# Programming with Python Questions

## Fragen zum Programmieren mit Python

Thomas Weise (汤卫思)

October 24, 2025


**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* [90], 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* [90], 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

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

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

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

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

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)

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

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

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

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

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

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

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

EN Write a Python3 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 Python3-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)

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

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

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

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

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

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

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.

# Chapter 5

# Boolean Expressions / Boolesche Ausrücke

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

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

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

DE What is die Bedeutung von `True` und `False`? Zu welchem Datentyp gehören sie?

## 5.3 Program Understanding / Programm Verstehen

Listing 5.1: The program bool_01.py (src)

```
1  """Working with Boolean expressions."""
2
3  a: bool = 1 < 5 <= 5 < 6   # Line 3
4  print(a)                   # Line 4: What does this print?
5
6  b: str  = "Hello World!"   # Line 6
7  c: bool = b is not b       # Line 7
8  print(c)                   # Line 8: What does this print?
9
10 print(a or c)              # Line 10: What does this print?
11 print(a and c)             # Line 11: What does this print?
12 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)

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

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 Program Understanding / Programm Verstehen

Listing 6.1: The program str_01.py (src)

```
1 """Printing some strings."""
2
3 print("Hello World!")            # Line 3: What output does it produce?
4 print("Hello " + "World!")       # Line 4: What output does it produce?
5 print("Hello\nWorld!")           # Line 5: What output does it produce?
6 print("Hello " * 3 + "World!")   # Line 6: What output does it produce?
7 print(f"{8 / 2}")                # Line 7: What output does it produce?
8 print(f"{8 / 2 = }")             # Line 8: What output does it produce?
9 print("Hello\nWorld!")           # Line 9: 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?

## 6.4 Program Understanding / Programm Verstehen

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?

## 6.5 f-Strings

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

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

# Chapter 7

# Nothing / Nichts

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

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

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

# Variable

## 8.1 What is it?

EN What is a variable in Python?

DE Was ist eine Variable in Python?

## 8.2 Program Understanding / Programm Verstehen

Listing 8.1: 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 8.1, 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 8.1, 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?

## 8.3 Type Hints

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 9

# Tools / Werkzeuge

## 9.1 MyPy

EN What is Mypy?

DE Was ist Mypy?

## 9.2 MyPy

Listing 9.1: The program mypy_01.py (src)

```
"""An example for using MyPy."""

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

Listing 9.2: The result of Mypy applied to Listing 9.1.

```
$ 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 1.18.2 failed with exit code 1.
```

EN Listing 9.2 shows the output of Mypy applied to program mypy_01.py given in Listing 9.1. Explain this output.

DE Listing 9.2 zeigt die Ausgabe von Mypy angewandt auf Programm mypy_01.py in Listing 9.1. Erklären Sie diese Ausgabe.

## 9.3 Linter

EN What is a linter?

DE Was ist ein Linter?

## 9.4 Ruff

EN What is Ruff?

DE Was ist Ruff?

## 9.5 Ruff

Listing 9.3: The program ruff_01.py (src)

```
"""An example for using Ruff."""

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

Listing 9.4: The result of Ruff applied to Listing 9.3.

```
$ 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 0.14.2 failed with exit code 1.
```

EN Listing 9.4 shows the output of Ruff applied to program ruff_01.py given in Listing 9.3. Explain this output.

DE Listing 9.4 zeigt die Ausgabe von Ruff angewandt auf Programm ruff_01.py in Listing 9.3. Erklären Sie diese Ausgabe.

## 9.6 Pylint

EN What is Pylint?

DE Was ist Pylint?

## 9.7 Pylint

Listing 9.5: The program pylint_01.py (src)

```
"""An example for using Pylint."""

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

Listing 9.6: The result of Pylint applied to Listing 9.5.

```
$ 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 4.0.2 failed with exit code 4.
```

EN Listing 9.6 shows the output of Pylint applied to program pylint_01.py given in Listing 9.5. Explain this output.

DE Listing 9.6 zeigt die Ausgabe von Pylint angewandt auf Programm pylint_01.py in Listing 9.5. Erklären Sie diese Ausgabe.

# Chapter 10

# Equality and Identity / Gleichheit und Identität

## 10.1 Understanding / Verstehen

Listing 10.1: Program `identity_01.py` (stored in file `identity_01.py`; output in Listing 10.2)

```
"""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 10.2: The stdout of the program `identity_01.py` given in Listing 10.1.

```
2.718281828459045
2.718281828459045
True
False
```

EN Explain the output of program `identity_01.py` in Listing 10.2.

DE Erklären Sie die Ausgabe des Programms `identity_01.py` in Listing 10.2.

# Chapter 11

# Lists / Listen

## 11.1 What is it?

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.

## 11.2 Program Understanding / Programm Verstehen

Listing 11.1: The program `list_01.py` (src)

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

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

print(a)
```

EN Which output will the program `list_01.py` in Listing 11.1 produce?

DE Welche Ausgabe produziert das Programm `list_01.py` in Listing 11.1?

# Chapter 12

# Tuple / Tuple

## 12.1 What is it?

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.

## 12.2 What is it?

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?

## 12.3 Program Understanding / Programm Verstehen

Listing 12.1: 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 12.1 produce?

DE Welche Ausgabe produziert das Programm `tuple_01.py` in Listing 12.1?

# Chapter 13

# Sets / Mengen

## 13.1 What is it?

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.

## 13.2 Program Understanding / Programm Verstehen

Listing 13.1: 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 13.1 produce?

DE Welche Ausgabe produziert das Programm set_01.py in Listing 13.1?

## 13.3 Program Understanding / Programm Verstehen

Listing 13.2: 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 13.3: One independent execution of set_02.py.

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

Listing 13.4: Another independent execution of set_02.py.

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

EN Listing 13.3 and Listing 13.4 show the outputs of two independent executions of set_02.py in Listing 13.2. Why are they different?

DE Listing 13.3 und Listing 13.4 zeigen die Ausgaben von zwei unabhängigen Ausführungen von Programm set_02.py in Listing 13.2. Warum sind sie verschieden?

# Chapter 14

# Dictionaries

## 14.1 What is it?

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 (`list`) in Python?Nennen Sie vier Eigenschaften bzw. Dinge die man mit Dictionaries in Python machen kann.

## 14.2 Program Understanding / Programm Verstehen

Listing 14.1: The program dict_01.py (src)

```
1  """Lists: 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 14.1 produce?

DE Welche Ausgabe produziert das Programm dict_01.py in Listing 14.1?

# Chapter 15

# Alternatives / Alternativen

## 15.1 What is it?

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.

## 15.2 What is it?

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.

## 15.3 What is it?

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.

## 15.4 Program Understanding / Programm Verstehen

Listing 15.1: 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 15.1 produce?

DE Welche Ausgabe produziert das Programm if_01.py in Listing 15.1?

## 15.5 Program Understanding / Programm Verstehen

Listing 15.2: 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 15.2 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm if_02.py in Listing 15.2 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

## 15.6 Programming / Programmieren

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.

# Chapter 16

# For-Loop / For-Schleife

## 16.1 What is it?

EN Explain the syntax *and* purpose/use of an `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.

## 16.2 Program Understanding / Programm Verstehen

Listing 16.1: The program `for_01.py` (src)

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

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

EN Explain program `for_01.py` in Listing 16.1 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `for_01.py` in Listing 16.1 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

## 16.3 Program Understanding / Programm Verstehen

Listing 16.2: The program `if_01.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 16.2 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `for_02.py` in Listing 16.2 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

# Chapter 17

# While-Loop / While-Schleife

## 17.1 What is it?

EN Explain the syntax *and* purpose/use of an `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.

## 17.2 Program Understanding / Programm Verstehen

Listing 17.1: The program `while_01.py` (src)

```
""" While-Loop Example. """

a, b, c, d, e = 5, 3, 2, 1, 0
counter = 0

while not (a < b < c < d < e):
    counter += 1
    if a > b:
        a, b = b, a
    if b > c:
        b, c = c, b
    if c > d:
        c, d = d, c
    if d > e:
        d, e = e, d

print(f"Loop finished after {counter} iterations.")
```

EN Explain program `while_01.py` in Listing 17.1 *line-by-line*, i.e., every single line. What output does it produce?

DE Erklären Sie das Programm `while_01.py` in Listing 17.1 Zeile für Zeile, also jede einzelne Zeile. Welche Ausgabe produziert es?

# Chapter 18

# Functions / Funktionen

## 18.1 What is it?

EN Explain the syntax *and* purpose/use of an `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.

## 18.2 Program Understanding / Programm Verstehen

Listing 18.1: 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 20: What does it print?
print(compute([25, -34, 6, -5, 9]))   # Line 21: What does it print?
```

EN Carefully read the program `function_01.py` in Listing 18.1.

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 18.1 genau.

1. Was macht die Funktion `compute`?
2. Welche Ausgabe produziert das Programm?
3. Ist das program 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.)

## 18.3 Programming / Programmieren

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{18.1}$$

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

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

## 18.4 Program Understanding / Programm Verstehen

Listing 18.2: 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 16: What does it print?
print(compute(7))  # Line 17: What does it print?
print(compute(0))  # Line 18: What does it print?
```

EN Carefully read the program `function_02.py` in Listing 18.2.

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 18.2 genau.

1. Was macht die Funktion `compute`?
2. Welche Ausgabe produziert das Programm?
3. Ist das program 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.)

# Chapter 19

# Unit Tests

## 19.1 Unit Tests

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?

## 19.2 Program Understanding / Programm Verstehen

Listing 19.1: 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 19.2: 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 19.3: 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 8.4.2 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 19.3.

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 19.3 gelistet ist.

# 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., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$ [13, 23, 47].

$\pi$ 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 [28, 39, 56], 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` [52].

**Bash** is a the 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, 55, 96]. Learn more at `https://www.gnu.org/software/bash`.

**C** is a programming language, which is very successful in system programming situations [22, 66].

**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, 46, 61, 68, 71]. 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* [89].

**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 [95].

$e$ is Euler's number [26], the base of the natural logarithm. $e \in \mathbb{R}$ is an irrational and transcendental number [28, 39], 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` [52].

**f-string** let you include the results of expressions in strings [11, 29–31, 53, 76]. 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"..."`.

**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 [75, 85]. Learn more at `https://git-scm.com`.

**GitHub** is a website where software projects can be hosted and managed via the Git VCS [63, 85]. Learn more at https://github.com.

**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.

**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, 35].

**linter** A linter is a tool for analyzing program code to identify bugs, problems, vulnerabilities, and inconsistent code styles [38, 73]. 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 for Microsoft Windows [3, 34, 74, 84, 88]. 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.

**MariaDB** An open source relational database management system that has forked off from MySQL [1, 2, 4, 24, 49, 69]. 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 [45] that makes use of type hints. Learn more at https://github.com/python/mypy and in [90].

**MySQL** An open source relational database management system [8, 24, 70, 82, 92]. MySQL is famous for its use in the LAMP Stack. See https://www.mysql.com for more information.

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

**PostgreSQL** An open source object-relational DBMS [27, 57, 65, 82]. 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 [86, 93, 94]. It comes in a free community 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 [67]. Learn more at https://www.pylint.org.

**pytest** is a framework for writing and executing unit tests in Python [21, 41, 59, 62, 93]. Learn more at https://pytest.org.

**Python** The Python programming language [36, 44, 48, 90], i.e., what you will learn about in our book [90]. Learn more at https://python.org.

$\mathbb{R}$ 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 [16, 32, 33, 77, 81, 89, 91].

**Ruff** is a linter and code formatting tool for Python [50, 51]. Learn more at https://docs.astral.sh/ruff or in [90].

**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" [42].

**SQL** The *Structured Query Language* is basically a programming language for querying and manipulating relational databases [14, 17–19, 37, 54, 78–81]. It is understood by many DBMSes. You find the SQL commands supported by `PostgreSQL` in the reference [78].

`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`) [40]. 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 standard error stream (`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`) [40]. 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`) [40]. 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, 15]. 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 [43, 87]. `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` [15, 35]. 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, 58, 60, 62, 72, 83]. 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 [85]. 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, 20]

$\mathbb{Z}$ 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 $\mathbb{Z} \subset \mathbb{R}$.

# 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. 32, 36).

[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. 32, 36).

[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. 32, 33).

[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. 32).

[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. 33).

[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. 34).

[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. 31).

[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. 32).

[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. 32).

[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. 31).

[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. 31).

[12] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, June 2022 (cit. on pp. 32, 33).

[13] 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. 31).

[14] 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. 33).

[15] 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. 33).

[16] 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. 32).

[17] *Database Language SQL*. Tech. rep. ANSI X3.135-1986. Washington, D.C., USA: American National Standards Institute (ANSI), 1986 (cit. on p. 33).

[18] 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. 33).

[19] *Database Language SQL*. International Standard ISO 9075-1987. Geneva, Switzerland: International Organization for Standardization (ISO), 1987 (cit. on p. 33).

[20] 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. 34).

[21] 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. 32).

[22] 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. 31).

[23] 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. 31).

[24] Russell J.T. Dyer. *Learning MySQL and MariaDB*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2015. ISBN: 978-1-4493-6290-4 (cit. on p. 32).

[25] 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 [26]. (Cit. on p. 37).

[26] 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 [25] for a translation. (Cit. on pp. 31, 37).

[27] 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. 32).

[28] Michael Filaseta. "The Transcendence of $e$ and $\pi$". 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 p. 31).

[29] "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. 31).

[30] 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. 31).

[31] 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. 31).

[32] 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. 32).

[33] 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. 32).

[34] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (cit. on p. 32).

[35] 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. 32, 33).

[36] 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. 32).

[37] *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. 33).

[38] 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. 32).

[39] Arthur Jones, Kenneth R. Pearson, and Sidney A. Morris. "Transcendence of $e$ and $\pi$". 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 p. 31).

[40] "stderr, stdin, stdout – Standard I/O Streams". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Ed. by Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) and San Francisco, CA, USA: The Open Group, Aug. 8, 2024. URL: https://pubs.opengroup.org/onlinepubs/9799919799/functions/stdin.html (visited on 2024-10-30) (cit. on p. 33).

[41] 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. 32).

[42] Jay LaCroix. *Mastering Ubuntu Server*. 4th ed. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2022. ISBN: 978-1-80323-424-3 (cit. on p. 33).

[43] 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. 33).

[44] 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. 32).

[45] 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. 32).

[46] 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. 31).

[47] 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. 31).

[48] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 32).

[49] *MariaDB Server Documentation*. Milpitas, CA, USA: MariaDB, 2025. URL: https://mariadb.com/kb/en/documentation (visited on 2025-04-24) (cit. on p. 32).

[50] Charlie Marsh. "Ruff". In: URL: https://pypi.org/project/ruff (visited on 2025-08-29) (cit. on p. 32).

[51] 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. 32).

[52] "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 p. 31).

[53] 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. 31).

[54] 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. 33).

[55] 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. 31).

[56] Ivan Niven. "The Transcendence of $\pi$". *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. 31).

[57] 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. 32).

[58] 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. 33).

[59] 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. 32).

[60] 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. 33).

[61] 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. 31).

[62] 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. 32, 33).

[63] 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. 32).

[64] *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).

[65] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2024 (cit. on p. 32).

[66] *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. 31).

[67] Pylint Contributors. *Pylint*. Toulouse, Occitanie, France: Logilab, 2003–2024. URL: https://pylint.readthedocs.io/en/stable (visited on 2024-09-24) (cit. on p. 32).

[68] 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. 31).

[69] Federico Razzoli. *Mastering MariaDB*. Birmingham, England, UK: Packt Publishing Ltd, Sept. 2014. ISBN: 978-1-78398-154-0 (cit. on p. 32).

[70] 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. 32).

[71] 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. 31).

[72] 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. 33).

[73] 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. 32).

[74] 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. 32).

[75] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2023. ISBN: 978-1-0981-3391-7 (cit. on p. 31).

[76] 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. 31).

[77] 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. 32).

[78] "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. 33).

[79] 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. 33, 40).

[80] 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 [79] (cit. on p. 33).

[81] Allen Taylor. *Introducing SQL and Relational Databases*. New York, NY, USA: Apress Media, LLC, Sept. 2018. ISBN: 978-1-4842-3841-7 (cit. on pp. 32, 33).

[82] 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. 32).

[83] 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. 33).

[84] 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. 32).

[85] 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. 31, 32, 34).

[86] 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. 32).

[87] 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. 33).

[88] Sander van Vugt. *Linux Fundamentals*. 2nd ed. Hoboken, NJ, USA: Pearson IT Certification, June 2022. ISBN: 978-0-13-792931-3 (cit. on p. 32).

[89] 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), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 31, 32).

[90] 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), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 1, ii, 32).

[91] *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. 32).

[92] 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. 32).

[93] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (cit. on p. 32).

[94] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Oct. 2022. ISBN: 978-1-83763-244-2 (cit. on p. 32).

[95] 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. 31).

[96] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, June 2017. ISBN: 978-1-78439-687-9 (cit. on p. 31).