Programming with Python Questions

Fragen zum Programmieren mit Python

Thomas Weise (汤卫思)

June 17, 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

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* [104], 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* [104], 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 / Fließkommaartithmetik

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.

Chapter 5

Boolean Expressions / Boolesche Ausrücke

5.1 Data type(s) / Datentyp(en) (Q21)

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 (Q22)

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 (Q23)

Listing 5.1: The program bool_01.py (src)

```python
"""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 5.1 produce?
2. The last line of the program is a bonus...

DE

1. Welche Ausgabe produziert das Programm bool_01.py in Listing 5.1?
2. Die letzte Zeile ist ein Bonus...

Chapter 6

Text

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

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 (Q25)

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 (Q26)

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 (Q27)

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 (Q28)

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 (Q29)

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 (Q30)

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 (Q31)

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.

Chapter 7

Nothing / Nichts

7.1 Data type(s) / Datentyp(en) (Q32)

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 (Q33)

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 (Q34)

EN What does the function `print`?

DE Was macht die Funktion `print`?

8.2 Rounding / Runden (Q35)

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 (Q36)

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.

Chapter 9

Variable

9.1 What is it? / Was ist das? (Q37)

EN What is a variable in Python?

DE Was ist eine Variable in Python?

9.2 Program Understanding / Programm Verstehen (Q38)

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 (Q39)

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.

Chapter 10

Tools / Werkzeuge

10.1 MyPy (Q40)

EN What is Mypy?

DE Was ist Mypy?

10.2 MyPy (Q41)

Listing 10.1: The program mypy_01.py (src)

```
"""An example for 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 2.1.0 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 (Q42)

EN What is a linter?

DE Was ist ein Linter?

10.4 Ruff (Q43)

EN What is Ruff?

DE Was ist Ruff?

10.5 Ruff (Q44)

Listing 10.3: The program ruff_01.py (src)

```
"""An example for 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 0.15.17 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 (Q45)

EN What is Pylint?

DE Was ist Pylint?

10.7 Pylint (Q46)

Listing 10.5: The program pylint_01.py (src)

```
"""An example for 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 4.0.6 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 (Q47)

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? (Q48)

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 (Q49)

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

```
1  """Lists: What output is produced by this program?"""
2
3  a: list[int] = [1, 2, 3, 4]
4  a.append(5)
5  a.remove(3)
6  del a[2]
7  a.extend([5, 2])
8  a.sort()
9
10 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?

12.3 Programming / Programmieren (Q50)

EN Assume that a list `values` contains integer and floating point numbers. Write Python code that computes and prints the arithmetic mean of these values. The arithmetic mean – often also called average – is the sum of the values divided by the number of values. Do not use any built-in Python functions for summation or for computing the arithmetic mean. Handle the special case of an empty list in a reasonable way.

DE Nehmen Sie an, dass eine Liste `values` Ganzzahlen und Fließkommazahlen beinhaltet. Schreiben Sie Python-Kode der den arithmetischen Mittelwert dieser Werte berechnet und ausgibt. Der arithmetische Mittelwert – oft auch Durchschnitt genannt – ist die Summe der Werte geteilt durch deren Anzahl. Verwenden Sie keine built-in Python-Funktionen zum Berechnen der Summe oder des Mittelwerts. Beachten und behandeln Sie den Sonderfall einer leeren Liste auf sinnvolle Art.

Chapter 13

Tuple / Tuple

13.1 What is it? / Was ist das? (Q51)

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? (Q52)

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 (Q53)

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 (Q54)

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 (Q55)

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? (Q56)

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 (Q57)

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 (Q58)

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.

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

Listing 14.4: Another independent execution of set_02.py.

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

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 (Q59)

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? (Q60)

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 (Q61)

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 (Q62)

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 (Q63)

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 (Q64)

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 (Q65)

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? (Q66)

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? (Q67)

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? (Q68)

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 (Q69)

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 (Q70)

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 (Q71)

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 (Q72)

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 (Q73)

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 (Q74)

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? (Q75)

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 (Q76)

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

```
1 """ For-Loop Example. """
2
3 for i in range(5):
4     for j in range(5):
5         if i < j:
6             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 (Q77)

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 (Q78)

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 (Q79)

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.

17.6 Programming / Programmieren (Q80)

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 `****`.

17.7 Programming / Programmieren (Q81)

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).

17.8 Programming / Programmieren (Q82)

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? (Q83)

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 (Q84)

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 (Q85)

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 (Q86)

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

EN The sequence defined by the equation above is called Collatz Problem [50]. 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 [50]. 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.

18.5 Programming / Programmieren (Q87)

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

```
1 """ Convert a `while`-loop to a `for`-loop. """
2
3 index: int = 0
4 while index < 5:
5     print(f"The index is {index}.")
6     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.

18.6 Programming / Programmieren (Q88)

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).

18.7 Programming / Programmieren (Q89)

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.

18.8 Programming / Programmieren (Q90)

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.

18.9 Programming / Programmieren (Q91)

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? (Q92)

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 (Q93)

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 (Q94)

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 (Q95)

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

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 $\binom{n}{k}$.

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 $\binom{n}{k}$ zurückliefert.

19.5 Program Understanding / Programm Verstehen (Q96)

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 (Q97)

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 (Q98)

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 (Q99)

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.

Chapter 20

Unit Tests

20.1 Unit Tests (Q100)

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 (Q101)

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 0.03s ===========================
# pytest 9.1.0 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 (Q102)

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 (Q103)

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 (Q104)

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 (Q105)

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

EN The equation above was given by Kellogg in [46] 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 [46] 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 (Q106)

Listing 20.4: The program doctest_01.py (src)

```python
""" 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 0.02s ===========================
# pytest 9.1.0 with pytest-timeout 2.4.0 failed with exit code 1.
```

20.7 Program Understanding / Programm Verstehen (Q106) (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? (Q107)

EN What are `Exceptions` in `Python`?

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

21.2 What is it? / Was ist das? (Q108)

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? (Q109)

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? (Q110)

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 (Q111)

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 (Q112)

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 (Q113)

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: integer modulo 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 (Q114)

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 (Q115)

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(
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 (Q116)

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: integer modulo 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? (Q117)

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? (Q118)

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? (Q119)

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 (Q120)

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 (Q121)

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 (Q122)

```
[[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 (Q123)

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 (Q124)

```
[(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? (Q125)

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? (Q126)

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? (Q127)

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 (Q128)

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 (Q129)

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 (Q130)

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

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

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

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

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 $\mathrm{i}^2 = -1$ [98].

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 [98].

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 (Q131)

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

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

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

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

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 (Q132)

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 (Q133)

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 (Q134)

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 (Q135)

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., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$ [14, 25, 55].

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 [30, 42, 64], which is approximately $\pi \approx 3.141\,592\,653\,589\,793\,238\,462\,643$. In Python, it is provided by the `math` module as constant `pi` with value `3.141592653589793`. In PostgreSQL, it is provided by the Structured Query Language (SQL) function `pi()` with value `3.141592653589793` [60].

$i..j$ 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. For example, $5..9$ is equivalent to $\{5, 6, 7, 8, 9\}$

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 [10, 63, 110]. Learn more at `https://www.gnu.org/software/bash`.

C is a programming language, which is very successful in system programming situations [23, 75].

$\mathbb{C}$ 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 $\mathrm{i}^2 = -1$ [98]. .

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 for 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 [7, 54, 69, 77, 80]. These requests and responses are usually sent over network connections. A typical example for 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.

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* [103].

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 [109].

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 [33]. They must be delimited by `"""..."""` [33, 101].

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` [24] or tools such as `pytest` [47]. 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 [28], the base of the natural logarithm. $e \in \mathbb{R}$ is an irrational and transcendental number [30, 42], which is approximately $e \approx 2.718\,281\,828\,459\,045\,235\,360$. In Python, it is provided by the `math` module as constant `e` with value `2.718281828459045`. In PostgreSQL, you can obtain it via the SQL function `exp(1)` as value `2.718281828459045` [60].

exit code When a process terminates, it can return a single integer value (the exit status code) to indicate success or failure [43]. 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-string let you include the results of expressions in strings [11, 31, 32, 34, 61, 88]. 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"..."`.

GIGO Garbage In–Garbage Out, see, e.g., [72]

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes [86, 97]. Learn more at `https://git-scm.com`.

GitHub is a website where software projects can be hosted and managed via the Git VCS [71, 97]. Learn more at `https://github.com`.

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 $\mathrm{i}^2 = -1$ [98].

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.

iOS is the operating system that powers Apple iPhones [13, 87]. Learn more at `https://www.apple.com/ios`.

iPadOS is the operating system that powers Apple iPads [13]. Learn more at `https://www.apple.com/ipados`.

IT information technology

LAMP Stack A system setup for web applications: Linux, Apache (a web server), MySQL, and the server-side scripting language PHP [12, 38].

linter A linter is a tool for analyzing program code to identify bugs, problems, vulnerabilities, and inconsistent code styles [41, 83]. Ruff is an example for a linter used in the Python world.

Linux is the leading open source operating system, i.e., a free alternative to Microsoft Windows [3, 37, 85, 96, 102]. 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.

macOS or Mac OS is the operating system that powers Apple Mac(intosh) computers [81, 87]. Learn more at `https://www.apple.com/macos`.

MariaDB An open source relational database management system that has forked off from MySQL [1, 2, 4, 26, 57, 78]. See `https://mariadb.org` for more information.

Microsoft Windows is a commercial proprietary operating system [9]. 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.

Mypy is a static type checking tool for Python [53] that makes use of type hints. Learn more at https://github.com/python/mypy and in [104].

MySQL An open source relational database management system [8, 26, 79, 94, 106]. MySQL is famous for its use in the LAMP Stack. See https://www.mysql.com for more information.

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

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

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

PostgreSQL An open source object-relational DBMS [29, 65, 74, 94]. 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 [99, 107, 108]. 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 [76]. Learn more at https://www.pylint.org.

pytest is a framework for writing and executing unit tests in Python [22, 48, 67, 70, 107]. Learn more at https://pytest.org.

Python The Python programming language [39, 52, 56, 104], i.e., what you will learn about in our book [104]. 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 $\mathbb{Q} \subset \mathbb{R}$.

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

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 [17, 35, 36, 89, 93, 103, 105].

Ruff is a linter and code formatting tool for Python [58, 59]. Learn more at https://docs.astral.sh/ruff or in [104].

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 for servers are web servers [12] 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" [49].

SQL The *Structured Query Language* is basically a programming language for querying and manipulating relational databases [15, 18–20, 40, 62, 90–93]. It is understood by many DBMSes. You find the SQL commands supported by PostgreSQL in the reference [90].

`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) [45]. 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`) [45]. 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`) [45]. 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 for string interpolation is turning `f"Rounded {1.234:.2f}"` to `"Rounded 1.23"`.

terminal A terminal is a text-based window where you can enter commands and execute them [3, 16]. 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.

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 [51, 100]. `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` [16, 38]. 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`.

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 [5, 66, 68, 70, 82, 95]. 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.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code [97]. 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.

WWW World Wide Web [6, 21]

Xcode is offers the tools for developing, testing, and distributing applications as well as an IDE for Apple platforms such as `macOS` and `iOS` [84].

$\mathbb{Z}$ 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 $\overline{z} = a - b\mathrm{i}$ [98]. You get the complex conjugate of a complex number by flipping the sign of the imaginary part.

Python Commands

Bibliography

[1] 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 [2] (cit. on pp. 72, 77).

[2] 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 [1] (cit. on pp. 72, 77).

[3] 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. 72, 74).

[4] 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. 72).

[5] Kent L. Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., Sept. 2004. ISBN: 978-0-596-00743-0 (cit. on p. 74).

[6] 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. 74).

[7] 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. 71).

[8] 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. 73).

[9] 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. 73).

[10] Ron Brash and Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, July 2018. ISBN: 978-1-78862-936-2 (cit. on p. 71).

[11] Florian Bruhin. *Python f-Strings*. Winterthur, Switzerland: Bruhin Software, May 31, 2023. URL: https://fstring.help (visited on 2024-07-25) (cit. on p. 72).

[12] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, June 2022 (cit. on pp. 72, 73).

[13] 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. 72).

[14] 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. 71).

[15] 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. 73).

[16] 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 p. 74).

[17] 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. 73).

[18] *Database Language SQL*. Tech. rep. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (cit. on p. 73).

[19] 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. 73).

[20] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (cit. on p. 73).

[21] 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. 74).

[22] 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. 73).

[23] 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. 71).

[24] "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. 71).

[25] 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. 71).

[26] 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 pp. 72, 73).

[27] 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 [28]. (Cit. on p. 78).

[28] 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 [27] for a translation. (Cit. on pp. 72, 78).

[29] 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. 73).

[30] 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. 71, 72).

[31] "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. 72).

[32] 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. 72).

[33] 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. 71).

[34] 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. 72).

[35] 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. 73).

[36] 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. 73).

[37] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (cit. on p. 72).

[38] 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. 72, 74).

[39] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 73).

[40] *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. 73).

[41] 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. 72).

[42] 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. 71, 72).

[43] "`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. 72).

[44] 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).

[45] "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 pp. 73, 74).

[46] 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. 50).

[47] 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. 71).

[48] 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. 73).

[49] Jay LaCroix. *Mastering Ubuntu Server*. 4th ed. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2022. ISBN: 978-1-80323-424-3 (cit. on p. 73).

[50] 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. 39).

[51] 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. 74).

[52] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 73).

[53] 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. 73).

[54] 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. 71).

[55] 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. 71).

[56] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 73).

[57] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (visited on 2025-04-24) (cit. on p. 72).

[58] Charlie Marsh. "Ruff". In: URL: https://pypi.org/project/ruff (visited on 2025-08-29) (cit. on p. 73).

[59] 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. 73).

[60] "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. 71, 72).

[61] 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. 72).

[62] 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. ISBN: 978-1-55860-456-8 (cit. on p. 73).

[63] 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. 71).

[64] 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. 71).

[65] 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. 73).

[66] 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. 74).

[67] 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. 73).

[68] 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. 74).

[69] 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. 71).

[70] 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. 73, 74).

[71] 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. 72).

[72] 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. 72).

[73] *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).

[74] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2024 (cit. on p. 73).

[75] *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. 71).

[76] Pylint Contributors. *Pylint*. Toulouse, Occitanie, France: Logilab, 2003–2024. URL: https://pylint.readthedocs.io/en/stable (visited on 2024-09-24) (cit. on p. 73).

[77] 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. 71).

[78] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2014. ISBN: 978-1-78398-154-0 (cit. on p. 72).

[79] 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. 73).

[80] 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. 71).

[81] 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. 72).

[82] 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. 74).

[83] 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. 72).

[84] 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. 74).

[85] 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. 72).

[86] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2023. ISBN: 978-1-0981-3391-7 (cit. on p. 72).

[87] 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 p. 72).

[88] 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. 72).

[89] 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. 73).

[90] "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. 73).

[91] Ryan K. Stephens and Ronald R. Plew. *Sams Teach Yourself SQL in 21 Days*. 4th ed. Sams Tech 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. 73, 82).

[92] 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 [91] (cit. on p. 73).

[93] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sept. 2018. ISBN: 978-1-4842-3841-7 (cit. on p. 73).

[94] 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 p. 73).

[95] 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. 74).

[96] 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. 72).

[97] 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. 72, 74).

[98] 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. 63, 71, 72, 75).

[99] 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. 73).

[100] 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. 74).

[101] 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. 71).

[102] Sander van Vugt. *Linux Fundamentals*. 2nd ed. Hoboken, NJ, USA: Pearson IT Certification, June 2022. ISBN: 978-0-13-792931-3 (cit. on p. 72).

[103] 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. 71, 73).

[104] 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), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 1, iii, 73).

[105] *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. 73).

[106] 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. 73).

[107] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (cit. on p. 73).

[108] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2022. ISBN: 978-1-83763-244-2 (cit. on p. 73).

[109] 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. 71).

[110] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, June 2017. ISBN: 978-1-78439-687-9 (cit. on p. 71).