1.introduce()

p2 = Person("Confucius", "551 BCE")
p2.introduce()
```

↓ `python3 class_01.py` ↓

Listing 23.2: The `stdout` of the program `class_01.py` given in Listing 23.1.

```
Hello. My name is Voltaire and my birthday is 1694-11-21.
Hello. My name is Confucius and my birthday is 551 BCE.
```

EN The output of program `class_01.py` is given in Listing 23.2 above. It imports a class `Person` from another module. Write a class `Person` in `Python` that has the necessary attributes, initializer, and method to produce exactly the observed behavior and output.

DE Die Ausgabe des Programms `class_01.py` ist oben in Listing 23.2 gegeben. Es importiert die Klasse `Person` aus einem anderen Modul. Schreiben Sie die Klasse `Person` in `Python` mit den notwendigen Attributen, den notwendigen Initialisierer und der notwendigen Methode so dass genau das beobachtete Verhalten bzw. die beobachtete Ausgabe entsteht.

23.6 Programming / Programmieren (Q144)

$$z_1 + z_2 = (a_1 + b_1\mathrm{i}) + (a_2 + b_2\mathrm{i}) = (a_1 + a_2) + (b_1 + b_2)\mathrm{i} \tag{23.1}$$

$$z_1 - z_2 = (a_1 + b_1\mathrm{i}) - (a_2 + b_2\mathrm{i}) = (a_1 - a_2) + (b_1 - b_2)\mathrm{i} \tag{23.2}$$

$$z_1 * z_2 = (a_1 + b_1\mathrm{i}) * (a_2 + b_2\mathrm{i}) = (a_1a_2 - b_1b_2) + (a_1b_2 + a_2b_1)\mathrm{i} \tag{23.3}$$

$$\overline{z_1} = \overline{a_1 + b_1\mathrm{i}} = a_1 - b_1\mathrm{i} \quad (\textit{conjugate}) \tag{23.4}$$

EN The above equations demonstrate mathematical operations for one to two complex numbers $z_1, z_2 \in \mathbb{C}$ such that $z_1 = a_1 + b_1\mathrm{i}$ and $z_2 = a_2 + b_2\mathrm{i}$, where i is the imaginary unit with $\mathrm{i}^2 = -1$ [317].

1. Write a `Python` class for representing complex numbers.
2. The instances of the class should be immutable.
3. The class should implement the four operations `plus`, `minus`, `multiply`, and `conjugate` as defined by the above equations.
4. The class must be annotated with a docstring which contains a `doctest`.
5. All attributes of the class must be annotated with type hints.

DE Die Gleichungen oben stellen Operationen für ein bis zwei komplexe Zahlen $z_1, z_2 \in \mathbb{C}$ mit $z_1 = a_1 + b_1\mathrm{i}$ und $z_2 = a_2 + b_2\mathrm{i}$ dar, wobei i die imaginäre Einheit mit $\mathrm{i}^2 = -1$ ist [317].

1. Schreiben Sie eine `Python`-Klasse um komplexe Zahlen darstellen zu können.
2. Die Instanzen der Klasse sollen unveränderlich sein.
3. Die Klasse soll die vier Operationen `plus`, `minus`, `multiply` und `conjugate` wie in den Gleichungen oben definiert implementieren.
4. Die Klasse soll mit einem DocString annotiert werden, der einen DocTest beinhaltet.
5. Alle Attribute der Klasse müssen mit Type-Hints annotiert werden.

23.7 Programming / Programmieren (Q145)

$$q_1 + q_2 = \frac{a_1}{b_1} + \frac{a_2}{b_2} = \frac{a_1 b_2 + a_2 b_1}{b_1 b_2} \quad \text{reduce/kürzen!} \tag{23.5}$$

$$q_1 - q_2 = \frac{a_1}{b_1} - \frac{a_2}{b_2} = \frac{a_1 b_2 - a_2 b_1}{b_1 b_2} \quad \text{reduce/kürzen!} \tag{23.6}$$

$$q_1 * q_2 = \frac{a_1}{b_1} * \frac{a_2}{b_2} = \frac{a_1 a_2}{b_1 b_2} \quad \text{reduce/kürzen!} \tag{23.7}$$

$$q_1 / q_2 = \frac{a_1}{b_1} / \frac{a_2}{b_2} = \begin{cases} \texttt{ZeroDivisionError} & \text{if/wenn } q_2 = 0 \\ \frac{a_1 b_2}{b_1 a_2} & \text{otherwise/else} \end{cases} \quad \text{reduce/kürzen!} \tag{23.8}$$

EN The above equations demonstrate mathematical operations for two rational numbers $q_1, q_2 \in \mathbb{Q}$ such that $q_1 = \frac{a_1}{b_1}$ and $q_2 = \frac{a_2}{b_2}$. In other words, q_1 and q_2 are fractional numbers where $a_1, a_2, b_1, b_2 \in \mathbb{Z}$ (integer numbers) and $b1, b2 \neq 0$.

1. Write a `Python` class for representing rational numbers.
2. The class should properly represent 0 and negative numbers.
3. The class should always ensure that the fractions are always reduced as far as possible. In other words, with the exception of 0, it holds that $gcd(a, b) = 1$ for all $\frac{a}{b}$. (You can use the function `gcd` from the module `math`.)
4. The instances of the class should be immutable.
5. The class should implement the four operations `plus`, `minus`, `multiply`, and `divide` as defined by the above equations.
6. A division by 0 should raise an `ZeroDivisionError`.
7. The class must be annotated with a docstring which contains a `doctest`.
8. All attributes of the class must be annotated with type hints.

DE Die Gleichungen oben demonstrieren die mathematischen Operationen für zwei rationale Zahlen $q_1, q_2 \in \mathbb{Q}$ mit $q_1 = \frac{a_1}{b_1}$ und $q_2 = \frac{a_2}{b_2}$. Mit anderen Worten, q_1 und q_2 sind Brüche mit $a_1, a_2, b_1, b_2 \in \mathbb{Z}$ (sind Ganzzahlen) und $b_1, b_2 \neq 0$.

1. Schreiben Sie eine `Python`-Klasse, um rationale Zahlen darzustellen.
2. Die Klasse sollte sowohl 0 als auch negative Zahlen korrekt darstellen.
3. Die Klasse sollte die Brüche immer so weit wie möglich gekürzt darstellen. Also mit der Ausnahme von 0 gilt immer das $gcd(a, b) = 1$ für alle $\frac{a}{b}$. (Sie können die Funktion `gcd` aus dem Modul `math` verwenden.)
4. Die Instanzen der Klasse sollen unveränderlich sein.
5. Die Klasse soll die vier Operationen `plus`, `minus`, `multiply` und `divide` wie in den Gleichungen oben definiert implementieren.
6. Eine Division durch 0 sollte einen `ZeroDivisionError` auslösen.
7. Die Klasse soll mit einem DocString annotiert werden, der einen DocTest beinhaltet.
8. Alle Attribute der Klasse müssen mit Type-Hints annotiert werden.

23.8 Programming / Programmieren (Q146)

EN Let's build a class hierarchy for life forms. Write code for everything below.

1. The base class be `LifeForm`, and it has one attribute *name*, which represents the name of a life form as string.
2. The direct subclasses of `LifeForm` are `Plant` and `Animal`.
3. The class `Plant` has a method `grow`, which prints `"(name) grows!"`, where `(name)` is to be replaced by the value of the *name* attribute.
4. The class `Animal` has a method `eat`, which prints `"(name) eats!"`, where `(name)` is to be replaced by the value of the *name* attribute.
5. Write code that instantiates `Plant` and creates a plant named `Willi`. Call its `grow` method.
6. Create a subclass of `Animal` named `Lion`. This subclass gets a method `do_something(self, hungry: bool)`, which invokes the `eat` method if `hungry == True`.

DE Lassen Sie uns eine Klassenhierarchie für Lebensformen bauen. Schreiben Sie Kode für *alles* Folgendes:

1. Die Basisklasse ist `LifeForm` und sie hat ein Attribut *name*, welches den Name der Lebensform als String repräsentiert.
2. Die direkten Subklassen von `LifeForm` sind `Plant` (DE: *Pflanze*) und `Animal` (DE: *Tier*).
3. Die Klasse `Plant` hat eine Methode `grow`, die ausgibt `"(name) grows!"`, wobei `(name)` mit dem Wert der *name*-Attributs zu ersetzen ist.
4. Die Klasse `Animal` hat eine Methode `eat`, die ausgibt `"(name) eats!"`, wobei `(name)` mit dem Wert des Attributs *name* zu ersetzen ist.
5. Schreiben Sie Kode der die Klasse `Plant` instantiiert und eine Pflanze namens `Willi` erzeugt. Rufen Sie deren `grow`-Methode auf.
6. Erstellen Sie eine Subklasse von `Animal`, die Sie `Lion` (DE: *Löwe*) nennen. Diese Subklasse bekommt eine Methode `do_something(self, hungry: bool)`, die die `eat`-Methode aufruft wenn `hungry == True`.

Chapter 24

Underhanded / Hinterhältig

EN Critical reading of code is very important. It can help us to spot errors. Therefore, in this section, we try to manually find situations in which code will behave unexpectedly. This will deeply improve understanding of code, the importance of checking parameters, as well as our debugging skills.

DE Das kritische Lesen von Kode ist sehr wichtig. Es kann uns helfen, Fehler zu finden. Deshalb versuchen wir in diesem Abschnitt per Hand Situationen zu finden, in denen sich Kode auf unerwartete Art verhält. Das wird unser Verständnis von Kode, der Wichtigkeit vom Prüfen von Parameterwerten, sowie unsere Debugging/Fehlersuche-Skills signifikant verbessern.

24.1 Underhanded Python (Q147)

Listing 24.1: The program underhanded_01.py (src)

```python
"""
Underhanded Python.

Find am argument value `X` for parameter `temperature` for which
`check_temperature` returns `True` even though `X` is not a floating
point number from the interval [0.0, 100.0].
"""

def check_temperature(temperature: int | float) -> bool:
    """
    Check whether a temperature setting for a water cooker is OK.

    :param temperature: the goal temperature that should be set
    :return: `True` if the temperature is permitted, `False` if not.
    """
    if (temperature < 0.0) or (temperature > 100.0):
        return False
    return True
```

EN The Python file underhanded_01.py above implements a function `check_temperature`, which is supposed to return `True` if a `temperature` is from the interval $[0, 100]$ and `False` otherwise.

1. Find a value X for parameter `temperature` for which the function `check_temperature` returns `True` even though X is *not* a floating point number from the interval $[0.0, 100.0]$.
2. How could we change the function to protect it against this?

DE Die Python-Datei underhanded_01.py oben implementiert eine Funktion `check_temperature`, die `True` zurückliefern soll, wenn eine `temperature` aus dem Intervall $[0, 100]$ ist und sonst `False`.

1. Finden Sie einen Wert X für Parameter `temperature` für den die Funktion `check_temperature` `True` liefert, obwohl X *keine* Fließkommazahl aus dem interval $[0.0, 100.0]$ ist.
2. Wie könnten wir die Funktion verändern, so dass das nicht mehr geht?

24.2 Underhanded Python (Q148)

Listing 24.2: The program underhanded_02.py (src)

```python
"""
Underhanded Python.

Find arguments `lower` and `upper` for which the function `all_squares`
never returns.
"""
from math import isqrt


def all_squares(lower: int, upper: int) -> list[int]:
    """
    Compute the list of all square numbers in `[lower,upper)`.

    :param lower: the lower (inclusive) interval end
    :param upper: the upper (exclusive) interval end
    :return: the list of all square numbers `s` with
        `lower <= s < upper`.

    >>> all_squares(1, 20)
    [1, 4, 9, 16]
    >>> all_squares(10, 36)  # 36 is not included: exclusive upper end!
    [16, 25]
    >>> all_squares(10, 10)
    []
    """
    # Perform a sanity check of the interval boundaries:
    # Negative lower limits are forbidden and so are large upper limits.
    if (lower <= 0) or (upper > 1_000_000) or (not isinstance(
            lower, int)) or (not isinstance(upper, int)):
        raise ValueError("Illegal Arguments!")

    result: list[int] = []  # Start with empty list.
    while lower != upper:   # We use `lower` as the variable to iterate.
        if (isqrt(lower) ** 2) == lower:  # Check if lower is a square.
            result.append(lower)  # It is! Add it to the list.
        lower += 1          # Increment lower.
    return result           # Return the list of all the square numbers.
```

EN The Python file underhanded_02.py above provides a function `all_squares`, which returns a list of all square numbers from the interval `lower`..`upper` − 1. A square number here be a number $z = x^2$ with $x \in \mathbb{N}_0$. The function passes all the doctests provided in its docstring.

1. Find input values `lower` and `upper` for which the function `all_squares` never returns.
2. How could we change the function to protect it against this?

DE Die Python-Datei underhanded_02.py oben implementiert eine Funktion `all_squares`, welche eine Liste aller Quadratzahlen aus dem Intervall `lower`..`upper` − 1 erstellt. Eine Quadratzahl sei hier eine Zahl $z = x^2$ mit $x \in \mathbb{N}_0$. Die Funktion besteht alle Doc-Tests, die im DocString angegeben sind.

1. Finden Sie Eingabewerte `lower` und `upper` für die die Funktion `all_squares` never returns.
2. Wie könnten wir die Funktion verändern, so dass das nicht mehr geht?

24.3 Underhanded Python (Q149)

Listing 24.3: The program underhanded_03.py (src)

```
"""
Underhanded Python.

Create an argument for the `to_str` function that causes it to crash.
"""

class Entry:
    """A class for a linked list."""

    def __init__(self, value: str, next: "Entry | None" = None) -> None:
        """
        Create the list entry.
        :param value: the value string of the list element
        :param next: the next element in the list
        """
        self.value: str = value
        self.next: "Entry | None" = next


def to_str(head: Entry | None) -> str:
    """
    Concatenate all value strings in the linked list `head`.

    :param head: the head of the list
    :return: the concatenation of all values in the linked list.

    >>> to_str(None)
    ''
    >>> to_str(Entry("x"))
    'x'
    >>> to_str(Entry("x", Entry("y")))
    'xy'
    >>> to_str(Entry("x", Entry("y", Entry("z"))))
    'xyz'
    """
    result: str = ""  # Set the initial result to the empty strong.

    while head is not None:   # Iterate until the end of the list.
        result += head.value  # Append the current value to the result.
        head = head.next      # Move to the next element.

    return result  # Return the result string.
```

EN The Python file underhanded_03.py above provides a class `Entry` representing a linked list. Each element of the linked list stores a value string `value` and a pointer `next` to the next list element. If `next` is `None`, the end of the list is reached. The function `to_str` receives the head of a linked list as input. It returns the concatenation of all of its value strings. The function passes all the doctests provided in its docstring.

1. Find an input value `head` for which the function `to_str` crashes.
2. How could we change the function to protect it against this?

DE Die Python-Datei underhanded_03.py oben bietet eine Klasse `Entry` für verkettete Listen. Jedes Element der verketten Liste speichert einen Wert-String `value` und einen Zeiger `next` auf das nächste Element. Wenn `next` `None` ist, ist das Ende der Liste erreicht. Die Funktion `to_str` bekommt den Kopf einer solchen verketten Liste als Input. Sie liefert die Konkatenation/Aneinanderhängung aller WertStrings zurück. Die Funktion besteht alle Doc-Tests, die im DocString angegeben sind.

1. Finden Sie einen Eingabewert `head` für den die Funktion `to_str` abstürzt.
2. Wie könnten wir die Funktion verändern, so dass das nicht mehr geht?

Backmatter

Glossary

$i!$ The factorial $a!$ of a natural number $a \in \mathbb{N}_1$ is the product of all positive natural numbers less than or equal to a, i.e., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$ [56, 98, 209].

ϕ The golden ratio (or golden section) ϕ is the irrational number $\frac{1+\sqrt{5}}{2}$. It is the ratio of a line segment cut into two pieces of different lengths such that the ratio of the whole segment to that of the longer segment is equal to the ratio of the longer segment to the shorter segment [50, 104]. The golden ratio is approximately $\phi \approx 1.618\,033\,988\,749\,894\,848\,204\,586\,834$ [286]. Represented as `float` in Python, its value is `1.618033988749895`.

π is the ratio of the circumference U of a circle and its diameter d, i.e., $\pi = U/d$. $\pi \in \mathbb{R}$ is an irrational and transcendental number [111, 168, 236], which is approximately $\pi \approx 3.141\,592\,653\,589\,793\,238\,462\,643$. An early method to approximate π was developed by the Chinese mathematician LIU Hui (刘徽) based on inscribing regular e-gons into a circle somewhere in the third century Common Era (CE) [238, 354] in his commentary to the famous Chinese mathematics book *Jiu Zhang Suanshu* (九章算术) [72, 74, 172, 238, 296]. In Python, π is provided by the `math` module as variable `pi` with value `3.141592653589793`. In PostgreSQL, it is provided by the Structured Query Language (SQL) function `pi()` with value `3.141592653589793` [218].

$(1+1)$ **EA** The $(1+1)$ EA is a local search algorithm that retains the best solution x_c discovered so far during the search [52, 96]. In each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then the $(1+1)$ EA uses a unary search operator that flips each bit independently with probability m/n, where usually $m = 1$. This operator is the main difference to randomized local search (RLS). The $(1+1)$ EA is a special case of the $(\mu+\lambda)$ evolutionary algorithm ($(\mu+\lambda)$ EA) where $\mu = \lambda = 1$.

$(1+1)$ **FEA** The $(1+1)$ evolutionary algorithm ($(1+1)$ EA) with Frequency Fitness Assignment (FFA) plugged in [336, 337, 356]. On the discrete bit-string domains, this algorithm has a very specific structure: Its unary search operator flips each bit with probability $1/n$, where n is the dimension of the search space, distinguishing it from the unary operator in RLS with FFA (FRLS) which flips exactly one bit. On other domains, such as permutations, the term is used synonymously with FRLS.

$(\mu+\lambda)$ **EA** The $(\mu+\lambda)$ EA is an evolutionary algorithm (EA) where, in each generation, λ offspring solutions are generated from the current population of μ parent solutions. The offspring and parent populations are merged, yielding $\mu+\lambda$ solutions, from which then the best μ solutions are ratained to form the parent population of the next generation. If the search space is the bit strings of length n, then this algorithm usually applies a mutation operator flipping each bit independently with probability $1/n$.

$i..j$ with $i, j \in \mathbb{Z}$ and $i \leq j$ is the set that contains all integer numbers in the inclusive range from i to j. It always holds that $i..j \subset \mathbb{Z}$. For example, 5..9 is equivalent to $\{5, 6, 7, 8, 9\}$

BinInt The *Binary Integer Problem* is a benchmark problem for bit-string based black-box search [309, 356]. It is very similar to OneMax: It tries to maximize the number of 1 bits in the bit string of a chosen dimension n. Different from OneMax, where each bit has the same weight in the objective function, the bits of BinInt have a exponentially increasing weight: The bit string is treated as a binary-encoded integer whose value should be maximized. The objective function $f(x)$ of the minimization variant of this problem can be defined as $f(x) = 2^n - 1 - \sum_{i=1}^{n} 2^{i-1} \cdot x_i$. The

problem is unimodal, i.e., has only a single global optimum and no local optimum. It is fully separable, i.e., we can optimize each variable separately.

LinHarm The Linear Harmonic Function is a benchmark problem for bit-string based black-box search [90, 96, 356]. It is very similar to OneMax: It tries to maximize the number of 1 bits in the bit string of a chosen dimension n. Different from OneMax, where each bit has the same weight in the objective function, the bits of LinHarm have a linear increasing weight. The objective function $f(x)$ of the minimization variant of this problem can be defined as $f(x) = \frac{1}{2}n(n+1) - \sum_{i=1}^{n} i \cdot x_i$. The problem is unimodal, i.e., has only a single global optimum and no local optimum. It is fully separable, i.e., we can optimize each variable separately. The $(1+1)$ EA can solve this problem with an expected runtime in $\mathbf{\Theta}(n \ln n)$ [90, 96].

TSP Suite is a open source benchmarking framework for the Traveling Salesperson Problem (TSP) written in `Java`[332]. It uses the instances of the TSPLib and offers both automated experiment execution as well as several metrics to measure runtime. It is available at `https://thomasweise.github.io/tspSuite` and has been used in many research works, such as [338]. It can be viewed as a predecessor of `moptipy`.

2BP In the two-dimensional Bin Packing Problem (BP), the goal is to pack rectangles into rectangular bins without overlap and such that their edges are parallel to the edges of the bins [156, 201, 203]. The 2BP is strongly $\mathcal{NP}$-hard [203]. The variants of this problem include the 2BP|O|F, 2BP|R|F, 2BP|O|G, and 2BP|R|G [202, 203]. Different objective function for the 2BP|O|F and 2BP|R|F are presented in [358]. 2BPs have many practical applications, including cutting (wood, glas) [203], packing (transportation, warehouse) [203], and newspaper page layout [297].

2BP|O|F In the oriented two-dimensional Bin Packing Problem (2BP) with free cuts, the rectangles to pack have fixed orientations, i.e., cannot be rotated, but can otherwise be arranged freely [202, 203, 217, 359]. Permutation-based search spaces can be used for this problem by using the Improved Bottom Left (IBL) encoding. The state-of-the-art (SOTA) for this problem comprises [30, 35, 63, 130, 248, 359].

2BP|O|G In the oriented 2BP with guillotine cuts, the rectangles to pack have fixed orientations, i.e., cannot be rotated, and must be arranged such that they can be extracted by edge-to-edge cuts [202, 203].

2BP|R|F In this variant of the 2BP with free cuts, the rectangles to pack can be rotated by 90° if need be and can freely be arranged [26, 202, 203, 358, 359]. Besides packing and cutting, this problem also occurs in scheduling and build formation in additive manufacturing [194, 256]. The almost-squares-in-almost-squares (Asqas) problem [318] is also a 2BP|R|F. Search spaces based on signed permutations can be used for this problem by using the IBL encoding. The SOTA for this problem comprises [30, 63, 130, 358, 359].

2BP|R|G In this variant of the 2BP with guillotine cuts, the rectangles to pack can be rotated by 90° if need be, but must be arranged such that they can be extracted by edge-to-edge cuts [202, 203].

A the $n \times n$ distance matrix. $A[a, b]$ indicates the distance between locations a and b with $a, b \in 1..n$. A distance matrix can be symmetric, in which case $A[a, b] = A[b, a]\ \forall a, b \in 1..n$. Distance matrices represent the problem instance data in TSPs and part of the instance data of Quadratic Assignment Problems (QAPs).

ACO *Ant Colony Optimization* is a nature-inspired optimization method for combinatorial problems where solutions are generated by "ants" that move from node to node in a graph choosing edges based on (1) the simulated pheromone on the edges and (2) a per-edge heuristic value [92–94]. If an ant produced a good solution, "pheromone" is distributed over the edges it visited, making it more likely to be re-visited by other ants.

AI Artificial Intelligence, see, e.g., [273]

API An *Application Programming Interface* is a set of rules or protocols that enables one software application or component to use or communicate with another [135].

Asqas An "almost-square" be a rectangle with size $i \times (i+1)$ with $i \in \mathbb{N}_1$. The *almost-squares-in-almost-squares* problem [318] is a special case of the 2BP|R|F. Here, almost-square rectangles of sizes 1×2, 2×3, ..., $n \times (n+1)$ must be placed into an almost-square bin without waste. This requires that there exists a natural number $p \in \mathbb{N}_1$ such that $p(p+1) = \sum_{i=1}^{n} i(i+1)$ [318]. There are exacty five Asqas instances [318], namely with bin sizes 1×2 for $n = 1$, $p = 1$, 5×4 for $n = 3$, $p = 4$, 15×16 for $n = 8$, $p = 15$, 55×56 for $n = 20$, $p = 55$, and 119×120 for $n = 34$, $p = 119$.

ATSP The general case for TSPs are *asymmetric Traveling Salesperson Problems*, where the $n \times n$ distance matrix A does not need to be symmetric. In other words, it is possible that $\exists a, b \in 1..n$ with $A[a, b] \neq A[b, a]$. Notice that while such index tuple *may* exist, this is not always or necessarily the case. In this case, one may consider the STSP as a special case of the ATSP, while ATSPs may be considered as synonymous to TSPs.

B the $n \times n$ flow matrix. $B[a, b]$ is the flow of goods between locations a and b with $a, b \in 1..n$. Flow matrices are one half of the problem instance data of QAPs.

B&B *Branch and bound* is a family of optimization methods that combines systematic exploration of the search space with both and objective function f subject to minimization and a lower bound function. The idea is that we divide the search space into two sets. We then pick the set that seems more interesting and divide it again. We do this recursively until we have arrived at a set of size 1, which is our first solution y. We can compute its objective value $z = f(y)$. Now we continue with the sets that we have created and not yet explored. However, we can discard all sets for which the lower bound of the solution quality is $\geq z$, because they can never give us a better solution than y. If we have divided another set down to size 1 and it yields a better solution than y, of cause we update these values. Eventually, we have either discarded or fully expanded all sets – and then the solution we got must be optimal. B&B-algorithms have been applied to the TSP [199], the Job Shop Scheduling Problem (JSSP) [49], the oriented two-dimensional Bin Packing Problem with free cuts (2BP|O|F) [217], and maritime logistics [349], for instance.

banker's rounding means rounding a real number to the closest integer and, in case of ties, to the closest *even* integer [11, 251]. Since ties can only occur for numbers ending in `.5`, this means that all numbers of the form `z.5` will be rounded to `z+1` if `z` is odd and to `z` if `z` is even. This is different from the "normal" understanding of rounding as we often learn in school, where all numbers of the form `z.5` are rounded to `z+1`. Banker's rounding has the advantage that it is not biased in any direction [151, 219]. If `z.5` is always rounded toward `z+1` and we have many numbers of the form `z.5`, then the sum or average of the results tends to be somewhat larger than the sum or average of the unrounded numbers. Banker's rounding avoids this problem by sometimes rounding numbers ending in `5` to larger and sometimes to smaller values. In the IEEE Floating-Point Arithmetic Standard 754 [153], this method corresponds to the *roundTiesToEven* direction introduced in its section *4.3.1 Rounding-direction attributes to nearest*. `Python` uses banker's rounding in its `round` function [43], whereas `Java`'s `Math.round` uses the "school-type" rounding [161].

Bash is a shell used under `Ubuntu Linux`, i.e., the program that "runs" in the terminal and interprets your commands, allowing you to start and interact with other programs [40, 234, 355]. Learn more at `https://www.gnu.org/software/bash`.

BCE The time notation *before Common Era* is a non-religious but chronological equivalent alternative to the traditional *Before Christ (BC)* notation, which refers to the years *before* the birth of Jesus Christ [53]. The years BCE are counted down, i.e., the larger the year, the farther in the past. The year 1 BCE comes directly before the year 1 CE [278, 360].

BP The *Bin Packing Problem* [99, 277] is one of the classical problems from Operations Research (OR). The goal is to pack a set of objects into containers, using as few containers as possible. Different flavors of BPs can be distinguished, e.g., by the number of dimensions [80, 201] that the objects and containers have (see, e.g., the 2BP), and/or whether the objects can be rotated [359], as well as other constraints and limitations. Both the one- and two-dimensional BP are strongly $\mathcal{NP}$-hard [203].

BRKGA The *Biased Random Key Genetic Algorithm* is a EA with a continuous search space $\mathbb{X} \subset [0,1]^n$ and a decoding function γ translating this space to something else and biased, uniform, and elitist mating strategies [132, 206].The *bias* comes from the fact that one of the parents used for the binary operation is biased to be of higher fitness than the other parent [132]. Often, γ is the Random Keys encoding [18, 19] procedure translating $\mathbb{X}$ to permutations of the first n natural numbers [131]. Sometimes, it is actually a Memetic Algorithm (MA) further refining the translation result with a local search (LS) [131]. Sometimes the permutations are then further translated to more complex data structures, such as Gantt charts for the JSSP [131], or packing plans for 2BP 3D packing [130].

C is a programming language, which is very successful in system programming situations [85, 260, 330].

$\mathbb{C}$ the set of complex numbers is defined as $\mathbb{C} = \{(a,b) : a,b \in \mathbb{R}\}$. Complex numbers z are often written in the standard form $z = a+b\mathrm{i}$, where i is the so-called imaginary unit with $\mathrm{i}^2 = -1$ [317].
.

CE The time notation *Common Era* is a non-religious but chronological equivalent alternative to the traditional *Anno Domini (AD)* notation, which refers to the years *after* the birth of Jesus Christ [53]. The years CE are counted upwards, i.e., the smaller they are, the farther they are in the past. The year 1 CE comes directly after the year 1 before Common Era (BCE) [278, 360].

client In a client-server architecture, the client is a device or process that requests a service from the server. It initiates the communication with the server, sends a request, and receives the response with the result of the request. Typical examples of clients are web browsers in the internet as well as clients for database management systems (DBMSes), such as `psql`.

client-server architecture is a system design where a central server receives requests from one or multiple clients [29, 207, 243, 264, 269]. These requests and responses are usually sent over network connections. A typical example of such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as `psql`, or the different applications that access the DBs.

Concorde is an exact solver for the TSP[4, 67, 68], which could even solve the largest instance of TSPLib (with 85 900 cities) to optimality.

CSP The *Cutting Stock Problem* is about cutting shapes of a larger shape while reducing waste [203, 211]. It is closely related to the BP.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases* [327].

DBMS A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB [352].

denominator The number b of a fraction $\frac{a}{b} \in \mathbb{Q}$ is called the *denominator*.

docstring Docstrings are special string constants in `Python` that contain documentation for modules or functions [133]. They must be delimited by `"""..."""` [133, 322].

doctest *doctests* are unit tests in the form of as small pieces of code in the docstrings that look like interactive `Python` sessions. The first line of a statement in such a `Python` snippet is indented with `Python»>` and the following lines by `...`. These snippets can be executed by modules like `doctest` [86] or tools such as `pytest` [181]. Their output is the compared to the text following the snippet in the docstring. If the output matches this text, the test succeeds. Otherwise it fails.

e is Euler's number [106], the base of the natural logarithm. $e \in \mathbb{R}$ is an irrational and transcendental number [111, 168], which is approximately $e \approx 2.718\,281\,828\,459\,045\,235\,360$. In Python, it is provided by the `math` module as variable `e` with value `2.718281828459045`. In PostgreSQL, you can obtain it via the SQL function `exp(1)` as value `2.718281828459045` [218].

EA An *evolutionary algorithm* is a metaheuristic optimization method that maintains a population of candidate solutions, which undergo selection (where better solutions are chosen with higher probability) and reproduction (where mutation and recombination create a new candidate solution from one or two existing ones, repectively) [10, 328].

exit code When a process terminates, it can return a single integer value (the exit status code) to indicate success or failure [169]. Per convention, an exit code of 0 means success. Any non-zero exit code indicates an error. Under Python, you can terminate the current process at any time by calling `exit` and optionally passing in the exit code that should be returned. If `exit` is not explicitly called, then the interpreter will return an exit code of 0 once the process normally terminates.If the process was terminated by an uncaught `Exception`, a non-zero exit code, usually 1, is returned.

$f(y)$ The *objective function* $f : \mathbb{Y} \mapsto \mathbb{R}$ can compute a cost or rating for each candidate solution y in the solution space $\mathbb{Y}$. The smaller this value, the better the solution, i.e., in the context of this work, objective functions are subject to minimization. Objective functions are not necessarily continuous and it is also quite common that they have integer or natural numbers as results, e.g., we often have $f : \mathbb{Y} \mapsto \mathbb{N}_0$ instead of $f : \mathbb{Y} \mapsto \mathbb{R}$.

f-string let you include the results of expressions in strings [42, 120, 123, 136, 220, 288]. They can contain expressions (in curly braces) like `f"a{6-1}b"` that are then transformed to text via (string) interpolation, which turns the string to `"a5b"`. F-strings are delimited by `f"..."`.

FFA *Frequency Fitness Assignment* (中文: 频率适应度分配) is a algorithm plugin for optimization methods applied to discrete or combinatorial problems with not-too-many different possible objective values [334, 336, 337]. It replaces the objective values in all comparisons with their absolute encounter frequency so far during the search. FFA has successfully been applied to the Maximum Satisfiability (MaxSAT) [336, 337], the TSP [195, 196], and the QAP [60, 311], amongst others. Its properties have been intensely researched on discrete benchmark functions such as OneMax, LeadingOnes, Trap, TwoMax, Jump, and Plateau [336, 337, 356]. The most common realizations of FFA so far are the $(1+1)$ FEA and FRLS.

FSSP The *Flow Shop Scheduling Problem* is a variant of a JSSP where all products go through the same sequence of machines [137, 166].

$\gamma(x)$ The decoding function $\gamma : \mathbb{X} \mapsto \mathbb{Y}$ translates point from the search space $\mathbb{X}$ to one point y in the solution space $\mathbb{Y}$. It is a very common case that $\mathbb{X} = \mathbb{Y}$ and that the decoding function γ is the identity mapping. Examples for encodings that translate from a simple search space $\mathbb{X}$ to a complex solution data structure $\mathbb{Y}$ are the IBL algorithm that translates (potentially signed) permutations to two-dimensional packings on the 2BP, the permutation-based encoding of Gantt charts for the JSSP from [31, 32, 124, 280, 333], or the permutation-based encoding of game plans for the Traveling Tournament Problem (TTP) from [348]. Encodings can also translate between types of search spaces. The Random Keys encoding, for example, translates from $[0,1]^n$ to permutations of the first n natural numbers..

GA A *Genetic Algorithm* is an EA usually for bit-string based search spaces [10, 77, 129, 224, 230, 328] (although many researchers use the terms EA and GA synonymously). It is a population-based metaheuristic applying a unary and a binary search operater as well as a selection scheme. The original GA is the Standard Genetic Algorithm (SGA), which today is considered outdated. Modern variants are the Greedy $(2+1)$ Genetic Algorithm (Greedy $(2+1)$ GA) and the Self-Adjusting $(1+(\lambda,\lambda))$ Genetic Algorithm (Self-Adjusting $(1+(\lambda,\lambda))$ GA) as well as bit-string based $(\mu+\lambda)$ EAs.

Gantt chart Gantt charts are two-dimensional diagrams illustrating the steps of a process [175, 345]. Their horizontal axis represents the time, the vertical axis represents work packages, machines, or workers. In the diagram area, bars located at certain vertical locations indicate that the work

package, machine, or worker is processed or works at the time range covered by the bar as projected to the horizontal axis.

GECCO The *Genetic and Evolutionary Computation Conference* is one of the largest conferences in the field of EAs. It has been established in 1999 and its proceedings are published by ACM. Learn more at https://sig.sigevo.org/GECCOs.

$\mathrm{geom}(A)$ The *geometric mean* $\mathrm{geom}(A)$ is the n^{th} root of the product of n *positive* values in a dataset $A = (a_0, a_1, \ldots, a_{n-1})$ with $a_i > 0$ for all $i \in 0..n$, i.e., $\mathrm{geom}(A) = \sqrt[n]{\prod_{i=0}^{n-1} a_i} = \exp\left(\frac{1}{n}\sum_{i=0}^{n-1} \ln a_i\right)$. The formula using $\exp$ and $\ln$ is equivalent to the product formula. However, due to the limited range of floating point numbers on computers [153], the product formula may lead to an overflow or underflow and the $\exp$-$\ln$ method is more reliable. In [115], it is shown that the geometric mean is the only reasonable measure of average when dealing with normalized data, where the arithmetic mean or median become unreliable. In Python, you can use function `geometric_mean` from the module `statistics` to compute the geometric mean.

GIGO Garbage In–Garbage Out, see, e.g., [254]

Git is a distributed Version Control System (VCS) which allows multiple users to work on the same code while preserving the history of the code changes [285, 315]. Learn more at https://git-scm.com.

GitHub is a website where software projects can be hosted and managed via the Git VCS [252, 315]. Learn more at https://github.com.

GRASP A *Greedy Randomized Adaptive Search Procedure* is a metaheuristic where each cycle consists of two main components [107, 109, 110]: First, a starting point is generated using a heuristic method (as opposed to being randomly sampled). Then, a LS is applied to refine the solution. Then, another cycle starts and proceeds in the same way. The overall best solution is retained as result. Many examples of applications of GRASPs are listed in [110]. GRASP in combination with VND has been applied to the 2BP|O|F in [248].

Greedy $(2+1)$ **GA** The Greedy $(2+1)$ GA [51, 52, 299, 300] is an EA for bit-string based search spaces, i.e., a Genetic Algorithm (GA). It was shown to be twice as fast as any mutation-only EA on OneMax [299, 300].

i The imaginary unit i, which can also be written as $\mathrm{i} = (0, 1) \in \mathbb{C}$, is defined as $\mathrm{i} = \sqrt{-1}$ and $\mathrm{i}^2 = -1$ [317].

IBL The *improved bottom left* heuristic is a one-shot algorithm constructing a solution for 2BP [200]. For the 2BP|O|F problem variant, the input of the IBL algorithm is a permutation of the first n natural numbers, each number identifying one rectangle to be packed. For the two-dimensional Bin Packing Problem with free cuts and rotation (2BP|R|F) problem flavor, the permutation elements are signed, where a negative sign indicates a rotation by 90° [200]. In [358, 359] it is adapted to a multi-bin situation as follows: The algorithm iterates over the (potentially signed) permutation x and places the rectangles into the packing plan in the prescribed order. It first places the rectangle outside on top of the current bin b, with its bottom-right corner onto the top-right corner of the current bin b. It then moves the rectangle downwards and leftwards as far as possible, prioritizing the downward movement whenever possible. If the rectangle cannot be moved any further, it checks if it is completely contained in the bin b. If yes, it can remain there. Otherwise, a new bin is opened ($b \leftarrow b+1$) and the rectangle is placed at its bottom-left corner. Once a new bin is opened, this bin is used for all further insertions. IBL is suitable as a decoding function γ inside an EA [200]. Historically, it improves upon the BL algorithm [160]: That algorithm cannot represent all possible optimal packings using a permutation, but the IBL can [200].

IDE An *Integrated Developer Environment* is a program that allows the user do multiple different activities required for software development in one single system. It often offers functionality such as editing source code, debugging, testing, or interaction with a distributed version control system. For Python, we recommend using PyCharm. On Apple systems, Xcode is often used.

ILS An *iterated local search* is a variant of the LS which applies cycles of local search and perturbation [298]. First, an initial solution is generated and refined by the local search. Then the cycle begins: A perturbation, i.e., a large search move that likely destroys a good part of the solution, is applied. The resulting new solution is refined by local search. Then, an acceptance criterion decides whether we should use this or the previous solution to continue in the next cycle. ILS has successfully been applied to the QAP.

iOS is the operating system that powers Apple iPhones [54, 287]. Learn more at `https://www.apple.com/ios`.

iPadOS is the operating system that powers Apple iPads [54]. Learn more at `https://www.apple.com/ipados`.

irrational An *irrational* number z is a real number that cannot be expressed as fraction $\frac{p}{q}$ of two integers $p \in \mathbb{Z}$ and $q \in \mathbb{N}_1$ [24, 339]. In other words, it holds that $z \in \mathbb{R}$ and $z \notin \mathbb{Q}$. The set of irrational numbers could be denoted as $\mathbb{R} \setminus \mathbb{Q}$. Irrational numbers have decimal expansions that neither terminate nor become periodic. Examples of irrational numbers are $\sqrt{2}$, ϕ, π, and e.

Ising Model The Ising models are benchmark problems for bit-string based black-box search [90, 112, 113, 356]. Here, a neighborhood is defined around each bit. The bit needs to have the same value as all bits in the neighborhood structure T. Thus, we try to achieve that all bits have the same value. The objective function $f(x)$ of the minimization variant of this problem family can be defined as $f(x,T) = \sum_{(i,j)\in T} |x_i - x_j|$, where T contains the index pairs of neighboring bits and is defined differently for the Ising model instances Ising 1D and Ising 2D. The problems are highly epistatic, as all variables interact with some other variables.

Ising 1D The 1D is a benchmark problem for bit-string based black-box search [90, 113, 356]. The bits are arranged in a one-dimensional ring and we try to minimize the number of bits that have different values than their neighbors. Hence, the neighborhood structure T is defined as $T_{1D} = \{(i+1, (i+1 \bmod n)+1) \mid i \in [0..n-1]\}$. The problems are highly epistatic, as all variables interact with some other variables. The $(1+1)$ EA has an expected runtime in $\mathcal{O}(n^3)$ on 1-Dimensional Ising Model (Ising 1D) [113].

Ising 2D The 2D is a benchmark problem for bit-string based black-box search [90, 112, 356]. The bits are arranged in a two-dimensional torus and we try to minimize the number of bits that have different values than their neighbors. Hence, the neighborhood structure $T = T_{2D}$ is defined as the set of all $\{(\alpha+\beta N, \gamma+\delta N)\}$ pairs with $\alpha, \beta, \gamma, \delta \in 0..N-1$ and either ()$\gamma = (\alpha+1) \bmod N$ and $\delta = \beta$) or ($\gamma = \alpha$ and $\delta = (\beta+1) \bmod N$) with $n = N^2$. The problems are highly epistatic, as all variables interact with some other variables. On a similar class of problems, the $(1+1)$ EA has an exponential expected runtime [112].

IT information technology

Java is another very successful programming language, with roots in the C family of languages [34, 208].

JSSP The *Job Shop Scheduling Problem* [33, 62, 137, 189] is one of the most prominent and well-studied scheduling tasks. In a JSSP instance, there are m machines and k jobs [301]. Each job must be processed once by each machine in a job-specific sequence and has a job-specific processing time on each machine. The goal is to find an assignment of jobs to machines that results in an overall shortest makespan, i.e., the schedule (Gantt chart) which can complete all the jobs in the shortest time. The JSSP is $\mathcal{NP}$-complete [58, 188, 189]. The 8 most common sets of JSSP benchmark instances are *abz* [1], *dmu* [81], *ft* [114], *la* [191], *orb* [5], *swv* [295], *ta* [301], and *yn* [350].Their data can be found (sometimes partially) in several repositories, such as the OR-Library [20, 21], [281], and [329], the latter of which also contains many results from existing works. Encodings based on permutations with repetition are proposed in [31, 32, 124, 280] and used in [336]. The SOTA of the JSSP includes [131, 147, 247, 250, 282, 326, 357].

Jump The Jump problem is a benchmark problem for bit-string based black-box search [96, 118, 356]. Like the OneMax, it tries to maximize the number of 1 bits in a bit string of chosen length n. However, the solutions representing the last ω bits around the optimum to flip are deceptive, meaning that they are actually worse and getting worse than surrounding solutions.

The objective function $f(x)$ of the minimization variant is $n - \|x\|_1$ if $\|x\|_1 = n$ or $\|x\|_1 \leq n - \omega$ hold, and $\omega + \|x\|_1$ otherwise, where $\|x\|_1 = \sum_{i=1}^{n} x_i$. This means that, the larger ω, the larger the deceptive basin around the optimum that an algorithm needs to cross. An $(1+1)$ EA will converge to the resulting local optimum and thus have an expected runtime of $\mathbf{\Theta}(n^\omega + n \ln n)$ [96].In experiments, the $(1+1)$ EA with FFA ($(1+1)$ FEA) achieves polynomial runtime on Jump, seemingly in $\mathcal{O}(n^2 \ln n)$ [336, 337, 356].

k the number of jobs in an instance of the JSSP.

LABS *Low-Autocorrelation Binary Sequences* are bit strings of length n whose autocorrelation is as low as possible [36, 37, 41, 83, 84, 121, 222, 225, 232, 244, 259]. This means that sub-sequences of any length $1 \leq k < n$ should ideally never repeat. The objective function $f : \{0,1\}^n \mapsto \mathbb{N}_0$ represents the energy of a bit string x and is subject to minimization. We can define it as $f(x) = \sum_{k=1}^{n-1} \left(\sum_{i=0}^{n-1-k} s_i \cdot s_{i+k} \right)^2$ with $s_j = 2x_j - 1$. Such sequences are needed for communication engineering [37, 41, 232], cryptography[83, 232], as well as modulation pulses in radar and sonar ranging[244]. The LABS is $\mathcal{NP}$-hard [313]. The SOTA on LABS contains [36, 41, 83, 84, 232].

LAMP Stack A system setup for web applications: `Linux`, Apache (a web server), `MySQL`, and the server-side scripting language PHP [48, 144].

LeadingOnes The LeadingOnes problem is a benchmark problem for bit-string based black-box search [271, 343, 356]. It tries to maximize the number of *leading* 1 bits in the bit string of a chosen dimension n. The objective function $f(x)$ of the minimization variant of this problem can be defined as $f(x) = n - \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$. Different from OneMax, the contribution to the objective value of the bit at index i thus depends on the state of all the bits *before* this index. The problem is unimodal, i.e., has only a single global optimum and no local optimum. Different from OneMax, it is not separable, i.e., we can optimize each variable separately. The expected runtime of $(1+1)$ EA on LeadingOnes is in $\mathbf{\Theta}(n^2)$ [96]. The average runtime $(1+1)$ FEA on LeadingOnes seems to be slower by factor proportional to n, i.e., seems to be in $\mathcal{O}(n^3)$ [336].

linter A linter is a tool for analyzing program code to identify bugs, problems, vulnerabilities, and inconsistent code styles [167, 274]. `Ruff` is an example of a linter used in the `Python` world.

Linux is the leading open source operating system, i.e., a free alternative to Microsoft `Windows` [12, 143, 284, 314, 323]. We recommend using it for this course, for software development, and for research. Learn more at `https://www.linux.org`. Its variant `Ubuntu` is particularly easy to use and install.

LK The *Lin-Kernighan* (LK) algorithm [198] is a stochastic local search (SLS) method for the TSP. The LK algorithm performs iterative k-opt moves, i.e., in each step, it constructs the two sets X and Y of k edges such that removing the edges in X from the current tour T and then adding the edges from Y to T results in a better tour.The LK algorithm is one of the most classical approaches to the TSP.

LKH The *Lin-Kernighan-Helsgaun* algorithm [68, 145, 146] is a SLS method for the TSP based on the Lin-Kernighan (LK) algorithm [198]. LKH-1 was proposed in [146] and LKH-2 in [145]. It holds the record in the *World TSP* [69] as of 2025.

LS A *local search* is an algorithm where a single "current" candidate solution is maintained and new solutions are selected from its neighborhood in each step. Most LSes are randomized, i.e., stochastic local search (SLS).

m the number of machines in an instance of the JSSP.

MA A *Memetic Algorithm* is an EA where each new solution is refined by a LS before it enters the population [328]. As LS, we can use, e.g., RLS, but Tabu Search (TS) [61, 116] is maybe a better choise. Some SOTA algorithms in some domains may be classified as MAs, e.g., Lin-Kernighan Helsgaun (LKH) basically uses a population that is refined via a local search. MAs also worked well on the JSSP [61, 147] and the TSP [116].

macOS or Mac OS is the operating system that powers Apple Mac(intosh) computers [270, 287]. Learn more at https://www.apple.com/macos.

MariaDB An open source relational database management system that has forked off from MySQL [6, 7, 13, 100, 214, 265]. See https://mariadb.org for more information.

MaxSAT The goal of satisfiability problems is to find an assignment for n Boolean variables that make a given Boolean formula $F : \{0,1\}^n \mapsto \{0,1\}$ become true. In the *Maximum Satisfiability* problem (MaxSAT) [148], F is given in conjunctive normal form, i.e., the variables appear either directly or negated in m "or" clauses, which are all combined into one "and." The objective function $f(x)$, subject to minimization, computes the number of clauses which are false under the variable setting x. If $f(x) = 0$, then all clauses of F are true, which solves the problem. The MaxSAT problem is $\mathcal{NP}$-complete [66] and exhibits low epistasis but deceptiveness [263], making it hard for local searches. Typical benchmark instances can be found in SATLIB, which contains smaller-scale instances, and the up-to-date MaxSAT Evaluation (MSE), providing SOTA benchmarks, algorithms, and results.

$\mathrm{mean}(A)$ The *arithmetic mean* $\mathrm{mean}(A)$ is an estimate of the expected value of a distribution from which a data sample A was, well, sampled. Its is computed on data sample $A = (a_0, a_1, \ldots, a_{n-1})$ as the sum of all n elements a_i in the sample data A divided by the total number n of values, i.e., $\mathrm{mean}(A) = \frac{1}{n}\sum_{i=0}^{n-1} a_i$. In Python, you can use function `mean` from the module `statistics` to compute the arithmetic mean.

$\mathrm{median}(A)$ The *median* $\mathrm{median}(A)$ is the value separating the bigger-valued half from the smaller-valued half of a data sample or distribution. Its estimate is the value right in the middle of a *sorted* data sample $A = (a_0, a_1, \ldots, a_{n-1})$ where $a_{i-1} \leq a_i \ \forall i \in 1 \ldots (n-1)$ with an odd number of elements and the mean of the two values in the middle if n is even. In other words, $\mathrm{median}(A) = a_{\frac{n-1}{2}}$ if n is odd and $\frac{1}{2}\left(a_{\frac{n}{2}-1} + a_{\frac{n}{2}}\right)$ otherwise, i.e., if n is even. In Python, you can use function `median` from the module `statistics` to compute the median.

metaheuristic A *metaheuristic* is an optimization algorithm which performs a cycle of trial and error during which candidate solutions are sampled from the solution space $\mathbb{Y}$ and evaluated using an objective function $f : \mathbb{Y} \mapsto \mathbb{R}$ [127, 328]. The results of the objective function are used to decide which solution is promising. The neighborhoods around promising solutions *or* similarly-structured solutions are then sampled in the next cycle. Prominent metaheuristics are SLSes, Simulated Annealing (SA), TS, and EAs.

Microsoft Windows is a commercial proprietary operating system [39]. It is widely spread, but we recommend using a Linux variant such as Ubuntu for software development and for our course. Learn more at https://www.microsoft.com/windows.

ML Machine Learning, see, e.g., [279]

moptipy is the *Metaheuristic Optimization in Python* library [335]. It has been used in several different research works, including [60, 195–197, 311, 348, 358, 359]. Algorithms implemented by moptipy include EAs, $(1+1)$ EA, RLS, SA, and the $(1+1)$ FEA, amongst others. Learn more at https://thomasweise.github.io/moptipy and https://thomasweise.github.io/moptipyapps.

MSE The *MaxSAT Evaluation* is a regularly updated set of benchmark instances for the MaxSAT [157]. It also includes solver software and benchmark results [324].

Mypy is a static type checking tool for Python [193] that makes use of type hints. Learn more at https://github.com/python/mypy and in [331].

MySQL An open source relational database management system [38, 100, 266, 306, 344]. MySQL is famous for its use in the LAMP Stack. See https://www.mysql.com for more information.

n We usually use the value $n \in \mathbb{N}_0$ to denote the *scale*, i.e., the size of the problem to solve. This notion is very general in computer science and not limited to optimization. For example, if we sort a list or search in a list, the scale n is the length of the list, i.e., the number of elements in it. In a TSP, this may be the number of cities. In a discrete benchmark problem such as OneMax or Trap, this may be the number of bits over which the problem is defined. In a QAP,

this may be the number of facilities and locations. In a MaxSAT problem, this may be the number of variables that are input to the Boolean formula that we want to become true. There are also problems where the scale is not defined straightforwardly like this. For example, in a JSSP, we have a number of machines and a number of jobs, which together define the size of the problem. But even then, the scale n relates to the size of the solution space $\mathbb{Y}$, and, hence, also of the search space $\mathbb{X}$, which usually grow exponentially with n. And this is already the main issue: The larger n, the (exponentially) larger the number of possible solutions and thus, the larger the runtime of algorithms. This is known as the "curse of dimensionality" [25, 59]. Scales n are also often used in conjunction with the $\mathcal{O}$-notation to denote algorithm complexity. For example, comparison-based sorting a list of n items takes $\mathbf{\Theta}(n \log n)$ steps [178] (because they need $\lceil \log_2 n! \rceil$ comparisons in the worst case).

$\mathbb{N}_0$ the set of the natural numbers *including* 0, i.e., 0, 1, 2, 3, and so on. It holds that $\mathbb{N}_0 \subset \mathbb{Z}$ and that $\mathbb{N}_0 = \mathbb{N}_1 \cup \{0\}$.

$\mathbb{N}_1$ the set of the natural numbers *excluding* 0, i.e., 1, 2, 3, 4, and so on. It holds that $\mathbb{N}_1 \subset \mathbb{Z}$ and that $\mathbb{N}_1 = \mathbb{N}_0 \setminus \{0\}$.

N-Queens The N-Queens Problem is a benchmark problem for bit-string based black-box search [90, 356]. The goal is to arrange N queens on an $N \times N$ chessboard so that none can attack the other. The search space has thus dimension $n = N^2$, with each bit standing for a field and encoding whether or not a queen is placed there. The objective function $f(x)$ of the minimization variant is $N - (\sum_{i=1}^{n} x_i) - N \sum_{\xi} \max\{0, Q_\xi(x) - 1\}$ where $Q_\xi(x)$ is the number of queens in the horizontal, vertical, or diagonale ξ. The problem is quite epistatic.

$\mathcal{NP}$ is the class of computational problems that can be solved in polynomial time by a non-deterministic machine and can be verified in polynomial time by a deterministic machine (such as a normal computer) [134].

$\mathcal{NP}$-complete A decision problem is $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and all problems in $\mathcal{NP}$ are reducible to it in polynomial time [134, 262]. A problem is $\mathcal{NP}$-complete if it is $\mathcal{NP}$-hard and if it is in $\mathcal{NP}$.

$\mathcal{NP}$-hard A problem is $\mathcal{NP}$-hard if all problems in $\mathcal{NP}$ are reducible to it in polynomial time [134]. Algorithms that guarantee to find the correct solutions of $\mathcal{NP}$-hard problems [58, 66, 189] have a worst-case runtime that grows exponentially with the problem scale.

numerator The number a of a fraction $\frac{a}{b} \in \mathbb{Q}$ is called the *numerator*.

$\mathbf{\Omega}(g(x))$ If $f(x) = \mathbf{\Omega}(g(x))$, then there exist positive numbers $x_0 \in \mathbb{R}^+$ and $c \in \mathbb{R}^+$ such that $f(x) \geq c * g(x) \geq 0$ for all $x \geq x_0$ [70, 176, 177]. In other words, $\mathbf{\Omega}(g(x))$ describes a lower bound for function growth.

$\mathcal{O}(g(x))$ If $f(x) = \mathcal{O}(g(x))$, then there exist positive numbers $x_0 \in \mathbb{R}^+$ and $c \in \mathbb{R}^+$ such that $0 \leq f(x) \leq c * g(x)$ for all $x \geq x_0$ [9, 70, 176, 177, 186]. In other words, $\mathcal{O}(g(x))$ describes an upper bound for function growth.

$\mathbf{\Theta}(g(x))$ If $f(x) = \mathbf{\Theta}(g(x))$, then $f(x) = \mathcal{O}(g(x))$ and $f(x) = \mathbf{\Omega}(g(x))$ [176, 177], meaning that there exist positive constants $c_1, c_2, x_0 \in \mathbb{R}^+$ such that $0 \leq c_1 g(x) \leq f(x) \leq c_2 g(x)$ for all $x \geq x_0$ [70]. In other words, $\mathbf{\Theta}(g(x))$ describes an exact order of function growth.

$\mathbf{o}(g(x))$ If $f(x) = \mathbf{o}(g(x))$, then there exist positive numbers $x_0 \in \mathbb{R}^+$ such that $0 \leq f(x) \leq c * g(x)$ for all $x \geq x_0$ *all* $c \in \mathbb{R}^+$ [70]. Different from $\mathcal{O}(g(x))$, $\mathbf{o}(g(x))$ holds for *all* constants $c > 0$, whereas $\mathcal{O}(g(x))$ implies the existence of *some* constant $c > 0$. This basically means that $f(x)$ becomes insignificant compared to $g(x)$ as x_0 grows toward infinity. It is often written that $\lim_{x_0 \to \infty} \frac{f(x)}{g(x)} = 0$. In other words, $\mathbf{o}(g(x))$ describes an upper bound for function growth that is not asymptotically tight, i.e., an upper bound that grows faster than $f(x)$.

OneMax OneMax is the simplest possible benchmark problem for bit-string based black-box search [97, 233, 356]. The goal is to maximize the number of 1 bits in the bit string of a chosen dimension n. The objective function $f(x)$ of the minimization variant of this problem can be defined as $f(x) = n - \sum_{i=1}^{n} x_i$. The problem is unimodal, i.e., has only a single global optimum and no local

optimum. It is fully separable, i.e., we can optimize each variable separately. The expected runtime of $(1+1)$ EA on OneMax is in $\mathbf{\Theta}(n \ln n)$ [88, 233]. The average runtime $(1+1)$ FEA on OneMax seems to be slower by factor proportional to n, i.e., seems to be in $\mathcal{O}(n^2 \ln n)$ [336].

OR Operations Research (or Operational Research) is the application of sciences such as mathematics and computer science to the management and organization of systems, organizations, enterprises, factories, or projects. It encompasses the development and application of problem-solving methods and techniques (such as mathematical optimization, simulation, queueing theory and other stochastic models) with the goal to improve decision-making and efficiency [78].

OR-Library is a library of benchmark instances for various OR problems [20, 21] located at `https://people.brunel.ac.uk/~mastjjb/jeb/info.html`. It contains instances of the JSSP, 1D- and 2D-Packing, Knapsack, periodic TSPs, amongst others.

Plateau The Plateau problem is a benchmark problem for bit-string based black-box search [3, 356]. It is structured exactly like Jump, but instead of being deceptive, the last ω bits to flip around the optimum are neutral. The objective function $f(x)$ of the minimization variant is $n - \|x\|_1$ if $\|x\|_1 = n$ or $\|x\|_1 \leq n - \omega$ hold, and ω otherwise, where $\|x\|_1 = \sum_{i=1}^{n} x_i$. This means that, the larger ω, the larger the neutral plane around the optimum that an algorithm needs to cross. An $(1+1)$ EA has an expected runtime of $\mathbf{\Theta}(n^\omega)$ on plateaus [3]. The average runtime of the $(1+1)$ FEA on Plateaus is slower by a factor seemingly proportional in n.

PostgreSQL An open source object-relational DBMS [108, 239, 258, 306]. See `https://postgresql.org` for more information.

psql is the client program used to access the `PostgreSQL` DBMS server.

PyCharm is the convenient `Python` IDE that we recommend for this course [319, 346, 351]. It comes in a free edition, so it can be downloaded and used at no cost. Learn more at `https://www.jetbrains.com/pycharm`.

Pylint is a linter for `Python` that checks for errors, enforces coding standards, and that can make suggestions for improvements [261]. Learn more at `https://www.pylint.org`.

pytest is a framework for writing and executing unit tests in `Python` [82, 182, 241, 245, 346]. Learn more at `https://pytest.org`.

Python The `Python` programming language [152, 192, 210, 331], i.e., what you will learn about in our book [331]. Learn more at `https://python.org`.

$\mathbb{Q}$ the set of the rational numbers, i.e., the set of all numbers that can be the result of $\frac{a}{b}$ with $a, b \in \mathbb{Z}$ and $b \neq 0$. a is called the numerator and b is called the denominator. It holds that $\mathbb{Z} \subset \mathbb{Q}$ and $\mathbb{Q} \subset \mathbb{R}$.

QAP The *Quadratic Assignment Problem* is an optimization problem where the goal is to assign a set of n facilities to a set of n locations [45, 60, 179, 204]. Such an assignment can be represented as a permutation x of the first n natural numbers, where x_i specifies the location where facility i should be placed. For each QAP instance, a distance matrix A is given, where $A[a, b]$ specifies the distance from location a to location b, as well as a flow matrix B, where $B[i, j]$ is the amount of material flowing (e.g., daily) from facility i to facility j. The objective function $f(x)$ then rates a permutation x as $f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} A[x_i, x_j] * B[i, j]$. The QAP is $\mathcal{NP}$-complete [95, 276] and even among this problem class, is considered as particularly hard for metaheuristics [23, 45, 204, 298]. The QAP has applications such as building layout [71, 102, 180], keyboard layout [47], circuit design [103], wiring [292], and scheduling [290]. Typical benchmark instances can be found in QAPLIB. SOTA algorithms for the QAP include [23, 116, 223, 229, 303, 304].

QAPLIB is a library of 136 benchmark instances for the Quadratic Assignment Problem (QAP) [44, 46], 28 of which have not yet been solved to optimality [2] (at least by 2023).

$\mathbb{R}$ the set of the real numbers.

$\mathbb{R}^+$ the set of the positive real numbers, i.e., $\mathbb{R}^+ = \{x \in \mathbb{R} : x > 0\}$.

Random Keys encoding is a decoding procedure $\gamma : \mathbb{X} \mapsto \mathbb{Y}$ that maps a continuous search space $\mathbb{X} \subset [0,1]^n$ which is a subset of the n-dimensional real numbers to the space $\mathbb{Y}$ of permutations of the first n natural numbers [18, 19]. For example, on $n = 5$, the vector $x = (0.49, 0.91, 0.33, 0.75, 0.51)$ is translated to the permutation $y = (3, 1, 5, 4, 2)$, because $x_3 = 0.33$ has the smallest value, putting 3 at the beginning, $x_1 = 0.49$ the second-smallest, placing 1 second, ..., and $x_2 = 0.91$ being the largest value meaning that the last number in y will be 2 [19]. The advantage of the encoding is that we can search in $\mathbb{X}$ applying arbitrary mathematical operators, crossover, and mutation procedures, but will always get correct permutations. The disadvantage is that $|\mathbb{X}| \gg |\mathbb{Y}|$ and that there actually is no meaning underpinning moves in $\mathbb{X}$ in relation to their projected result in $\mathbb{Y}$.

relational database A relational DB is a database that organizes data into rows (tuples, records) and columns (attributes), which collectively form tables (relations) where the data points are related to each other [65, 140, 142, 289, 305, 327, 342].

RLS Randomized local search retains the best solution x_c discovered so far during the search and, in each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then RLS uses a unary search operator that flips exactly one bit. This operator is the main difference to $(1+1)$ EA.

FRLS The RLS with FFA plugged in [356]. On the discrete bit-string domains, this algorithm has a very specific structure: Its unary search operator flips exactly one bit, distinguishing it from the unary operator in the $(1+1)$ FEA, which flips each bit with probability $1/n$. On other domains, such as permutations, we use the term synonymously with $(1+1)$ FEA.

RTS *Reactive Tabu Search* is a variant of TS where the length of the tabu list is adapted to the problem and to the current evolution of the search, and an escape strategy for diversifying the search is used when that is not sufficient [14–17]. It has successfully been applied to a wide range of problems, from the QAP [15, 16], to Knapsack problems [14, 16], the N-K landscape [14] to the training of neural networks [17].

Ruff is a linter and code formatting tool for `Python` [215, 216]. Learn more at `https://docs.astral.sh/ruff` or in [331].

SA *Simulated Annealing* is a SLS that sometimes accepts a worse solution [55, 159, 174, 255, 328]. The probability to do so decreases over time and with the difference in objective values, i.e.,is the lower the worse the new solution is. SA performed well on a variety of problems, including the QAP [227, 312] and the JSSP [247].

SATLIB is a library of benchmark instances for the Maximum Satisfiability (MaxSAT) available at `https://www.cs.ubc.ca/~hoos/SATLIB/index-ubc.html` [149, 150].

Self-Adjusting $(1+(\lambda,\lambda))$ GA The Self-Adjusting $(1+(\lambda,\lambda))$ GA is a theory-inspired EA for bit string-based search spaces [52]. It has linear runtime on the OneMax problem [87].

server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carring the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples of servers are web servers [48] in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the "server computer" [184].

SGA The Standard Genetic Algorithm [10, 77, 129, 224, 230, 328] was the first EA. It maintains a population of solutions and applies mutation and crossover to generate offspring solutions. It uses fitness proportionate selection to choose which solutions should "survive" into the next generation, which today is considered a very bad design choice, see, e.g., [343].

SLS *Stochastic local search* is a local search (LS), one of the simplest metaheuristic search paradigms [148]: A single "current" candidate solution is maintained and new solutions are *stochastically* sampled from its neighborhood, i.e., in a way that involves random numbers. If

these solutions are deemed interesting based on their objective value, they replace the "current". The cycle continues. Basic examples are RLS and $(1+1)$ EA. More complex examples that add mechanisms to escape local optima on top of the SLS paradigm are TS and SA.

SOTA State-of-the-Art, i.e., the best-performing methods for a field or problem type

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases [57, 73, 75, 76, 154, 221, 291, 293, 294, 305]. It is understood by many DBMSes. You find the SQL commands supported by `PostgreSQL` in the reference [291].

`stderr` The *standard error stream* is one of the three pre-defined streams of a console process (together with the standard input stream (`stdin`) and the `stdout`) [171]. It is the text stream to which the process writes information about errors and exceptions. If an uncaught `Exception` is raised in `Python` and the program terminates, then this information is written to `stderr`. If you run a program in a terminal, then the text that a process writes to its `stderr` appears in the console.

`stdin` The *standard input stream* is one of the three pre-defined streams of a console process (together with the `stdout` and the `stderr`) [171]. It is the text stream from which the process reads its input text, if any. The `Python` instruction `input` reads from this stream. If you run a program in a terminal, then the text that you type into the terminal while the process is running appears in this stream.

`stdout` The *standard output stream* is one of the three pre-defined streams of a console process (together with the `stdin` and the `stderr`) [171]. It is the text stream to which the process writes its normal output. The `print` instruction of `Python` writes text to this stream. If you run a program in a terminal, then the text that a process writes to its `stdout` appears in the console.

(string) interpolation In `Python`, string interpolation is the process where all the expressions in an f-string are evaluated and the final string is constructed. An example of string interpolation is turning `f"Rounded {1.234:.2f}"` to `"Rounded 1.23"`.

STSP A *symmetric Traveling Salesperson Problem* is a special case of the TSP, where the $n \times n$ distance matrix A is symmetric, i.e., $A[a,b] = A[b,a] \forall a, b \in 1..n$.

terminal A terminal is a text-based window where you can enter commands and execute them [12, 64]. Knowing what a terminal is and how to use it is very essential in any programming- or system administration-related task. If you want to open a terminal under Microsoft `Windows`, you can press ⊞+R, type in `cmd`, and hit ↵. Under `Ubuntu Linux`, Ctrl+Alt+T opens a terminal, which then runs a `Bash` shell inside.

transcendental A *transcendental* number is a number that is not the root of any integer polynomial, meaning that it is not an algebraic number of any degree [341]. Every real transcendental number must also be irrational, since a rational number is, by definition, an algebraic number of degree one [341]. Examples of transcendental numbers are π and e.

Trap The Trap problem is a benchmark problem for bit-string based black-box search [96, 235, 356]. It is structured exactly like OneMax, however the worst and the best solution are swapped. This makes the problem highly deceptive, as following the "gradient" in the search space leads the search away from the global optimum. For bit strings of length n, the objective function $f(x)$ of the minimization variant is 0 if $\|x\|_1 = 0$ and $n - \|x\|_1 + 1$ otherwise, where $\|x\|_1 = \sum_{i=1}^{n} x_i$. An $(1+1)$ EA will converge to the resulting local optimum and thus have an expected runtime of $\mathbf{\Theta}(n^n)$[96]. In experiments, the $(1+1)$ FEA achieves polynomial runtime on Trap, seemingly in $\mathcal{O}(n^2 \ln n)$ [336, 337, 356].

TS *Tabu Search* is one of the most successful SLSes [125, 126, 128, 162]. In order to avoid returning to previous solutions, it declares certain solutions or search moves as "tabu." For example, if a bit in a discrete problem has been flipped by the search operator, it may be forbidden to flip the bit again for some time, thus preventing the algorithm from getting trapped in a cycle. This rule may be overridden if a new best-so-far solution is discovered that outperforms all previous solutions, in which case it would be accepted (this is the so-called aspiration criterion). TS has performed well on the QAP [15, 16, 226, 229, 302, 303] and on the JSSP [237, 250, 253, 282, 357].

TSP In an instance of the *Traveling Salesperson Problem*, also known as *Traveling Salesman Problem*, a set of n cities or locations as well as the distances between them are given [4, 139, 190, 195, 332, 338]. The goal is to find the shortest round-trip tour that starts at one city, visits all the other cities one time each, and returns to the origin. The TSP is one of the most well-known $\mathcal{NP}$-hard combinatorial optimization problems [122, 139, 183]. The TSP has many applications[4], including in transportation, logistics, (circuit) plotting and drilling problems[138], telescope aiming, and genome sequencing. Solutions are often encoded as permutations x of the first n natural numbers, where each city gets a number and the order of numbers in the permutation corresponds to the order of cities in the tour. The distances between the n cities are usually given in a $n \times n$ distance matrix A, where $A[a, b]$ is the distance from city a to city b with $a, b \in 1..n$. Then, the objective function subject to minimization is $f(x) = A[x_n, x_1] + \sum_{i=1}^{n-1} A[x_i, x_{i+1}]$. Typical benchmark instances can be found in TSPLib [267, 268], the 8th DIMACS challenge[163–165], and the *World TSP* repository[69].

TSPLib is a library of benchmark instances for the Traveling Salesperson Problem (TSP) [267, 268]. Its original Uniform Resource Locator (URL) was available until 2025 and then became defunct. The repository `https://github.com/shredderzwj/TSPLIB` contains the data. Additionally, you may also check `https://github.com/thomasWeise/tspSuite`, which contains all the TSPLib instances as well as `Java` code to load them. The TSPLib contains 110 instances of the symmetric Traveling Salesperson Problems (STSPs) with scales n ranging from 14 to 85 900, but 93 instances have less than 2000 nodes. There also are 19 asymmetric Traveling Salesperson Problems (ATSPs) with scales $n \in 17..443$.

TTP The *Traveling Tournament Problem* is the combinatorial optimization problem of both efficiently and fairly organizing a tournament of n teams that play against each other in a pairwise fashion [101, 348]. The efficient part boils down to arranging the games such that the total travel length is short, which is somewhat similar to the classical TSP. Initially, each team is at its home location. On each day, a team needs to travel if its scheduled game is not at its present location. On the last day, each team may need to travel back home unless their last game is a home game. The total travel length sums up the lengths of all travels over all teams. The fair part is represented in several constraints, such as doubleRoundRobin, compactness, maxStreak, and noRepeat. The TTP is $\mathcal{NP}$-hard [325].

TwoMax The TwoMax problem is a benchmark problem for bit-string based black-box search [118, 320, 356]. It has a global and a local optimum, both of about the same size, at opposite ends of the search space. For bit strings of length n, the objective function $f(x)$ of the minimization variant is 0 if $\|x\|_1 = n$ and $1 + n - \max\{\|x\|_1, n - \|x\|_1\}$ otherwise, where $\|x\|_1 = \sum_{i=1}^{n} x_i$. An $(1+1)$ EA has a 50/50 chance to converge to the local or global optimum and thus has an expected runtime in $\mathbf{\Theta}(n^n)$ [117, 118]. In experiments, the $(1+1)$ FEA achieves polynomial runtime seemingly in $\mathcal{O}(n^2 \ln n)$ on TwoMax [336, 337, 356].

type hint are annotations that help programmers and static code analysis tools such as `Mypy` to better understand what type a variable or function parameter is supposed to be [187, 321]. `Python` is a dynamically typed programming language where you do not need to specify the type of, e.g., a variable. This creates problems for code analysis, both automated as well as manual: For example, it may not always be clear whether a variable or function parameter should be an integer or floating point number. The annotations allow us to explicitly state which type is expected. They are *ignored* during the program execution. They are a basically a piece of documentation.

Ubuntu is a variant of the open source operating system `Linux` [64, 144]. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at `https://ubuntu.com`. If you are in China, you can download it from `https://mirrors.ustc.edu.cn/ubuntu-releases`.

UCS Universal Coded Character Set, see Unicode

Unicode A standard for assigning characters to numbers [155, 308, 316]. The Unicode standard supports basically all characters from all languages that are currently in use, as well as many special symbols. It is the predominantly used way to represent characters in computers and is regularly updated and improved.

unit test Software development is centered around creating the program code of an application, library, or otherwise useful system. A *unit test* is an *additional* code fragment that is not part of that productive code. It exists to execute (a part of) the productive code in a certain scenario (e.g., with specific parameters), to observe the behavior of that code, and to compare whether this behavior meets the specification [22, 240, 242, 245, 272, 310]. If not, the unit test fails. The use of unit tests is at least threefold: First, they help us to detect errors in the code. Second, program code is usually not developed only once and, from then on, used without change indefinitely. Instead, programs are often updated, improved, extended, and maintained over a long time. Unit tests can help us to detect whether such changes in the program code, maybe after years, violate the specification or, maybe, cause another, depending, module of the program to violate its specification. Third, they are part of the documentation or even specification of a program.

URI A *Uniform Resource Identifier* is an identifier for an abstract or physical resource in the internet [353]. It can be a URL, a name, or both. URIs are supersets of URLs. The connection strings of the `PostgreSQL` DBMS are examples of Uniform Resource Identifiers (URIs).

URL A *Uniform Resource Locator* identifies a resource in the WWW and a way to obtain it by describing a network access mechanism. The most notable example of URLs is the text you write into web browsers to visit websites [28]. URLs are subsets of URIs.

UTF-8 The *UCS Transformation Format 8* is one standard for encoding Unicode characters into a binary format that can be stored in files [155, 353]. It is the world wide web's most commonly used character encoding, where each character is represented by one to four bytes. It is backwards compatible with ASCII.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code [315]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

VND *Variable Neighborhood Descend*; see Variable Neighborhood Search (VNS). VND in combination with GRASP has been applied to the 2BP|O|F in [248].

VNS *Variable Neighborhood Search* is a metaheuristic framework combining a systematic change of neighborhood with LS [141, 231]. A *neighborhood* structure in LS defines which solutions can be reached from a given solution within one search step. Usually, one uses a single unary operator and, hence, has a single neighborhood structure. However, for many problems, we can easily design many operators. For permutations, for example, one operator may swap two elements, another one may reverse a section of the permutation, and yet another one, while yet another one may rotate a subsequence one step to the left or right [332]. In the VNS framework, a multiple neighborhoods are investigated in a systematic fashion. Assume that we have a list of neighborhood structures. Variable Neighborhood Descend (VND) works in a deterministic fashion. It finds the best solution that can be reached in one step from the current solution under the current neighborhood. If this is better than the current solution, we accept it and continue with the first neighborhood. Otherwise, we continue with the next neighborhood in the list. In Basic VNS, instead of picking the best move from the current neighborhood, the first improving move is used. Also, at the beginning of each cycle, a random point from the current neighborhood around the current solution is chosen as starting point.

$\omega(g(x))$ If $f(x) = \omega(g(x))$, then for every positive constant $c > 0$, there exists a number $x_0 \in \mathbb{R}^+$ such that $0 \leq c * g(x) < f(x)$ for all $x \geq x_0$ [70]. In other words, $f(x) = \omega(g(x))$ if and only if $g(x) = \mathbf{o}(f(x))$. In other words, ω describes a lower bound that is not asymptotically tight.

WWW World Wide Web [27, 79]

$\mathbb{X}$ The search space $\mathbb{X}$ is the space through which an optimization algorithm navigates in order to find good solutions. The search operators of metaheuristics are used to sample points $x \in \mathbb{X}$, for example. On many benchmark problems, the objective functions f directly evaluate the elements $x \in \mathbb{X}$, i.e., are defined as $f : \mathbb{X} \mapsto \mathbb{R}$. However, very often these elements x are not the same data structures that the user is interested in and/or that the objective functions evaluate:

It is common that the search space $\mathbb{X}$ is a simple and plain data structure that can be tackled with standard operations or mathematical tools, such as real vectors $\mathbb{R}^n$, permutations, or bit strings, whereas the solution space $\mathbb{Y}$ can be something more complicated. Then, the user only cares about the solutions $y \in \mathbb{Y}$ and not about the search space $\mathbb{X}$. A decoding $\gamma : \mathbb{X} \mapsto \mathbb{Y}$ is employed to translate the $x \in \mathbb{X}$ to $y = \gamma(x)$ with $y \in \mathbb{Y}$.

x a point in the search space $\mathbb{X}$.

Xcode is offers the tools for developing, testing, and distributing applications as well as an IDE for Apple platforms such as `macOS` and `iOS` [275].

$\mathbb{Y}$ the set of candidate solutions of an optimization problem is called the *solution space* $\mathbb{Y}$. The solution space corresponds to the data structure holding the candidate solutions y. These are the elements that are passed to the user after the optimization process is completed. They are also the parameters of the objective function(s) $f : \mathbb{Y} \mapsto \mathbb{R}$.

y a candidate solution $y \in \mathbb{Y}$ is one possible solution to an optimization problem. After the optimization process is completed, one or multiple such solutions are passed to the user.

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., ..., -3, -2, -1, 0, 1, 2, 3, ..., and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.

$\overline{z}$ The conjugate $\overline{z}$ of a complex number $z \in \mathbb{C}$ with $z = a + b\mathrm{i}$ is $\overline{z} = a - b\mathrm{i}$ [317]. You get the complex conjugate of a complex number by flipping the sign of the imaginary part.

Python Commands

**, 4
.2f, 89
/, 3
//, 3
>>>, 80
..., 80
"""..."""", 80
2.718281828459045, 81
3.141592653589793, 77

def, 48
dict, 32, 33
doctest, 80

e, 81
elif, 36
else, 36
enumerate, 46
Exception, 81, 89
exit, 81

f"...", 81

False, 10
find, 15
float, 6, 52
for, 39, 46

geometric_mean, 82

if, 36
inf, 7
input, 89
int], 3

list, 27, 28, 41, 52

math, 77, 81
mean, 85
median, 85
Mypy, 85

nan, 7
None, 16

pi, 77
print, 41, 89

round, 8, 79
ruff, 88

set, 30
statistics, 82, 85
    geometric_mean, 82
    mean, 85
    median, 85
str, 12

True, 10, 62
try-except, 59
try-finally, 59
tuple, 28, 41

ValueError, 48, 62, 63

while, 44, 46
with, 59

Bibliography

[1] Joseph Adams, Egon Balas, and Daniel Zawack. "The Shifting Bottleneck Procedure for Job Shop Scheduling". *Management Science* 34(3):391–401, Mar. 1988. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0025-1909. doi:10.1287/mnsc.34.3.391 (cit. on p. 83).

[2] Miguel F. Anjos. *The QAPLIB Challenge*. Edinburgh, Scotland, UK: Miguel F. Anjos @ University of Edinburgh, Nov. 19, 2023. URL: https://www.miguelanjos.com/projects/2999 (visited on 2026-09-28). The document lists the 28 instances of QAPLIB [44, 46] that had not yet been solved to optimality by the time of its writing. (Cit. on p. 87).

[3] Denis Antipov and Benjamin Doerr. "Precise Runtime Analysis for Plateaus". In: *15th International Conference on Parallel Problem Solving from Nature (PPSN XV)*. Vol. 2. Sept. 8–12, 2018, Coimbra, Portugal. Ed. by Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and L. Darrell Whitley. Vol. 11102 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer, 2018, pp. 117–128. ISSN: 0302-9743. ISBN: 978-3-319-99258-7. doi:10.1007/978-3-319-99259-4_10 (cit. on p. 87).

[4] David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. 2nd ed. Vol. 17 of Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 978-0-691-12993-8. This book also discusses the Concorde TSP solver. (Cit. on pp. 80, 90).

[5] David Lee Applegate and William John Cook. "A Computational Study of the Job-Shop Scheduling Problem". *ORSA Journal on Computing* 3(2):149–156, May 1991. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.3.2.149. The JSSP instances used were generated in Bonn in 1986. (Cit. on p. 83).

[6] Adam Aspin and Karine Aspin. *Query Answers with MariaDB – Volume I: Introduction to SQL Queries*. Tetras Publishing, Oct. 2018. ISBN: 978-1-9996172-4-0. See also [7] (cit. on pp. 85, 94).

[7] Adam Aspin and Karine Aspin. *Query Answers with MariaDB – Volume II: In-Depth Querying*. Tetras Publishing, Oct. 2018. ISBN: 978-1-9996172-5-7. See also [6] (cit. on pp. 85, 94).

[8] Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and L. Darrell Whitley, eds. *15th International Conference on Parallel Problem Solving from Nature (PPSN XV)*. Vol. 2. Sept. 8–12, 2018, Coimbra, Portugal. Vol. 11102 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer, 2018. ISSN: 0302-9743. ISBN: 978-3-319-99258-7. doi:10.1007/978-3-319-99259-4.

[9] Paul Gustav Heinrich Bachmann. *Die Analytische Zahlentheorie / Dargestellt von Paul Bachmann*. Vol. Zweiter Theil of Zahlentheorie: Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Haupttheilen. Leipzig, Sachsen, Germany: B. G. Teubner, 1894. ISBN: 978-1-4181-6963-3. URL: http://gallica.bnf.fr/ark:/12148/bpt6k994750 (visited on 2023-12-13) (cit. on p. 86).

[10] Thomas Bäck, David B. Fogel, and Zbigniew "Zbyszek" Michalewicz, eds. *Handbook of Evolutionary Computation*. Bristol, England, UK: IOP Publishing Ltd and Oxford, Oxfordshire, England, UK: Oxford University Press, 1997. ISBN: 978-0-7503-0392-7 (cit. on pp. 81, 88).

[11] "Banker's Rounding". In: *Alipay$^+$ Documentation*. Singapore: Alipay Connect Pte. Lte., 2022. URL: https://docs.alipayplus.com/alipayplus/alipayplus/reconcile_mpp/bank_rounding (visited on 2025-07-18) (cit. on p. 79).

[12] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (cit. on pp. 84, 89).

[13] Daniel Bartholomew. *Learning the MariaDB Ecosystem: Enterprise-level Features for Scalability and Availability*. New York, NY, USA: Apress Media, LLC, Oct. 2019. ISBN: 978-1-4842-5514-8 (cit. on p. 85).

[14] Roberto Battiti and Giampietro Tecchiolli. "Local Search with Memory: Benchmarking RTS". *OR Spectrum: Quantitative Approaches in Management. Formerly* Operations-Research-Spektrum 17:67–86, June 1995. London, England, UK: Springer Nature Limited. ISSN: 0171-6468. doi:10.1007/BF01719249 (cit. on p. 88).

[15] Roberto Battiti and Giampietro Tecchiolli. "Simulated Annealing and Tabu Search in the Long Run: A Comparison on QAP Tasks". *Computers & Mathematics with Applications* 28(6):1–8, Sept. 1994. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0898-1221. doi:10.1016/0898-1221(94)00147-2 (cit. on pp. 88, 89).

[16] Roberto Battiti and Giampietro Tecchiolli. "The Reactive Tabu Search". 6(2):126–140, May 1994. doi:10.1287/ijoc.6.2.126. Applications of Reactive Tabu Search (RTS) to the 0-1 Knapsack and QAP. (Cit. on pp. 88, 89).

[17] Roberto Battiti and Giampietro Tecchiolli. "Training Neural Nets with the Reactive Tabu Search". *IEEE Transactions on Neural Networks* 6(5):1185–1200, Sept. 1995. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1045-9227. doi:10.1109/72.410361. See also "Training Neural Nets with the Reactive Tabu Search, Technical Report UTM 421, Trento, Alto Adige/Südtirol, Italy: University of Trento, 11 1993" (cit. on p. 88).

[18] James C. Bean. "Genetic Algorithms and Random Keys for Sequencing and Optimization". *ORSA Journal on Computing* 6(2):107–220, Spr. 1994. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.6.2.154. Applies the Random Keys encoding to scheduling and the QAP. See [19]. (Cit. on pp. 80, 88, 95).

[19] James C. Bean. *Genetics and Random Keys for Sequencing and Optimization*. Tech. rep. 92-43. Ann Arbor, MI, USA: University of Michigan, Department of Industrial & Operations Engineering, June 30, 1992. URL: https://www.math.wm.edu/~rrkinc/homework/RdmKeyGA.pdf (visited on 2026-10-01). See [18]. (Cit. on pp. 80, 88, 95).

[20] John Edward Beasley. "OR-Library: Distributing Test Problems by Electronic Mail". *The Journal of the Operational Research Society (JORS)* 41:1069–1072, Nov. 1990. London, England, UK: Taylor and Francis Ltd. and London, England, UK: Springer Nature Limited. ISSN: 0160-5682. doi:10.1057/jors.1990.166. See the website [21] (cit. on pp. 83, 87, 95).

[21] John Edward Beasley. *Welcome to OR-Library*. Uxbridge, Middlesex, England, UK: Brunel University of London, June 1990–Feb. 2018. URL: https://people.brunel.ac.uk/~mastjjb/jeb/info.html (visited on 2026-09-30). See the article [20]. Contains instances of the JSSP, 1D- and 2D-Packing, Knapsack, periodic TSPs, amongst others. (Cit. on pp. 83, 87, 95).

[22] Kent L. Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., Sept. 2004. ISBN: 978-0-596-00743-0 (cit. on p. 91).

[23] Andreas Beham, Michael Affenzeller, and Erik Pitzer. "Metaheuristic Algorithms for the Quadratic Assignment Problem: Performance and Comparison". In: *Innovative Technologies in Management and Science*. Ed. by Ryszard Klempous and Jan Nikodem. Vol. 10 of Topics in Intelligent Engineering and Informatics (TIEI). Cham, Switzerland: Springer, 2015, pp. 171–190. ISSN: 2193-9411. ISBN: 978-3-319-12651-7. doi:10.1007/978-3-319-12652-4_10. Algorithms like SLS/iterated local search (ILS), SA, TS, VNS, EAs, MAs are compared and, with the exception of SA and EA, all perform best on some instance groups of QAPLIB. (Cit. on p. 87).

[24] Richard Beigel. "Irrationality without Number Theory". *The American Mathematical Monthly* 98(4):332–335, Apr. 1991. London, England, UK: Taylor and Francis Ltd. ISSN: 1930-0972. doi:10.1080/00029890.1991.12000762. URL: https://cis.temple.edu/~beigel/papers/irrationality-amm.PS.gz (visited on 2026-11-04) (cit. on p. 83).

[25] Richard Ernest Bellman. *Dynamic Programming*. Princeton Landmarks in Mathematics and Physics. Princeton, NJ, USA: Princeton University Press, 1957–2003. ISBN: 978-0-691-14668-3 (cit. on p. 86).

[26] Bengt-Erik Bengtsson. "Packing Rectangular Pieces— A Heuristic Approach". *The Computer Journal* 25(3):353–357, Aug. 1982. Oxford, Oxfordshire, England, UK: Oxford University Press. ISSN: 0010-4620. doi:10.1093/comjnl/25.3.353 (cit. on p. 78).

[27] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) and Newsgroups: alt.hypertext, Aug. 6, 1991. URL: https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt (visited on 2025-02-05) (cit. on p. 91).

[28] Tim Berners-Lee, Larry Masinter, and Mark P. McCahill. *Uniform Resource Locators (URL)*. Request for Comments (RFC) 1738. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Dec. 1994. URL: https://www.ietf.org/rfc/rfc1738.txt (visited on 2025-02-05) (cit. on p. 91).

[29] Alex Berson. *Client/Server Architecture*. 2nd ed. Computer Communications Series. New York, NY, USA: McGraw-Hill, Mar. 29, 1996. ISBN: 978-0-07-005664-0 (cit. on p. 80).

[30] Muhammed Beyaz, Tansel Dökeroğlu, and Ahmet Cosar. "Robust Hyper-Heuristic Algorithms for the Offline Oriented/Non-Oriented 2D Bin Packing Problems". *Applied Soft Computing (ASOC)* 36:236–245, Nov. 2015. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 1568-4946. doi:10.1016/j.asoc.2015.06.063 (cit. on p. 78).

[31] Christian Bierwirth. "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms". *OR Spectrum: Quantitative Approaches in Management. Formerly* Operations-Research-Spektrum 17(2–3):87–92, June 1995. London, England, UK: Springer Nature Limited. ISSN: 0171-6468. doi:10.1007/BF01719250. URL: https://www.researchgate.net/publication/240263036 (visited on 2026-09-30). Proposes a permutations-with-repetitions based encoding for the JSSP, see also [32]. (Cit. on pp. 81, 83, 96, 104).

[32] Christian Bierwirth, Dirk C. Mattfeld, and Herbert Kopfer. "On Permutation Representations for Scheduling Problems". In: *4th International Conference on Parallel Problem Solving from Nature (PPSN IV)*. Sept. 22–24, 1996, Berlin, Germany. Ed. by Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel. Vol. 1141 of Lecture Notes in Computer Science (LNCS). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 1996, pp. 310–318. ISSN: 0302-9743. ISBN: 978-3-540-61723-5. doi:10.1007/3-540-61723-X_995. URL: https://www.researchgate.net/publication/2753293 (visited on 2026-09-30). Proposes a permutations-with-repetitions based encoding for the JSSP, see also [31]. (Cit. on pp. 81, 83, 96, 104).

[33] Jacek Błażewicz, Wolfgang Domschke, and Erwin Pesch. "The Job Shop Scheduling Problem: Conventional and New Solution Techniques". *European Journal of Operational Research* 93(1):1–33, Aug. 1996. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/0377-2217(95)00362-2 (cit. on p. 83).

[34] Joshua Bloch. *Effective Java*. Reading, MA, USA: Addison-Wesley Professional, May 2008. ISBN: 978-0-321-35668-0 (cit. on p. 83).

[35] Christian Blum and Verena Schmid. "Solving the 2D Bin Packing Problem by Means of a Hybrid Evolutionary Algorithm". In: *International Conference on Computational Science (ICCS'2013)*. June 5–7, 2013, Barcelona, Spain. Ed. by Vassil Alexandrov, Michael Lees, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot. Vol. 18 of Procedia Computer Science. Amsterdam, The Netherlands: Elsevier B.V., 2013, pp. 899–908. ISSN: 1877-0509. doi:10.1016/j.procs.2013.05.255. An EA for the 2BP|O|F using the heuristic by Wong and Lee [347]. (Cit. on p. 78).

[36] Borko Bošković and Janez Brest. "Computational Search of Long Skew-Symmetric Binary Sequences with High Merit Factors". *MENDEL – Soft Computing Journal* 28(2):17–24, Dec. 2022. Brno, Czech Republic: Brno University of Technology (Vysoké učení technické v Brně). ISSN: 1803-3814. doi:10.13164/mendel.2022.2.017 (cit. on p. 84).

[37] Borko Bošković, Franc Brglez, and Janez Brest. "Low-Autocorrelation Binary Sequences: On Improved Merit Factors and Runtime Predictions to Achieve Them". *Applied Soft Computing (ASOC)* 56:262–285, July 2017. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 1568-4946. doi:10.1016/J.ASOC.2017.02.024 (cit. on p. 84).

[38] Silvia Botros and Jeremy Tinley. *High Performance MySQL*. 4th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Nov. 2021. ISBN: 978-1-4920-8051-0 (cit. on p. 85).

[39] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (cit. on p. 85).

[40] Ron Brash and Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, July 2018. ISBN: 978-1-78862-936-2 (cit. on p. 79).

[41] Janez Brest and Borko Bošković. "A Heuristic Algorithm for a Low Autocorrelation Binary Sequence Problem with Odd Length and High Merit Factor". *IEEE Access* 6:4127–4134, Jan. 2018. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 2169-3536. doi:10.1109/ACCESS.2018.2789916 (cit. on p. 84).

[42] Florian Bruhin. *Python f-Strings*. Winterthur, Switzerland: Bruhin Software, May 31, 2023. URL: https://fstring.help (visited on 2024-07-25) (cit. on p. 81).

[43] "Built-in Functions: `round(number, ndigits=None)`". In: *Python 3 Documentation*. *The Python Standard Library*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/builtins/functions.html#round (visited on 2026-10-25) (cit. on p. 79).

[44] Rainer E. Burkard, Eranda Çela, Stefan E. Karisch, Franz Rendl, Miguel F. Anjos, and Peter M. Hahn. *QAPLIB – A Quadratic Assignment Problem Library – Problem Instances and Solutions*. Edinburgh, Scotland, UK: University of Edinburgh, Apr. 20, 2012–Mar. 31, 2022. doi:10.7488/ds/3428. This is the 2012 status of the QAPLIB dataset from [46] (cit. on pp. 87, 94, 97).

[45] Rainer E. Burkard, Eranda Çela, Panos Miltiades Pardalos, and Leonidas S. Pitsoulis. "The Quadratic Assignment Problem". In: *Handbook of Combinatorial Optimization*. Ed. by Panos Miltiades Pardalos, Ding-Zhu Du, and Ronald Lewis Graham. 1st ed. Boston, MA, USA: Springer, 1998, pp. 1713–1809. ISBN: 978-1-4613-7987-4. doi:10.1007/978-1-4613-0303-9_27 (cit. on p. 87).

[46] Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl. "QAPLIB – A Quadratic Assignment Problem Library". *Journal of Global Optimization* 10(4):391–403, June 1997. London, England, UK: Springer Nature Limited. ISSN: 0925-5001. doi:10.1023/A:1008293323270. This article introduces QAPLIB, a set of benchmark problems for the QAP, now hosted at [44]. (Cit. on pp. 87, 94, 97).

[47] Rainer E. Burkard and Josef Offermann. "Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme". *Zeitschrift für Operations Research (ZOR). now* Mathematical Methods of Operations Research (MMOR) 21:B121–B132, Aug. 1977. Heidelberg, Baden-Württemberg, Germany: Physica-Verlag; now London, England, UK: Springer Nature Limited. ISSN: 0340-9422. doi:10.1007/BF01918175. Keyboard layout modeled as QAPs. (Cit. on p. 87).

[48] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, June 2022 (cit. on pp. 84, 88).

[49] Jacques Carlier and Éric Pinson. "An Algorithm for Solving the Job-Shop Problem". *Management Science* 35(2):164–176, 1989. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0025-1909. doi:10.1287/mnsc.35.2.164. A branch and bound (B&B) method for the JSSP. (Cit. on p. 79).

[50] Stephan C. Carlson and The Editors of Encyclopaedia Britannica. *Golden Ratio*. Ed. by The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., Oct. 21, 2024. URL: https://www.britannica.com/science/golden-ratio (visited on 2024-12-14) (cit. on p. 77).

[51] Eduardo Carvalho Pinto and Carola Doerr. "A Simple Proof for the Usefulness of Crossover in Black-Box Optimization". In: *15th International Conference on Parallel Problem Solving from Nature (PPSN XV)*. Vol. 2. Sept. 8–12, 2018, Coimbra, Portugal. Ed. by Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and L. Darrell Whitley. Vol. 11102 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer, 2018, pp. 29–41. ISSN: 0302-9743. ISBN: 978-3-319-99258-7. doi:10.1007/978-3-319-99259-4_3 (cit. on p. 82).

[52] Eduardo Carvalho Pinto and Carola Doerr. *Towards a More Practice-Aware Runtime Analysis of Evolutionary Algorithms*. arXiv.org: Computing Research Repository (CoRR) abs/1812.00493. Ithaca, NY, USA: Cornell Universiy Library, Dec. 3, 2018. doi:10.48550/arXiv.1812.00493. URL: https://arxiv.org/abs/1812.00493 (visited on 2025-08-08). arXiv:1812.00493v1 [cs.NE] 3 Dec 2018 (cit. on pp. 77, 82, 88).

[53] Antonio Cavacini. "Is the CE/BCE notation becoming a standard in scholarly literature?" *Scientometrics* 102(2):1661–1668, July 2015. London, England, UK: Springer Nature Limited. ISSN: 0138-9130. doi:10.1007/s11192-014-1352-1 (cit. on pp. 79, 80).

[54] Josh Centers. *Take Control of iOS 18 and iPadOS 18*. San Diego, CA, USA: Take Control Books, Dec. 2024. ISBN: 978-1-990783-55-5 (cit. on p. 83).

[55] Vladimír Černý. "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm". *Journal of Optimization Theory and Applications* 45(1):41–51, Jan. 1985. New York, NY, USA: Springer Science+Business Media, LLC. ISSN: 0022-3239. doi:10.1007/BF00940812 (cit. on p. 88).

[56] Noureddine Chabini and Rachid Beguenane. "FPGA-Based Designs of the Factorial Function". In: *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2022)*. Sept. 18–20, 2022, Halifax, NS, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 16–20. ISBN: 978-1-6654-8432-9. doi:10.1109/CCECE49351.2022.9918302 (cit. on p. 77).

[57] Donald D. Chamberlin. "50 Years of Queries". *Communications of the ACM (CACM)* 67(8):110–121, Aug. 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/3649887. URL: https://cacm.acm.org/research/50-years-of-queries (visited on 2025-01-09) (cit. on p. 89).

[58] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. "A Review of Machine Scheduling: Complexity, Algorithms and Approximability". In: *Handbook of Combinatorial Optimization*. Ed. by Panos Miltiades Pardalos, Ding-Zhu Du, and Ronald Lewis Graham. 1st ed. Boston, MA, USA: Springer, 1998, pp. 1493–1641. ISBN: 978-1-4613-7987-4. doi:10.1007/978-1-4613-0303-9_25. See also pages 21–169 in volume 3/3 by Norwell, MA, USA: Kluwer Academic Publishers. Discusses, amongst others, that the JSSP is $\mathcal{NP}$-hard. (Cit. on pp. 83, 86).

[59] Stephen Chen, James Montgomery, and Antonio Bolufé-Röhler. "Measuring the Curse of Dimensionality and its Effects on Particle Swarm Optimization and Differential Evolution". *Applied Intelligence – The International Journal of Research on Intelligent Systems for Real Life Complex Problems* 42:514–526, Apr. 2015. London, England, UK: Springer Nature Limited. ISSN: 0924-669X. doi:10.1007/s10489-014-0613-2 (cit. on p. 86).

[60] Jiayang Chen (陈嘉阳), Zhize Wu (吴志泽), Sarah Louise Thomson, and Thomas Weise (汤卫思). "Frequency Fitness Assignment: Optimization Without Bias for Good Solution Outperforms Randomized Local Search on the Quadratic Assignment Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 27–37. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888600003837 (cit. on pp. 81, 85, 87).

[61] T. C. Edwin Cheng, Bo Peng, and Zhipeng Lü. "A Hybrid Evolutionary Algorithm to Solve the Job Shop Scheduling Problem". *Annals of Operations Research* 242(2):223–237, July 2016. London, England, UK: Springer Nature Limited. ISSN: 0254-5330. doi:10.1007/s10479-013-1332-5. A MA utlizing TS (cit. on p. 84).

[62] Philippe Chrétienne, Edward G. Coffman, Jan Karel Lenstra, and Zhen Liu, eds. *Scheduling Theory and Its Applications*. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., Sept. 1995. ISBN: 978-0-471-94059-3 (cit. on p. 83).

[63] Nestor M. Cid-Garcia and Yasmin A. Rios-Solis. "Positions and Covering: A Two-Stage Methodology to Obtain Optimal Solutions for the 2D-Bin Packing Problem". *PLoS ONE* 15(4):e0229358, Apr. 2020. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1932-6203. doi:10.1371/journal.pone.0229358. An exact method for the 2BP|R|F and the 2BP|O|F. (Cit. on p. 78).

[64] David Clinton and Christopher Negus. *Ubuntu Linux Bible*. 10th ed. Bible Series. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., Nov. 10, 2020. ISBN: 978-1-119-72233-5 (cit. on pp. 89, 90).

[65] Edgar Frank "Ted" Codd. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM (CACM)* 13(6):377–387, June 1970. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/362384.362685. URL: https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf (visited on 2025-01-05) (cit. on p. 88).

[66] Stephen Arthur Cook. "The Complexity of Theorem-Proving Procedures". In: *Third Annual ACM Symposium on Theory of Computing (STOC'1971)*. May 3–5, 1971, Shaker Heights, OH, USA. Ed. by Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman. New York, NY, USA: Association for Computing Machinery (ACM), 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. doi:10.1145/800157.805047 (cit. on pp. 85, 86).

[67] William John Cook. *Concorde TSP Solver*. Waterloo, ON, Canada: University of Waterloo, 1997. URL: https://math.uwaterloo.ca/tsp/concorde (visited on 2026-09-26) (cit. on p. 80).

[68] William John Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton, NJ, USA: Princeton University Press, 2012–Nov. 9, 2014. ISBN: 978-0-691-16352-9 (cit. on pp. 80, 84).

[69] William John Cook. *World TSP*. Waterloo, ON, Canada: University of Waterloo, 2013–Oct. 25, 2025. URL: http://www.math.uwaterloo.ca/tsp/world (visited on 2026-01-03) (cit. on pp. 84, 90).

[70] Thomas H. Cormen, Charles E. Leiserson, Ronald Linn Rivest, and Clifford Stein. "Asymptotic Notation". In: *Introduction to Algorithms*. 3rd ed. Cambridge, MA, USA: MIT Press, 2009. Chap. 3.1, pp. 43–53. ISBN: 978-0-262-03384-8. Discusses Θ, $\mathcal{O}$, Ω, o, and ω (cit. on pp. 86, 91).

[71] Cemre Çubukçuoğlu, Pirouz Nourian, M. Fatih Tasgetiren, I. Sevil Sariyildiz, and Shervin Azadi. "Hospital Layout Design Renovation as a Quadratic Assignment Problem with Geodesic Distances". *Journal of Building Engineering* 44:102952, Dec. 2021. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 2352-7102. doi:10.1016/j.jobe.2021.102952 (cit. on p. 87).

[72] Christopher Cullen. "Learning from Liu Hui? A Different Way to Do Mathematics". *Notices of the American Mathematical Society* 49(7):783–790, Aug. 2002. Providence, RI, USA: American Mathematical Society (AMS). ISSN: 1088-9477. URL: https://www.ams.org/notices/200207/comm-cullen.pdf (visited on 2024-08-09) (cit. on p. 77).

[73] *Database Language SQL*. Tech. rep. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (cit. on p. 89).

[74] Joseph W. Dauben. "Archimedes and Liu Hui on Circles and Spheres". *Ontology Studies (Cuadernos de Ontología)* 10:21–38, 2010. Leioa, Bizkaia, Spain: Universidad del País Vasco / Euskal Herriko Unibertsitatea. ISSN: 1576-2270. URL: https://ddd.uab.cat/pub/ontstu/ontstu_a2010n10/ontstu_a2010n10p21.pdf (visited on 2026-11-03). Also discusses LIU Hui's (刘徽) method of approximating π (cit. on p. 77).

[75] Matt David and Blake Barnhill. *How to Teach People SQL*. San Francisco, CA, USA: The Data School, Chart.io, Inc., Dec. 10, 2019–Apr. 10, 2023. URL: https://dataschool.com/how-to-teach-people-sql (visited on 2025-02-27) (cit. on p. 89).

[76] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (cit. on p. 89).

[77] Kenneth Alan De Jong. *Evolutionary Computation: A Unified Approach*. Vol. 4 of Complex Adaptive Systems. Cambridge, MA, USA: MIT Press, 2006. ISBN: 978-0-262-04194-2. URL: https://www.researchgate.net/publication/220740669 (visited on 2025-08-08) (cit. on pp. 81, 88).

[78] *Definition of Operations Research*. University of Western Ontario, London, ON, Canada: International Federation of Operational Research Societies (IFORS), 2020. URL: https://www.ifors.org/what-is-or (visited on 2026-01-01) (cit. on p. 87).

[79] Paul Deitel, Harvey Deitel, and Abbey Deitel. *Internet & World Wide Web: How to Program*. 5th ed. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (cit. on p. 91).

[80] Maxence Delorme, Manuel Iori, and Silvano Martello. "Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms". *European Journal of Operational Research* 255(1):1–20, Nov. 2016. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/j.ejor.2016.04.030 (cit. on p. 79).

[81] Ebru Demirkol, Sanjay V. Mehta, and Reha Uzsoy. "Benchmarks for Shop Scheduling Problems". *European Journal of Operational Research* 109(1):137–141, Aug. 1998. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/S0377-2217(97)00019-2 (cit. on p. 83).

[82] Alfredo Deza and Noah Gift. *Testing In Python*. San Francisco, CA, USA: Pragmatic AI Labs, Feb. 2020. ISBN: 979-8-6169-6064-1 (cit. on p. 87).

[83] Miroslav Dimitrov. *New Classes of Binary Sequences with High Merit Factor*. arXiv.org: Computing Research Repository (CoRR). Ithaca, NY, USA: Cornell Universiy Library, June 24, 2022. doi:10.48550/arXiv.2206.12070. URL: https://arxiv.org/pdf/2206.12070 (visited on 2026-09-21). arXiv:2206.12070v1 [cs.IT] 24 Jun 2022 (cit. on p. 84).

[84] Miroslav Dimitrov, Tsonka Baitcheva, and Nikolay Nikolov. "Efficient Generation of Low Autocorrelation Binary Sequences". *IEEE Signal Processing Letters* 27:341–345, 2020. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1558-2361. doi:10.1109/LSP.2020.2972127 (cit. on p. 84).

[85] Slobodan Dmitrović. *Modern C for Absolute Beginners: A Friendly Introduction to the C Programming Language*. New York, NY, USA: Apress Media, LLC, Mar. 2024. ISBN: 979-8-8688-0224-9 (cit. on p. 80).

[86] "Doctest – Test Interactive Python Examples". In: *Python 3 Documentation*. *The Python Standard Library*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/library/doctest.html (visited on 2024-11-07) (cit. on p. 80).

[87] Benjamin Doerr and Carola Doerr. "Optimal Static and Self-Adjusting Parameter Choices for the $(1+(\lambda,\lambda))$ Genetic Algorithm". *Algorithmica* 80(5):1658–1709, May 2018. London, England, UK: Springer Nature Limited. ISSN: 0178-4617. doi:10.1007/s00453-017-0354-9 (cit. on p. 88).

[88] Benjamin Doerr, Carola Doerr, and Franziska Ebel. "From Black-Box Complexity to Designing New Genetic Algorithms". *Theoretical Computer Science* 567:87–104, Feb. 2015. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1016/J.TCS.2014.11.028 (cit. on p. 87).

[89] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, and Ofer M. ShirThomas Bäck. "Benchmarking Discrete Optimization Heuristics with IOHprofiler". In: *21st Genetic and Evolutionary Computation Conference Companion (GECCO'2019)*. July 13–17, 2019, Prague, Czech Republic. Ed. by Manuel López-Ibáñez, Anne Auger, and Thomas Stützle. New York, NY, USA: Association for Computing Machinery (ACM), 2019, pp. 1798–1806. ISBN: 978-1-4503-6748-6. doi:10.1145/3319619.3326810. See also [90, 91] (cit. on p. 100).

[90] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, and Ofer M. ShirThomas Bäck. "Benchmarking Discrete Optimization Heuristics with IOHprofiler". *Applied Soft Computing (ASOC)* 88:106027, Mar. 2020. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 1568-4946. doi:10.1016/j.asoc.2019.106027. See also [89, 91] (cit. on pp. 78, 83, 86, 100).

[91] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, and Ofer M. ShirThomas Bäck. "Benchmarking Discrete Optimization Heuristics with IOHprofiler". (abs/1912.09237), Jan. 31, 2021. doi:10.48550/arXiv.1912.09237. URL: https://arxiv.org/abs/1912.09237 (visited on 2024-06-27). arXiv:1912.09237v2 [cs.NE] 31 Jan 2021, see also [89, 90] (cit. on p. 100).

[92] Marco Dorigo, Mauro Birattari, and Thomas Stützle. "Ant Colony Optimization". *IEEE Computational Intelligence Magazine (CIM)* 1(4):28–39, Nov. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1556-603X. doi:10.1109/MCI.2006.329691. URL: https://www.researchgate.net/publication/308953674 (visited on 2025-07-25) (cit. on p. 78).

[93] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant System: Optimization by a Colony of Cooperating Agents". *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* 26(1):29–41, Feb. 1996. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1083-4419. doi:10.1109/3477.484436. URL: https://www.researchgate.net/publication/5589170 (visited on 2025-07-26) (cit. on p. 78).

[94] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004. ISBN: 978-0-262-04219-2 (cit. on p. 78).

[95] Johann Dréo, Alain Pétrowski, Patrick Siarry, and Éric D. Taillard. *Metaheuristics for Hard Optimization – Methods and Case Studies*. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2006. ISBN: 978-3-540-23022-9. doi:10.1007/3-540-30966-7. Discusses SA, TS, EAs, and Ant Colony Optimization (ACO). (Cit. on p. 87).

[96] Stefan Droste, Thomas Jansen, and Ingo Wegener. "On the Analysis of the $(1+1)$ Evolutionary Algorithm". *Theoretical Computer Science* 276(1-2):51–81, Apr. 2002. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1016/S0304-3975(01)00182-7 (cit. on pp. 77, 78, 83, 84, 89).

[97] Stefan Droste, Thomas Jansen, and Ingo Wegener. "Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization". *Theoretical Computer Science* 39(4):525–544, July 2006. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1007/S00224-004-1177-Z (cit. on p. 86).

[98] Jacques Dutka. "The Early History of the Factorial Function". *Archive for History of Exact Sciences* 43(3):225–249, Sept. 1991. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany. ISSN: 0003-9519. doi:10.1007/BF00389433. Communicated by Umberto Bottazzini (cit. on p. 77).

[99] Harald Dyckhoff and Ute Finke. *Cutting and Packing in Production and Distribution: A Typology and Bibliography*. Contributions to Management Science. Heidelberg, Baden-Württemberg, Germany: Physica-Verlag, 1992. ISSN: 1431-1941. ISBN: 978-3-642-63487-1. doi:10.1007/978-3-642-58165-6 (cit. on p. 79).

[100] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2015. ISBN: 978-1-4493-6290-4 (cit. on p. 85).

[101] Kelly Easton, George L. Nemhauser, and Michael A. Trick. "The Traveling Tournament Problem Description and Benchmarks". In: *7th International Conference on Principles and Practice of Constraint Programming (CP'01)*. Nov. 26–Dec. 1, 2001, Paphos, Cyprus. Ed. by Toby Walsh. Vol. 2239 of Lecture Notes in Computer Science (LNCS). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2001, pp. 580–584. ISSN: 0302-9743. ISBN: 978-3-540-42863-3. doi:10.1007/3-540-45578-7_43 (cit. on p. 90).

[102] Alwalid N. Elshafei. "Hospital Layout as a Quadratic Assignment Problem". *The Journal of the Operational Research Society (JORS)* 28(1):167–179, Apr. 1977. London, England, UK: Taylor and Francis Ltd. and London, England, UK: Springer Nature Limited. ISSN: 0160-5682. doi:10.1057/jors.1977.29 (cit. on p. 87).

[103] Bernhard Eschermann and Hans-Joachim Wunderlich. "Optimized Synthesis of Self-Testable Finite State Machines". In: *20th International Symposium Fault-Tolerant Computing (FTCS'1990): Digest of Papers*. June 26–28, 1990, Newcastle Upon Tyne, England, UK. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 1990. ISBN: 978-0-8186-2051-5. doi:10.1109/FTCS.1990.89393 (cit. on p. 87).

[104] Euclid of Alexandria ($Εὐκλείδης$). *Euclid's Elements of Geometry ($Στοιχεῖα$). The Greek Text of J.L. Heiberg (1883-1885) from Euclidis Elementa, Edidit et Latine Interpretatus est I.L. Heiberg in Aedibus B. G. Teubneri, 1883-1885. Edited, and provided with a modern English translation, by Richard Fitzpatrick*. Ed. by Richard Fitzpatrick. Trans. by Johan Ludvig Heiberg. revised and corrected. Austin, TX, USA: The University of Texas at Austin, 2008. ISBN: 978-0-615-17984-1. URL: https://farside.ph.utexas.edu/Books/Euclid/Elements.pdf (visited on 2024-09-30) (cit. on p. 77).

[105] Leonhard Euler. "An Essay on Continued Fractions". Trans. by Myra F. Wyman and Bostwick F. Wyman. *Mathematical Systems Theory* 18(1):295–328, Dec. 1985. New York, NY, USA: Springer Science+Business Media, LLC. ISSN: 1432-4350. doi:10.1007/BF01699475. URL: https://www.researchgate.net/publication/301720080 (visited on 2024-09-24). Translation of [106]. (Cit. on p. 102).

[106] Leonhard Euler. "De Fractionibus Continuis Dissertation". *Commentarii Academiae Scientiarum Petropolitanae* 9:98–137, 1737–1744. Petropolis (St. Petersburg), Russia: Typis Academiae. URL: https://scholarlycommons.pacific.edu/cgi/viewcontent.cgi?article=1070 (visited on 2024-09-24). See [105] for a translation. (Cit. on pp. 81, 101).

[107] Thomas A. Feo and Mauricio G. C. Resende. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization* 6(2):109–133, Mar. 1995. London, England, UK: Springer Nature Limited. ISSN: 0925-5001. doi:10.1007/BF01096763. Introduces Greedy Randomized Adaptive Search Procedures (GRASPs) and applies them to the set covering problem, among others, and lists many problem domains. (Cit. on p. 82).

[108] Luca Ferrari and Enrico Pirozzi. *Learn PostgreSQL*. 2nd ed. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2023. ISBN: 978-1-83763-564-1 (cit. on p. 87).

[109] Paola Festa and Mauricio G. C. Resende. "An Annotated Bibliography of GRASP—Part I: Algorithms". *International Transactions on Operational Research (ITOR)* 16(1):1–24, Jan. 2009. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd. ISSN: 0969-6016. doi:10.1111/j.1475-3995.2009.00663.x. See also [110] (cit. on pp. 82, 102).

[110] Paola Festa and Mauricio G. C. Resende. "An Annotated Bibliography of GRASP—Part II: Applications". *International Transactions on Operational Research (ITOR)* 16(2):131–172, Mar. 2009. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd. ISSN: 0969-6016. doi:10.1111/j.1475-3995.2009.00664.x. See also [109] (cit. on pp. 82, 102).

[111] Michael Filaseta. "The Transcendence of e and π". In: *Math 785: Transcendental Number Theory*. Columbia, SC, USA: University of South Carolina, Spr. 2011. Chap. 6. URL: https://people.math.sc.edu/filaseta/gradcourses/Math785/Math785Notes6.pdf (visited on 2024-07-05) (cit. on pp. 77, 81).

[112] Simon Fischer. "A Polynomial Upper Bound for a Mutation-Based Algorithm on the Two-Dimensional Ising Model". In: *6th Genetic and Evolutionary Computation Conference (GECCO'2004), Part I*. June 26–30, 2004, Seattle, WA, USA. Ed. by Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell. Vol. 3102 of Lecture Notes in Computer Science (LNCS). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2004, pp. 1100–1112. ISSN: 0302-9743. ISBN: 978-3-540-22344-3. doi:10.1007/978-3-540-24854-5_108 (cit. on p. 83).

[113] Simon Fischer and Ingo Wegener. "The One-Dimensional Ising Model: Mutation versus Recombination". *Theoretical Computer Science* 344(2-3):208–225, Nov. 2005. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1016/J.TCS.2005.04.002 (cit. on p. 83).

[114] Henry Fisher and Gerald L. Thompson. "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules". In: *Industrial Scheduling*. Ed. by John F. Muth and Gerald L. Thompson. Englewood Cliffs, NJ, USA: Prentice-Hall, 1963. Chap. 3.2, pp. 225–251 (cit. on p. 83).

[115] Philip J. Fleming and John J. Wallace. "How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results". *Communications of the ACM (CACM)* 29(3):218–221, Mar. 1986. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/5666.5673. This paper also presents that the geometric mean is the only meaningful average measure when dealing with normalized data (as opposed to the arithmetic mean or median (cit. on p. 82).

[116] Charles Fleurent and Jacques A. Ferland. "Genetic Hybrids for the Quadratic Assignment Problem". In: *Quadratic Assignment and Related Problems, Proceedings of a DIMACS Workshop*. May 20–21, 1993, New Brunswick, NJ, USA: Rutgers University. Ed. by Panos Miltiades Pardalos and Henry Wolkowicz. Providence, RI, USA: American Mathematical Society (AMS), 1994, pp. 173–187. ISBN: 978-0-8218-6607-8. A MA using the TS from [303] as local search performs well. (Cit. on pp. 84, 87).

[117] Tobias Friedrich, Pietro Simone Oliveto, Dirk Sudholt, and Carsten Witt. "Analysis of Diversity-Preserving Mechanisms for Global Exploration". *Evolutionary Computation* 17(4):455–476, Win. 2009. Cambridge, MA, USA: MIT Press. ISSN: 1063-6560. doi:10.1162/evco.2009.17.4.17401 (cit. on p. 90).

[118] Tobias Friedrich, Francesco Quinzan, and Markus Wagner. "Escaping Large Deceptive Basins of Attraction with Heavy-Tailed Mutation Operators". In: *20th Genetic and Evolutionary Computation Conference (GECCO'2018)*. July 15–19, 2018, Kyōto, Kyōto-fu, Japan. Ed. by Hernán E. Aguirre and Keiki Takadama. New York, NY, USA: Association for Computing Machinery (ACM), 2018, pp. 293–300. ISBN: 978-1-4503-5618-3. doi:10.1145/3205455.3205515 (cit. on pp. 83, 90).

[119] J. Newton Friend. "The Origin of Fahrenheit's Thermometric Scale". *Nature* 139(3514):395–398, Mar. 6, 1937. London, England, UK: Springer Nature Limited. ISSN: 0028-0836. doi:10.1038/139395a0 (cit. on pp. 42, 115).

[120] "Formatted String Literals". In: *Python 3 Documentation. The Python Tutorial*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. Chap. 7.1.1. URL: https://docs.python.org/3/tutorial/inputoutput.html#formatted-string-literals (visited on 2024-07-25) (cit. on p. 81).

[121] José E. Gallardo, Carlos Cotta, and Antonio J. Fernández. "A Memetic Algorithm for the Low Autocorrelation Binary Sequence Problem". In: *9th Genetic and Evolutionary Computation Conference (GECCO'2007)*. July 7–11, 2007, London, England, UK. Ed. by Hod Lipson. New York, NY, USA: Association for Computing Machinery (ACM), 2007, pp. 1226–1233. ISBN: 978-1-59593-697-4. doi:10.1145/1276958.1277195 (cit. on p. 84).

[122] Michael R. Garey and David Stifler Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman and Company, 1979. ISBN: 978-0-7167-1045-5 (cit. on p. 90).

[123] Bhavesh Gawade. "Mastering F-Strings in Python: Efficient String Handling in Python Using Smart F-Strings". In: *C O D E B*. Mumbai, Maharashtra, India: Code B Solutions Pvt Ltd, Apr. 25–June 3, 2025. URL: https://code-b.dev/blog/f-strings-in-python (visited on 2025-08-04) (cit. on p. 81).

[124] Mitsuo Gen, Yasuhiro Tsujimura, and Erika Kubota. "Solving Job-Shop Scheduling Problems by Genetic Algorithm". In: *Humans, Information and Technology: Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics (SMC'1994)*. Vol. 2. Oct. 2–5, 1994, San Antonio, TX, USA. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 1994, pp. 1577–1582. ISBN: 978-0-7803-2129-8. doi:10.1109/ICSMC.1994.400072. Proposes a permutations-with-repetitions based encoding for the JSSP. (Cit. on pp. 81, 83, 104).

[125] Fred Glover. "Tabu Search – Part I". *ORSA Journal on Computing* 1(3):190–206, Sum. 1989. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.1.3.190 (cit. on p. 89).

[126] Fred Glover. "Tabu Search – Part II". *ORSA Journal on Computing* 2(1):4–32, Win. 1990. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.2.1.4 (cit. on p. 89).

[127] Fred Glover and Gary A. Kochenberger, eds. *Handbook of Metaheuristics*. Vol. 57 of International Series in Operations Research & Management Science (ISOR). Cham, Switzerland: Springer, 2003. ISSN: 0884-8289. ISBN: 978-0-306-48056-0. doi:10.1007/b101874 (cit. on p. 85).

[128] Fred Glover, Éric D. Taillard, and Dominique de Werra. "A User's Guide to Tabu Search". *Annals of Operations Research* 41(1):3–28, Mar. 1993. London, England, UK: Springer Nature Limited. ISSN: 0254-5330. doi:10.1007/BF02078647. Special issue on TS. (Cit. on p. 89).

[129] David Edward Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1989. ISBN: 978-0-201-15767-3 (cit. on pp. 81, 88).

[130] José Fernando Gonçalves and Mauricio G. C. Resende. "A Biased Random Key Genetic Algorithm for 2BP and 3D Bin Packing Problems". *International Journal of Production Economics (IJPE)* 145(2):500–510, Oct. 2013. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0925-5273. doi:10.1016/j.ijpe.2013.04.019. Applies the Biased Random Key Genetic Algorithm (BRKGA) to the 2BP|R|F and 2BP|O|F as well as 3D packing. Continuous vectors are translated to permutations and orientation data which is then the input of a placement heuristic. (Cit. on pp. 78, 80).

[131] José Fernando Gonçalves and Mauricio G. C. Resende. "An Extended Akers Graphical Method with a Biased Random-Key Genetic Algorithm for Job-Shop Scheduling". *International Transactions on Operational Research (ITOR)* 21(2):215–246, Mar. 2014. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd. ISSN: 0969-6016. doi:10.1111/itor.12044. URL: http://mauricio.resende.info/doc/brkga-jss2011.pdf (visited on 2026-09-30). The BRKGA basically is a MA using a continuous encoding of permutations that are decoded to schedules using the method from [31, 32, 124, 280] and then refined via a TS. (Cit. on pp. 80, 83).

[132] José Fernando Gonçalves and Mauricio G. C. Resende. "Biased Random-Key Genetic Algorithms for Combinatorial Optimization". *Journal of Heuristics* 17:487–525, Oct. 2010. London, England, UK: Springer Nature Limited. ISSN: 1381-1231. doi:10.1007/s10732-010-9143-1 (cit. on p. 80).

[133] David Goodger and Guido van Rossum. *Docstring Conventions*. Python Enhancement Proposal (PEP) 257. Beaverton, OR, USA: Python Software Foundation (PSF), May 29–June 13, 2001. URL: https://peps.python.org/pep-0257 (visited on 2024-07-27) (cit. on p. 80).

[134] Michael T. Goodrich. *A Gentle Introduction to NP-Completeness*. Irvine, CA, USA: University of California, Irvine, Apr. 2022. URL: https://ics.uci.edu/~goodrich/teach/cs165/notes/NPComplete.pdf (visited on 2025-08-01) (cit. on p. 86).

[135] Michael Goodwin. *What is an API?* Armonk, NY, USA: International Business Machines Corporation (IBM), Apr. 9, 2024. URL: https://www.ibm.com/topics/api (visited on 2024-12-12) (cit. on p. 78).

[136] Olaf Górski. "Why f-strings are awesome: Performance of different string concatenation methods in Python". In: *DEV Community*. Sacramento, CA, USA: DEV Community Inc., Nov. 8, 2022. URL: https://dev.to/grski/performance-of-different-string-concatenation-methods-in-python-why-f-strings-are-awesome-2e97 (visited on 2025-08-04) (cit. on p. 81).

[137] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and Alexander Hendrik George Rinnooy Kan. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey". Annals of Discrete Mathematics (5):287–326. Ed. by Peter Ladislaw Hammer, Ellis Lane Johnson, and Bernhard H. Korte, 1979. ISSN: 0167-5060. doi:10.1016/S0167-5060(08)70356-X. An early review on JSSPs and FSSPs. (Cit. on pp. 81, 83).

[138] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. "Optimal Control of Plotting and Drilling Machines: A Case Study". *Zeitschrift für Operations Research (ZOR). now* Mathematical Methods of Operations Research (MMOR) 35(1):61–84, Jan. 1991. Heidelberg, Baden-Württemberg, Germany: Physica-Verlag; now London, England, UK: Springer Nature Limited. ISSN: 0340-9422. doi:10.1007/BF01415960. This paper tackles circuit board plotting and drilling problems by modelling (parts of) them as STSPs. (Cit. on p. 90).

[139] Gregory Z. Gutin and Abraham P. Punnen, eds. *The Traveling Salesman Problem and its Variations*. Vol. 12 of Combinatorial Optimization (COOP). New York, NY, USA: Springer New York, May 2002. ISSN: 1388-3011. doi:10.1007/b101971 (cit. on p. 90).

[140] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. 3rd ed. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, July 2024. ISBN: 978-0-443-23791-1 (cit. on p. 88).

[141] Pierre Hansen, Nenad Mladenović, and José Andrés Moreno Pérez. "Variable Neighbourhood Search: Methods and Applications. Invited Survey". *4OR* 6(4):319–360, Dec. 2008. ISSN: 1619-4500. doi:10.1007/s10288-008-0089-1. URL: https://www.researchgate.net/publication/225155945 (visited on 2026-10-02) (cit. on p. 91).

[142] Jan L. Harrington. *Relational Database Design and Implementation*. 4th ed. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, Apr. 2016. ISBN: 978-0-12-849902-3 (cit. on p. 88).

[143] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (cit. on p. 84).

[144] Matthew Helmke. *Ubuntu Linux Unleashed 2021 Edition*. 14th ed. Reading, MA, USA: Addison-Wesley Professional, Aug. 2020. ISBN: 978-0-13-668539-5 (cit. on pp. 84, 90).

[145] Keld Helsgaun. "General k-opt Submoves for the Lin–Kernighan TSP Heuristic". *Mathematical Programming Computation* 1(2-3):119–163, July 2009. London, England, UK: Springer Nature Limited. ISSN: 1867-2949. doi:10.1007/s12532-009-0004-6. URL: https://www2.seas.gwu.edu/~simhaweb/champalg/tsp/papers/Helsgaun2.pdf (visited on 2026-09-26) (cit. on p. 84).

[146] Keld Helsgaun. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic". 126(1):106–130, Oct. 2000. doi:10.1016/S0377-2217(99)00284-2. URL: http://akira.ruc.dk/~keld/research/LKH/LKH-2.0/DOC/LKH_REPORT.pdf (visited on 2026-09-26) (cit. on p. 84).

[147] André Henning. "Praktische Job-Shop Scheduling-Probleme". PhD thesis. Jena, Thüringen, Germany: Friedrich-Schiller-Universität Jena, Aug. 2002. URL: http://www.db-thueringen.de/servlets/DocumentServlet?id=873 (visited on 2026-09-30). Alternate url: https://nbn-resolving.org/urn:nbn:de:gbv:27-20060809-115700-4. Applies MA-like algorithms to the JSSP. (Cit. on pp. 83, 84).

[148] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Amsterdam, The Netherlands: Elsevier B.V., 2005. ISBN: 978-1-55860-872-6. doi:10.1016/B978-1-55860-872-6.X5016-1 (cit. on pp. 85, 88).

[149] Holger H. Hoos and Thomas Stützle. *SATLIB – The Satisfiability Library*. 1.4.4. Vancouver, BC, Canada: The University of British Columbia (UBC), May 11, 2001. URL: https://www.cs.ubc.ca/~hoos/SATLIB/index-ubc.html (visited on 2026-09-20). See paper [150] (cit. on pp. 88, 105).

[150] Holger H. Hoos and Thomas Stützle. "SATLIB: An Online Resource for Research on SAT". In: *SAT2000 – Highlights of Satisfiability Research in the Year 2000*. Ed. by Ian Philip Gent, Toby Walsh, and Hans van Maaren. Vol. 63 of Frontiers in Artificial Intelligence and Applications. Amsterdam, The Netherlands: IOS Press BV, Jan. 2000, pp. 283–292. ISSN: 0922-6389. ISBN: 978-1-58603-061-2. See the website [149] (cit. on pp. 88, 105).

[151] "Why does Python 3 round half to even?" In: *Stack Overflow*. Ed. by wjandrea. New York, NY, USA: Stack Exchange Inc., May 31, 2012–June 13, 2025. URL: https://stackoverflow.com/questions/10825926 (visited on 2025-07-18). Closely related to banker's rounding (cit. on p. 79).

[152] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISSN: 1863-7310. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 87).

[153] *IEEE Standard for Floating-Point Arithmetic*. IEEE Std 754™-2019 (Revision of IEEE Std 754-2008). New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), June 13, 2019 (cit. on pp. 79, 82).

[154] *Information Technology – Database Languages – SQL – Part 1: Framework (SQL/Framework), Part 1*. International Standard ISO/IEC 9075-1:2023(E), Sixth Edition, (ANSI X3.135). Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), June 2023. URL: https://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_9075-1_2023_ed_6_-_id_76583_Publication_PDF_(en).zip (visited on 2025-01-08). Consists of several parts, see https://modern-sql.com/standard for information where to obtain them. (Cit. on p. 89).

[155] *Information Technology – Universal Coded Character Set (UCS)*. International Standard ISO/IEC 10646:2020. Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Dec. 2020 (cit. on pp. 90, 91, 120).

[156] Manuel Iori, Vinícius Loti de Lima, Silvano Martello, Flávio K. Miyazaw, and Michele Monaci. "Exact Solution Techniques for Two-Dimensional Cutting and Packing". *European Journal of Operational Research* 289(2):399–415, Mar. 2021. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/j.ejor.2020.06.050. See also arXiv:2004.12619v1 [math.OC] 27 Apr 2020 at doi:10.1016/j.ejor.2020.06.050. (Cit. on p. 78).

[157] Christoph Jabs, Dieter Vandesande, Jeremias Berg, Johannes Fichte, George Katsirelos, Ruben Martins, André Schidler, and Dominik Schreiber. *MaxSAT Evaluation: Complete Benchmark Collection*. 2026th ed. Geneva, Switzerland: zenodo (hosted by European Organization for Nuclear Research (CERN)), July 9, 2026. doi:10.5281/zenodo.21283148. All versions of this dataset can be reached directly via doi:10.5281/zenodo.21283147. See also https://github.com/maxsat-evaluations/data and https://maxsat-evaluations.github.io. (Cit. on p. 85).

[158] Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. "Monte Carlo Techniques in Code Optimization". In: *15th Annual Workshop on Microprogramming (MICRO 15)*. Oct. 5–7, 1982. Ed. by Joseph Allen Fisher, William J. Tracz, and Bill Hopkins. Palo Alto, CA, USA: Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and New York, NY, USA: Association for Computing Machinery (ACM), Oct. 1982, pp. 143–148. doi:10.5555/800036.800944. See [159] (cit. on p. 106).

[159] Dean Jacobs, Jan Prins, Peter Siegel, and Kenneth Wilson. "Monte Carlo Techniques in Code Optimization". *ACM SIGMICRO Newsletter* 13(4):143–148, Dec. 1982. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1050-916X. doi:10.1145/1014194.800944. See [158] (cit. on pp. 88, 106).

[160] Stefan Jakobs. "On Genetic Algorithms for the Packing of Polygons". *European Journal of Operational Research* 88(1):165–181, Jan. 1996. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/0377-2217(94)00166-9 (cit. on p. 82).

[161] *Java® Platform, Standard Edition & Java Development Kit Version 22 API Specification*. Redwood Shores, CA, USA: Oracle Corporation, Apr. 9, 2024. URL: https://docs.oracle.com/en/java/javase/22/docs/api (visited on 2024-07-07) (cit. on p. 79).

[162] Wassim Jaziri, ed. *Tabu Search*. London, England, UK: IntechOpen Limited, Aug. 1, 2008. ISBN: 978-3-902613-34-9. doi:10.5772/65. URL: https://www.intechopen.com/books/6114 (visited on 2026-09-30) (cit. on p. 89).

[163] David Stifler Johnson, Gregory Z. Gutin, Lyle Andrew McGeoch, Anders Yeo, Weixiong Zhang, and Alexei Zverovitch. *Experimental Analysis of Heuristics for the ATSP*. Ed. by Gregory Z. Gutin and Abraham P. Punnen. New York, NY, USA, May 2002. doi:10.1007/0-306-48213-4_10. URL: https://archive.dimacs.rutgers.edu/Challenges/TSP/papers/atspchap.pdf (visited on 2026-09-25) (cit. on p. 90).

[164] David Stifler Johnson and Lyle Andrew McGeoch. *Experimental Analysis of Heuristics for the STSP*. Ed. by Gregory Z. Gutin and Abraham P. Punnen. New York, NY, USA, May 2002. doi:10.1007/0-306-48213-4_9. URL: https://archive.dimacs.rutgers.edu/Challenges/TSP/papers/stspchap.pdf (visited on 2026-09-25) (cit. on p. 90).

[165] David Stifler Johnson, Lyle Andrew McGeoch, Fred Glover, and Cesar Rego. *8th DIMACS Implementation Challenge: The Traveling Salesman Problem*. June 1, 2000–Nov. 19, 2013. URL: https://archive.dimacs.rutgers.edu/Challenges/TSP (visited on 2026-09-25) (cit. on p. 90).

[166] Selmer Martin Johnson. "Optimal Two- and Three-Stage Production Schedules with Setup Times Included". *Naval Research Logistics Quarterly* I(1):61–68, Mar. 1954. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd. ISSN: 0894-069X. doi:10.1002/nav.3800010110. URL: https://www.rand.org/content/dam/rand/pubs/papers/2008/P402.pdf (visited on 2023-12-06) (cit. on p. 81).

[167] Stephen Curtis Johnson. *Lint, a C Program Checker*. Computing Science Technical Report 78-1273. New York, NY, USA: Bell Telephone Laboratories, Incorporated, Oct. 25, 1978. URL: https://wolfram.schneider.org/bsd/7thEdManVol2/lint/lint.pdf (visited on 2024-08-23) (cit. on p. 84).

[168] Arthur Jones, Kenneth R. Pearson, and Sidney A. Morris. "Transcendence of e and π". In: *Abstract Algebra and Famous Impossibilities*. Universitext (UTX). New York, NY, USA: Springer New York, 1991. Chap. 9, pp. 115–161. ISSN: 0172-5939. ISBN: 978-1-4419-8552-1. doi:10.1007/978-1-4419-8552-1_8 (cit. on pp. 77, 81).

[169] "exit – Terminate a Process". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Ed. by Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and San Francisco, CA, USA: The Open Group, Aug. 8, 2024. URL: https://pubs.opengroup.org/onlinepubs/9799919799/functions/exit.html (visited on 2024-10-30) (cit. on p. 81).

[170] Andrew Josey, ed. *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and San Francisco, CA, USA: The Open Group, Aug. 8, 2024. URL: https://pubs.opengroup.org/onlinepubs/9799919799 (visited on 2024-10-30).

[171] "stderr, stdin, stdout – Standard I/O Streams". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Ed. by Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and San Francisco, CA, USA: The Open Group, Aug. 8, 2024. URL: https://pubs.opengroup.org/onlinepubs/9799919799/functions/stdin.html (visited on 2024-10-30) (cit. on p. 89).

[172] Shen Kangshen, John Newsome Crossley, and Anthony W.-C. Lun. *The Nine Chapters on the Mathematical Art: Companion and Commentary*. Oxford, Oxfordshire, England, UK: Oxford University Press, Oct. 7, 1999. ISBN: 978-0-19-853936-0. doi:10.1093/oso/9780198539360.001.0001. Translation of the historical Chinese mathematics book *Jiu Zhang Suanshu* (九章算术), including the commentaries of LIU Hui (刘徽). (Cit. on p. 77).

[173] Ansel N. Kellogg. "Empirical formulæ for Approximate Computation". *The American Mathematical Monthly* 4(2):39–49, Feb. 1987. London, England, UK: Taylor and Francis Ltd. ISSN: 1930-0972. doi:10.2307/2970040 (cit. on p. 56).

[174] Scott Kirkpatrick, C. Daniel Gelatt, Jr., and Mario P. Vecchi. "Optimization by Simulated Annealing". *Science Magazine* 220(4598):671–680, May 13, 1983. Washington, D.C., USA: American Association for the Advancement of Science (AAAS). ISSN: 0036-8075. doi:10.1126/science.220.4598.671. URL: https://www.researchgate.net/publication/6026283 (visited on 2025-08-08) (cit. on p. 88).

[175] Robert Klein. *Scheduling of Resource-Constrained Projects*. Vol. 10 of Operations Research/Computer Science Interfaces Series (ORCS). New York, NY, USA: Springer New York, 2000. ISSN: 1387-666X. ISBN: 978-0-7923-8637-7. doi:10.1007/978-1-4615-4629-0 (cit. on p. 81).

[176] Donald Ervin Knuth. "Big Omicron and Big Omega and Big Theta". *ACM SIGACT News* 8(2):18–24, Apr.–June 1976. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0163-5700. doi:10.1145/1008328.1008329 (cit. on p. 86).

[177] Donald Ervin Knuth. *Fundamental Algorithms*. 3rd ed. Vol. 1 of The Art of Computer Programming. Reading, MA, USA: Addison-Wesley Professional, 1997. ISBN: 978-0-201-89683-1 (cit. on p. 86).

[178] Donald Ervin Knuth. *Sorting and Searching*. Vol. 3 of The Art of Computer Programming. Reading, MA, USA: Addison-Wesley Professional, 1998. ISBN: 978-0-201-89685-5 (cit. on p. 86).

[179] Tjalling Charles Koopmans and Martin Beckmann. "Assignment Problems and the Location of Economic Activities". *Econometrica* 25(1):53–76, 1957. New Haven, CT, USA: The Econometric Society and Chichester, West Sussex, England, UK: John Wiley and Sons Ltd. ISSN: 0012-9682. doi:10.2307/1907742 (cit. on p. 87).

[180] Jakob Krarup and Peter Mark Pruzan. "Computer-aided Layout Design". In: *Mathematical Programming in Use*. Ed. by Michel Louis Balinski and Claude Lemaréchal. Vol. 9 of. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 1978. Chap. 6, pp. 75–94. ISSN: 0303-3929. ISBN: 978-3-642-00796-5. doi:10.1007/BFb0120827. Uses QAPs as model for designing architectural layouts. (Cit. on p. 87).

[181] Holger Krekel and pytest-Dev Team. "How to Run Doctests". In: *pytest Documentation*. Release 8.4. Freiburg, Baden-Württemberg, Germany: merlinux GmbH. Chap. 2.8, pp. 65–69. URL: https://docs.pytest.org/en/stable/how-to/doctest.html (visited on 2024-11-07) (cit. on p. 80).

[182] Holger Krekel and pytest-Dev Team. *pytest Documentation*. Release 8.4. Freiburg, Baden-Württemberg, Germany: merlinux GmbH. URL: https://readthedocs.org/projects/pytest/downloads/pdf/latest (visited on 2024-11-07) (cit. on p. 87).

[183] Mark W. Krentel. "The Complexity of Optimization Problems". *Journal of Computer and System Sciences* 36(3):490–509, June 1988. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0022-0000. Explores the hardness (optimization) problems such TSP, Clique, and Coloring, where computing the optimal value is $\mathcal{NP}$-complete. (Cit. on p. 90).

[184] Jay LaCroix. *Mastering Ubuntu Server*. 4th ed. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2022. ISBN: 978-1-80323-424-3 (cit. on p. 88).

[185] Jeffrey C. Lagarias, ed. *The Ultimate Challenge: The* $3x + 1$ *Problem*. Providence, RI, USA: American Mathematical Society (AMS), 2010. ISBN: 978-1-4704-7289-4 (cit. on p. 45).

[186] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*. Leipzig, Sachsen, Germany: B. G. Teubner, 1909. ISBN: 978-0-8218-2650-8 (cit. on p. 86).

[187] Łukasz Langa. *Literature Overview for Type Hints*. Python Enhancement Proposal (PEP) 482. Beaverton, OR, USA: Python Software Foundation (PSF), Jan. 8, 2015. URL: https://peps.python.org/pep-0482 (visited on 2024-10-09) (cit. on p. 90).

[188] Eugene Leighton Lawler, Jan Karel Lenstra, and Alexander Hendrik George Rinnooy Kan. "Recent Developments in Deterministic Sequencing and Scheduling: A Survey". In: *Deterministic and Stochastic Scheduling: Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems*. June 6–17, 1981, Durham, England, UK. Ed. by Michael Anthony Howe Dempster, Jan Karel Lenstra, and Alexander Hendrik George Rinnooy Kan. Vol. 84 of Nato Science Series C: Mathematical and Physical Science (ASIC). London, England, UK: Springer Nature Limited, Apr. 1982, pp. 35–73. ISBN: 978-90-277-1397-1. doi:10.1007/978-94-009-7801-0_3. The JSSP is $\mathcal{NP}$-hard. (Cit. on p. 83).

[189] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. "Sequencing and Scheduling: Algorithms and Complexity". In: *Production Planning and Inventory*. Ed. by Stephen C. Graves, Alexander Hendrik George Rinnooy Kan, and Paul H. Zipkin. Vol. IV of Handbooks of Operations Research and Management Science. Amsterdam, The Netherlands: Elsevier B.V., 1993. Chap. 9, pp. 445–522. ISSN: 0927-0507. ISBN: 978-0-444-87472-6. doi:10.1016/S0927-0507(05)80189-6. URL: http://alexandria.tue.nl/repository/books/339776.pdf (visited on 2023-12-06). Discusses, amongst others, that the JSSP is $\mathcal{NP}$-hard. (Cit. on pp. 83, 86).

[190] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, England, UK: Wiley Interscience, Sept. 1985. ISSN: 0277-2698. ISBN: 978-0-471-90413-7 (cit. on p. 90).

[191] Stephen R. Lawrence. "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques". PhD thesis. Pittsburgh, PA, USA: Graduate School of Industrial Administration (GSIA), Carnegie-Mellon University, 1984 (cit. on p. 83).

[192] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISSN: 1863-7310. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 87).

[193] Jukka Lehtosalo, Ivan Levkivskyi, Jared Hance, Ethan Smith, Guido van Rossum, Jelle "JelleZijlstra" Zijlstra, Michael J. Sullivan, Shantanu Jain, Xuanda Yang, Jingchen Ye, Nikita Sobolev, and Mypy Contributors. *Mypy – Static Typing for Python*. San Francisco, CA, USA: GitHub Inc, 2024. URL: https://github.com/python/mypy (visited on 2024-08-17) (cit. on p. 85).

[194] Xueping Li and Kaike Zhang. "Single Batch Processing Machine Scheduling with Two-Dimensional Bin Packing Constraints". *International Journal of Production Economics (IJPE)* 196:113–121, Feb. 2018. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0925-5273. doi:10.1016/j.ijpe.2017.11.015. Mentions some scheduling tasks of Additive Manufacturing jobs can be viewed as instance of the 2BP|R|F. (Cit. on p. 78).

[195] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, Sarah Louise Thomson, and Thomas Weise (汤卫思). "Addressing the Traveling Salesperson Problem with Frequency Fitness Assignment and Hybrid Algorithms". *Soft Computing* 28(17-18):9495–9508, July 2024. London, England, UK: Springer Nature Limited. ISSN: 1432-7643. doi:10.1007/S00500-024-09718-8 (cit. on pp. 81, 85, 90).

[196] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, and Thomas Weise (汤卫思). "Solving the Traveling Salesperson Problem using Frequency Fitness Assignment". In: *IEEE Symposium Series on Computational Intelligence (SSCI'2022)*. Dec. 4–7, 2022, Singapore. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022. ISBN: 978-1-6654-8769-6. doi:10.1109/SSCI51031.2022.10022296 (cit. on pp. 81, 85).

[197] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Matthias Thürer, Markus Wagner, and Thomas Weise (汤卫思). "Generating Small Instances with Interesting Features for the Traveling Salesperson Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 173–180. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888800003837 (cit. on p. 85).

[198] Shen Lin and Brian W. Kernighan. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". *Operations Research* 21(2):498–516, Mar.–Apr. 1973. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0030-364X. doi:10.1287/opre.21.2.498. URL: https://www.cs.princeton.edu/~bwk/btl.mirror/tsp.pdf (visited on 2026-09-26) (cit. on p. 84).

[199] John D. C. Little, Katta G. Murty, Dura W. Sweeny, and Caroline Karel. *An Algorithm for the Traveling Salesman Problem*. Sloan Working Papers 07-63. Massachusetts Institute of Technology (MIT): Massachusetts Institute of Technology (MIT), Sloan School of Management, Mar. 1963. URL: http://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf (visited on 2026-01-03) (cit. on p. 79).

[200] Dequan Liu (刘德全) and Hongfei Teng (滕弘飞). "An Improved BL-Algorithm for EAGenetic Algorithm of the Orthogonal Packing of Rectangles". *European Journal of Operational Research* 112(2):413–420, Jan. 1999. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/S0377-2217(97)00437-2 (cit. on p. 82).

[201] Andrea Lodi, Silvano Martello, and Michele Monaci. "Two-Dimensional Packing Problems: A Survey". *European Journal of Operational Research* 141(2):241–252, Sept. 2002. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/S0377-2217(02)00123-6 (cit. on pp. 78, 79).

[202] Andrea Lodi, Silvano Martello, and Daniele Vigo. "Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems". *INFORMS Journal on Computing* 11(4):345–357, Aut. 1999. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 1091-9856. doi:10.1287/ijoc.11.4.345 (cit. on p. 78).

[203] Andrea Lodi, Silvano Martello, and Daniele Vigo. "Recent Advances on Two-Dimensional Bin Packing Problems". *Discrete Applied Mathematics* 123(1–3):379–396, Nov. 2002. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0166-218X. doi:10.1016/S0166-218X(01)00347-X (cit. on pp. 78–80).

[204] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter M. Hahn, and Tania Querido. "A Survey for the Quadratic Assignment Problem". *European Journal of Operational Research* 176(2):657–690, 2007. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/j.ejor.2005.09.032 (cit. on p. 87).

[205] Mariana A. Londe, Luciana S. Pessoa, Carlos E. Andrade, and Mauricio G. C. Resende. *Biased Random-Key Genetic Algorithms: A Review*. arXiv.org: Computing Research Repository (CoRR) abs/2312.00961v2. Ithaca, NY, USA: Cornell Universiy Library, Dec. 6, 2023. doi:10.48550/arXiv.2312.00961. URL: https://arxiv.org/abs/2312.00961v2 (visited on 2024-06-27). arXiv:2312.00961v2 [cs.NE] 06 Dec 2023; See [206] (cit. on p. 109).

[206] Mariana A. Londe, Luciana S. Pessoa, Carlos E. Andrade, and Mauricio G. C. Resende. "Biased Random-Key Genetic Algorithms: A Review". *European Journal of Operational Research* 321(1):1–22, Feb. 2025. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/j.ejor.2024.03.030. See [205] (cit. on pp. 80, 109).

[207] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra, and William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Ed. by The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., Jan. 3, 2025. URL: https://www.britannica.com/technology/client-server-architecture (visited on 2025-01-20) (cit. on p. 80).

[208] Marc Loy, Patrick Niemeyer, and Daniel Leuck. *Learning Java*. 5th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2020. ISBN: 978-1-4920-5627-0 (cit. on p. 83).

[209] Peter Luschny. *A New Kind of Factorial Function*. Highland Park, NJ, USA: The OEIS Foundation Inc., Oct. 4, 2015. URL: https://oeis.org/A000142/a000142.pdf (visited on 2024-09-29) (cit. on p. 77).

[210] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 87).

[211] Rita Macedo, Cláudio Alves, and J. M. Valério de Carvalho. "Arc-Flow Model for the Two-Dimensional Guillotine Cutting Stock Problem". *Computers & Operations Research* 37(6):991–1001, June 2010. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0305-0548. doi:10.1016/j.cor.2009.08.005 (cit. on p. 80).

[212] Reinhard Männer and Bernard Manderick, eds. *Parallel Problem Solving from Nature 2 (PPSN-II)*. Sept. 28–30, 1992, Brussels, Belgium. Amsterdam, The Netherlands: Elsevier B.V., 1992.

[213] Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe, eds. *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0000195000003837.

[214] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (visited on 2025-04-24) (cit. on p. 85).

[215] Charlie Marsh. "Ruff". In: URL: https://pypi.org/project/ruff (visited on 2025-08-29) (cit. on p. 88).

[216] Charlie Marsh. *ruff: An Extremely Fast Python Linter and Code Formatter, Written in Rust*. New York, NY, USA: Astral Software Inc., Aug. 28, 2022. URL: https://docs.astral.sh/ruff (visited on 2024-08-23) (cit. on p. 88).

[217] Silvano Martello and Daniele Vigo. "Exact Solution of the Two-Dimensional Finite Bin Packing Problem". 44(3):388–399, Mar. 1998. doi:10.1287/mnsc.44.3.388 (cit. on pp. 78, 79).

[218] "Mathematical Functions and Operators". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), Feb. 20, 2025. Chap. 9.3. URL: https://www.postgresql.org/docs/17/functions-math.html (visited on 2025-02-27) (cit. on pp. 77, 81).

[219] Clive Maxfield and Alvin Brown. "Rounding Algorithms 101". In: *DIY Calculator*. Huntsville, AL, USA: DIY Calculator, 2005. URL: https://www.clivemaxfield.com/diycalculator/popup-m-round.shtml (visited on 2025-07-18). Also discusses banker's rounding (cit. on p. 79).

[220] Aaron Maxwell. *What are f-strings in Python and how can I use them?* Oakville, ON, Canada: Infinite Skills Inc, June 2017. ISBN: 978-1-4919-9486-3 (cit. on p. 81).

[221] Jim Melton and Alan R. Simon. *SQL: 1999 – Understanding Relational Language Components*. The Morgan Kaufmann Series in Data Management Systems. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, June 2001. ISSN: 1046-1698. ISBN: 978-1-55860-456-8 (cit. on p. 89).

[222] Stephan Mertens. "Exhaustive Search for Low-Autocorrelation Binary Sequences". *Journal of Physics A: Mathematical and General* 29(18):L473, Sept. 1996. Bristol, England, UK: IOP Publishing Ltd. ISSN: 0305-4470. doi:10.1088/0305-4470/29/18/005 (cit. on p. 84).

[223] Peter Merz and Bernd Freisleben. "A Comparison of MAMemetic Algorithms, TSTabu Search, and ACOAnt Colonies for the QAPQuadratic Assignment Problem". In: *Congress on Evolutionary Computation (CEC'1999)*. July 6–9, 1999, Washington, D.C., USA. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 1999, pp. 2063–2070. doi:10.1109/CEC.1999.785529 (cit. on p. 87).

[224] Zbigniew "Zbyszek" Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 1996. ISBN: 978-3-540-58090-4. doi:10.1007/978-3-662-03315-9 (cit. on pp. 81, 88).

[225] Burkhard Militzer, Michele Zamparelli, and Dieter Beule. "Evolutionary Search for Low Autocorrelated Binary Sequences". *IEEE Transactions on Evolutionary Computation* 2(1):34–39, Apr. 1998. Los Alamitos, CA, USA: IEEE Computer Society. ISSN: 1089-778X. doi:10.1109/4235.728212 (cit. on p. 84).

[226] Alfonsas Misevičius. "A Tabu Search Algorithm for the Quadratic Assignment Problem". *Computational Optimization and Applications* 30:95–111, Jan. 2005. London, England, UK: Springer Nature Limited. ISSN: 0926-6003. doi:10.1007/s10589-005-4562-x (cit. on p. 89).

[227] Alfonsas Misevičius. "A Modified Simulated Annealing Algorithm for the Quadratic Assignment Problem". *Informatica* 14(4):497–514, Jan. 2003. Vilnius, Lithuania: Vilnius University, Institute of Data Science and Digital Technologies (DMSTI), Lithuanian Academy of Sciences. ISSN: 0868-4952. doi:10.15388/Informatica.2003.037 (cit. on p. 88).

[228] Alfonsas Misevičius. *An Implementation of the Iterated Tabu Search Algorithm for the Quadratic Assignment Problem*. Working Paper. Kaunas, Lithuania: Kaunas University of Technology, 2008. See also [229] (cit. on p. 111).

[229] Alfonsas Misevičius. "An Implementation of the Iterated Tabu Search Algorithm for the Quadratic Assignment Problem". *OR Spectrum: Quantitative Approaches in Management. Formerly* Operations-Research-Spektrum 34(3):665–690, July 2012. London, England, UK: Springer Nature Limited. ISSN: 0171-6468. doi:10.1007/s00291-011-0274-z. See also [228] (cit. on pp. 87, 89, 111).

[230] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. Cambridge, MA, USA: MIT Press, Feb. 1998. ISBN: 978-0-262-13316-6. URL: http://boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf (visited on 2025-08-08) (cit. on pp. 81, 88).

[231] Nenad Mladenović and Pierre Hansen. "Variable Neighborhood Search". *Computers & Operations Research* 24(11):1097–1100, Nov. 1997. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0305-0548. doi:10.1016/S0305-0548(97)00031-2 (cit. on p. 91).

[232] Wai Ho Mow, Ke-Lin Du, and Wei Hsiang Wu. "New Evolutionary Search for Long Low Autocorrelation Binary Sequences". *IEEE Transactions on Aerospace and Electronic Systems* 51(1):290–303, Jan. 2015. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0018-9251. doi:10.1109/TAES.2014.130518 (cit. on p. 84).

[233] Heinz Mühlenbein. "How Genetic Algorithms Really Work: Mutation and Hillclimbing". In: *Parallel Problem Solving from Nature 2 (PPSN-II)*. Sept. 28–30, 1992, Brussels, Belgium. Ed. by Reinhard Männer and Bernard Manderick. Amsterdam, The Netherlands: Elsevier B.V., 1992, pp. 15–26. URL: https://www.researchgate.net/publication/284805798 (visited on 2025-08-15) (cit. on pp. 86, 87).

[234] Cameron Newham and Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3rd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (cit. on p. 79).

[235] Siegfried Nijssen and Thomas Bäck. "An Analysis of the Behavior of Simplified Evolutionary Algorithms on Trap Functions". *IEEE Transactions on Evolutionary Computation* 7(1):11–22, Feb. 2003. Los Alamitos, CA, USA: IEEE Computer Society. ISSN: 1089-778X. doi:10.1109/TEVC.2002.806169 (cit. on p. 89).

[236] Ivan Niven. "The Transcendence of π". *The American Mathematical Monthly* 46(8):469–471, Oct. 1939. London, England, UK: Taylor and Francis Ltd. ISSN: 1930-0972. doi:10.2307/2302515 (cit. on p. 77).

[237] Eugeniusz Nowicki and Czesław Smutnicki. "A Fast Taboo Search Algorithm for the Job Shop Problem". *Management Science* 42(6):783–938, June 1996. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0025-1909. doi:10.1287/mnsc.42.6.797 (cit. on p. 89).

[238] John J. O'Connor and Edmund F. Robertson. *Liu Hui*. St Andrews, Scotland, UK: University of St Andrews, School of Mathematics and Statistics, Dec. 2003. URL: https://mathshistory.st-andrews.ac.uk/Biographies/Liu_Hui (visited on 2024-08-10). Discusses LIU Hui's (刘徽) method of approximating π (cit. on p. 77).

[239] Regina O. Obe and Leo S. Hsu. *PostgreSQL: Up and Running*. 3rd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Oct. 2017. ISBN: 978-1-4919-6336-4 (cit. on p. 87).

[240] A. Jefferson Offutt. "Unit Testing Versus Integration Testing". In: *Test: Faster, Better, Sooner – IEEE International Test Conference (ITC'1991)*. Oct. 26–30, 1991, Nashville, TN, USA. Los Alamitos, CA, USA: IEEE Computer Society, 1991. Chap. Paper P2.3, pp. 1108–1109. ISSN: 1089-3539. ISBN: 978-0-8186-9156-0. doi:10.1109/TEST.1991.519784 (cit. on p. 91).

[241] Brian Okken. *Python Testing with pytest*. Flower Mound, TX, USA: Pragmatic Bookshelf by The Pragmatic Programmers, L.L.C., Feb. 2022. ISBN: 978-1-68050-860-4 (cit. on p. 87).

[242] Michael Olan. "Unit Testing: Test Early, Test Often". *Journal of Computing Sciences in Colleges (JCSC)* 19(2):319–328, Dec. 2003. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1937-4771. doi:10.5555/948785.948830. URL: https://www.researchgate.net/publication/255673967 (visited on 2025-09-05) (cit. on p. 91).

[243] Robert Orfali, Dan Harkey, and Jeri Edwards. *Client/Server Survival Guide*. 3rd ed. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., Jan. 25, 1999. ISBN: 978-0-471-31615-2 (cit. on p. 80).

[244] Tom Packebusch and Stephan Mertens. "Low Autocorrelation Binary Sequences". *Journal of Physics A: Mathematical and Theoretical* 49(16):165001, Mar. 2016. Bristol, England, UK: IOP Publishing Ltd. ISSN: 1751-8121. doi:10.1088/1751-8113/49/16/165001 (cit. on p. 84).

[245] Ashwin Pajankar. *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. New York, NY, USA: Apress Media, LLC, Dec. 2021. ISBN: 978-1-4842-7854-3 (cit. on pp. 87, 91).

[246] Panos Miltiades Pardalos, Ding-Zhu Du, and Ronald Lewis Graham, eds. *Handbook of Combinatorial Optimization*. 1st ed. Boston, MA, USA: Springer, 1998. ISBN: 978-1-4613-7987-4.

[247] Panos Miltiades Pardalos, Oleg V. Shylo, and Alkis Vazacopoulos. "Solving Job Shop Scheduling Problems Utilizing the Properties of Backbone and "Big Valley"". *Computational Optimization and Applications* 47(1):61–76, Sept. 2010. London, England, UK: Springer Nature Limited. ISSN: 0926-6003. doi:10.1007/s10589-008-9206-5. An algorithm similar to SA. (Cit. on pp. 83, 88).

[248] Francisco Parreño, Ramón Alvarez-Valdés, José Fernando Oliveira, and José Manuel Tamarit. "A Hybrid GRASP/VND Algorithm for Two- and Three-Dimensional Bin Packing". *Annals of Operations Research* 179(1):203–220, Sept. 2010. London, England, UK: Springer Nature Limited. ISSN: 0254-5330. doi:10.1007/s10479-008-0449-4 (cit. on pp. 78, 82, 91).

[249] Julio Pellicer, M. Amparo Gilabert, and Ernesto Lopez-Baez. "The Evolution of the Celsius and Kelvin Temperature Scales and the State of the Art". *Journal of Chemical Education* 76(7):911–913, July 1999. Washington, D.C., USA: ACS Publications. ISSN: 0021-9584. doi:10.1021/ed076p911. URL: https://www.uv.es/~juliop/docencia/JCE99.pdf (visited on 2026-10-08) (cit. on pp. 42, 115).

[250] Bo Peng, Zhipeng Lü, and T. C. Edwin Cheng. "A Tabu Search/Path Relinking Algorithm to Solve the Job Shop Scheduling Problem". *Computers & Operations Research* 53:154–164, Jan. 2015. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0305-0548. doi:10.1016/j.cor.2014.08.006. URL: http://arxiv.org/abs/1402.5613 (visited on 2026-09-30) (cit. on pp. 83, 89).

[251] Yun Peng, Yubing Wang, Yingxu Zhang, Lei Liang, Jun Zhang, Jianwei Zhang, Qin Li, and Lijun Wang. "Hardware Implementation of Improved Banker's Fixed-Point Rounding Algorithm". *IEEE Access* 13:36679–36686, Feb. 19, 2025. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 2169-3536. doi:10.1109/ACCESS.2025.3543540 (cit. on p. 79).

[252] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin, and Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), July 14, 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (cit. on p. 82).

[253] Ferdinando Pezzella and Emanuela Merelli. "A Tabu Search Method Guided by Shifting Bottleneck for the Job Shop Scheduling Problem". *European Journal of Operational Research* 120(2):297–310, Jan. 2000. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/S0377-2217(99)00158-7. URL: https://www2.cs.sfu.ca/CourseCentral/827/havens/papers/topic%5C%2310(JobShop)/Tabu%5C%20With%5C%20Shifting.pdf (visited on 2026-09-30) (cit. on p. 89).

[254] Amit Phalgun, Cory Kissinger, Margaret M. Burnett, Curtis R. Cook, Laura Beckwith, and Joseph R. Ruthruff. "Garbage in, Garbage out? An Empirical Look at Oracle Mistakes by End-User Programmers". In: *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'2005)*. Sept. 21–24, 2005, Dallas, TX, USA. Ed. by Martin Erwig and Andy Schürr. Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 45–52. ISSN: 1943-6092. ISBN: 978-0-7695-2443-6. doi:10.1109/VLHCC.2005.40 (cit. on p. 82).

[255] Martin Pincus. "Letter to the Editor – A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems". *Operations Research* 18(6):1225–1228, Nov.–Dec. 1970. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0030-364X. doi:10.1287/opre.18.6.1225 (cit. on p. 88).

[256] Marcelo Pinto, Cristóvão Silva, Matthias Thürer, and Samuel Moniz. "Survey in Operations Research and Management Science. Nesting and Scheduling Optimization of Additive Manufacturing Systems: Mapping the Territory". *Computers & Operations Research* 165(106592), Feb. 2024. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0305-0548. doi:10.1016/j.cor.2024.106592. Mentions some scheduling tasks of Additive Manufacturing jobs can be viewed as instance of the 2BP|R|F. (Cit. on p. 78).

[257] *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), Feb. 2025. URL: https://www.postgresql.org/docs/17/index.html (visited on 2025-02-25).

[258] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2024 (cit. on p. 87).

[259] S. D. Prestwich. *Improved Branch-and-Bound for Low Autocorrelation Binary Sequences*. arXiv.org: Computing Research Repository (CoRR) abs/1305.6187. Ithaca, NY, USA: Cornell Universiy Library, Apr. 3, 2024. doi:10.48550/arXiv.1305.6187. URL: https://arxiv.org/abs/1305.6187 (visited on 2013-07-23). arXiv:1305.6187v2 [cs.AI] 23 Jul 2013 (cit. on p. 84).

[260] *Programming Languages – C, Working Document of SC22/WG14*. International Standard ISO/3IEC9899:2017 C17 Ballot N2176. Geneva, Switzerland: International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Nov. 2017. URL: https://files.lhmouse.com/standards/ISO%20C%20N2176.pdf (visited on 2024-06-29) (cit. on p. 80).

[261] Pylint Contributors. *Pylint*. Toulouse, Occitanie, France: Logilab, 2003–2024. URL: https://pylint.readthedocs.io/en/stable (visited on 2024-09-24) (cit. on p. 87).

[262] Sanatan Rai and George Vairaktarakis. "NP-Complete Problems and Proof Methodology". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos Miltiades Pardalos. 2nd ed. Boston, MA, USA: Springer, Sept. 2008, pp. 2675–2682. ISBN: 978-0-387-74758-3. doi:10.1007/978-0-387-74759-0_462 (cit. on p. 86).

[263] Soraya B. Rana and L. Darrell Whitley. "Genetic Algorithm Behavior in the MAXSAT Domain". In: *5th International Conference on Parallel Problem Solving from Nature (PPSN V)*. Sept. 27–30, 1998, Amsterdam, The Netherlands. Ed. by Ágoston Endre Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel. Lecture Notes in Computer Science (LNCS). 1998, pp. 785–794. ISSN: 0302-9743. ISBN: 978-3-540-65078-2. doi:10.1007/BFb0056920 (cit. on p. 85).

[264] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu, and Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6 (cit. on p. 80).

[265] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2014. ISBN: 978-1-78398-154-0 (cit. on p. 85).

[266] Mike Reichardt, Michael Gundall, and Hans D. Schotten. "Benchmarking the Operation Times of NoSQL and MySQL Databases for Python Clients". In: *47th Annual Conference of the IEEE Industrial Electronics Society (IECON'2021*. Oct. 13–15, 2021, Toronto, ON, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2021, pp. 1–8. ISSN: 2577-1647. ISBN: 978-1-6654-3554-3. doi:10.1109/IECON48115.2021.9589382 (cit. on p. 85).

[267] Gerhard Reinelt. "TSPLIB – A Traveling Salesman Problem Library". *ORSA Journal on Computing* 3(4):376–384, Aut. 1991. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0899-1499. doi:10.1287/ijoc.3.4.376 (cit. on p. 90).

[268] Gerhard Reinelt. *TSPLIB95*. Tech. rep. Heidelberg, Baden-Württemberg, Germany: Universität Heidelberg, Institut für Angewandte Mathematik, 1995–Apr. 28, 2016. URL: https://github.com/shredderzwj/TSPLIB (visited on 2026-09-25). The original URL of the repository has become defunct in 2026. Therefore, we provide the repository listed as main URL which contains the data. Additionally, you may also check https://github.com/thomasWeise/tspSuite of our own project *TSPSuite*, which contains all the TSPLib instances as well as Java code to load them. (Cit. on p. 90).

[269] Mark Richards and Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: 978-1-4920-4345-4 (cit. on p. 80).

[270] Ernest E. Rothman, Rich Rosen, and Brian Jepson. *Mac OS X for Unix Geeks*. 4th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Sept. 2008. ISBN: 978-0-596-52062-5 (cit. on p. 85).

[271] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Vol. 35 of Forschungsergebnisse zur Informatik. Hamburg, Germany: Verlag Dr. Kovač, 1997. ISSN: 1435-6260. ISBN: 978-3-86064-554-3 (cit. on p. 84).

[272] Per Runeson. "A Survey of Unit Testing Practices". *IEEE Software* 23(4):22–29, July–Aug. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0740-7459. doi:10.1109/MS.2006.91 (cit. on p. 91).

[273] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (AIMA)*. 4th ed. Hoboken, NJ, USA: Pearson Education, Inc. ISBN: 978-1-292-40113-3. URL: https://aima.cs.berkeley.edu (visited on 2024-06-27) (cit. on p. 78).

[274] Yeonhee Ryou, Sangwoo Joh, Joonmo Yang, Sujin Kim, and Youil Kim. "Code Understanding Linter to Detect Variable Misuse". In: *37th IEEE/ACM International Conference on Automated Software Engineering (ASE'2022)*. Oct. 10–14, 2022, Rochester, MI, USA. New York, NY, USA: Association for Computing Machinery (ACM), 2022, 133:1–133:5. ISBN: 978-1-4503-9475-8. doi:10.1145/3551349.3559497 (cit. on p. 84).

[275] Ahmad Sahar. *iOS 26 Programming for Beginners*. 10th ed. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2025. ISBN: 978-1-80602-393-6 (cit. on p. 92).

[276] Sartaj Sahni and Teofilo Francisco Gonzalez Arce. "P-complete Approximation Problems". *Journal of the ACM (JACM)* 23(3):555–565, July 1976. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0004-5411. doi:10.1145/321958.321975. URL: https://sites.cs.ucsb.edu/~teo/papers/JACM-PCom.pdf (visited on 2026-02-21) (cit. on p. 87).

[277] Guntram Scheithauer. *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*. Vol. 263 of International Series in Operations Research & Management Science (ISOR). Cham, Switzerland: Springer, 2018. ISSN: 0884-8289. ISBN: 978-3-319-64402-8. doi:10.1007/978-3-319-64403-5 (cit. on p. 79).

[278] Syamal K. Sen and Ravi P. Agarwal. "Existence of year zero in astronomical counting is advantageous and preserves compatibility with significance of AD, BC, CE, and BCE". In: *Zero – A Landmark Discovery, the Dreadful Void, and the Ultimate Mind*. Amsterdam, The Netherlands: Elsevier B.V., 2016. Chap. 5.5, pp. 94–95. ISBN: 978-0-08-100774-7. doi:10.1016/C2015-0-02299-7 (cit. on pp. 79, 80).

[279] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, England, UK: Cambridge University Press (CUP), July 2014. ISBN: 978-1-107-05713-5. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning (visited on 2024-06-27) (cit. on p. 85).

[280] Guoyong Shi, Hitoshi Iima, and Nobuo Sannomiya. "New Encoding Scheme for Solving Job Shop Problems by Genetic Algorithm". In: *35th IEEE Conference on Decision and Control (CDC'96)*. Vol. 4. Dec. 11–13, 1996. Kōbe, Hyōgo-ken, Japan: Institute of Electrical and Electronics Engineers (IEEE), 1997, pp. 4395–4400. ISBN: 978-0-7803-3590-5. doi:10.1109/CDC.1996.577484. URL: https://www.researchgate.net/publication/224238934 (visited on 2026-09-30). Proposes a permutations-with-repetitions based encoding for the JSSP. (Cit. on pp. 81, 83, 104).

[281] Oleg V. Shylo. *Job Shop Scheduling (Personal Homepage)*. Knoxville, TN, USA: Oleg V. Shylo, Aug. 2019. URL: http://optimizizer.com/jobshop.php (visited on 2026-09-30). Contains many JSSP benchmark instances. (Cit. on pp. 83, 115).

[282] Oleg V. Shylo and Hesam Shams. *Boosting Binary Optimization via Binary Classification: A Case Study of Job Shop Scheduling*. arXiv.org: Computing Research Repository (CoRR) abs/1808.10813. Proposes a TS using regression to adapt the tabu tenures for each variable of the solution vector of a JSSP separately. Many results are available in the GitHub repository https://github.com/quasiquasar/gta-jobshop-data; other results are published Shylo's website http://optimizizer.com/jobshop.php. Ithaca, NY, USA: Cornell Universiy Library, Aug. 31, 2018. doi:10.48550/arXiv.1808.10813. URL: https://arxiv.org/abs/1808.10813 (visited on 2026-09-30) (cit. on pp. 83, 89).

[283] *SI Units – Temperature*. Gaithersburg, MD, USA: U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Apr. 12, 2010–June 17, 2026. URL: https://www.nist.gov/pml/owm/si-units-temperature (visited on 2026-10-08). See [119, 249] for discussions of the temperature scales. (Cit. on p. 42).

[284] Ellen Siever, Stephen Figgins, Robert Love, and Arnold Robbins. *Linux in a Nutshell*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Sept. 2009. ISBN: 978-0-596-15448-6 (cit. on p. 84).

[285] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2023. ISBN: 978-1-0981-3391-7 (cit. on p. 82).

[286] Neil James Alexander Sloane. *Decimal Expansion of Golden Ratio phi (or tau) = (1 + sqrt(5))/2*. Ed. by John Horton Conway. Vol. A001622 of The On-Line Encyclopedia of Integer Sequences. Highland Park, NJ, USA: The OEIS Foundation Inc., Dec. 13, 2024. URL: https://oeis.org/A001622 (visited on 2024-12-14) (cit. on p. 77).

[287] Drew Smith. *Modern Apple Platform Administration – macOS and iOS Essentials (2025)*. Birmingham, England, UK: Packt Publishing Ltd, Feb. 2025. ISBN: 978-1-80580-309-6 (cit. on pp. 83, 85).

[288] Eric V. "ericvsmith" Smith. *Literal String Interpolation*. Python Enhancement Proposal (PEP) 498. Beaverton, OR, USA: Python Software Foundation (PSF), Nov. 6, 2016–Sept. 9, 2023. URL: https://peps.python.org/pep-0498 (visited on 2024-07-25) (cit. on p. 81).

[289] John Miles Smith and Philip Yen-Tang Chang. "Optimizing the Performance of a Relational Algebra Database Interface". *Communications of the ACM (CACM)* 18(10):568–579, Oct. 1975. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/361020.361025 (cit. on p. 88).

[290] Hossein M. Soroush. "Scheduling in Stochastic Bicriteria Single Machine Systems with Set-up Times". *International Journal of Planning and Scheduling* 1(1–2):109–145, Jan. 2011. Geneva, Switzerland: Inderscience Publishers. ISSN: 2044-494X. doi:10.1504/IJPS.2011.044605. A scheduling problem is phrased as QAP. (Cit. on p. 87).

[291] "SQL Commands". In: *PostgreSQL Documentation*. 17.4. The PostgreSQL Global Development Group (PGDG), Feb. 20, 2025. Chap. Part VI. Reference. URL: https://www.postgresql.org/docs/17/sql-commands.html (visited on 2025-02-25) (cit. on p. 89).

[292] Leon Steinberg. "The Backboard Wiring Problem: A Placement Algorithm". *SIAM Review* 3(1):37–50, Jan. 1961. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics Publications (SIAM). ISSN: 0036-1445. doi:10.1137/1003003. The problem is modeled as a QAP (cit. on p. 87).

[293] Ryan K. Stephens and Ronald R. Plew. *Sams Teach Yourself SQL in 21 Days*. 4th ed. Sams Teach Yourself. Indianapolis, IN, USA: SAMS Technical Publishing and Hoboken, NJ, USA: Pearson Education, Inc., Oct. 2002. ISBN: 978-0-672-32451-2 (cit. on pp. 89, 116).

[294] Ryan K. Stephens, Ronald R. Plew, Bryan Morgan, and Jeff Perkins. *SQL in 21 Tagen. Die Datenbank-Abfragesprache SQL vollständig erklärt (in 14/21 Tagen)*. 6th ed. Burgthann, Bayern, Germany: Markt+Technik Verlag GmbH, Feb. 1998. ISBN: 978-3-8272-2020-2. Translation of [293] (cit. on p. 89).

[295] Robert H. Storer, S. David Wu, and Renzo Vaccari. "New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling". *Management Science* 38(10):1495–1509, Oct. 1992. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0025-1909. doi:10.1287/mnsc.38.10.1495 (cit. on p. 83).

[296] Philip D. Straffin Jr. "Liu Hui and the First Golden Age of Chinese Mathematics". *Mathematics Magazine* 71(3):163–181, June 1998. London, England, UK: Taylor and Francis Ltd. ISSN: 0025-570X. doi:10.2307/2691200. URL: https://www.researchgate.net/publication/237334342 (visited on 2024-08-10) (cit. on p. 77).

[297] Thomas Strecker and Leonhard Hennig. "Automatic Layouting of Personalized Newspaper Pages". In: *Operations Research Proceedings 2008: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*. Sept. 3–5, 2008, Augsburg, Bayern, Germany: University of Augsburg. Ed. by Bernhard Fleischmann, Karl-Heinz Borgwardt, Robert Klein, and Axel Tuma. Vol. 2008 of Operations Research Proceedings (ORP), Gesellschaft für Operations Research e.V., (GOR). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, 2009, pp. 469–474. ISSN: 0721-5924. doi:10.1007/978-3-642-00142-0_76 (cit. on p. 78).

[298] Thomas Stützle. "Iterated Local Search for the Quadratic Assignment Problem". *European Journal of Operational Research* 174(3):1519–1539, Nov. 2006. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/j.ejor.2005.01.066 (cit. on pp. 83, 87).

[299] Dirk Sudholt. "Crossover Speeds up Building-Block Assembly". In: *14th Genetic and Evolutionary Computation Conference (GECCO'2012)*. July 7–11, 2012, Philadelphia, PA, USA. Ed. by Terence Soule and Jason Hall Moore. New York, NY, USA: Association for Computing Machinery (ACM), 2012, pp. 689–702. ISBN: 978-1-4503-1177-9. doi:10.1145/2330163.2330260. This paper is on the Greedy $(2+1)$ GA, see also [300] (cit. on pp. 82, 116).

[300] Dirk Sudholt. "How Crossover Speeds up Building Block Assembly in Genetic Algorithms". *Evolutionary Computation* 25(2):237–274, 2017. Cambridge, MA, USA: MIT Press. ISSN: 1063-6560. doi:10.1162/EVCO_a_00171. This paper is on the Greedy $(2+1)$ GA, see also [299] (cit. on pp. 82, 116).

[301] Éric D. Taillard. "Benchmarks for Basic Scheduling Problems". *European Journal of Operational Research* 64(2):278–285, Jan. 1993. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/0377-2217(93)90182-M. URL: https://arodes.hes-so.ch/record/8224/files/Preprint.pdf (visited on 2026-02-20) (cit. on p. 83).

[302] Éric D. Taillard. "Comparison of Iterative Searches for the Quadratic Assignment Problem". *Location Science. Now incorporated into* Computers & Operations Research 3(2):87–105, Aug. 1995. Amsterdam, The Netherlands: Elsevier B.V. doi:10.1016/0966-8349(95)00008-6 (cit. on p. 89).

[303] Éric D. Taillard. "Robust Taboo Search for the Quadratic Assignment Problem". *Parallel Computing* 17(4–5):443–455, July 1991. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0167-8191. doi:10.1016/S0167-8191(05)80147-4 (cit. on pp. 87, 89, 102, 116).

[304] Éric D. Taillard and Luca M. Gambardella. *Adaptive Memories for the Quadratic Assignment Problem*. Tech. rep. IDSIA-87-97. Lugano, Switzerland: Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale (IDSIA), Sept. 10, 1997. URL: https://mistic.iict-heig-vd.ch/taillard/articles.dir/TaillardGambardella1997.pdf (visited on 2026-09-28). A MA using TS from [303] as well as a ACO-based approach refining its solutions with a very simple SLS are presented. (Cit. on p. 87).

[305] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sept. 2018. ISBN: 978-1-4842-3841-7 (cit. on pp. 88, 89).

[306] Alkin Tezuysal and Ibrar Ahmed. *Database Design and Modeling with PostgreSQL and MySQL*. Birmingham, England, UK: Packt Publishing Ltd, July 2024. ISBN: 978-1-80323-347-5 (cit. on pp. 85, 87).

[307] The Editors of Encyclopaedia Britannica, ed. *Encyclopaedia Britannica*. Chicago, IL, USA: Encyclopædia Britannica, Inc.

[308] *The Unicode Standard, Version 15.1: Archived Code Charts*. South San Francisco, CA, USA: The Unicode Consortium, Aug. 25, 2023. URL: https://www.unicode.org/Public/15.1.0/charts/CodeCharts.pdf (visited on 2024-07-26) (cit. on p. 90).

[309] Dirk Thierens, David Edward Goldberg, and Angela Guimarães Pereira. "Domino Convergence, Drift, and the Temporal-Salience Structure of Problems". In: pp. 535–540. doi:10.1109/ICEC.1998.700085 (cit. on p. 77).

[310] George K. Thiruvathukal, Konstantin Läufer, and Benjamin Gonzalez. "Unit Testing Considered Useful". *Computing in Science & Engineering* 8(6):76–87, Nov.–Dec. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2006.124. URL: https://www.researchgate.net/publication/220094077 (visited on 2024-10-01) (cit. on p. 91).

[311] Sarah Louise Thomson, Gabriela Ochoa, Daan van den Berg, Tianyu Liang (梁天宇), and Thomas Weise (汤卫思). "Entropy, Search Trajectories, and Explainability for Frequency Fitness Assignment". In: *Parallel Problem Solving from Nature (PPSN XVIII)*. Vol. 1. Sept. 14–18, 2024, Hagenberg, Mühlkreis, Austria. Ed. by Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck. Vol. 15148 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer. ISSN: 0302-9743. ISBN: 978-3-031-70054-5. doi:10.1007/978-3-031-70055-2_23. Explores RLS, FRLS, and SA on the QAP. (Cit. on pp. 81, 85).

[312] Ulrich Wilhelm Thonemann and Andreas Bölte. *An Improved Simulated Annealing Algorithm for the Quadratic Assignment Problem*. Working Paper. Paderborn, Nordrhein-Westfalen, Germany: School of Business, Department of Production and Operations Research, University of Paderborn, 1994 (cit. on p. 88).

[313] Marco Tomassini. "Complex Networks Analysis of the Energy Landscape of the Low Autocorrelation Binary Sequences Problem". *Physica A: Statistical Mechanics and its Applications* 577:126089, Sept. 1, 2021. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0378-4371. doi:10.1016/j.physa.2021.126089. Lists the LABS as $\mathcal{NP}$-hard. (Cit. on p. 84).

[314] Linus Torvalds. "The Linux Edge". *Communications of the ACM (CACM)* 42(4):38–39, Apr. 1999. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/299157.299165 (cit. on p. 84).

[315] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, Mar. 2024. ISBN: 979-8-8688-0215-7 (cit. on pp. 82, 91).

[316] *Unicode®15.1.0*. South San Francisco, CA, USA: The Unicode Consortium, Sept. 12, 2023. ISBN: 978-1-936213-33-7. URL: https://www.unicode.org/versions/Unicode15.1.0 (visited on 2024-07-26) (cit. on p. 90).

[317] Jan van de Craats. *An Introduction to Complex Numbers*. Amsterdam, The Netherlands: Universiteit van Amsterdam, 2017–Apr. 25, 2022. URL: https://staff.science.uva.nl/j.vandecraats/ComplexNumbers.pdf (visited on 2025-11-11) (cit. on pp. 69, 80, 82, 92).

[318] Daan van den Berg, Florian Braam, Mark Moes, Emiel Suilen, and Sandjai Bhulai. "Almost Squares in Almost Squares: Solving the Final Instance". In: *5th International Conference on Data Analytics (DATA ANALYTICS'2016)*. Oct. 9–13, 2016, Venice, Venezia, Veneto, Italy. Ed. by Sandjai Bhulai and Ivana Semanjski. Wilmington, DE, USA: International Academy, Research, and Industry Association (IARIA), 2016, pp. 69–74. ISSN: 2308-4464. ISBN: 978-1-61208-510-4. URL: https://math.vu.nl/~sbhulai/publications/data_analytics2016b.pdf (visited on 2026-10-01). This poses a very interesting 2BP|R|F problem, where "almost squars," i.e., rectangles with sizes $(i, i+1)$, must be placed in an almost square. (Cit. on pp. 78, 79).

[319] Bruce M. Van Horn II and Quan Nguyen. *Hands-On Application Development with PyCharm*. 2nd ed. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2023. ISBN: 978-1-83763-235-0 (cit. on p. 87).

[320] Clarissa Van Hoyweghen, David Edward Goldberg, and Bart Naudts. "From Twomax To The Ising Model: Easy And Hard Symmetrical Problems". In: *4th Genetic and Evolutionary Computation Conference (GECCO'2002)*. July 9–13, 2002, New York, NY, USA. Ed. by William B. Langdon, Erick Cantú-Paz, Keith E. Mathias, Rajkumar Roy, David Davis, Riccardo Poli, Karthik Balakrishnan, Vasant G. Honavar, Günter Rudolph, Joachim Wegener, Larry Bull, Mitchell A. Potter, Alan C. Schultz, Julian F. Miller, Edmund K. Burke, and Natasa Jonoska. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, 2002, pp. 626–633. ISBN: 978-1-55860-878-8 (cit. on p. 90).

[321] Guido van Rossum and Łukasz Langa. *Type Hints*. Python Enhancement Proposal (PEP) 484. Beaverton, OR, USA: Python Software Foundation (PSF), Sept. 29, 2014. URL: https://peps.python.org/pep-0484 (visited on 2024-08-22) (cit. on p. 90).

[322] Guido van Rossum, Barry Warsaw, and Alyssa Coghlan. *Style Guide for Python Code*. Python Enhancement Proposal (PEP) 8. Beaverton, OR, USA: Python Software Foundation (PSF), July 5, 2001. URL: https://peps.python.org/pep-0008 (visited on 2024-07-27) (cit. on p. 80).

[323] Sander van Vugt. *Linux Fundamentals*. 2nd ed. Hoboken, NJ, USA: Pearson IT Certification, June 2022. ISBN: 978-0-13-792931-3 (cit. on p. 84).

[324] Dieter Vandesande, Christoph Jabs, Jeremias Berg, Johannes Fichte, George Katsirelos, Ruben Martins, André Schidler, and Dominik Schreiber. *MaxSAT Evaluation 2026: Solvers, Benchmarks & Results*. v1. Geneva, Switzerland: zenodo (hosted by European Organization for Nuclear Research (CERN)), July 14, 2026. doi:10.5281/zenodo.21358014. See also https://maxsat-evaluations.github.io. (Cit. on p. 85).

[325] Kristian Verduin, Sarah Louise Thomson, and Daan van den Berg. "Too Constrained for Genetic Algorithms. Too Hard for Evolutionary Computing. The Traveling Tournament Problem". In: *15th International Joint Conference on Computational Intelligence (IJCCI'23)*. Nov. 13–15, 2023, Rome, Italy. Ed. by Niki van Stein, Francesco Marcelloni, H. K. Lam, Marie Cottrell, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2023, pp. 246–257. ISSN: 2184-3236. ISBN: 978-989-758-674-3. doi:10.5220/0012192100003595 (cit. on p. 90).

[326] Petr Vilím, Philippe Laborie, and Paul Shaw. "Failure-Directed Search for Constraint-Based Scheduling". In: *12th International Conference on the Integration of AI and OR Techniques in Constraint Programming (CPAIOR'2015)*. Ed. by Laurent Michel. Vol. 9075 of Theoretical Computer Science and General Issues (LNTCS), sub-series of Lecture Notes in Computer Science (LNCS). Barcelona, Spain: Springer, 2015, pp. 437–453. ISSN: 2512-2010. ISBN: 978-3-319-18007-6. doi:10.1007/978-3-319-18008-3_30. URL: https://www.researchgate.net/publication/280223626 (visited on 2026-09-30) (cit. on p. 83).

[327] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 80, 88).

[328] Thomas Weise (汤卫思). *Global Optimization Algorithms – Theory and Application*. self-published, 2009. URL: https://www.researchgate.net/publication/200622167 (visited on 2025-07-25) (cit. on pp. 81, 84, 85, 88).

[329] Thomas Weise (汤卫思). *jsspInstancesAndResults: Results, Data, and Instances of the Job Shop Scheduling Problem*. San Francisco, CA, USA: GitHub Inc, 2019–2020. URL: https://github.com/thomasWeise/jsspInstancesAndResults (visited on 2025-08-20) (cit. on p. 83).

[330] Thomas Weise (汤卫思). *Programming with C*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2026. URL: https://thomasweise.github.io/programmingWithC (visited on 2026-05-10) (cit. on p. 80).

[331] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), 2024–2026. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 1, iii, 85, 87, 88).

[332] Thomas Weise (汤卫思), Raymond Chiong, Jörg Lässig, Ke Tang (唐珂), Shigeyoshi Tsutsui, Wenxiang Chen (陈文祥), Zbigniew "Zbyszek" Michalewicz, and Xin Yao (姚新). "Benchmarking Optimization Algorithms: An Open Source Framework for the Traveling Salesman Problem". *IEEE Computational Intelligence Magazine (CIM)* 9(3):40–52, Aug. 2014. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1556-603X. doi:10.1109/MCI.2014.2326101. Applies a multitude of algorithms including SLSes, EAs, MAs, ACO, and EDAs to the TSPLib instances. (Cit. on pp. 78, 90, 91).

[333] Thomas Weise (汤卫思), Xinlu Li (李新路), Yan Chen (陈岩), and Zhize Wu (吴志泽). "Solving Job Shop Scheduling Problems without using a Bias for Good Solutions". In: *23rd Genetic and Evolutionary Computation Conference (GECCO'2021), Companion Volume*. July 10–14, 2021, Lille, France. Ed. by Krzysztof Krawiec. New York, NY, USA: Association for Computing Machinery (ACM), 2021, pp. 1459–1466. ISBN: 978-1-4503-8351-6. doi:10.1145/3449726.3463124. Applies FFA to the JSSP. (Cit. on p. 81).

[334] Thomas Weise (汤卫思), Mingxu Wan (万明绪), Pu Wang (王璞), Ke Tang (唐珂), Alexandre Devert, and Xin Yao (姚新). "Frequency Fitness Assignment". *IEEE Transactions on Evolutionary Computation* 18(2):226–243, Apr. 2014. Los Alamitos, CA, USA: IEEE Computer Society. ISSN: 1089-778X. doi:10.1109/TEVC.2013.2251885. The article introducing FFA. (Cit. on p. 81).

[335] Thomas Weise (汤卫思) and Zhize Wu (吴志泽). "Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms". In: *25th Conference on Genetic and Evolutionary Computation (GECCO'2023), Companion Volume*. July 15–19, 2023, Lisbon, Portugal. Ed. by Sara Silva and Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM), 2023, pp. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306. Introduces the experiment execution and data collection facilities of `moptipy`. (Cit. on p. 85).

[336] Thomas Weise (汤卫思), Zhize Wu (吴志泽), Xinlu Li (李新路), and Yan Chen (陈岩). "Frequency Fitness Assignment: Making Optimization Algorithms Invariant under Bijective Transformations of the Objective Function Value". *IEEE Transactions on Evolutionary Computation* 25(2):307–319, Apr. 2021. Los Alamitos, CA, USA: IEEE Computer Society. ISSN: 1089-778X. doi:10.1109/TEVC.2020.3032090 (cit. on pp. 77, 81, 83, 84, 87, 89, 90).

[337] Thomas Weise (汤卫思), Zhize Wu (吴志泽), Xinlu Li (李新路), Yan Chen (陈岩), and Jörg Lässig. "Frequency Fitness Assignment: Optimization without Bias for Good Solutions can be Efficient". *IEEE Transactions on Evolutionary Computation* 27(4):980–992, 2023. Los Alamitos, CA, USA: IEEE Computer Society. ISSN: 1089-778X. doi:10.1109/TEVC.2022.3191698. Applies many FFA-based algorithm variants to problems such as OneMax, LeadingOnes, Linear Harmonic Function (LinHarm), TwoMax, Trap, Jump, Plateau, Ising Model, and N-Queens. (Cit. on pp. 77, 81, 84, 89, 90).

[338] Thomas Weise (汤卫思), Yuezhong Wu (吴越钟), Raymond Chiong, Ke Tang (唐珂), and Jörg Lässig. "Global versus Local Search: The Impact of Population Sizes on Evolutionary Algorithm Performance". *Journal of Global Optimization* 66(3):511–534, Feb. 2016. London, England, UK: Springer Nature Limited. ISSN: 0925-5001. doi:10.1007/s10898-016-0417-5 (cit. on pp. 78, 90).

[339] Eric Wolfgang Weisstein. "Irrational Number". In: *MathWorld – A Wolfram Web Resource*. Champaign, IL, USA: Wolfram Research, Inc., Aug. 22, 2024. URL: https://mathworld.wolfram.com/IrrationalNumber.html (visited on 2026-11-04) (cit. on p. 83).

[340] Eric Wolfgang Weisstein. *MathWorld – A Wolfram Web Resource*. Champaign, IL, USA: Wolfram Research, Inc., Aug. 22, 2024. URL: https://mathworld.wolfram.com (visited on 2024-09-24).

[341] Eric Wolfgang Weisstein. "Transcendental Number". In: *MathWorld – A Wolfram Web Resource*. Champaign, IL, USA: Wolfram Research, Inc., Aug. 22, 2024. URL: https://mathworld.wolfram.com/TranscendentalNumber.html (visited on 2026-11-04) (cit. on p. 89).

[342] *What is a Relational Database?* Armonk, NY, USA: International Business Machines Corporation (IBM), Oct. 20, 2021–Dec. 12, 2024. URL: https://www.ibm.com/think/topics/relational-databases (visited on 2025-01-05) (cit. on p. 88).

[343] L. Darrell Whitley. "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best". In: *3rd International Conference on Genetic Algorithms (ICGA'1989)*. June 1989, Fairfax, VA, USA: George Mason University. Ed. by J. David Schaffer. Burlington, MA, USA/San Mateo, CA, USA: Morgan Kaufmann Publishers, 1989, pp. 116–123. ISBN: 978-1-55860-066-9. URL: https://www.researchgate.net/publication/2527551 (visited on 2025-08-08) (cit. on pp. 84, 88).

[344] Ulf Michael "Monty" Widenius, David Axmark, and Uppsala, Sweden: MySQL AB. *MySQL Reference Manual – Documentation from the Source*. Sebastopol, CA, USA: O'Reilly Media, Inc., July 9, 2002. ISBN: 978-0-596-00265-7 (cit. on p. 85).

[345] James M. Wilson. "Gantt Charts: A Centenary Appreciation". *European Journal of Operational Research* 149(2):430–437, Sept. 2003. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0377-2217. doi:10.1016/S0377-2217(02)00769-5 (cit. on p. 81).

[346] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (cit. on p. 87).

[347] L. Wong and Lai-Soon Lee. "Heuristic Placement Routines for Two-Dimensional Bin Packing Problem". 5(4):334–341, Dec. 2009. doi:/10.3844/jmssp.2009.334.341. URL: https://www.thescipub.com/pdf/jmssp.2009.334.341 (visited on 2026-10-02). A heuristic for the 2BP|R|F and the 2BP|O|F (cit. on p. 96).

[348] CAO Xiang (曹翔), Zhize Wu (吴志泽), Daan van den Berg, and Thomas Weise (汤卫思). "Randomized Local Search vs. NSGA-II vs. Frequency Fitness Assignment on The Traveling Tournament Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 38–49. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012891500003837 (cit. on pp. 81, 85, 90).

[349] Zhou Xu and Chung-Yee Lee. "New Lower Bound and Exact Method for the Continuous Berth Allocation Problem". *Operations Research* 66(3):ii–iv, 597–892, May–June 2018. Catonsville, MD, USA: The Institute for Operations Research and the Management Sciences (INFORMS). ISSN: 0030-364X. doi:10.1287/opre.2017.1687. URL: https://ira.lib.polyu.edu.hk/bitstream/10397/78451/1/Xu_New_Lower_Bound.pdf (visited on 2026-10-01) (cit. on p. 79).

[350] Takeshi Yamada and Ryohei Nakano. "A Genetic Algorithm Applicable to Large-Scale Job-Shop Instances". In: *Parallel Problem Solving from Nature 2 (PPSN-II)*. Sept. 28–30, 1992, Brussels, Belgium. Ed. by Reinhard Männer and Bernard Manderick. Amsterdam, The Netherlands: Elsevier B.V., 1992, pp. 281–290. URL: https://www.researchgate.net/publication/220701684 (visited on 2026-09-30) (cit. on p. 83).

[351] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2022. ISBN: 978-1-83763-244-2 (cit. on p. 87).

[352] Kinza Yasar and Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., June 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (visited on 2025-01-11) (cit. on p. 80).

[353] François Yergeau. *UTF-8, A Transformation Format of ISO 10646*. Request for Comments (RFC) 3629. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Nov. 2003. URL: https://www.ietf.org/rfc/rfc3629.txt (visited on 2025-02-05). See Unicode and [155] (cit. on p. 91).

[354] Wenqi Ying (应雯棋), ed. *Commemoration of Ancient Chinese Mathematical Master Liu Hui for his Timeless Influence on Mathematics and Civilizational Exchange*. Vol. 48 (Special Issue) of CAST Newsletter. China, Beijing (中国北京市): 中国科学技术协会 (China Association for Science and Technology, CAST), Nov. 2024. URL: https://english.cast.org.cn/cms_files/filemanager/1941250207/attach/202412/8f23655a82364d19ad7874eb37b23035.pdf (visited on 2025-08-24). Proofreader: Yumeng Wei (魏雨萌), Designer: Shan Zhang (张珊) (cit. on p. 77).

[355] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, June 2017. ISBN: 978-1-78439-687-9 (cit. on p. 79).

[356] Jiazheng Zeng (曾嘉政), Thomas Weise (汤卫思), Zhize Wu (吴志泽), and Markus Wagner. "When Frequency Fitness Assignment Fails: Trapped States in Frequency-Guided Local Search". In: *28th Genetic and Evolutionary Computation Conference (GECCO'2026)*. July 13–17, 2026, San José, Costa Rica. Ed. by Leonardo Trujillo and Ting Hu. New York, NY, USA: Association for Computing Machinery (ACM), 2026, pp. 233–242. ISBN: 979-8-4007-2487-9. doi:10.1145/3795095.3805098. Explores $(1+1)$ EA, $(1+1)$ FEA, RLS, and FRLS as well as hybrid algorithms on a variety of discrete benchmarks such as OneMax, LeadingOnes, LinHarm, Binary Integer (BinInt), TwoMax, Trap, Jump, Plateau, Ising Model, N-Queens and also applies them to the search for Low-Autocorrelation Binary Sequences (LABS). (Cit. on pp. 77, 78, 81, 83, 84, 86–90).

[357] Chaoyong Zhang, Xinyu Shao, Yunqing Rao, and Haobo Qiu. "Some New Results on Tabu Search Algorithm Applied to the Job-Shop Scheduling Problem". In: *Tabu Search*. Ed. by Wassim Jaziri. London, England, UK: IntechOpen Limited, Aug. 1, 2008. Chap. 8. ISBN: 978-3-902613-34-9. doi:10.5772/5593. URL: https://www.intechopen.com/chapters/4596 (visited on 2026-09-30) (cit. on pp. 83, 89).

[358] Rui Zhao (赵睿), Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, and Thomas Weise (汤卫思). "Randomized Local Search on the 2D Rectangular Bin Packing Problem with Item Rotation". In: *26th Genetic and Evolutionary Computation Conference (GECCO'2024)*. July 14–18, 2024, Melbourne, VIC, Australia. Ed. by Xiaodong Li and Julia Handl. New York, NY, USA: Association for Computing Machinery (ACM), 2024, pp. 235–238. ISBN: 979-8-4007-0494-9. doi:10.1145/3638530.3654139 (cit. on pp. 78, 82, 85).

[359] Rui Zhao (赵睿), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, Tianyu Liang (梁天宇), Ming Tan (檀明), and Thomas Weise (汤卫思). "Randomized Local Search for Two-Dimensional Bin Packing and a Negative Result for Frequency Fitness Assignment". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 15–26. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888500003837 (cit. on pp. 78, 79, 82, 85).

[360] Nicola Abdo Ziadeh, Michael B. Rowton, A. Geoffrey Woodhead, Wolfgang Helck, Jean L.A. Filliozat, Hiroyuki Momo, Eric Thompson, E.J. Wiesenberg, and Shih-ch'ang Wu. "Chronology – Christian History, Dates, Events". In: *Encyclopaedia Britannica*. Ed. by The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., July 26, 1999–Mar. 20, 2024. URL: https://www.britannica.com/topic/chronology/Christian (visited on 2025-08-27) (cit. on pp. 79, 80).