Technische Universität Braunschweig

Institut für Nachrichtentechnik

Einführung in die Programmierung für Nicht Informatiker*innen

Phil Keier
Institut für Nachrichtentechnik (IfN)
Technische Universität Braunschweig

Lösungen 1. Tutorial

31 mögliche Punkte

Aufgabe Datentypen

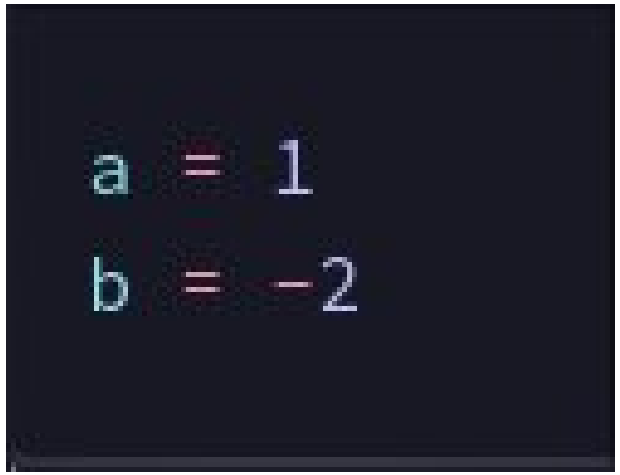
Gebe den Text Hallo Python aus, Nutze dabei die print-funktion. 1 Punkt

```
print("Hallo Python")
```

Aufgabe Datentypen

Aufgabe 2 Punkte:

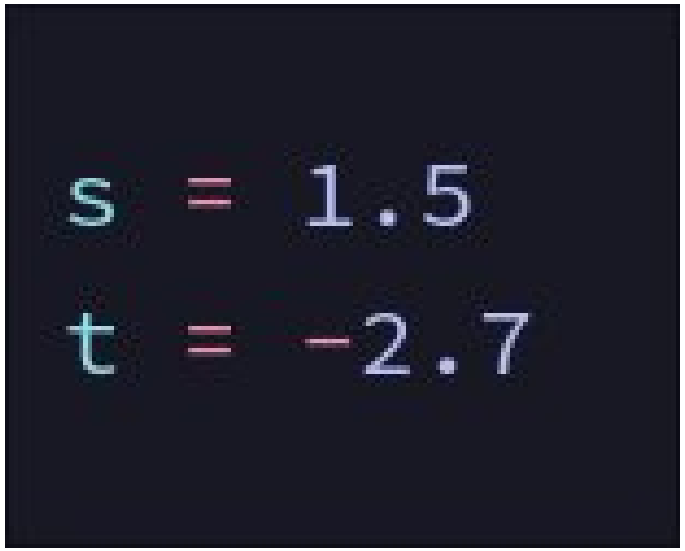
Definieren Sie zunächst die zwei Variablen a und b und initialisieren diese mit einem Integerwert ungleich 0:



Aufgabe Datentypen

Aufgabe 2 Punkte:

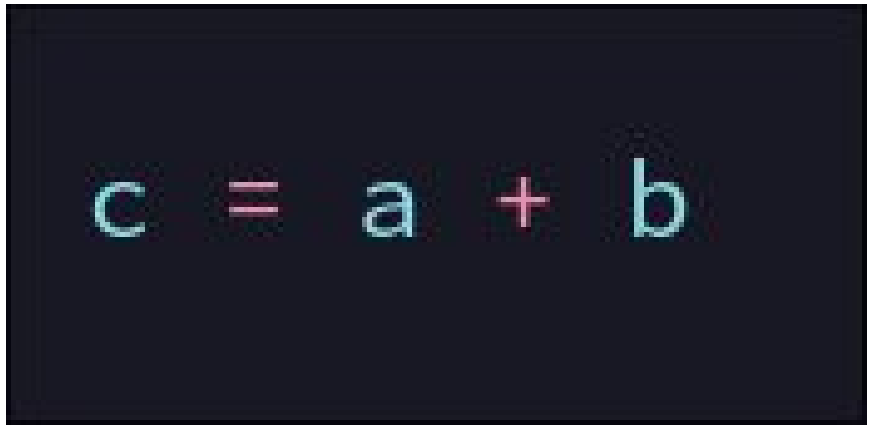
Definieren Sie zwei Variablen s und t und initialisieren diese mit einem Floatwert ungleich 0:



Aufgabe Datentypen

Aufgabe 2 Punkte:

Addieren Sie die Werte der Variablen a und b und speichern Sie das Ergebnis in der Variable c:



Aufgabe Datentypen

Aufgabe 5 Punkte:

Nutzen Sie die Variablen a & b und Speichern Sie die Ergebnisse für die Multiplikation, Division, Ganzzahldivision, Exponentiation und den Modulo-Operator in den unten angegebenen Variablen:

$$m = a \cdot b$$
$$d = \frac{a}{b}$$
$$i = \lfloor \frac{a}{b} \rfloor$$
$$e = a^b$$
$$r = a \bmod b$$

```
m = a * b
d = a / b
i = a // b
e = a ** b
r = a % b
```

Aufgabe Datentypen

Aufgabe 2 Punkte:

Ein String-Objekt (Text) können sie mit Hilfe von 'Some Text' oder "Some Text2" definieren. Definieren sie die Variable text mit einem beliebigen Text.

```
text = "Hi Mom, I am on TV!"
```

Aufgabe Datentypen

Aufgabe 1 Punkte:

Geben Sie die Variablen a & b aus Aufgabe 1 im format "a = 12 und b = 12" (Die Werte sollen dann den Werten aus ihrer Definition entsprechen. 12 ist hier nur ein Beispiel) aus.

```
print("a = {} und b = {}".format(a, b))
```

Aufgabe Listen

Aufgabe 1 Punkte:

Definieren Sie die Variable l und weisen Sie dieser Variable eine Liste mit aufsteigenden Integerwerten von 0 bis 4 zu.

```
l = list(range(5))
l = [0, 1, 2, 3, 4]
```

Aufgabe Listen

Aufgabe 1 Punkte:

Hängen Sie der Liste l noch den Wert 42 an.

Hinweis: Nutzen Sie dafür die Methode .append.

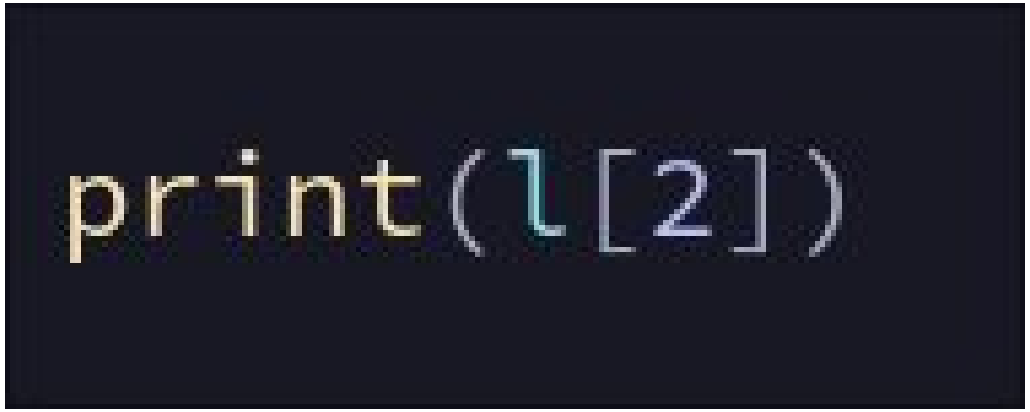
```
l.append(42)
```

Aufgabe Listen

Aufgabe 1 Punkte:

Geben Sie das dritte Element der Liste `l` aus.

Hinweis: Achten Sie darauf das der erste Index immer `0` ist.



Aufgabe Listen

Aufgabe 1 Punkte:

Geben Sie das vorletzte Element der Liste ` aus.

Hinweis: Achten Sie darauf das der letzte Index mit -1 ausgegeben wird.

```
print(l[-2])
```

Aufgabe Dictionaries

Das Dictionary ist ein Datentyp, welcher Schlüssel-Werte-Paare speichert. Dabei wird ein Dictionary mit dict() oder {"Schlüssel1": "Wert1"} initalisiert. Wichtig ist hierbei das ein Dictionary nicht mit {} initialisiert werden kann da dies die Notation für das Set Objekt ist.

Aufgabe 1 Punkte:

Initialisieren Sie die Dictionary Variable `my_dict` mit folgendem Mapping:

Key	Value
„apple“	„Apfel“
„banana“	„Banane“
„cherry“	„Kirsche“

```
my_dict = {
    "apple": "Apfel",
    "banana": "Banane",
    "cherry": "Kirsche"
    }
```

Aufgabe Dictionaries

Aufgabe 1 Punkte:

Fügen Sie nun das Key-Value Paar "pear": "Birne" zu my_dict hinzu.

```
my_dict["pear"] = "Birne"
my_dict.update({"pear": "Birne"})
```

Aufgabe Dictionaries

Aufgabe 2 Punkte:

Speichere die Werte des Dictionaries my_dict in der Variablen values.

Speichern Sie die Elemente des Dictionaries my_dict, welche mit der Funktion .items() ausgegeben werden, in der Variablen items.

```
values = my_dict.values()
items = my_dict.items()
```

Aufgabe Funktionen

Aufgabe 1 Punkte*:

Schreibe eine Funktion successor die auf jede Eingabe +1 rechnet.

Schreibe eine Funktion add mit den Eingabeparametern a & b, welche die Werte von a & b miteinander addiert.

```
def successor(n):
    return n+1

def add(a,b):
    return a+b
```

Aufgabe Funktionen

Aufgabe 1 Punkte:

Schreibe eine Funktion is_odd mit einem Eingabeparameter n die prüft ob die eingegebene Zahl ungerade ist.

Wenn die Zahl gerade ist gebe den Text "Gerade Zahl" und bei ungerade "Ungerade Zahl" zurück.

```
def is_odd(n):
    if n % 2 == 0:
        return "Gerade Zahl"
    else:
        return "Ungerade Zahl"
```

Aufgabe Funktionen

7 Punkte.

Aufgabe: Schreibe eine Funktion fubar mit Eingabeparameter n.
Die Funktion soll wie folgt definiert sein:

- Der Eingabeparameter n ist ein Integer, Floats geben False zurück
- Negative zahlen & 0 beenden die Funktion und geben False zurück
- Die Funktion zählt bis einschließlich dem Eingabeparameter n = 9 -> 1, 2, 3, ..., 9
- Bei jedem Schleifendurchlauf soll die Zahl bei der sich die Schleife gerade befindet mittels print ausgegeben werden werden.
- Ist der zurzeitige Schleifendurchlauf durch 3 teilbar, gebe mittels print denn String Foo aus.
- Ist der zurzeitige Schleifendurchlauf durch 5 teilbar, gebe mittels print denn String Bar aus.
- Ist der zurzeitge Schleifendurrchlauf durch 3 & 5 teilbar, gebe mittels print den String FooBar aus.

Aufgabe Funktionen

```python
def fubar(n: int):
    if isinstance(n, float) or n < 1:
        return False

    count = 1
    while count <= n:
        msg = count
        if count % 3 == 0:
            msg = "Foo"
        if count % 5 == 0:
            msg = "Bar"
        if count % 15 == 0:
            msg = "FooBar"

        count += 1
        print(msg, end=', ')
```

Lösungen 2. Tutorial

17+1 mögliche Punkte

Aufgabe - Kontrollstruckturen

3 Punkte

Schreibe eine Funktion sum_up mit Eingabeparameter n, welcher die Zahlen von 1...n aufsummiert.

Nutze dafür einen for-loop.

```
def sum_up(n: int) -> int:
    count = 0
    for i in range(1,n+1):
        count += i
    return count
```

Aufgabe - Kontrollstruckturen

2 Punkte

Gegeben ist das Dictionary dict2.
Ändere jeden Wert in diesem Dictionary entsprechend der Formel f(x) = x^3-1 mittels for-loop.

```python
dict2 = {"a": 56, 5: 12, "python": 9, 3.14: 1.141414}

# Möglichkeit 1 - Dictionary Comprehension
dict2 = {key: value**3-1 for key, value in dict2.items()}

# Möglichkeit 2 - For Loop
for key, value in dict2.items():
    dict2[key] = value**3 - 1
```

Zusatzaufgabe - Kontrollstruckturen

Keine Punkte (Extra Punkt möglich)

Erstelle eine Liste mittels List Comprehension, welche die Zahlen 1…6 auf deren kubische Zahl 1…216 also der Funktion f(x) = x^3 abbildet.

```
cubics = [n**3 for n in range(1,7)]
```

Aufgabe – System Interactions

2 Punkte

Erstelle und Öffne eine Datei testfile.txt mit der open Funktion, nutze dafür das with - Statement.

Schreibe in diese Datei 100 mal den String "Python\n".

```python
# Möglichkeit 1
with open('testfile.txt', 'w') as f:
    for _ in range(100):
        f.write("Python\n")
```

```python
# Möglichkeit 2
with open('testfile.txt', 'w') as f:
    f.write("Python\n" * 100)
```

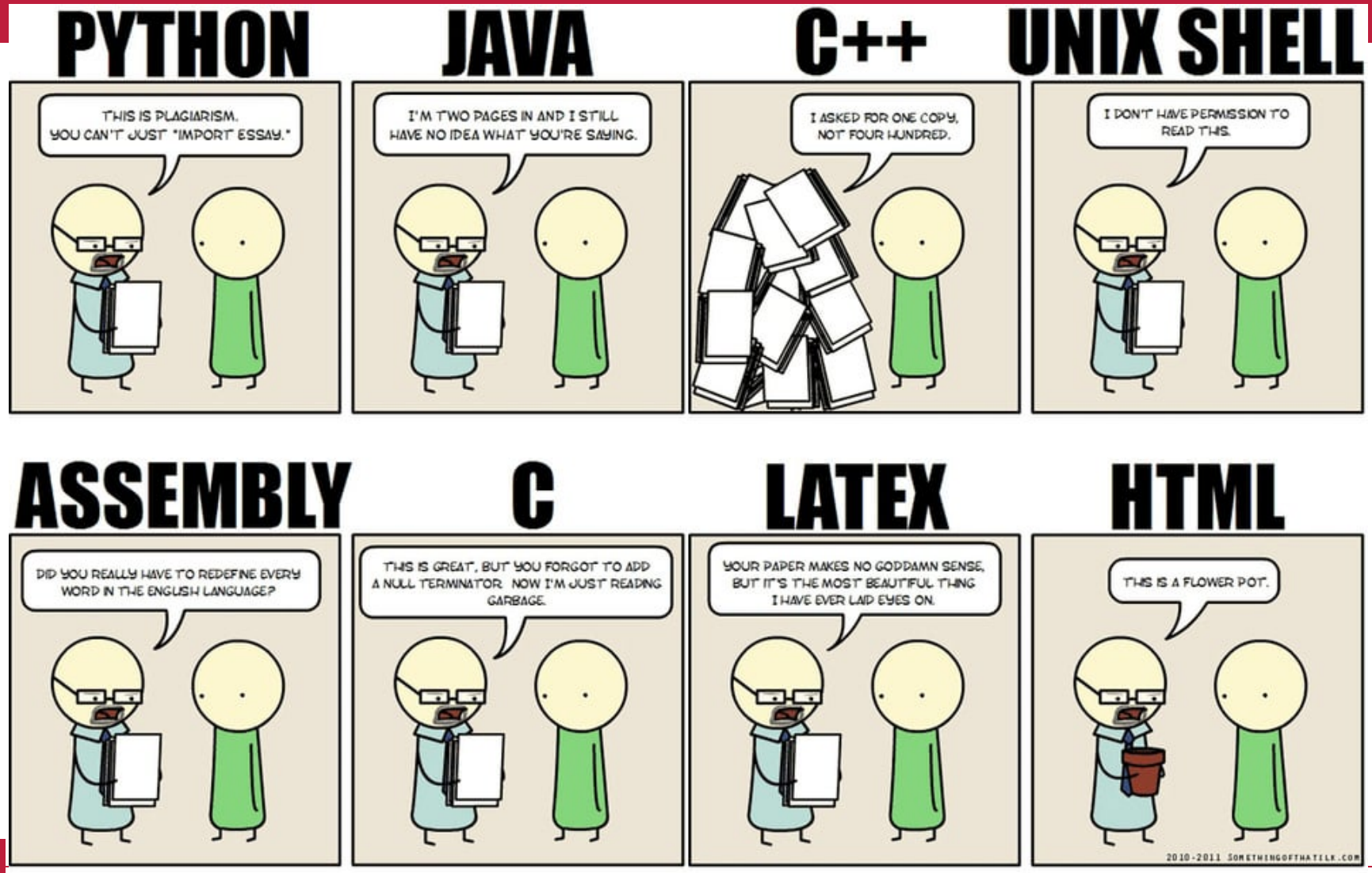
Aufgabe – System Interactions

2 Punkte

Öffne die zuvor erstellte Datei testfile.txt im Lesemodus und weiße den Inhalt der .readlines() Funktion der Variabeln lines zu.

```
with open('testfile.txt', 'r') as f:
    lines = f.readlines()
```

Memes


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
2010-2011 SOMETHINGOFTHATILK.COM

Aufgabe - Importing

3 Punkte

Importiere Pythons Built-In Library random und rufe zuerst aus dem Modul die Funktion seed auf mit dem Eingabewert 42, und weiße danach der Variable rand den Wert des Funktionsaufrufes von randint(1,100) zu.

```
import random
random.seed(42)
rand = random.randint(1,100)
```

Aufgabe - Importing

1 Punkt

Importiere die Built-In Library datetime als dt.

```
import datetime as dt
```

Aufgabe - Importing

2 Punkte

Importiere die Funktion sqrt aus dem Built-In Modul math.
Berechne sqrt(4). Speicher das Ergebnis in der variablen s4.

```
from math import sqrt
s4 = sqrt(4)
```

Aufgabe - Importing

2 Punkte

Importiere das Objekt TextCalendar aus dem Built-In Module calendar als TC.

```
from calendar import TextCalendar as TC

# Output
   November 2025
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 [21] 22 23
24 25 26 27 28 29 30

21.11.2025 - Nächste Vorlesung
```

Lösungen 3. Extended Applications

26 mögliche Punkte

Aufgabe Generatoren

3 Punkte

Schreibe einen Generator *square_count* mit dem Eingabeparameter n, der die Quadratzahlen von 1 … $(n-1)^2$ erzeugt.

Wichtig: Wenn *square_count* kein Generator ist, gibt es 0 Punkte.

Hinweis: Bei einer Eingabe von 5 sollen die Werte 1 4 9 16 ausgegeben werden.

Aufgabe Generatoren

```python
def square_count(n):
    for i in range(1, n):
        yield i*i

# Test
for n in range(2, 7):
    print(f"Square Numbers from 0 to {n-1}:", *square_count(n))
```

Ausgabe:

Square Numbers from 0 to 1: 1
Square Numbers from 0 to 2: 1 4
Square Numbers from 0 to 3: 1 4 9
Square Numbers from 0 to 4: 1 4 9 16
Square Numbers from 0 to 5: 1 4 9 16 25

Aufgabe Generatoren

3 Punkte

Schreibe einen Generator *naturals*, der bei jedem Aufruf die nächste natürliche Zahl ausgibt, beginnend mit 1.

- Es sind **keine Eingabeparameter** erforderlich.
- Wenn die Funktion **kein Generator** ist, gibt es **0 Punkte**.

Hinweis: Orientiere dich am Beispiel *faktoriel_gen*.

```
def faktoriel_gen():
    curr = 1
    count = 1
    while 1:
        curr = count * curr
        count += 1
        yield curr
```

Aufgabe Generatoren

```python
# Test if generator works as intended
import random
test_n = random.randint(5, 17)
test_gen = naturals()
for i in range(1, test_n):
    number = next(test_gen)
    print(number, end=', ')
    assert i == number
```

Ausgabe (test_n = 14)**:**
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,

```python
def naturals():
    curr = 1
    while 1:
        yield curr
        curr += 1
```

Aufgabe Datenklassen

8 Punkte

Erstelle eine Dataclass *Student*.

- Die Hinweise sind aktuell nicht in logischer Reihenfolge.
  Struckturier den Code sinnvoll!

Beispielstudent:

Attribut	Wert
vorname	Martin
nachname	Le
mat_nr	520420
semester	5

Aufgabe Datenklassen

1. Importiere aus dem dataclasses-Modul die Funktion asdict.
2. Die Dataclass soll die Werte *vorname*, *nachname*, *semester* und *mat_nr* speichern. Wählen Sie für jedes Attribut den geeigneten Datentyp (z. B. macht es Sinn, die Semesteranzahl als *int* zu speichern, nicht als *float*).
   - **Wichtig:** Wenn *Student* keine Dataclass ist, gibt es **0 Punkte**.
   - Alle Variablen sollen mit *Type Hints* versehen werden.
   - Hat eines der Attribute keinen optimalen Datentyp, gibt es trotzdem Punkte, solange die Mehrheit korrekt ist.
3. Erstelle ein Objekt mit den Werten des Beispielstudenten und weise das Ergebnis von *asdict(<beispielstudent>)* der Variablen *stud* zu.
4. Füge Kommentare ein, die sich auf die jeweilige PEP 8 Regel beziehen.

Aufgabe Datenklassen

```python
# Imports on top of the File
from dataclasses import asdict

# Newlines between logic blocks

# Docstring classes to improve understandabillity
# Classes are written in Title Case
@dataclass
class Student:
    '''
    Dataclass to store student Informations
    '''
    vorname: str
    nachname: str
    mat_nr: int
    semester: int
```

```python
# Variabls should be Type Hinted,
# if not obvious
stud: dict = asdict(
    # Proper spacing for readabillity
    Student(
        vorname='Martin',
        nachname='Le',
        mat_nr=520420,
        semester=5
        )
    )
```

Aufgabe Datenklassen

6 Punkte

Gegeben sind zwei Listen *colorn* und *colorv_hex*, die zueinander indexsortiert sind.

1. Erstelle eine Dataclass *Color* mit den Attributen *name* und *value* und versehe sie mit passenden Type Hints.

2. Erzeuge anschließend eine Liste, die die Werte aus *colorn* und *colorv_hex* in Instanzen der Dataclass *Color* umwandelt, und speichere diese Liste in der Variablen *colors*.

```
colorn = ['RED', 'GREEN', 'BLUE', 'YELLOW', 'PURPLE']
colorv_hex = ['#FF0000', '#00FF00', '#0000FF', '#FFFF00', '#FF00FF']
```

Aufgabe Datenklassen

```python
@dataclass
class Color:
    name: str
    value: str
```

Möglichkeit 2:

```python
colors: list = []

for i in range(len(color_hex)):
    current_color = Color(
        name=colorn[i],
        value=color_hex[i]
        )
    colors.append(current_color)
```

Möglichkeit 1:

```python
colors: list = [Color(n, w) for n, w in zip(colorn, colorv_hex)]
```

Aufgabe Datenklassen

6 Punkte
Gegeben ist ein Dictionary *dict_obj*.

1. Erstelle eine Dataclass *Transaction*, die die Originalstruktur des Dictionaries abbildet.
   Achte dabei auf **korrekte Type Hint** für die einzelnen Attribute.

2. Schreibe **unterhalb der Dataclass in einer Markdown-Zeile** deine Begründung, warum du diese Struktur gewählt hast. **(!! Nicht als Kommentar !!)**

```python
dict_obj: dict = {
    "IBAN": "DE19500211001598716891", # FAKE!
    "Transaction Number": random.randint(1, 8000),
    "Recipient": {
        "First Name": "Phil",
        "Last Name": "Keier",
    },
    "Amount": 3.14,
}
```

Aufgabe Datenklassen

```python
@dataclass
class Recipient:
    first_name: str
    last_name: str

@dataclass
class Transaction:
    iban: str
    trans_nr: int
    recipient: Recipient
    amount: float
```

„Die vorliegende Struktur bildet die Grundlage für die Verarbeitung finanzieller Transaktionen.

Für die Repräsentation, aufgeteilt auf zwei separate, jedoch logisch miteinander verknüpfte Datenklassen, wurde bewusst entschieden, um eine hohe Modularität und Wiederverwendbarkeit der Daten sicherzustellen.

Das gewählte Modell ermöglicht es, unabhängige Analysen der Empfänger (hier: Recipient) effizient und ohne direkte Abhängigkeit von transaktionsspezifischen Daten durchzuführen, wodurch eine klare Trennung der Verantwortlichkeiten sowie eine verbesserte Wartbarkeit des Systems erreicht wird.“

KI-Regeln

KI Regeln - Programmierung

1. Die Verantwortung für KI-generierten Code obliegt dem/der Studierenden

2. Python Objekte (Funktionen/Klassen/Logische Blöcke) sind mittels *Docstring* kommentiert. Hierbei gilt:
   - Angabe des verwendeten KI Modells inklusive Zugriffsdatum
   - Angabe des verwendeten Prompts
   - Zweck der Nutzung ist zu erläutern
   - Zitations- sowie Verständniskommentare der KI sind zu entfernen
   - Die Einhaltung von PEP8 ist zu überprüfen

3. KI Generierter Code unterliegt gesonderten Styling. **Alle** logischen Blöcke sind mit einem Kommentar zu versehen, der den jeweiligen Zweck kurz beschreibt.

KI Regeln - Programmierung

Schreibe mir einen Generator, welcher mir die Fakultät einer Zahl berechnet. Nutze dafür Python.

Hier ist ein **Python-Generator**, der die Fakultät einer Zahl berechnet und dabei die Zwischenergebnisse schrittweise liefert:

python Copy code

```python
def fakultaet_generator(n):
    """Generator, der die Fakultät von n Schritt für Schritt berechnet."""
    if n < 0:
        raise ValueError("Die Fakultät ist nur für nicht-negative Zahlen definiert.")

    ergebnis = 1
    for i in range(1, n + 1):
        ergebnis *= i
        yield ergebnis
```

Beispielverwendung:

python Copy code

```python
for wert in fakultaet_generator(5):
    print(wert)
```

Ausgabe:

Copy code

```
1
2
6
24
120
```

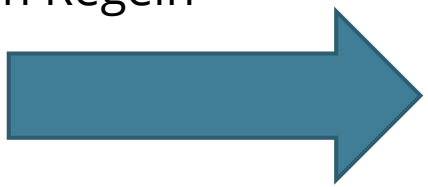
👉 Der Generator gibt bei jedem Schritt den aktuellen Fakultätswert zurück, bis `n!` erreicht ist.

Wenn du stattdessen nur den finalen Wert (ohne Zwischenschritte) möchtest, sag gerne Bescheid!

KI Regeln - Programmierung

```python
def fakultaet_generator(n):
    """Generator, der die Fakultät von n Schritt für Schritt berechnet."""
    if n < 0:
        raise ValueError("Die Fakultät ist nur für nicht-negative Zahlen definiert.")

    ergebnis = 1
    for i in range(1, n + 1):
        ergebnis *= i
        yield ergebnis
```

Transformiert nach Regeln



```python
def fakultaet_generator(n: int) -> int:
    """
    Generiert mittels ChatGPT (26.11.2025)

    Prompt: "..."

    - Berechnung der Fakultät
    - Robuste Fehlerbehandlung
    - Benötigt für Berechnung XXX
    """

    # Robuste Fehlerbehandlung bei Werten kleiner 0
    if n < 0:
        raise ValueError("Die Fakultät ist nur für nicht-negative Zahlen definiert.")

    # Berechnung der Fakultät mittels aufmultiplizieren
    ergebnis = 1
    for i in range(1, n + 1):
        ergebnis *= i
        yield ergebnis
```

KI Regeln - Texte

1. Die Verantwortung für KI-generierten Text obliegt dem/der Studierenden

2. KI generierter/bearbeiteter Text muss gekennzeichnet werden (Datum & Model).

3. Falls KI nur zur Korrektur verwendet wird muss am Ende des Textes der Satz stehen:
   „*Korrektur der Sätze erfolgte mit <KI-Model> am <Datum>*“

   Bsp.:

   (ChatGPT am 26.11.2025)

   „Diese Funktion kann verwendet werden, um die Fakultät einer Zahl schrittweise zu berechnen und dabei jeden Zwischenwert zu erhalten. Sie ist besonders nützlich, wenn große Fakultäten berechnet werden sollen und man den Fortschritt beobachten oder Zwischenergebnisse weiterverarbeiten möchte. Außerdem eignet sie sich gut für speichereffiziente Berechnungen, da sie die Werte nacheinander generiert, anstatt alles auf einmal zu speichern.“

Lösungen 4. NumPy & MatPlotLib

45 mögliche Punkte

Aufgabe Arrays

1 Punkt

Erstelle ein NumPy Array, welches 600 Nullen reserviert und speicher das Array in der Variablen only_zeros.

```
only_zeros = np.zeros(600)
```

Aufgabe Arrays

1 Punkt

Erstelle ein NumPy Array mit 11 Elementen mithilfe der Funktion linspace. Der Startwert soll -6 und der Endwert 16.5 betragen.

Speichere das Ergebnis in der Variablen x_scale.

```
x_scale = np.linspace(-6, 16.5, num=11)
```

Aufgabe Arrays

6 Punkt

Erstelle ein NumPy Array und bearbeite es anschließend wie folgt:

1. Erzeuge ein eindimensionales NumPy Array numbers mit 20 gleichmäßig verteilten Zahlen zwischen 10 und 50 mithilfe von np.linspace.
6. Gebe als Kommentar die NumPy Referenz zu den verwendeten Funktionen an. (Keine Kommentare = 0 Punkte)

https://numpy.org/doc/stable/reference/generated/numpy.linspace.html#numpy-linspace

```
numbers = np.linspace(10, 50, 20)
```

Aufgabe Arrays

2. Berechne ein neues Array squared_numbers, das die Quadrate der Werte aus numbers enthält.

https://numpy.org/doc/stable/reference/generated/numpy.square.html

```
squared_numbers = numbers**2
squared_numbers = np.square(numbers)
```

Aufgabe Arrays

3. Finde alle Elemente in numbers, die größer als 30 sind, und speichere sie in einem neuen Array numbers_gt_30.

https://numpy.org/doc/stable/user/basics.indexing.html#boolean-indexing

```
numbers_gt_30 = numbers[numbers > 30]
```

Aufgabe Arrays

4. Berechne die Summe aller Werte in squared_numbers und speichere das Ergebnis in total_squared.

https://numpy.org/doc/stable/reference/generated/numpy.sum.html

```
total_squared = np.sum(squared_numbers)
```

Aufgabe Arrays

5. Sortiere numbers absteigend und speichere das Ergebnis in numbers_sorted_desc.

https://numpy.org/doc/stable/reference/generated/numpy.sort.html#numpy.sort

```
numbers_sorted_desc = np.sort(numbers)[::-1]
```

Aufgabe Ploting - Squareroot

3 Punkte

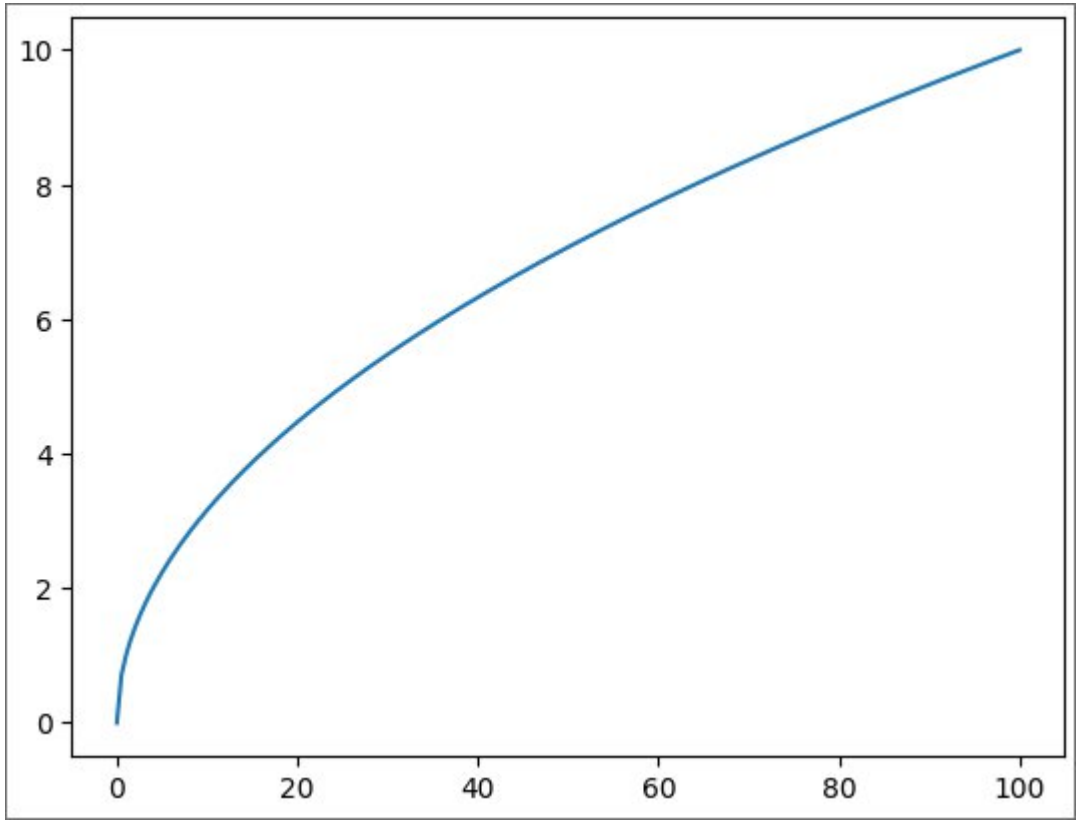
Plotte im Folgenden die Funktion Square Root, mathematisch definiert als f(x) = sqrt(x), $x \geq 0$.

Gehe dabei wie folgt vor:
1. Definiere einen geeigneten Linspace für den Zahlenraum von 0 bis 100. (Tipp: Achte darauf, dass die Quadratwurzel nur für nicht-negative Zahlen definiert ist.)

2. Berechne die Werte für die Quadratwurzel mithilfe der Funktion np.sqrt.

3. Plotte das Ergebnis mit plt.plot und gebe den Plot mit plt.show() aus.

Squareroot




```
xs = np.linspace(0, 100, num=200)
ys = np.sqrt(xs)

plt.plot(xs, ys)
plt.show()
```

Aufgabe Ploting - Multiplot

8 Punkte

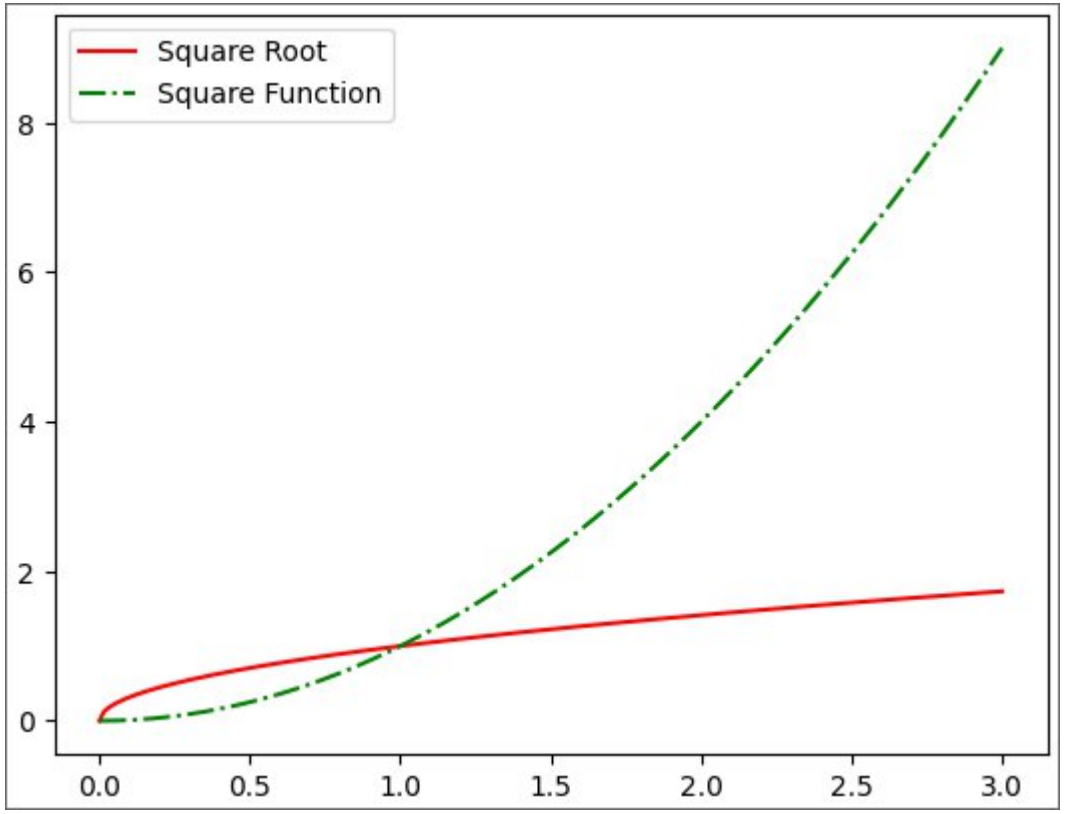
In der nächsten Aufgabe willst du zwei Funktionen gleichzeitig plotten: $f(x) = \sqrt{x}$; $x \geq 0$ und $g(x) = x^2$.

Gehe dabei wie folgt vor:
1. Definieren einen geeigneten Linspace für den Zahlenraum von 0 bis 3. (Tipp: Achte darauf, dass die Quadratwurzel nur für nicht-negative Zahlen definiert ist.)
2. Berechne die Werte für die Quadratwurzel mit np.sqrt.
3. Berechne die Werte für die Quadratzahlen mit np.square.
4. Gib den beiden Plots die Farben Grün und Rot. Nutze hierfür die Color Shorthands aus der Dokumentation.

Aufgabe Ploting - Multiplot

5. Plotte die Quadratfunktion mit dem Linestyle dashdot, wie in der Dokumentation zu Linestyles beschrieben.
6. Vergebe angemessene Labels für beide Plots.
7. Füge die Legende hinzu.
8. Plotte das Ergebnis mit plt.show().

Multiplot




```
xt = np.linspace(0, 3, num=200)
sqrt = np.sqrt(xt)
square = np.square(xt)

plt.plot(xt, sqrt, color='r', label="Square Root")
plt.plot(xt, square, color='g', label="Square Function",
linestyle="dashdot")
plt.legend()
plt.show()
```

Aufgabe – Pseudo Randomness

$X_{n+1} = (a X_n + c) \bmod m, n \geq 0$

Wenn c = 0 gilt, spricht man auch vom **Multiplicative Congruent Generator (MCG)**.

Die einzelnen Werte haben folgende Bedeutung:

- **X_n** ist der Startwert, auch **Seed** genannt.
- **X_n+1** ist der Folgewert, der im nächsten Schritt als **X_n** verwendet wird.
- **a** ist der Vorfaktor des Startwerts, wird auch als **Skalar** bezeichnet, da er den Wert skaliert.
- **c** ist das hinzuaddierte **Offset**.
- **m** ist der Restklassenring oder **Modulus**, der die Werte in einem festen Bereich hält.

Aufgabe – Pseudo Randomness

6 Punkte

Schreibe einen **Linear Congruent Generator** mit dem Funktionsnamen lcg.

- Verwende die oben gegebene mathematische Definition
- Prüfe, dass die Eingabewerte gültig sind.
- lcg soll als unendlicher Generator implementiert werden, der bei jedem Aufruf den nächsten Wert der Pseudo-Zufallsfolge liefert.
- Ekläre in der Markdownzeile die Funktionsweise des Generators. (Keine Erklärung = 0 Punkte)

Linear Congruent Generator

Python

```python
def lcg(seed: int, scalar: int, modulus: int, offset: int) -> int:
    '''
    Linear Congruential Generators

    X(n+1) = (a X(n) + c) mod m; n >= 0

    m > 0;
    0 <= a < m;
    c > 0; a > 0

    '''
    assert modulus > 0, "Modulus must be greater than 0"
    assert 0 <= scalar and scalar < modulus, "Scalar must be in range 0 <=
a < m"

    while seed > 1:
        seed = (scalar*seed+offset) % modulus
        assert seed >= 0
        yield seed
```

Linear Congruent Generator - Funktionsweise

Der Generator implementiert einen linearen Kongruenzgenerator (LCG), der aus einem Startwert seed durch die rekursive Formel (siehe Definition) eine Folge pseudozufälliger Zahlen erzeugt. Bei jedem Durchlauf wird der aktuelle Zustand mit dem Skalar a, dem Offset c und dem Modulus m zu einem neuen Wert transformiert, der anschließend ausgegeben wird.

1. 3089810780120156248
2. 8356396685252565260
3. 1921117399837525548
4. 14806858147081821235
5. 2557599628047639428

```python
# Test
rand = lcg(
    3_935_559_000_370_003_845,
    3_203_021_881_815_356_449,
    2**64-1,
    11_742_185_885_288_659_963
    )

for i in range(5):
    print(f"{i+1}. {next(rand)}")
```

Aufgabe – Plotting a PCG

6 Punkte

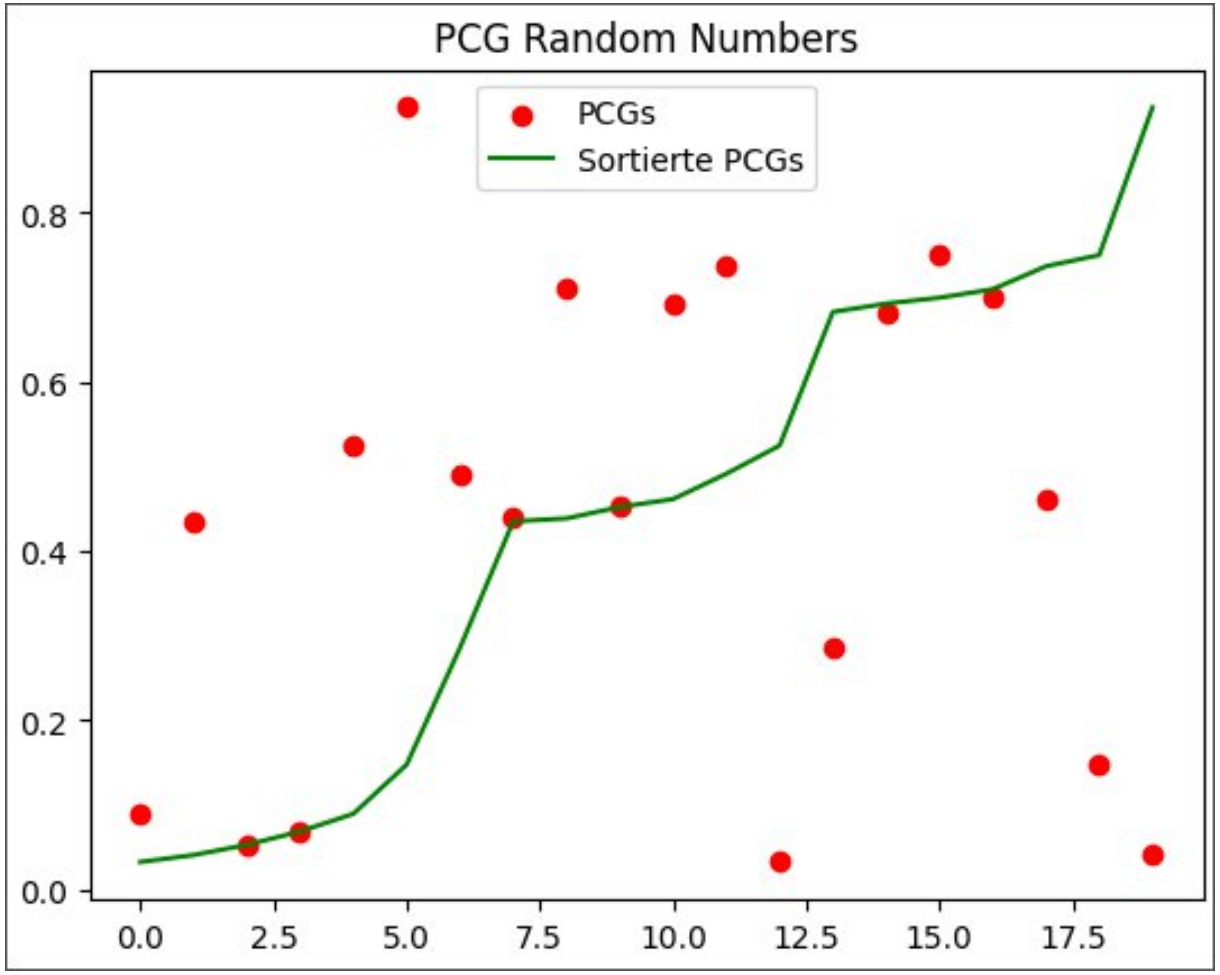
Plotte die Zufallszahlen eines _Permuted Congruent Generators_ (PCG) mithilfe von NumPy und Matplotlib.

- Gegeben ist der Anfangszustand des Generators.
- Verwende die NumPy-Dokumentation um 20 Zufallszahlen zu erzeugen und speichere sie in der Variablen pcgs (Tipp: als NumPy Array).
- Sortiere anschließend die Werte in pcgs und speichere das Ergebnis in pcgs_sorted.
- Plotte sowohl die ursprünglichen als auch die sortierten Zufallszahlen in einem sinnvollen Diagramm (z. B. Linienplot oder Scatterplot) und gestalte den Plot übersichtlich.
– Erkläre in der gegebenen Markdown Zeile deinen Plot. (Keine Erklärung = 0 Punkte)

Permuted Congruent Generator

```python
np.random.seed(42) # Setting a fixed start Value for the Generator
pcgs: np.array = None
pcgs_sorted: np.array = None

gen = np.random.default_rng()
pcgs = np.array([gen.random() for _ in range(20)])
pcgs_sorted = np.sort(pcgs)

# Plot
plt.scatter(np.arange(len(pcgs_sorted)), pcgs, color='r', label='PCGs')
plt.plot(np.arange(len(pcgs_sorted)), pcgs_sorted, color='g',
label='Sortierte PCGs')
plt.title("PCG Random Numbers")
plt.legend()
plt.show()
```

Permuted Congruent Generator



Zu sehen sind in Rot die mittels eines permuted congruential generators (PCG) erzeugten Zufallszahlen im Bereich von 0.0 bis 1.0; in Grün erscheinen dieselben Werte, jedoch sortiert und als Linie geplottet. Ein erkennbares Muster entsteht dabei nicht. Lediglich der sortierte Verlauf zeigt – rein rechnerisch bedingt – einen monoton steigenden Graphen, was für zufällige Werte in dieser Darstellung zu erwarten ist.

Aufgabe – Bar Plot

5 Punkte

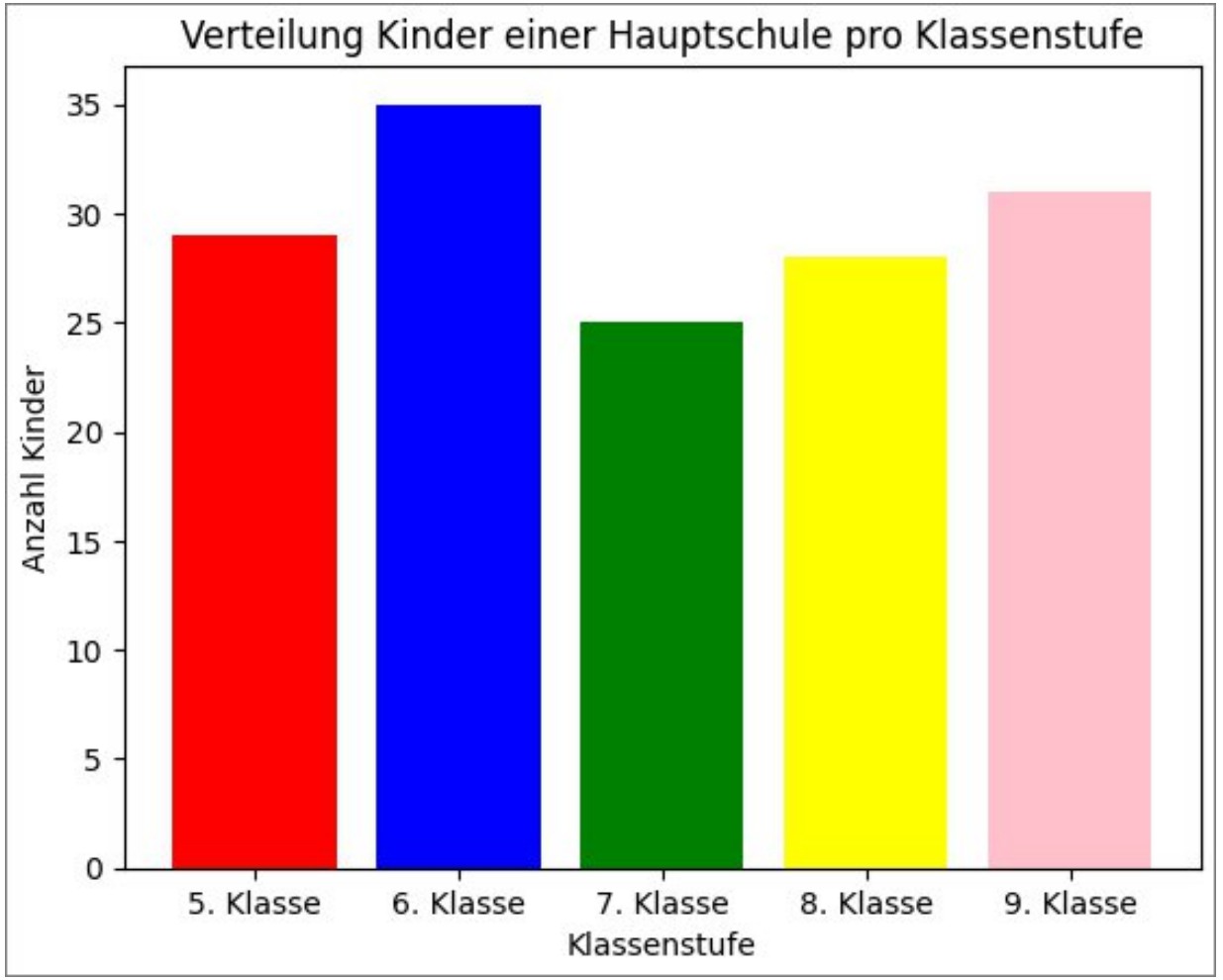
Dir liegt ein Datenset sec_school einer Hauptschule vor, das die Klassenstufen von 5 bis 9 auf die Anzahl ihrer Schüler im jeweiligen Jahrgang abbildet.

Erstelle einen Barplot und gehe dabei wie folgt vor:
1. Wähle ein passendes Farbschema, um die einzelnen Klassenstufen anschaulich darzustellen.
2. Extrahiere die Schlüssel (Klassenstufen) und Werte (Anzahl der Schüler) aus dem Datenset und übergebe diese zusammen mit den Farbwerten an plt.bar().
3. Setze geeignete Beschriftungen für die X- und Y-Achse, um die Darstellung verständlich zu machen.
4. Vergiss nicht, einen passenden Titel für den Plot zu setzen.
5. Gib den Plot mit plt.show() aus, um die Werte zu visualisieren.

Bar Plot - Hauptschule

```python
sec_school = {
    '5. Klasse': 29,
    '6. Klasse': 35,
    '7. Klasse': 25,
    '8. Klasse': 28,
    '9. Klasse': 31
}

colors = ["red", "blue", "green", "yellow", "pink"]

plt.bar(sec_school.keys(), sec_school.values(), color=colors)
plt.xlabel("Klassenstufe")
plt.ylabel("Anzahl Kinder")
plt.title("Verteilung Kinder einer Hauptschule pro Klassenstufe")
plt.show()
```

Bar Plot - Hauptschule

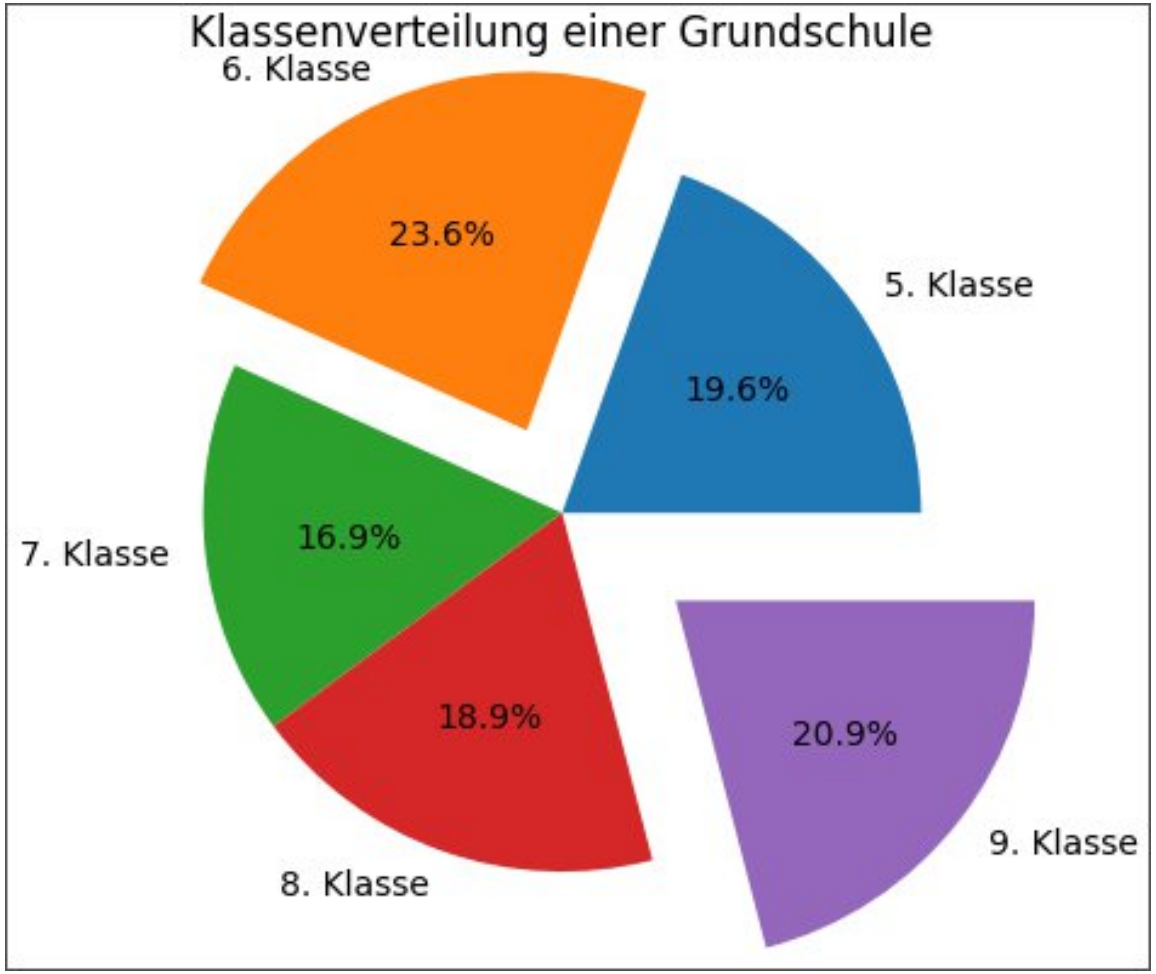


Aufgabe – Pie Plot

5 Punkte

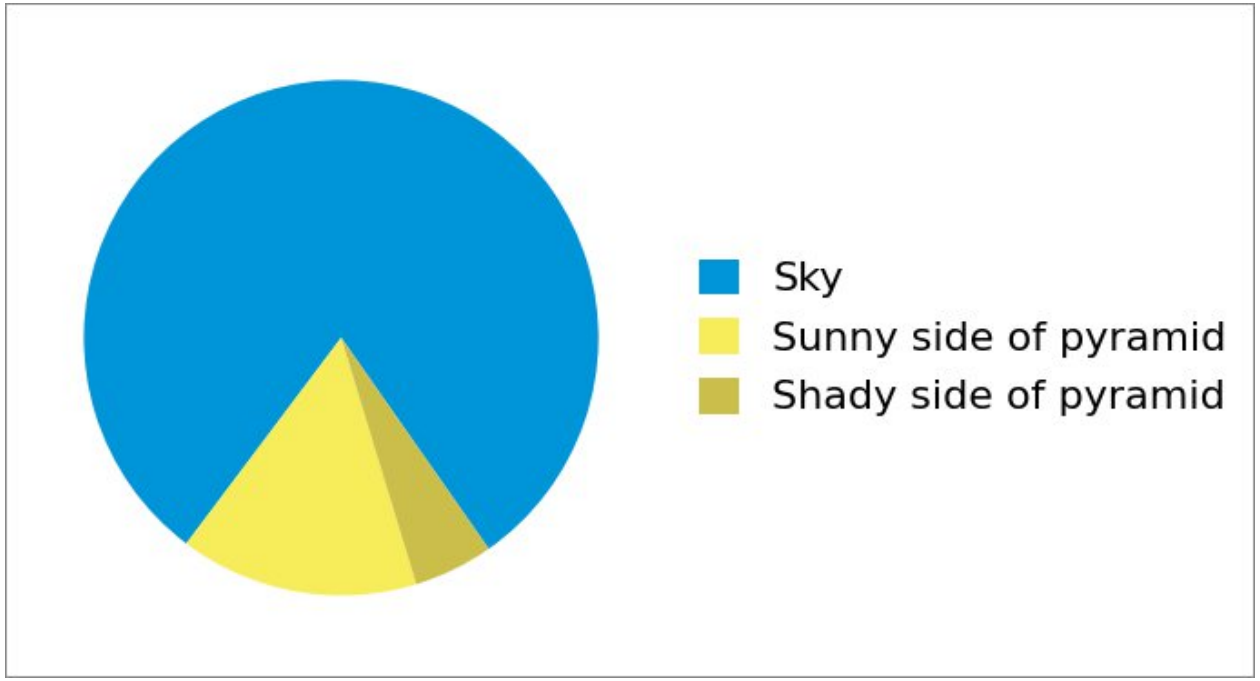
Erstelle einen Pieplot und gehe dabei wie folgt vor:
1. Wähle ein passendes Farbschema, um die einzelnen Klassenstufen anschaulich darzustellen.
2. Extrahiere die Schlüssel (Klassenstufen) und Werte (Anzahl der Schüler) aus dem Datenset und übergebe diese zusammen mit den Farbwerten an plt.pie(). Nutze autopct='%1.1f%%', um die Prozentwerte der Segmente anzuzeigen.
3. Lasse die 6. Klasse um 25 % und die 9. Klasse um 40 % aus dem Kuchen hervortreten, indem du den Parameter explode entsprechend setzt.
4. Setze einen geeigneten Titel für den Plot.
5. Gib den Plot mit plt.show() aus, um die Werte zu visualisieren.

Pie Plot - Hauptschule




```python
sec_school = {
    '5. Klasse': 29,
    '6. Klasse': 35,
    '7. Klasse': 25,
    '8. Klasse': 28,
    '9. Klasse': 31
}

plt.pie(
    sec_school.values(),
    labels=sec_school.keys(),
    autopct='%1.1f%%',
    explode=[0, 0.25, 0, 0, 0.4]
)

plt.title("Klassenverteilung einer Grundschule")

plt.show()
```

Bonus - Meme




```python
from matplotlib.patches import Patch

colors = ['#0095D9','#F7EC59','#CBBE4A'] # Color Picker helps

plt.pie(
    [0.8, 0.15, 0.05], # Try and Error ratios
    colors=colors,
    startangle=-55
)

plt.legend(
    labels=["Sky", "Sunny side of pyramid", "Shady side of pyramid"],
    # Set right color and size for Patches
    handles=[Patch(facecolor=color) for color in colors],
    handlelength=1,
    handleheight=1,
    # Move Legend to the right
    loc="center left",
    bbox_to_anchor=(1, 0.5),
    fontsize=16,
    # Disable background and edge
    facecolor='none',
    edgecolor='none'
)

plt.tight_layout() # Layout shouldnt Overflow

plt.show()
```

Lösungen 5. SciPy

45 mögliche Punkte

Aufgabe – Lineare Regression

7 Punkte

Bestimme mittels Linearer Regression die **best fit** Funktion für die beiden gegebenen Datensets x_data & y_data, unter beachtung folgender Punkte:

- Plotte das Ergebnis angemessen
- Nutze SciPys linregress Funktion, speichere den Output vor dem entpacken in der Variablen l
- Definiere die Funktion reg_line mit einem Eingabeparameter
- Bestimme die Werte für -0.3 & 3.4 speichere diese als liste in variablen future

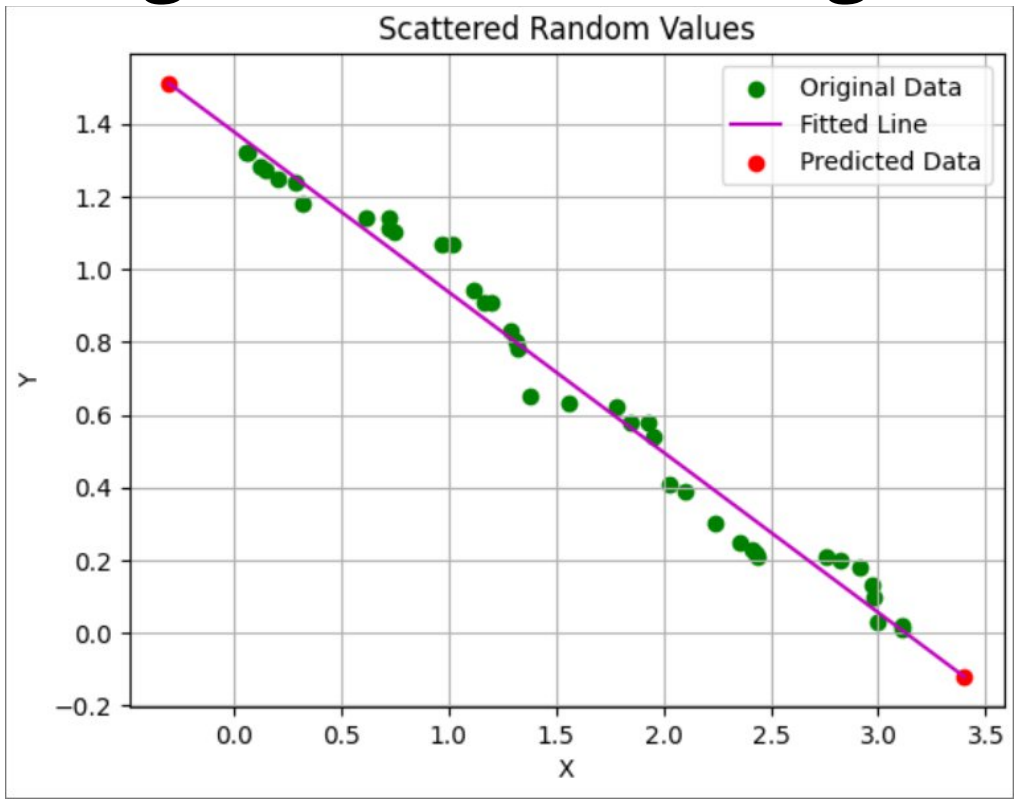
Aufgabe – Lineare Regression

```python
import numpy as np
random = np.random.default_rng(420)

# two scuffed up One-Liners :)
x_data: np.array = np.sort(np.round(random.random(40)*np.pi, decimals=2))
y_data: np.array = np.flip(np.sort(np.round(random.random(40)*np.sqrt(2),
decimals=2)))
```

Aufgabe – Lineare Regression

```python
slope, intercept, _, _, stderr = stats.linregress(x_data, y_data)

def reg_line(x: float) -> float:
    return float(np.round(slope*x+intercept, decimals=2))

ext: tuple = (-0.3, 3.4)
rl: np.array = np.vectorize(reg_line)(ext)

future: list = [reg_line(x) for x in ext]
```

Aufgabe – Lineare Regression



„Der Plot zeigt eine klar negative lineare Beziehung zwischen X und Y, obwohl die Datenpunkte zunächst zufällig verstreut wirken. Die Regressionslinie beschreibt diesen Trend präzise und fügt der scheinbaren Unordnung eine statistische Ordnung hinzu.“

```
plt.title("Scattered Random Values")
plt.grid()

plt.xlabel("X")
plt.ylabel("Y")

plt.scatter(
    x_data,
    y_data,
    color='g',
    label="Original Data"
)
plt.plot(
    ext,
    rl,
    color='m',
    label="Fitted Line"
)
plt.scatter(
    ext,
    future,
    color='r',
    label="Predicted Data"
)

plt.legend()
plt.show()
```

Aufgabe – Lineare Regression

Markdown Zeile (0 Punkte bei nicht beantworten):

- Erkläre die berechneten Werte `slope`, `intercept` & `stderr` in eigenen Worten und setze diese in den Kontext stochastischer Analytik.

- Interpretiere den resultierenden Plot nutze hierfür gerne Reale Beispiele.

Slope: Steigung der Geraden.
Intercept: Schnittwert mit der Y-Achse.
Standard Error: Streuung der Fehler. Je kleiner desto besser passt die Gerade.

Reales Beispiel: Stromverbrauch eines Computer Akkus.

Aufgabe – Geburtenschätzung Braunschweig

15 Punkte

Gegeben ist die Geburtenzahl insgesamt der Stadt Braunschweig geburten_bs. (Quelle: Landesamt für Statistik Niedersachsen (LSN))

Bestimme mittels Linearer Regression die Geburtenzahl für das Jahr 2030. Nutze dir die dafür gegebenen Mittel. Achte darauf auch eine angemessene Visualisierung zu finden.

Bestimme auch die Geburtenrate für das Jahr 2028 mittels:
*Birth Rate = B/P * 1000*

Mit:
- *B* = Zahl der Lebendgeborenen
- *P* = Bevölkerung zur Jahresmitte

Aufgabe – Geburtenschätzung Braunschweig

Nutze für *P* den aktuellen Wert aus dem Jahr 2025 *254.867*.
(Quelle: G 2.01 Entwicklung der Einwohnerzahl seit 1988)

Erkläre & Interpretiere in der Markdownzeile (Keine Antwort 0 Punkte) deine Berechnung, übe auch Kritik an der verwendeten Methodik aus.

```
geburten_bs = {
    2000: 2091, 2001: 1989,
    2002: 2016, 2003: 1946,
    2004: 1987, 2005: 1994,
    2006: 1986, 2007: 2126,
    2008: 2048, 2009: 2042,
    2010: 2193, 2011: 2154,
    2012: 2208, 2013: 2199,
    2014: 2300, 2015: 2410,
    2016: 2567, 2017: 2371,
    2018: 2474, 2019: 2434,
    2020: 2282, 2021: 2471,
    2022: 2328, 2023: 2066,
    2024: 2041,
}
```

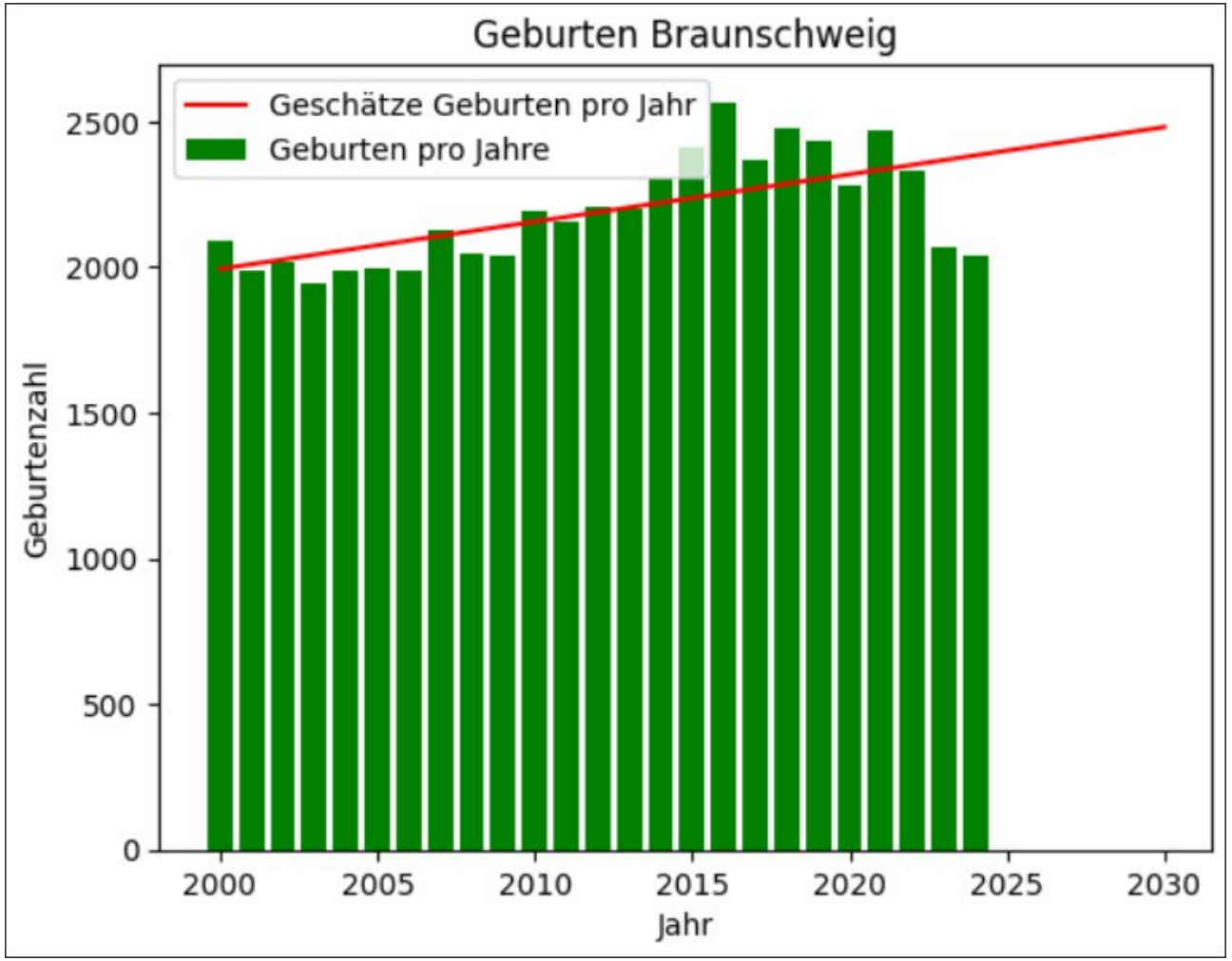
Aufgabe – Geburtenschätzung Braunschweig

```python
years = list(geburten_bs.keys())
births = list(geburten_bs.values())
slope, intercept, _, _, stderr = stats.linregress(years, births)

def reg_line(x: float) -> float:
    return float(np.round(slope*x+intercept, decimals=2))

births_2028 = reg_line(2028)
births_2030 = reg_line(2030)

p_2025 = 254867

birth_rate_2028 = births_2028 / p_2025 * 1000

print(births_2030)
print(birth_rate_2028)
```

Geschätze Geburten 2030:
2481.41

Geburtenrate 2028:
9.6

Aufgabe – Geburtenschätzung Braunschweig




```python
xs = np.linspace(min(years), 2030, 31)
ys = np.array(
    [reg_line(x) for x in range(2000,2031)]
)

plt.bar(
    years, births,
    label="Geburten pro Jahre",
    color="g"
)

plt.plot(
    xs, ys,
    label="Geschätze Geburten pro Jahr",
    color="r"
)

plt.title("Geburten Braunschweig")
plt.xlabel("Jahr")
plt.ylabel("Geburtenzahl")
plt.legend()

plt.show()
```

Aufgabe – Geburtenschätzung Braunschweig

„Die Berechnung extrapoliert die historischen Geburtenzahlen mittels linearer Regression und bestimmt daraus prognostizierte Werte für 2028 und 2030; die Geburtenrate 2028 ergibt sich durch Normierung der vorhergesagten Geburten auf die (konstant gesetzte) Bevölkerungszahl 2025. Interpretativ liefern die Resultate eine einfache Trendfortschreibung, jedoch ohne Aussage zu Unsicherheiten oder Modellgüte. Methodisch ist der Ansatz kritisch zu bewerten, da lineare Extrapolation demografische Dynamiken, potenzielle Nichtlinearitäten und strukturelle Brüche ausblendet und die Verwendung einer fixen Bevölkerungsbasis zusätzliche Verzerrungen erzeugt.“

Aufgabe – Normalverteilung Körpergrößen Frauen

20 Punkte

Gegeben sind die nach Altersgruppe aufgeschlüsselten Durschnittskörpergrößen (in cm) von Frauen in Deutschland. (Zu finden beim Statistischen Bundesamt)

```python
avg_height_per_woman = {
    "18 - 20": 167.6,
    "20 - 25": 167.7,
    "25 - 30": 167.3,
    "30 - 35": 167.2,
    "35 - 40": 167.3,
    "40 - 45": 167.5,
    "45 - 50": 167.1,
    "50 - 55": 167.1,
    "55 - 60": 166.9,
    "60 - 65": 165.4,
    "65 - 70": 164.5,
    "70 - 75": 163.9,
    "75+": 162.8
}

avg_height = None
norm_height = None
avg_percentile = None
```

Aufgabe – Normalverteilung Körpergrößen Frauen

Gehe wie folgt vor (8 Punkte):

- Berechne das arithmetische Mittel nutze dafür NumPy. und speichere das Ergebnis mit einer Genauigkeit von 1 Dezimalstelle nach dem Komma in der Variablen avg_height.

```
avg_height = np.round(
    np.mean(
        list(avg_height_per_woman.values())
    ),
        decimals=1
)
```

Aufgabe – Normalverteilung Körpergrößen Frauen

Gehe wie folgt vor (8 Punkte):

- Gegeben ist auch die Standardabweichung von 15cm, stelle die Normalverteilung mittels norm.pdf auf. Speichere den Wert in norm_height und finde einen geeigneten linespace zum plotten.

```
std_sigma = 15
norm_height = stats.norm(avg_height, std_sigma).pdf(norm_x)
norm_x = np.linspace(120, 220, 1000)
```

Aufgabe – Normalverteilung Körpergrößen Frauen

Gehe wie folgt vor (8 Punkte):

- Berechne folgend die Körpergröße unter die 80% aller Frauen (nach Datenset) fallen. Speichere den Wert in der Variablen avg_percentile.

```
avg_percentile = stats.norm(avg_height, std_sigma).ppf(0.8)

x_percentile = np.arange(norm_x[0], avg_percentile, 0.01)
y_percentile = stats.norm(avg_height, std_sigma).pdf(x_percentile)
```

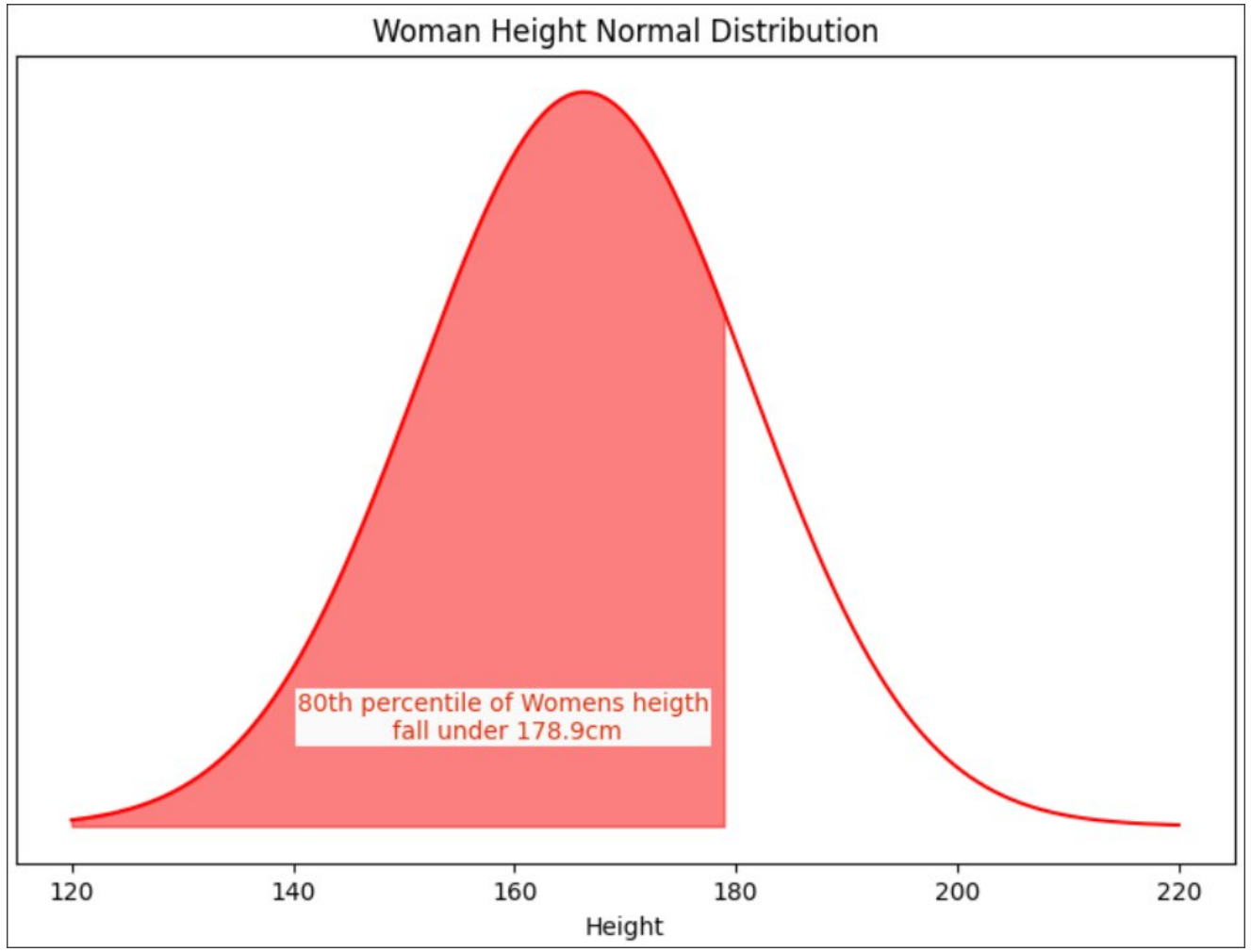
Aufgabe – Normalverteilung Körpergrößen Frauen

Gehe wie folgt vor (8 Punkte):

- Plotte das Ergebnis. Orientiere dich gerne an dem Bienenbeispiel. Finde eine geeignete Darstellung. Tipp: Da die Y-Achse in diesem Beispiel keinen Sinn ergibt kannst du sie einfach austellen mit plt.yticks([])

```python
# Plot
fig, ax = plt.subplots(figsize=(9,6))
ax.plot(norm_x, norm_height, color='r')

# filling under the curve
ax.fill_between(x_percentile, y_percentile, 0, alpha=.5, color='#fa0000')


# Set text
ax.text(0.4,0.18,
        f"80th percentile of Womens heigth\n fall under
{avg_percentile:.1f}cm",
        ha='center', va='center', transform=ax.transAxes,
        bbox={'facecolor':'#fafafa','alpha':1,'edgecolor':'none','pad':1},
        color='#de2e0b'
    )

# Show
plt.title("Woman Height Normal Distribution")
plt.xlabel("Height")
plt.yticks([]) # hide y
plt.show()
```

Aufgabe – Normalverteilung Körpergrößen Frauen



Aufgabe – Normalverteilung Körpergrößen Frauen

Markdown Zeile (0 Punkte bei nicht beantworten) (12 Punkte):

- Beschreibe die gegebenen Daten, nutze dazu die verlinkte und weitere Quellen. (Quellenangaben nicht vergessen)

- Beschreibe den resultierenden Plot und setze diesen in Kontext zum Datenset.

- Stelle die Annahmen der Aufgabe in Frage und setze diese in Kontext.

- Beurteile die Aussage aus resultierender Berechnung: "Frauen sind im durchschnitt ungeeignet Basketball zu spielen!"

Aufgabe – Normalverteilung Körpergrößen Frauen

Das Datenset enthält **durchschnittliche Körpergrößen (in cm) von Frauen in Deutschland** aufgeschlüsselt nach **Altersgruppen** basierend auf dem **Mikrozensus des Statistischen Bundesamts** (Erhebungsjahr 2021/2022). Die Werte repräsentieren Mittelwerte der Körpergröße für jede Altersklasse.

Der Plot zeigt eine Normalverteilung basierend auf dem Durchschnitt der über die Altersgruppen gemittelten Werte – allerdings ohne Berücksichtigung der tatsächlichen Bevölkerungsverteilung. Unterhalb der Verteilung ist zudem markiert, unter welcher Körpergröße gemäß dieser Berechnung rund 80 % der weiblichen Bevölkerung liegen würden.

Aufgabe – Normalverteilung Körpergrößen Frauen

Die Aufgabenstellung impliziert, dass die angegebenen Durchschnittswerte hinreichend repräsentativ sind. Allerdings fehlen sowohl Streuungsmaße als auch Fehlerwerte, was die Aussagekraft einschränkt. Besonders die Annahme einer Standardabweichung von 15 cm ist ohne entsprechende Quellenangabe methodisch fragwürdig.

Die Schlussfolgerung ist wissenschaftlich nicht haltbar. Basketballleistung ergibt sich aus einer Vielzahl relevanter Faktoren, von denen die Körpergröße lediglich einer ist. Alters- und Gesundheitsmerkmale sowie technische und taktische Fähigkeiten bleiben in der betrachteten Aussage vollständig unberücksichtigt.

Daraus ergibt sich, dass die Aussage auf Grundlage der vorhandenen Datenlage nicht belastbar ist.

Lösungen 6. Monte Carlo Simulationen

40 mögliche Punkte

Aufgabe – Arbeitszeit schätzen

15 Punkte

Du befindest dich erneut in der Situation, dass dein Chef dich beauftragt, zwei wichtige Aufgaben bis zum Ende des Arbeitstages zu bearbeiten. Gleichzeitig bist du abends auf einem Grillfest mit deinen Freunden verabredet, das um 18 Uhr beginnt.

Da du nun im Zwiespalt stehst, sowohl deinen beruflichen Aufgaben nachzukommen als auch das Grillfest pünktlich zu erreichen, fragst du dich, wie hoch die Wahrscheinlichkeit ist, dass du es in den nächsten 9 Stunden schaffst, beides unter einen Hut zu bringen.

Aufgabe – Arbeitszeit schätzen

Nach einigen Überlegungen stellst du Folgendes fest:

- Für die erste Aufgabe benötigst du zwischen 1–5 Stunden.
- Für die zweite Aufgabe benötigst du zwischen 2–6 Stunden.
- Egal wie schnell du eine Aufgabe erledigst, es gibt keine Auswirkungen auf die andere Aufgabe (die Aufgaben sind unabhängig voneinander).

Nun sollst du die Wahrscheinlichkeit berechnen, dass du pünktlich beim Grillfest ankommst.

Aufgabe – Arbeitszeit schätzen

Gehe dazu wie folgt vor:

1. Nimm an, dass beide Aufgaben gleichmäßig verteilt sind, und speichere die Verteilungen in den Variablen exc1 & exc2.

2. Verwende eine geeignete Anzahl an Samples und speichere diese in der Variablen sims.

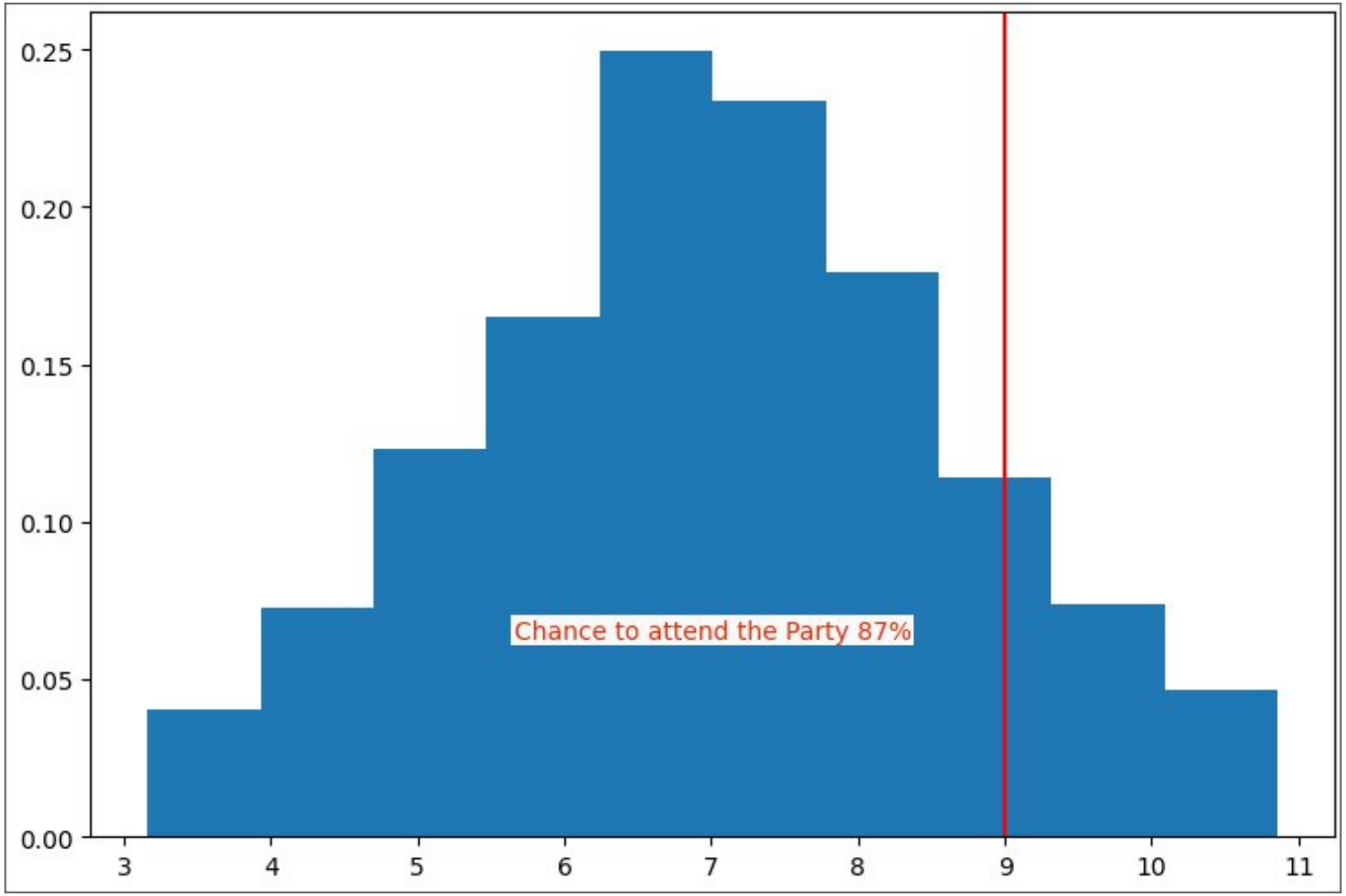
3. Speichere die Wahrscheinlichkeit, dass du pünktlich zum Grillfest kommst, in der Variablen chance mit einer Genauigkeit von zwei Nachkommastellen.

4. Plotte die Verteilungen der Aufgabenzeiten und beziehe dabei die Variable chance ein, z.B. als Annotation oder Beschriftung.

Aufgabe – Arbeitszeit schätzen

```python
import numpy as np
import matplotlib.pyplot as plt

rand = np.random.default_rng(42) # Use this rng!
sims = 1_000 # 1 Punkt

# 2 Punkte
exc1 = rand.uniform(1,5,sims)
exc2 = rand.uniform(2,6,sims)

duration = exc1 + exc2 # 1 Punkt
chance = float(np.round((duration > 9).sum()/sims, decimals=2)) # 1 Punkt

fig, ax = plt.subplots(figsize=(9,6))
ax.hist(duration, density=True) # 1 Punkt
ax.axvline(9, color='r') # 1 Punkt
ax.text(0.5,0.25,
        f"Chance to attend the Party {(1-chance)*100:.0f}%",
        ha='center', va='center', transform=ax.transAxes,
        bbox={'facecolor':'#fafafa','alpha':1,'edgecolor':'none','pad':1},
        color='#de2e0b'
    )
plt.show()
```

Aufgabe – Arbeitszeit schätzen



Aufgabe – Arbeitszeit schätzen

In der Markdownzelle (Keine Antwort = 0 Punkte):
- Stelle dein Ergebnis begründet dar.
- Interpretiere deinen Plot.
- Erkläre in eigenen Worten, warum dein Ergebnis aussagekräftig ist.

„Die Simulation zeigt bei 1000 Durchläufen, dass die Wahrscheinlichkeit, die Party rechtzeitig zu erreichen, bei 87 % liegt. Die Arbeitszeiten wurden hierfür mittels Monte-Carlo-Verfahren gleichverteilt gesampelt. Im Plot ist erkennbar, dass sich die resultierenden Gesamtzeiten einer Normalverteilung annähern. Die 9-Stunden-Marke ist klar gekennzeichnet und verdeutlicht, dass eine realistische Chance besteht, das Zeitlimit einzuhalten.

Das Ergebnis ist insofern aussagekräftig, als die zugrunde liegenden Annahmen plausibel gewählt sind. Die Simulation durchläuft eine Vielzahl möglicher zukünftiger Abfolgen von Ereignissen – eine davon wird letztlich eintreten.“

Aufgabe – Simulation Körpergewichtszunahme

25 Punkte

Gegeben sind die nach Altersgruppen aufgeschlüsselten Durchschnittskörpergewichte (in kg) von Männern in Deutschland.
Quelle: Statistisches Bundesamt

Ziel ist es, mittels Monte-Carlo-Simulation den mittleren wöchentlichen Gewichtszuwachs zu bestimmen und die Wahrscheinlichkeit zu finden, dass ein Durchschnittsmann innerhalb einer Woche 3 kg abnimmt.

```
avg_weight_per_men = {
    "18 - 20": 77.9,
    "20 - 25": 80.5,
    "25 - 30": 83.3,
    "30 - 35": 85.6,
    "35 - 40": 86.7,
    "40 - 45": 88.1,
    "45 - 50": 89.8,
    "50 - 55": 89.0,
    "55 - 60": 88.8,
    "60 - 65": 87.9,
    "65 - 70": 86.7,
    "70 - 75": 85.3,
    "75+": 81.0
}
```

Aufgabe – Simulation Körpergewichtszunahme

Vorgehensweise:

1. Bestimme das arithmetische Mittel der Durchschnittsgewichte über die Altersgruppen und speichere diesen Wert in avg_weight mit einer Genauigkeit von 1 Dezimalstelle.
2. Wähle eine geeignete Anzahl an Samples (sims) für die Simulation.
3. Nimm an, dass das Durchschnittsgewicht normalverteilt ist, mit einer Standardabweichung von 5% des Durchschnittsgewichts, und speichere die simulierten Werte in men_normal.

Aufgabe – Simulation Körpergewichtszunahme

```python
rand = np.random.default_rng(420)

avg_weight = np.round(
    np.mean(
        list(avg_weight_per_men.values()
    )), decimals=1
    )

sims = 1_000

sigma = np.round(avg_weight * 0.05, decimals=1)
men_normal = rand.normal(avg_weight, sigma, sims)
```

Aufgabe – Simulation Körpergewichtszunahme

Vorgehensweise:

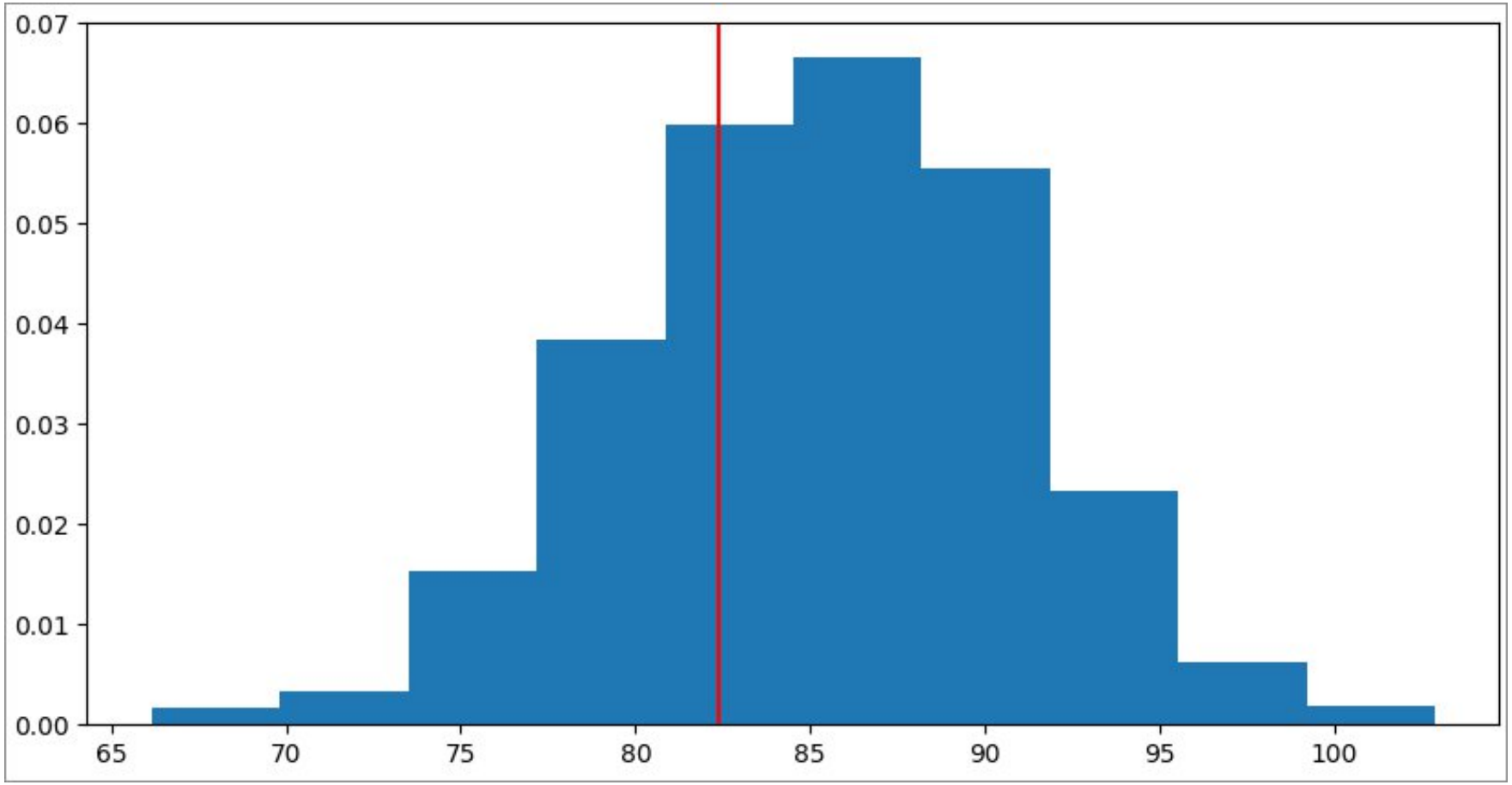
4. Nimm an, dass die tägliche Gewichtsschwankung eines Mannes gleichverteilt zwischen ±2.5 kg ist.
5. Simuliere die Gewichtszunahmen/-abnahmen für eine Woche (7 Tage) und speichere die Ergebnisse in gain_week.
6. Berechne die durchschnittliche wöchentliche Gewichtszunahme/-abnahme und speichere das Ergebnis in duration.
7. Bestimme die Wahrscheinlichkeit, dass ein Mann 3 kg oder mehr abnimmt, und speichere sie in gain_percent.
8. Plotte die Verteilung der simulierten Werte und markiere die relevanten Kennzahlen (Durchschnitt, Schwankungsbereich, Wahrscheinlichkeit).

Aufgabe – Simulation Körpergewichtszunahme

```
gain_week = [
    rand.uniform(-2.5, 2.5, sims)
    for _ in range(7)
]


duration = np.zeros(sims)
for gain in gain_week:
    duration += gain
duration += men

gain_percent = float(
    np.round((duration < avg_weight-3).sum()/sims, decimals=2)
    )

plt.figure(figsize=(10,5))
plt.hist(duration, density=True)
plt.axvline(avg_weight-3, color='r')
plt.show()
print(gain_percent)
```

gain_percent = 0.31
31%

Aufgabe – Simulation Körpergewichtszunahme



Aufgabe – Simulation Körpergewichtszunahme

In der Markdownzelle (Keine Antwort = 0 Punkte):
- Bewerte & Beurteile die gegebenen Daten, sowie Quelle.

- Erkläre & Interpretiere deinen Plot.

- Bewerte die angewandte Methodik im Zusammenhang mit der Aufgabe, beziehe dein Ergebnis anschaulich mit ein.

- Beurteile anhand deiner Simulation den Bild Artikel - "Ehe macht dick – vor allem die Männer"

Aufgabe – Simulation Körpergewichtszunahme

„Die Daten entstammen dem Statistischen Bundesamt und bilden das durchschnittliche Körpergewicht von Männern über verschiedene Altersgruppen hinweg ab. Sie erscheinen plausibel, da sie den erwartbaren Trend eines mit zunehmendem Alter ansteigenden Körpergewichts widerspiegeln.“

„Der Plot zeigt die über einen Zeitraum von einer Woche akkumulierten Gewichtsschwankungen. Die rote Linie markiert den Bereich ab −3 kg. Insgesamt weist die Verteilung eine annähernd normale Form auf.“

Aufgabe – Simulation Körpergewichtszunahme

„Die Monte-Carlo-Methodik erweist sich hier als sehr effektiv, da sie die Summe vieler einzelner Zufallsereignisse über einen Zeitraum von sieben Tagen kombiniert. Das Ergebnis ist aussagekräftig, da mit 1000 Simulationen eine solide statistische Basis vorliegt. Die Simulation zeigt, dass eine extreme Gewichtsabnahme von 3 kg allein durch zufällige Schwankungen vergleichsweise selten auftritt (ca. 31 %).“

„Die Simulation zeigt, dass das Körpergewicht im Mittel stabil bleibt. Die Aussage „Ehe macht dick“ lässt sich allein auf Grundlage dieser biologischen Zufallsschwankungen nicht belegen, da die Simulation keine systematische Verschiebung der Verteilung in Richtung Gewichtszunahme erkennen lässt.“

Lösungen 7. Pandas & Seaborn

25 mögliche Punkte

Aufgabe – Erstellen eines Dataframes

2 Punkte

Erstelle einen Pandas Data Frame mit dem namen *uni_addr*, nachdem Schema folgender Tabelle:

Address	Plz
Johannes-Selenka-Platz 1	38118 Braunschweig
Universitätspl. 2	38106 Braunschweig
Harburger Str. 6	21614 Buxtehude
Adolph-Roemer-Straße 2A	38678 Clausthal-Zellerfeld
Constantiapl. 4	26723 Emden
Weender Landstraße 3-7	37073 Göttingen
Wilhelmsplatz 1	37073 Göttingen

Aufgabe – Erstellen eines Dataframes

```
uni_addr = pd.DataFrame({
    "Address": [
        "Johannes-Selenka-Platz 1", "Universitätspl. 2",
        "Harburger Str. 6", "Adolph-Roemer-Straße 2A",
        "Constantiapl. 4", "Weender Landstraße 3-7",
        "Wilhelmsplatz 1",
        ],
    "plz": [
        "38118 Braunschweig", "38106 Braunschweig",
        "21614 Buxtehude", "38678 Clausthal-Zellerfeld",
        "26723 Emden", "37073 Göttingen",
        "37073 Göttingen",
        ]
})
```

Aufgabe – Extrahieren einer Series

1 Punkte

Exthahiere die Series *plz* aus dem zuvor erstelltem Data Frame *uni_addr* und speicher dein Ergebnis in *uni_plz* mit dem richtigen Type Hint.

```
uni_plz: pd.Series = uni_addr["plz"]
```

Aufgabe – Read CSV

1 Punkte

Lies das Datenset *unis_nd.csv* aus und speicher den resultierenden DataFrame in der Variabeln *unis_nd*.

Falls hilfe benötigt, lese gerne die Dokumentation im Getting Started Guide.

Hinweis: Die Datei liegt in keinem Ordner, sondern im selben wie dieses Notebook!

```
unis_nd: pd.DataFrame = pd.read_csv("unis_nd.csv")
```

Aufgabe – Read CSV

2 Punkte

Selektiere die Spalten *University name*, *Founding year* & *Number of students*, speichern sie ihr Ergebnis in der Variablen *select*.

Gebe danach die ersten 5 Werte von Oben der Selektion aus.

```
select: pd.DataFrame = unis_nd[[
    "University name",
    "Founding year",
    "Number of students"
]]
select.head(5)
```

Aufgabe – A Plot

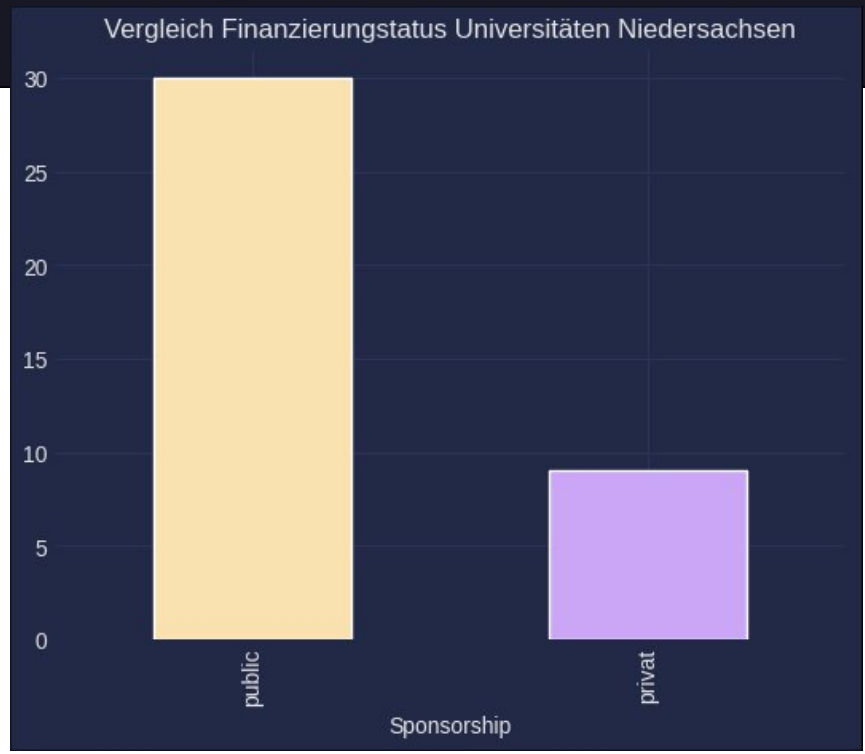
6 Punkte

Erstelle eine Visualisierung, welche die Unterschiede zwischen staatlichen & privaten Hochschulen in Niedersachsen zeigt.

Die dazugehörige Spalte im Datenset ist *Sponsorship*. Finde eine geeignete Darstellung.

Erkläre anschließend deine Visualisierung und warum du dich für diese Entschieden hast.
(0 Punkte bei keiner Erklärung)

```python
# Hacky One-Liner
unis_nd['Sponsorship'].value_counts().plot(
        kind='bar',
        title="Vergleich Finanzierungstatus Universitäten Niedersachsen",
        color=["#f9e2af", "#cba6f7"]
    )
plt.show()
```



Aufgabe – People In Germany

8 Punkte

Erstelle das Dataset *people_in_germany*.

Folgendes Szenario:

Du bist Part einer Massenhaft angelegten Studie um die bereits bekannten Zahlen des Statistischen Bundesamtes zu überprüfen.

Dazu sind 4 Größen bekannt aus den Angaben des Statistischen Bundesamt für die männliche Population & die weibliche Population:

	Körpergröße (in cm)	Gewicht (in Kg)
Männlich	178.9	85.8
Weiblich	165.8	69.2

Aufgabe – People In Germany

Gehe dabei wie folgt vor:

- Treffe annahmen über die Verteilung und finde geeignete Werte für μ & σ. **Erkläre** deine Annahmen mit einem kurzen Text. Quellen sind gerne gesehen.
- Die Samplegröße beträgt 1000. Sample dementsprechend aus den gegebenen Werten.
- Speichere die gesampleten Personen nach dem Schema: **|sex|height|weight|** in der File *people_in_germany.csv* als csv.
  -> Nutze für das **sex** Attribut den Datentyp **bool** mit der Kodierung:
  True = female & *False = male*.
- Stelle dein Ergebnis angemessen dar. Das theme **darkgrid** darf nicht verwendet werden.
- **Beschreibe** & **Interpretiere** den Plot. Gehe dabei von der Hypothese aus das es **keine** Unterschiede zwischen den Geschlechtern beim samplen geben sollte. (Auch hier 0 Punkte bei keiner Antwort)

Aufgabe – People in Germany

```python
from dataclasses import dataclass

@dataclass
class Individual:
    sex: bool
    height: np.float64
    weight: np.float64
```

Sources:
Our World in Data
Nako Gesundheitsstudie (Springer)

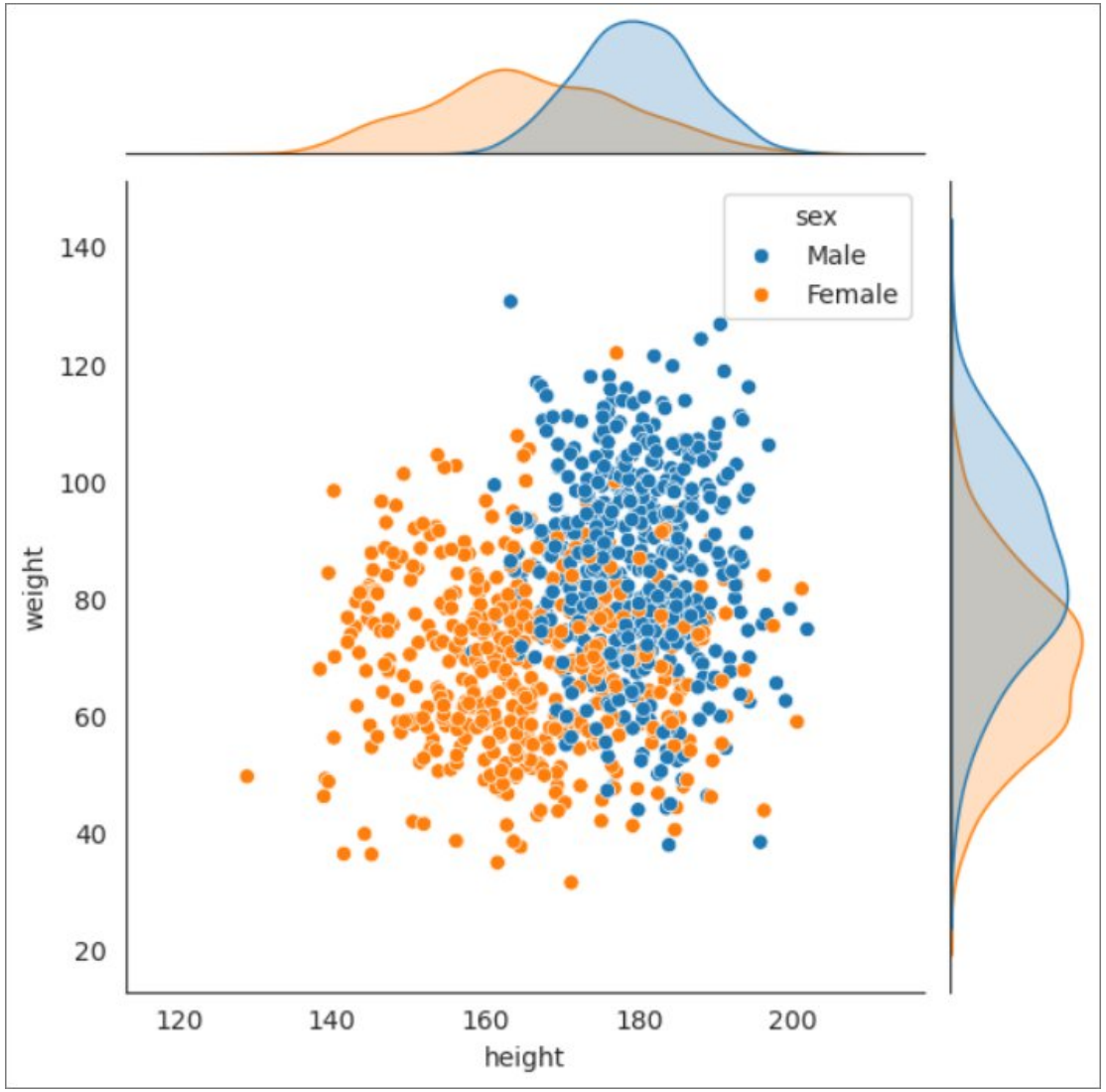
```python
rng = np.random.default_rng(42)
samples = 1000
individuals = list()

# Source: Destatis (2024)
mu_male_height = 178.9
mu_female_height = 165.8
mu_male_weight = 85.8
mu_female_weight = 69.2

# Source: Our World in Data
sigma_male_height = 7.6
sigma_female_height = 12.0

# Source: NAKO Gesundheitsstudie (2020)
sigma_male_weight = 15.1
sigma_female_weight = 14.6
```

Aufgabe – People in Germany

```python
# Samplen
for i in range(samples):
    if i % 2 == 0:
        sex = False
        height = rng.normal(mu_male_height, sigma_male_height)
        weight = rng.normal(mu_male_weight, sigma_male_weight)

    else:
        sex = True
        height = rng.normal(mu_female_height, sigma_female_height)
        weight = rng.normal(mu_female_weight, sigma_female_weight)

    height = np.round(height, decimals=1)
    weight = np.round(weight, decimals=1)

    individual = Individual(sex=sex,height=height,weight=weight)
    individuals.append(individual)
```

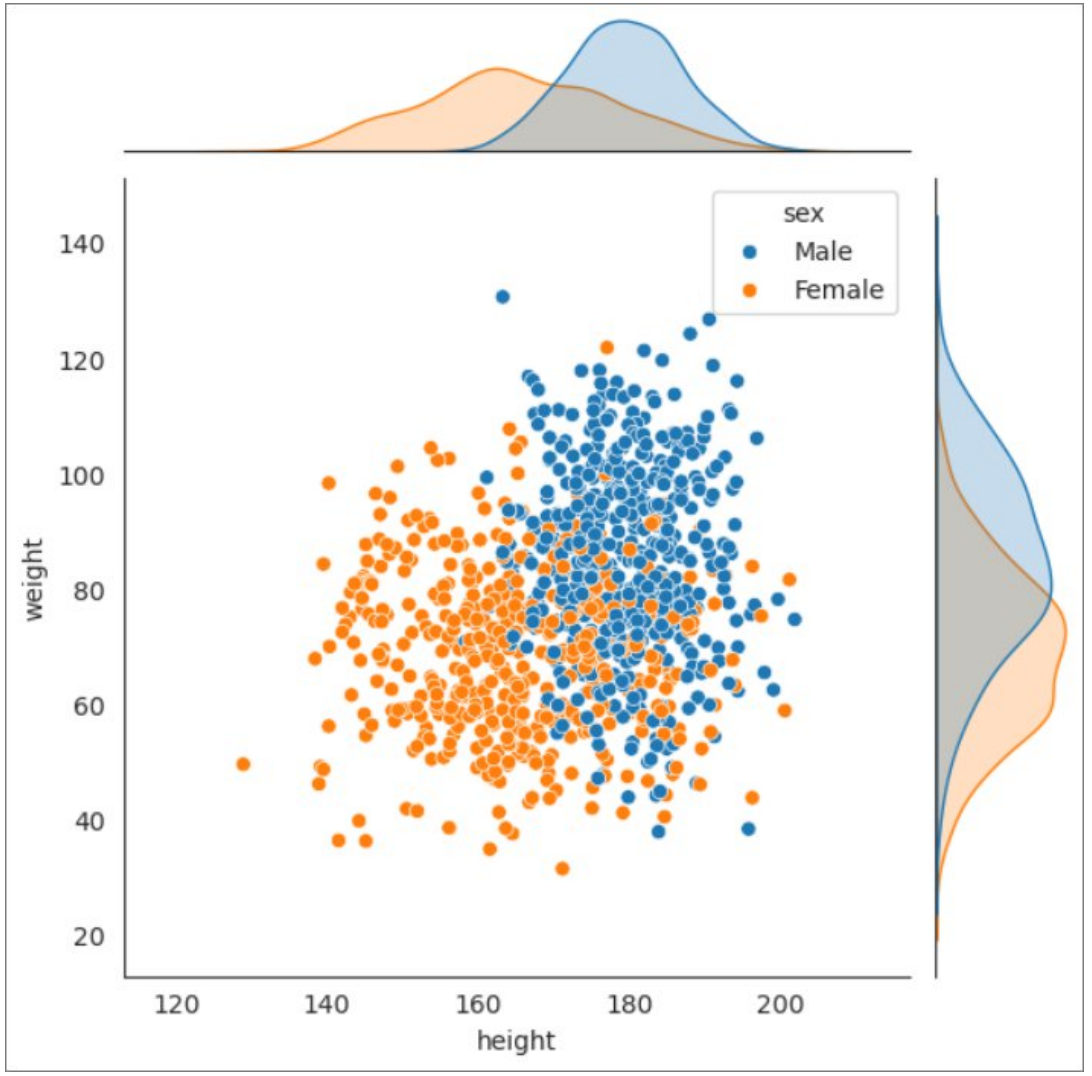
Aufgabe – People in Germany

```python
individuals = pd.DataFrame(individuals)
individuals.to_csv('people_in_germany.csv', index=False)

# Plot
sns.set_style('white')
sns.jointplot(
        data=individuals,
        x="height",
        y="weight",
        hue="sex"
    )
plt.show()
```



Aufgabe – People in Germany

- Der Scatterplot mit Randverteilungen zeigt eine deutliche Überlappung der Geschlechter, jedoch systematisch verschobene Verteilungen von Körpergröße & Gewicht.
- Männliche Samples weisen im Mittel höhere Werte für Größe & Gewicht auf als weibliche Beobachtungen.
- Unter der Nullhypothese eines geschlechtsneutralen Samplings wären vergleichbare Lageparameter und Verteilungen zu erwarten, was visuell nicht erfüllt ist.
- Die Darstellung liefert somit deskriptive Evidenz gegen die Annahme identischer Verteilungen zwischen den Geschlechtern.



Reasoning – Motivated Numeracy

Gedankenexperiment

Eine Studie mit 426 Teilnehmenden untersuchte die Wirksamkeit einer neuen Creme zur Behandlung von Hautausschlag.
Hier sind die Ergebnisse:

Frage: Hat die Creme den Ausschlag verbessert?

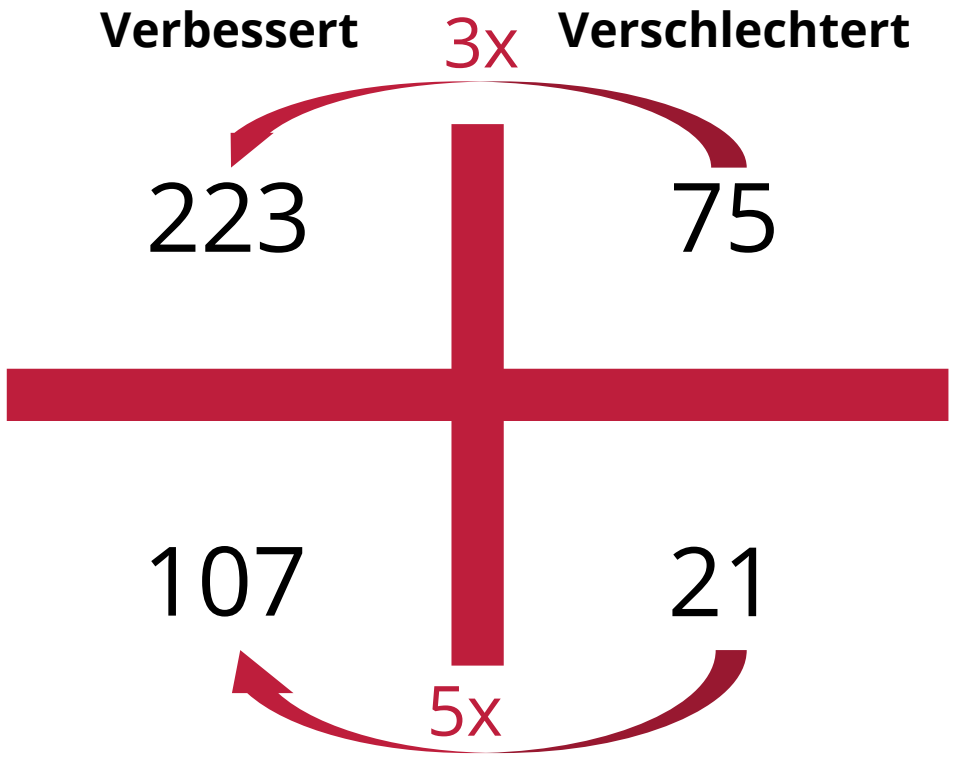
	Verbessert	**Verschlechtert**
	223	75
	107	21

Gedankenexperiment

Eine Studie mit 426 Teilnehmenden untersuchte die Wirksamkeit einer neuen Creme zur Behandlung von Hautausschlag.
Hier sind die Ergebnisse:

Frage: Hat die Creme den Ausschlag verbessert?

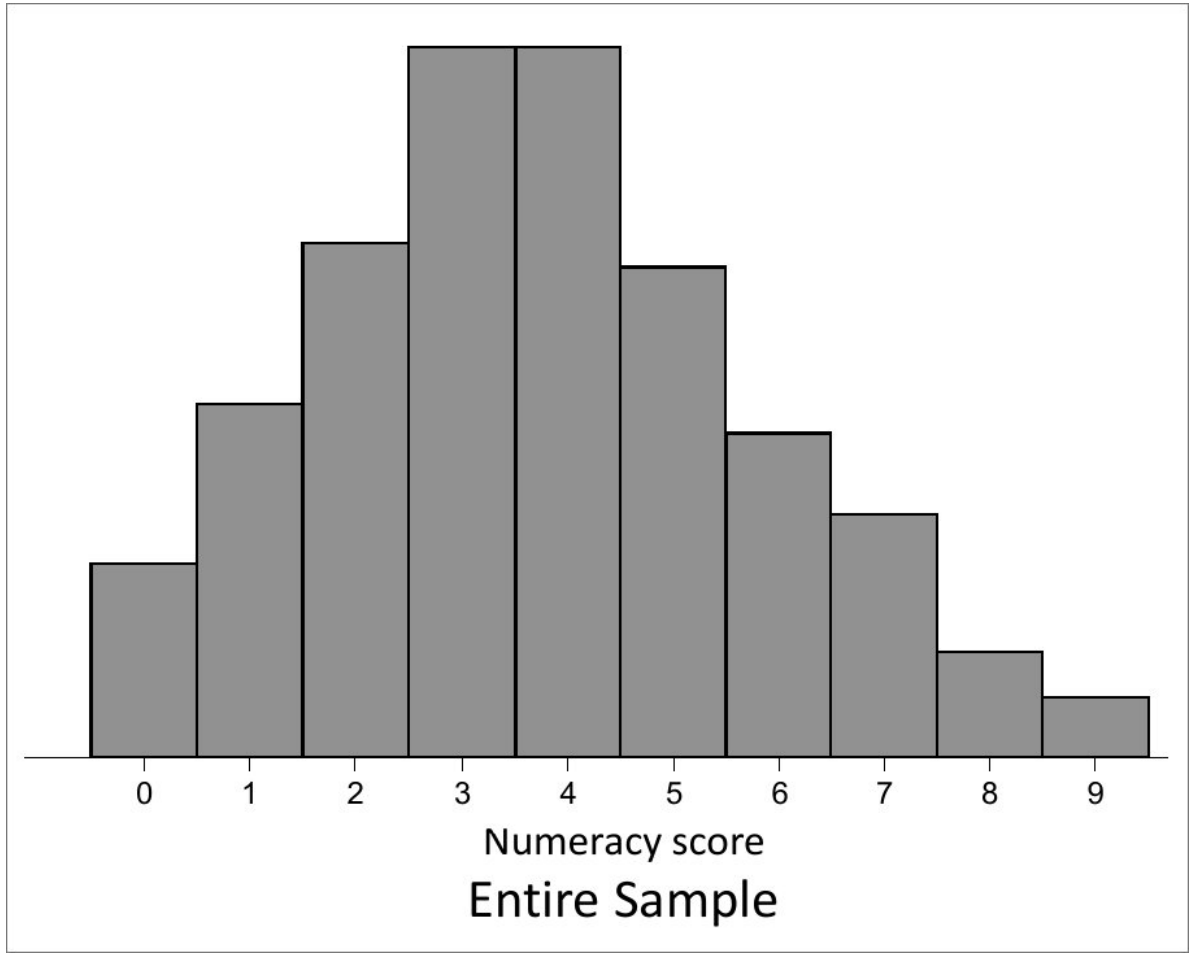
Nein!
Im Mittel verschlechtert sich sogar der Ausschlag.



Motivated Numeracy

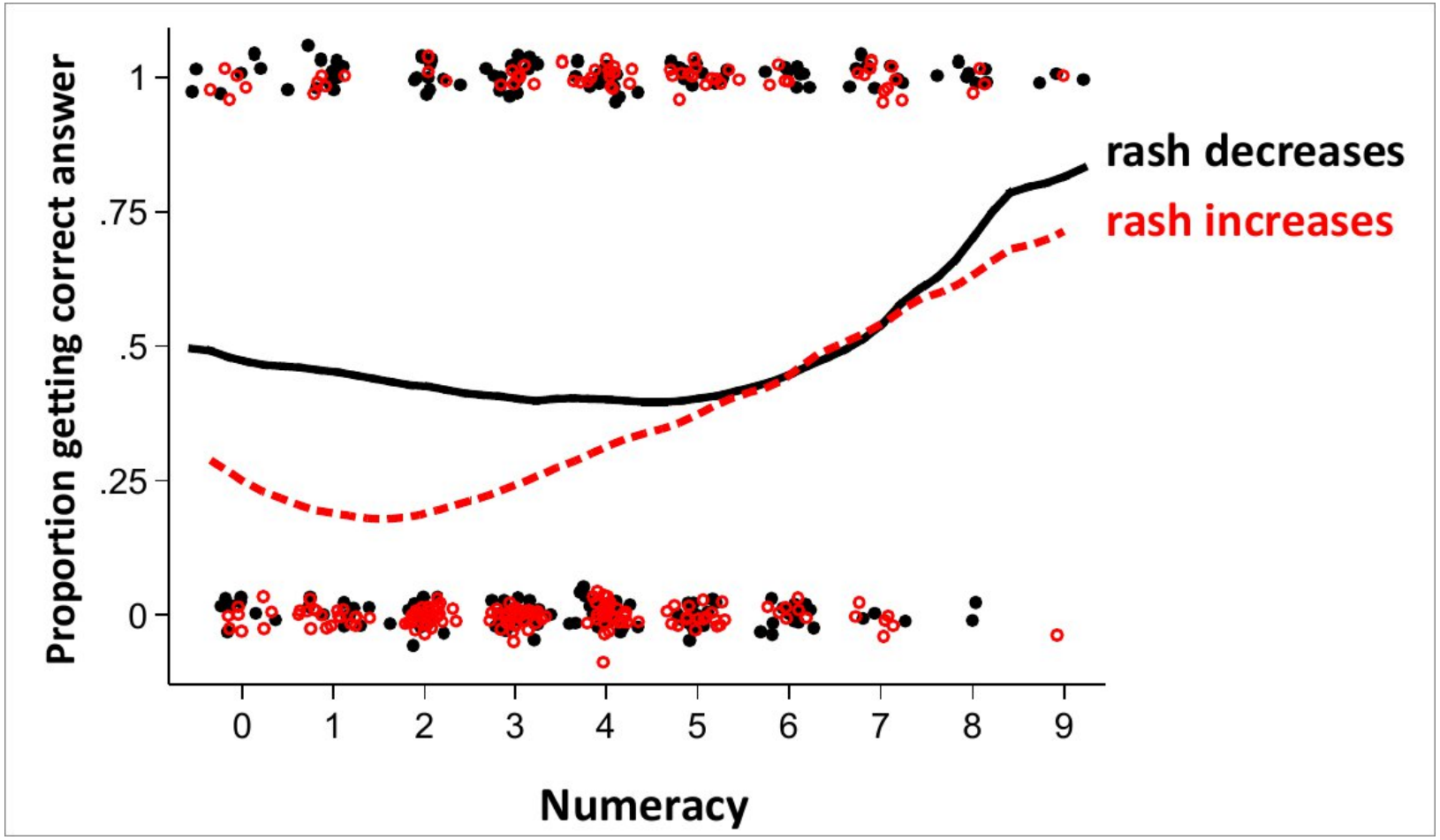
Quelle: Dan M. Kahan

- Er & seine Mitarbeiter haben selbige Frage 1,111 Teilnehmern in den USA gestellt.
- Vorher haben Sie jedoch die *Numeracy* (Rechenkompetenz) des Individuums ermittelt.
- **Numeracy**: Die Fertigkeit über Quantitative Größen zu urteilen.
- Welche Gruppe hat die Frage im Schnitt am meisten richtig beantwortet?



Motivated Numeracy

Wie (vielleicht) zu erwarten gibt es einen Zusammenhang zwischen einem hohen *Numeracy Score* und *richtiger Antwort.*



Gedankenexperiment (Abgewandelt)

Eine Studie mit 426 Teilnehmenden untersuchte die Sicherheit zu Silvester an verschiedenen Orten über die Verletzungsrate.
Hier sind die Ergebnisse:

Frage: Hat ein Böllerverbot zu weniger Verletzungen geführt?

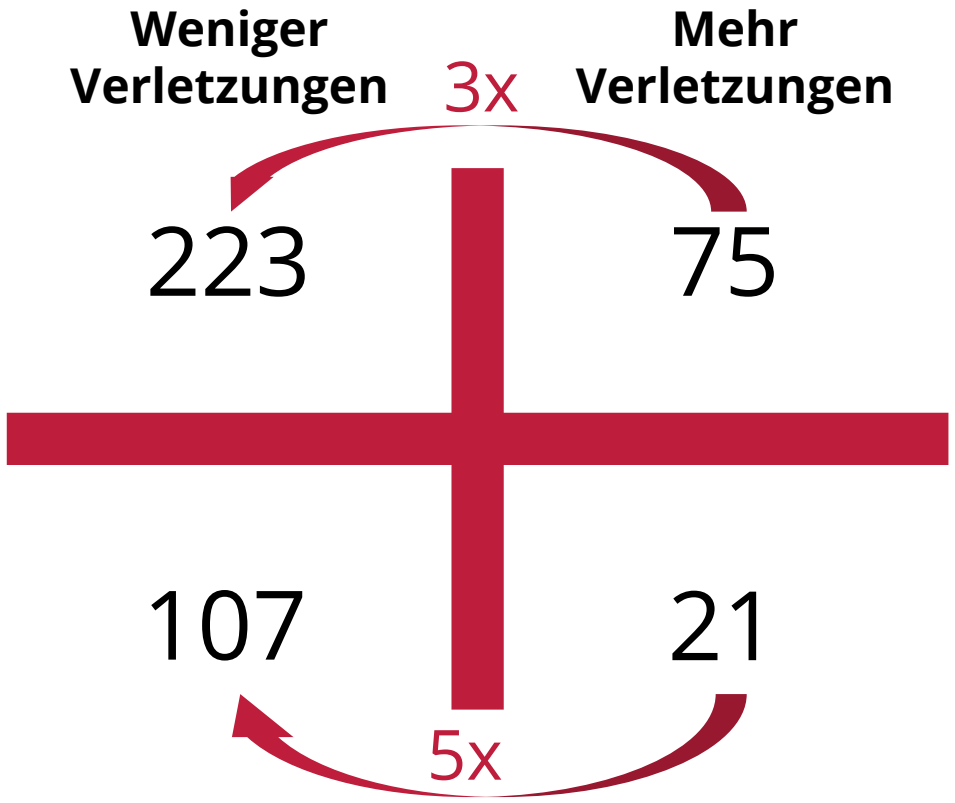
	Weniger Verletzungen	**Mehr Verletzungen**
	223	75
	107	21

Gedankenexperiment (Abgewandelt)

Eine Studie mit 426 Teilnehmenden untersuchte die Sicherheit zu Silvester an verschiedenen Orten über die Verletzungsrate.
Hier sind die Ergebnisse:

Frage: Hat ein Böllerverbot zu weniger Verletzungen geführt?

Ja!



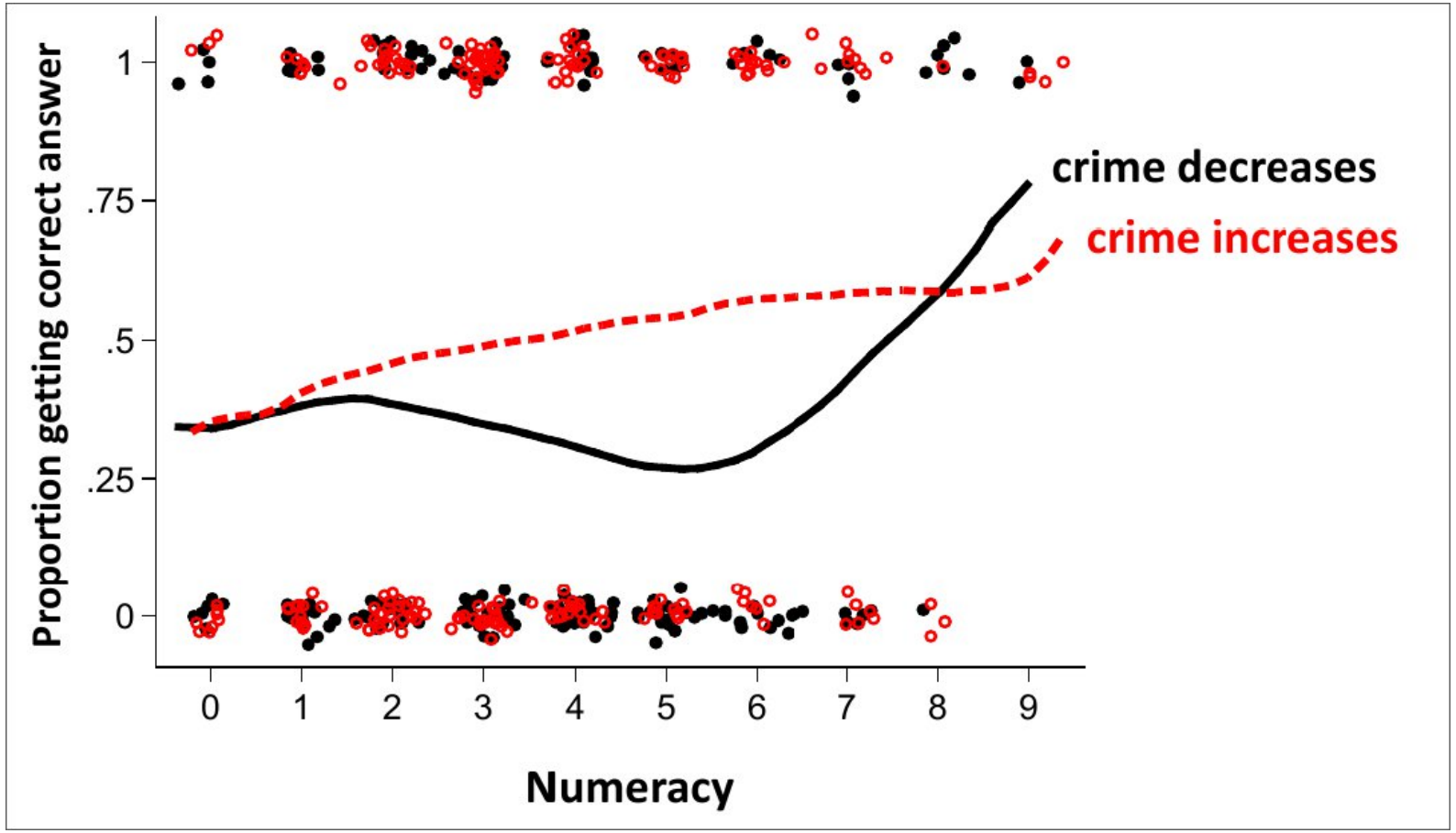
Gedankenexperiment (Ursprünglich)

Eine Studie mit 426 Teilnehmenden untersuchte die Kriminalitätsrate in deren Stadt. Hier sind die Ergebnisse:

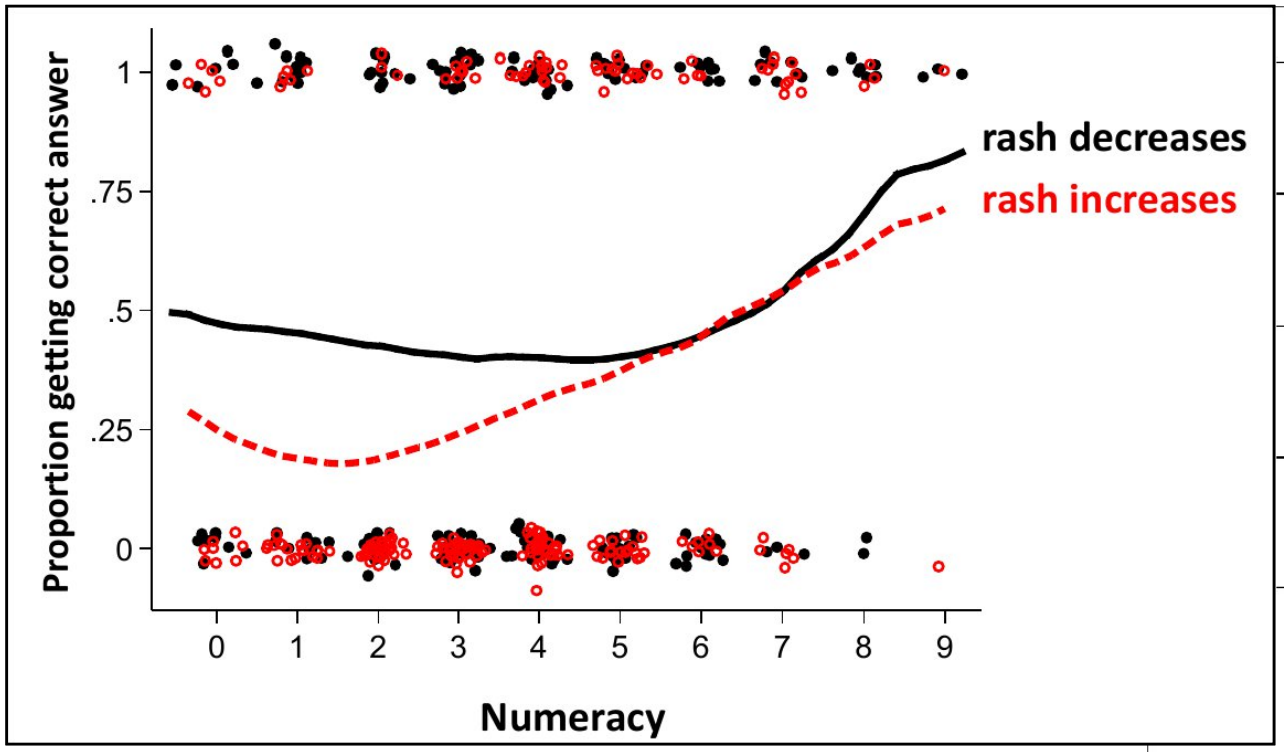
		Crime increased	Crime decreased
Frage: Hat ein Waffenverbot geholfen?		223	75
		107	21

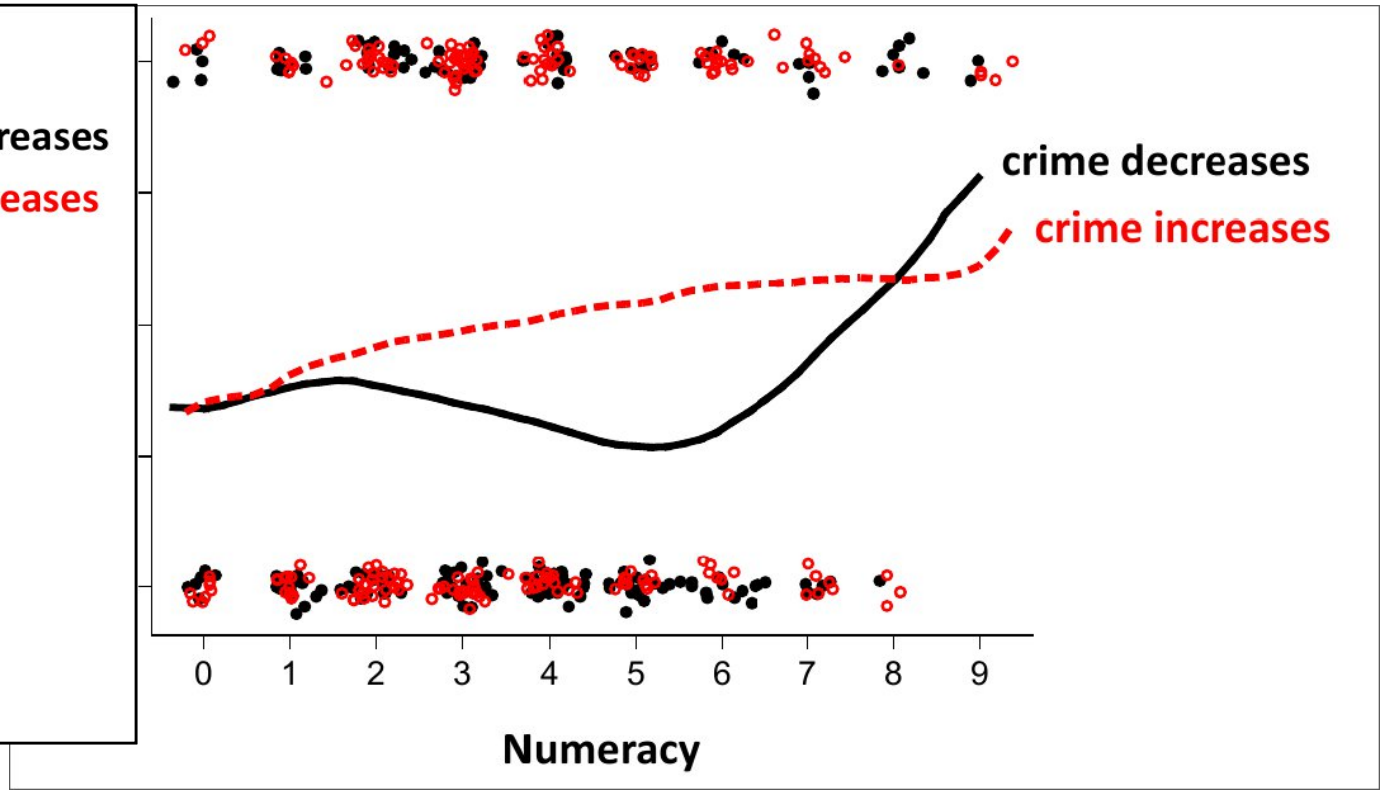
Motivated Numeracy

Kaum kommt Politik mit in die Befragung, fällt das Ergebnis dramatisch anders aus.



Motivated Numeracy

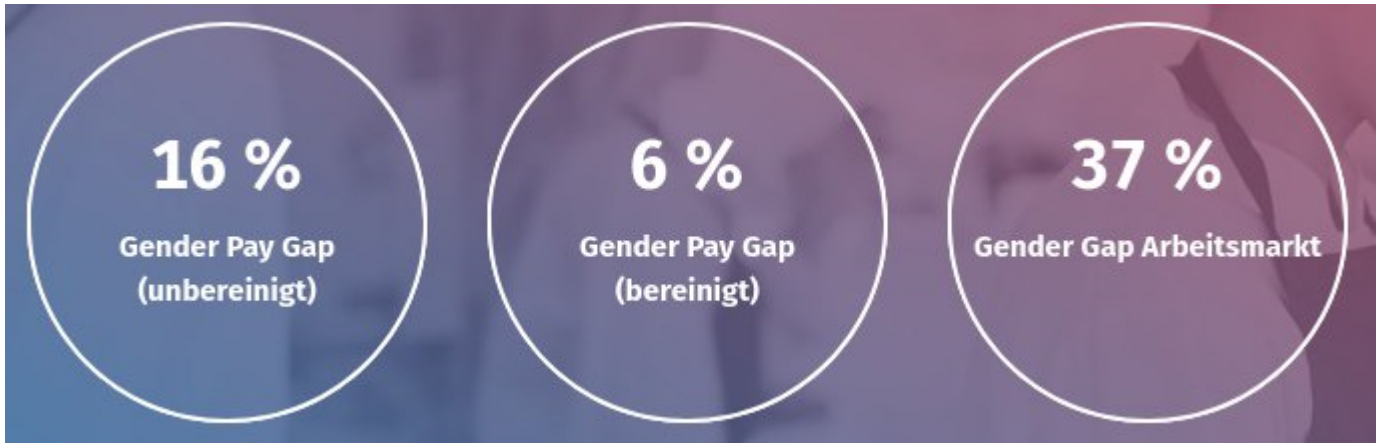




Gender Pay Gap

YouTube: Tagesschau (2019) - Equal Pay Day: Frauen verdienen weniger als Männer

Gender Pay Gap – Aktuellere Zahlen



Destatis – Gender Pay Gap

Was bedeutet in diesem Zusammenhang *unbereinigt* und *bereinigt* ?

Was könnten die Gründe für diese Lücken sein?

Gender Pay Gap – Definition & Berechnung

Unbereinigter Gender Pay Gap:

Wird ermittelt über die Funktion:

$$\frac{\text{Ø Bruttoverdienst Männer} - \text{Ø Bruttoverdient Frauen}}{\text{Ø Bruttoverdienst Männer}} * 100$$

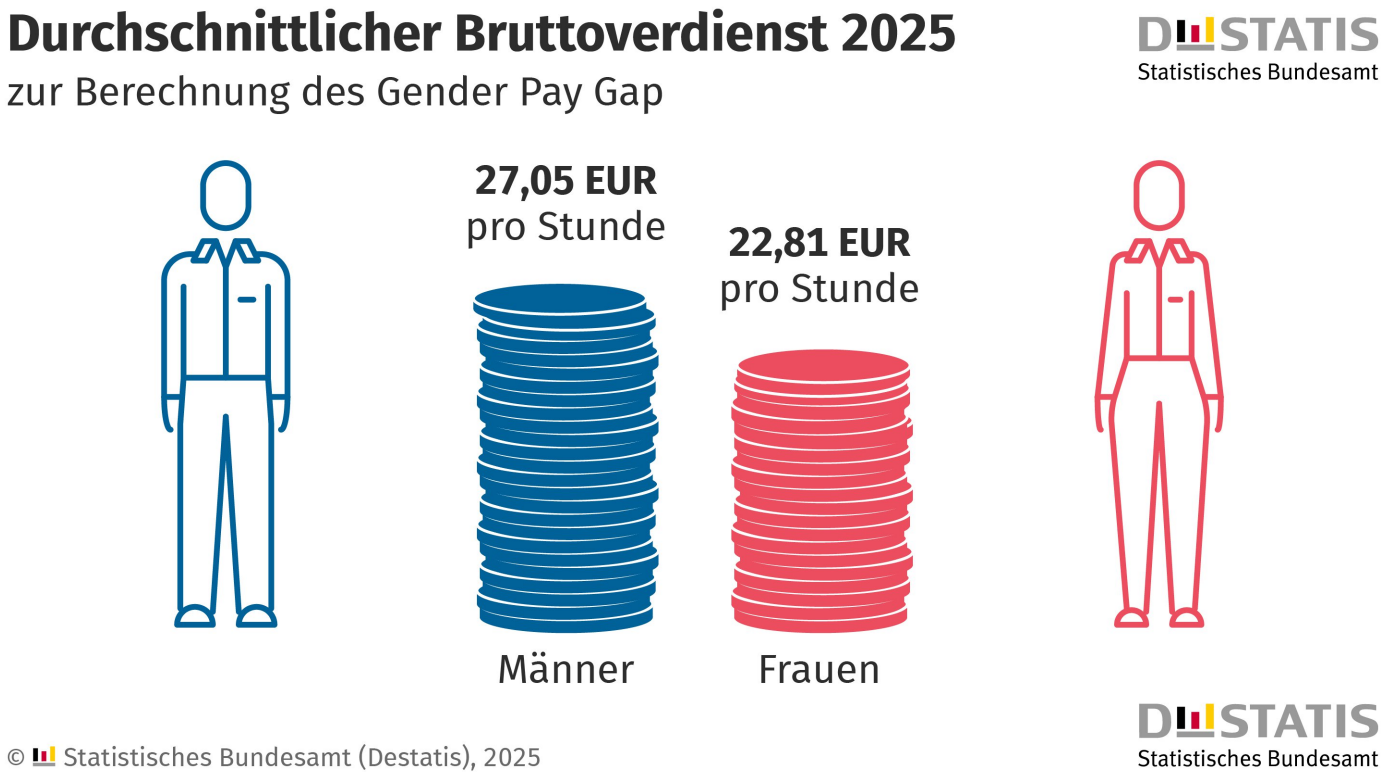
Was kann mit dieser Berechnung nicht Ausgesagt werden?

Bereinigter Gender Pay Gap:

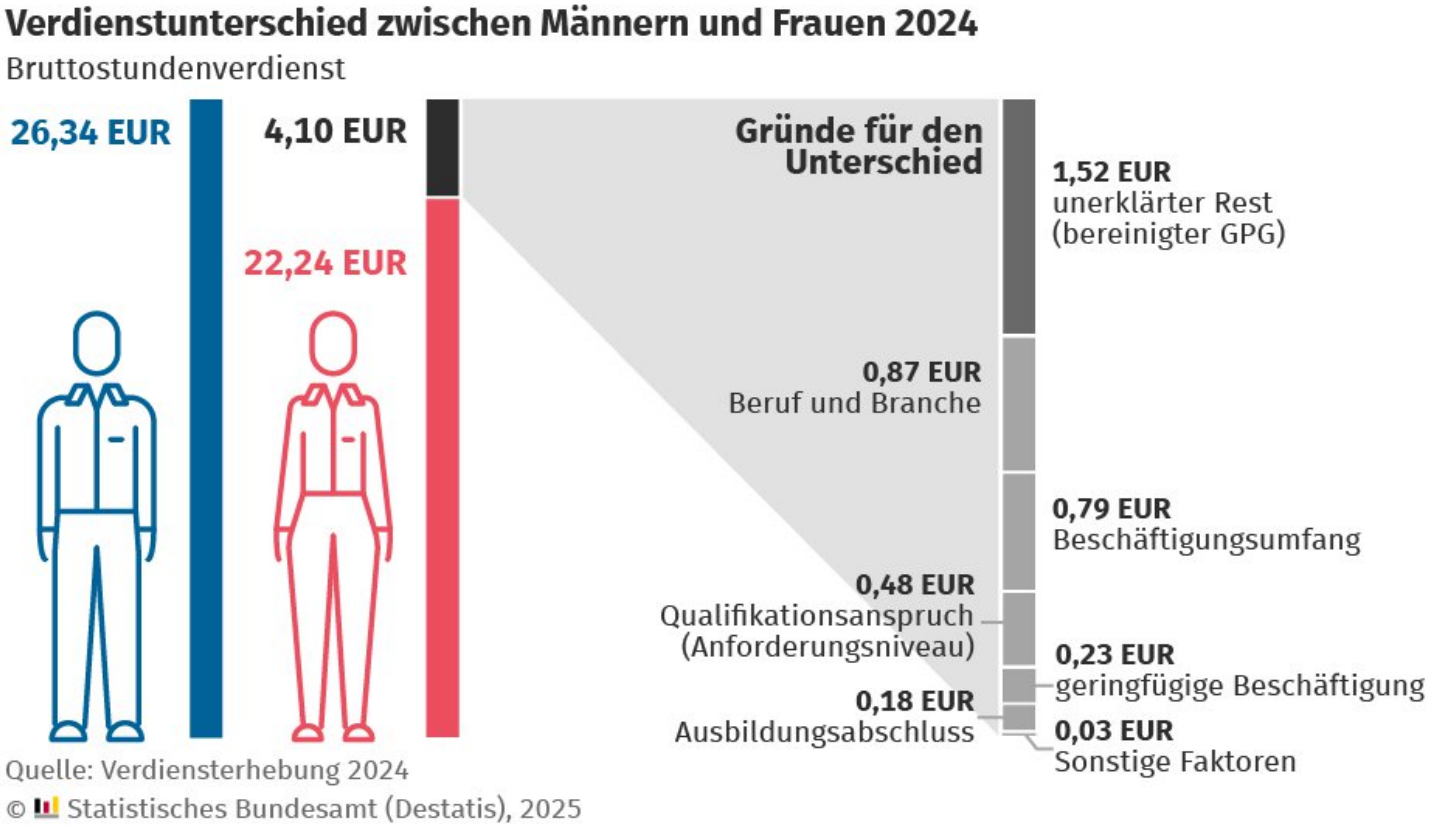
- Basis ist die vierteljährliche Verdienststruckturerhebung
- Jeder nicht auf strukturelle Unterschiede (Beruf, Branche, Beschäftigungsumfang, etc.) wird herausgerechnet
- Angaben zur Erwerbsunterbrechung (Bsp. Elternzeit) fehlen

Bereinigter GPG ≠ Verdienstdiskriminierung

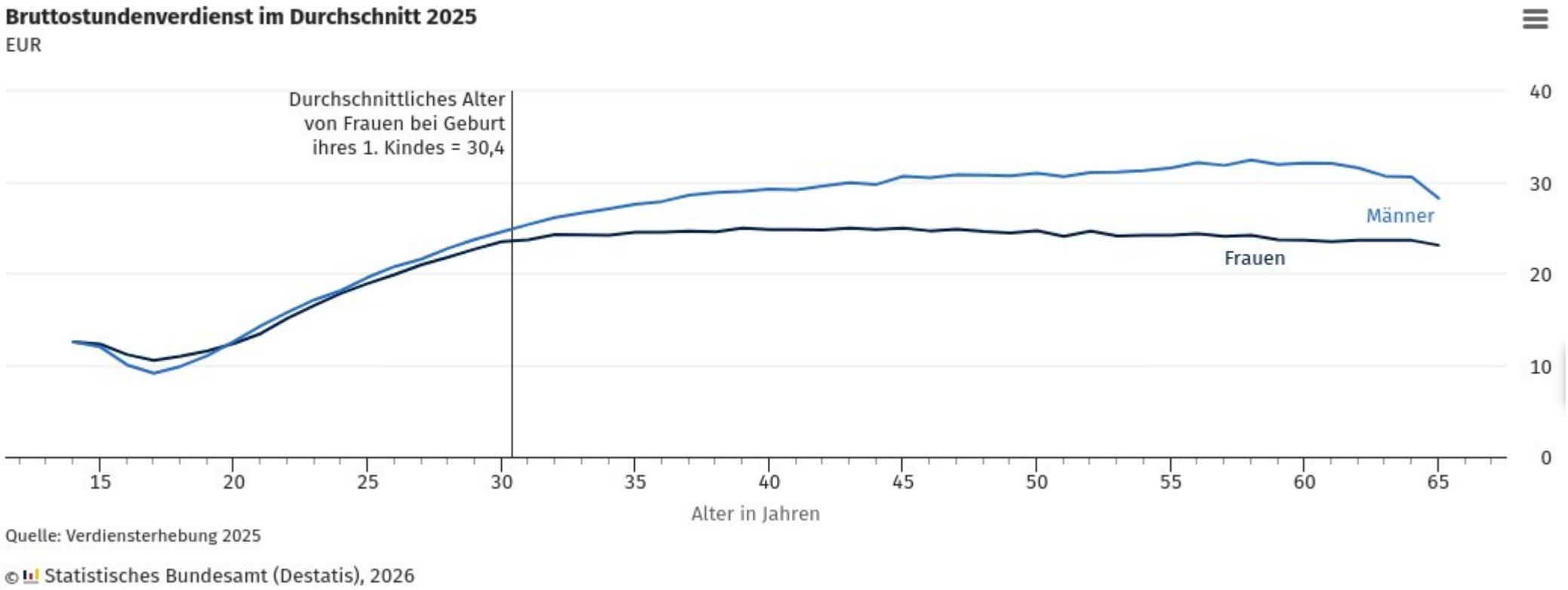
Gender Pay Gap – Aktuelle Zahlen



Gender Pay Gap – Aktuelle Zahlen



Gender Pay Gap – Aktuelle Zahlen



Quelle: Destatis 2025

Gender Pay Gap - Aussagen

Wie man es falsch macht:

„Werden Frauen schlechter bezahlt als Männer? Simple Mathematik und ein Minimum an Logik beweisen das Gegenteil. Erstabdruck in COMPACT 04/2021. _ von Steven Garcia
Eine Schweißerin soll um 19 Prozent weniger verdienen als ein Schweißer? Klingt empörend. Muss es aber nicht – denn niemand hat so einen Vergleich jemals angestellt. Es gibt nämlich keine weiblichen Beschäftigten in diesem Beruf, denn Frauen ist er schlicht zu anstrengend.“

- Löhne, Lügen und Larifari: Verdienen Frauen wirklich weniger? (25.03.2021)

Lösungen 8. Statistische Test Methoden

28 mögliche Punkte

Aufgabe – T-Test

12 Punkte

Datenset: *nominallohn_deutschland_2023-2024_nach_geschlecht.csv*
(DESTASIS Nominallohnindex)

Analysiere das Datenset, gehe dabei von der Nullhypothese aus das es keinen Signifikanten Unterschied in den Nominallöhnen zwischen den Geschlechtern gibt.

1. Stelle die Nullhypothese prüfbar auf.
-> Daten haben gleiche Varianz

H_0: μ_1 (Männlich) = μ_2 (Weiblich)

Aufgabe – T-Test

2. Importiere das Datenset und filtere angemessen.

```python
# Load dataset from CSV file
dataset: pd.DataFrame = pd.read_csv(
    'nominallohn_deutschland_2023-2024_nach_geschlecht.csv',
    delimiter=','
    )

# Split dataset by gender
male = dataset.loc[dataset["Geschlecht"] == "männlich"]
female = dataset.loc[dataset["Geschlecht"] == "weiblich"]
```

Aufgabe – T-Test

3. Berechne den von der Nullhypothese geforderten Populations Parameter μ.
4. Bestimme das Signifikanzlevel α unter der H_0 verworfen wird.

```
# Compute population mean (male wages)
# and define significance level
mu = male["Nominallohn"].mean()
alpha = 0.05
```

Aufgabe – T-Test

5. Bestimme mittels t-test den p-wert zur Nullhypothese.

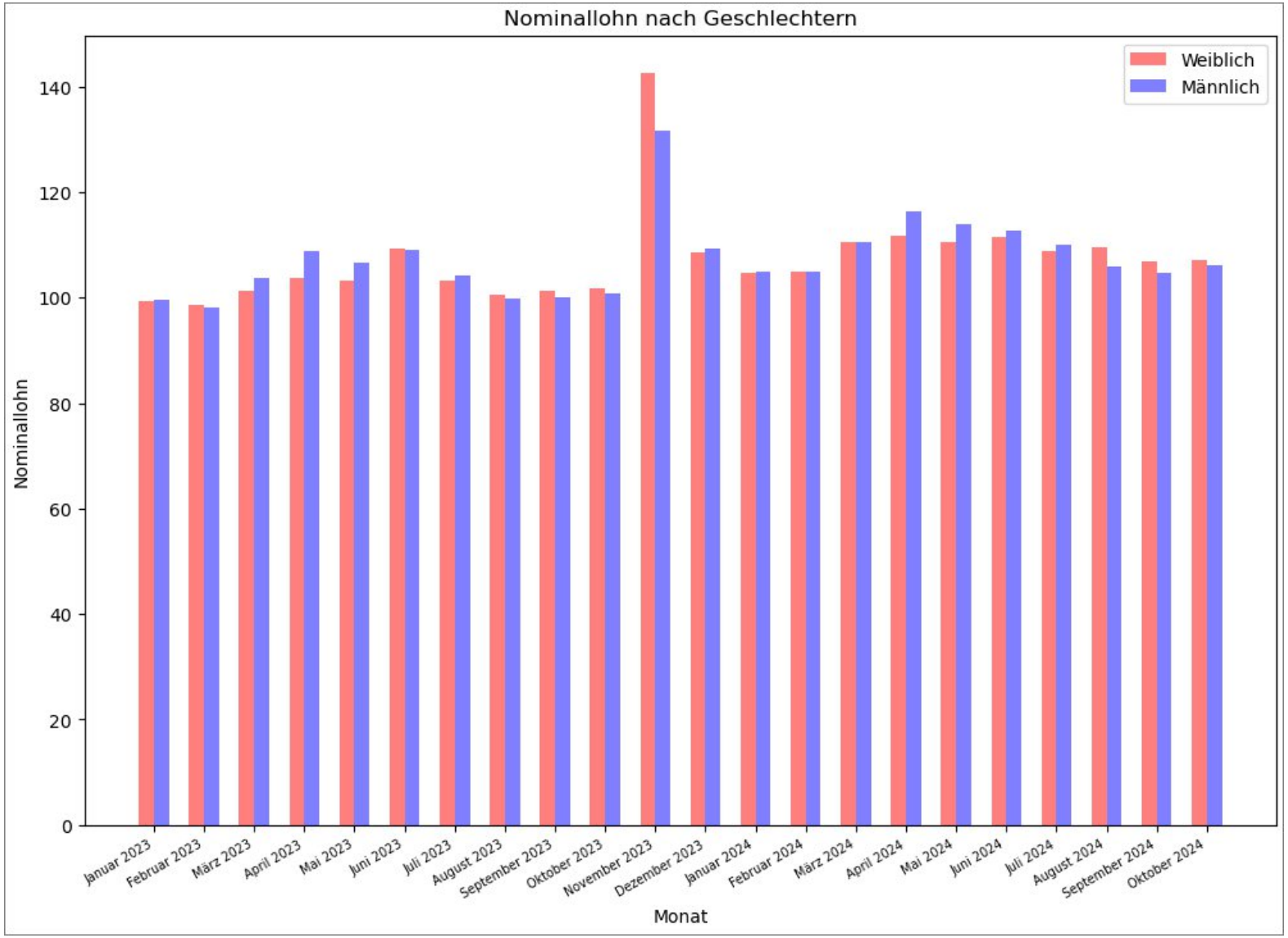
p_value: 0.95

„Fail to reject the null hypothesis; there is no significant difference between the sample mean and the hypothesized population mean.“

```python
# One-sample t-test: female wages against male population mean
t_stat, p_value = stats.ttest_1samp(female["Nominallohn"], mu)

# Hypothesis test decision based on p-value
if p_value < alpha:
    print('''
        Reject the null hypothesis;
        there is a significant difference between the sample mean
        and the hypothesized population mean.
        '''
         )
else:
    print('''
        Fail to reject the null hypothesis;
        there is no significant difference between the sample mean
        and the hypothesized population mean.
        '''
         )

print(f"P-Value: {p_value:.2}")
```

Aufgabe – T-Test

6. Stelle dein Ergebnis dar.



Aufgabe – T-Test

Nominallohn

das tatsächlich in € gezahlte Entgelt für geleistete Arbeit (Gegensatz: Reallohn). Der Nominallohn lässt keine Aussagen über die Kaufkraft zu, da die Preisentwicklung nicht berücksichtigt wird. Werden in einem Tarifvertrag z. B. Lohnsteigerungen von 3 % vereinbart, und die Inflationsrate beträgt gleichzeitig 2 %, so steigen die Löhne nominal um 3 %, tatsächlich (real) aber nur um 1 %.

Quelle: bpb.de

Aufgabe – Unbereinigter Gender Pay Gap

16 Punkte

Datenset: *bruttoverdiensterhebung_deutschland_april_22-23_nach_geschlecht.csv* (DESTASIS)

Analysiere das Datenset, gehe dabei von der Hypothese aus das einen keinen Bruttoeinkommensunterschied der Geschlechter in Deutschland für den Zeitraum 2022-2023 gibt.

Spalte	Mögliche Werte
Geschlecht	Weiblich | Männlich | Insgesamt
Branche	Verschiedenes (String)
Verdienst	Verschiedenes (Float)
Verdienstart	Alle: *Ohne Sonderz.* – Durchschn. Bruttomonatsverdienste – Durchschn. Bruttostundenverdienste – Mittlere Bruttomonatsverdienste – Mittlere Bruttostundenverdienste

Aufgabe – Unbereinigter Gender Pay Gap

1. Lies das Datenset ein und schaue dir die vorhandenen Werte an.
2. Filter aus der Spalte **Geschlecht** die benötigten Werte.

```python
# Read in
df = pd.read_csv(
'bruttoverdiensterhebung_deutschland_april_22-23_nach_geschlecht.csv'
)

# Filter
# "Verdienstart"
target = "Durchschn. Bruttomonatsverdienste ohne Sonderz."
df_filtered = df[
    (df['Geschlecht'].isin(['männlich', 'weiblich'])) &
    (df['Verdienstart'] == target)
].copy()
```

Aufgabe – Unbereinigter Gender Pay Gap

3. Führe einen Korrelationstest zwischen den Geschlechtern & dem Bruttoverdienst durch.

```python
# Code sex values
df_filtered["sex_coded"] = df_filtered["Geschlecht"]\
                           .map({"männlich": 0, "weiblich": 1})
df_filtered["male"] = (df_filtered["Geschlecht"] == "männlich")\
                      .astype(int)
df_filtered["female"] = (df_filtered["Geschlecht"] ==
"weiblich").astype(int)

# Select columns and remove NaN
cols = ["Verdienst", "sex_coded", "male", "female"]
data = df_filtered[cols].dropna()

# Calculate correlations
variables = cols
n = len(variables)
corr_matrix = np.zeros((n, n))

for i, var1 in enumerate(variables):
    for j, var2 in enumerate(variables):
        if i == j:
            corr_matrix[i, j] = 1.0
        else:
            corr, _ = stats.pearsonr(data[var1], data[var2])
            corr_matrix[i, j] = corr

# Create DataFrame for easier plotting
corr_df = pd.DataFrame(corr_matrix, index=variables, columns=variables)
```

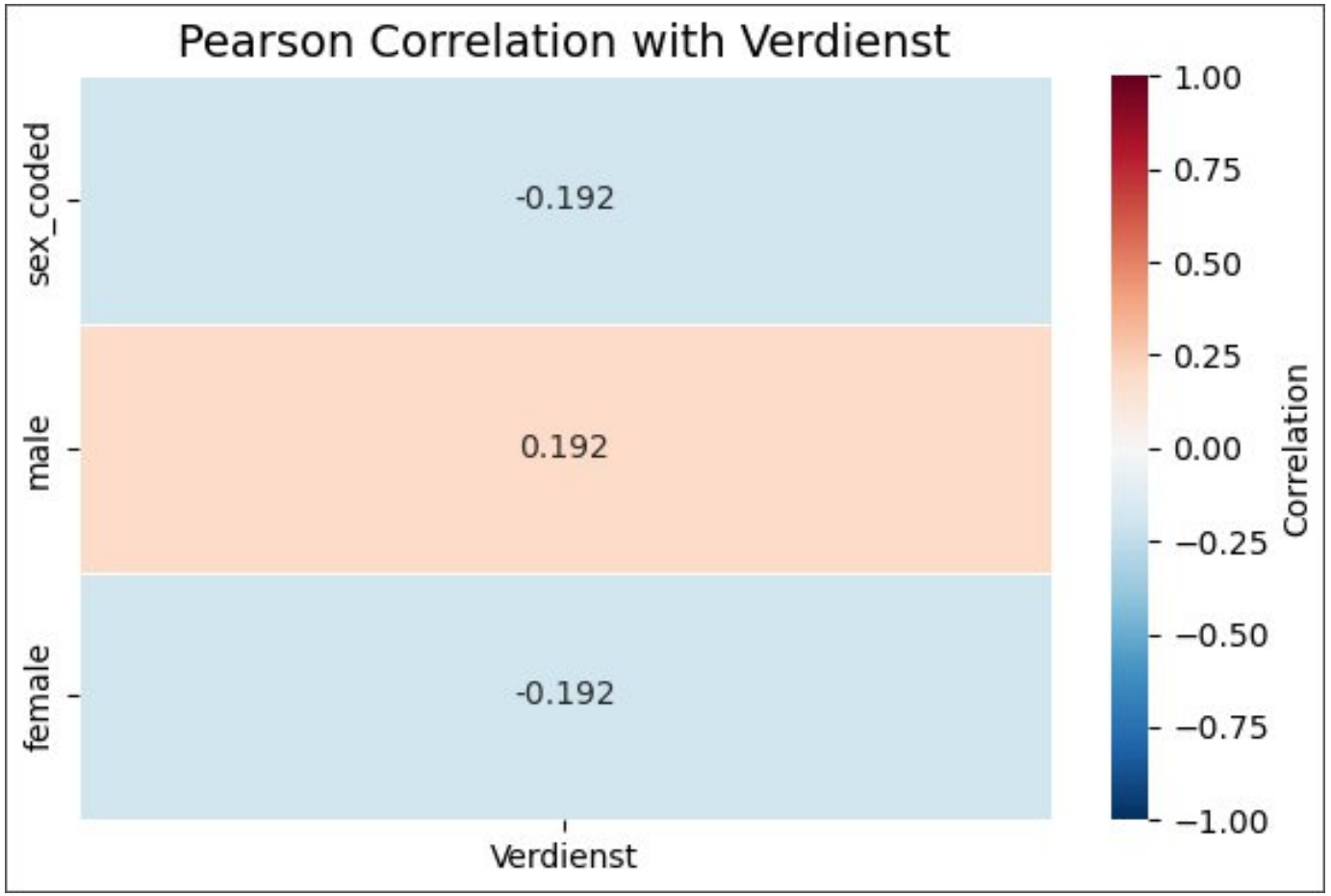
Aufgabe – Unbereinigter Gender Pay Gap

3. Führe einen Korrelationstest zwischen den Geschlechtern & dem Bruttoverdienst durch.

```
# Correlation Matrix Pearson
verdienst_corr = corr_df[["Verdienst"]].drop("Verdienst")

# Plot
fig, ax = plt.subplots(figsize=(6, 4))
sb.heatmap(verdienst_corr,
           cmap='RdBu_r',
           annot=True,
           linewidth=0.5,
           ax=ax,
           vmin=-1, vmax=1,
           fmt='.3f',
           cbar_kws={'label': 'Correlation'})

ax.set_title("Pearson Correlation with Verdienst", fontsize=14)
plt.tight_layout()
```

Aufgabe – Unbereinigter Gender Pay Gap

3. Führe einen Korrelationstest zwischen den Geschlechtern & dem Bruttoverdienst durch.



Aufgabe – Unbereinigter Gender Pay Gap

4. Berechne den Bruttoeinkommensunterschied.

5. Stelle den unbereinigten Gender Pay Gap auf.

```python
# Filter by "Geschlecht"
wage_mean = df_filtered.groupby('Geschlecht')['Verdienst'].mean()
wage_male = wage_mean["männlich"]
wage_female = wage_mean["weiblich"]

bruttoeinkommen_diff = wage_male - wage_female
unbereinigter_gpg = np.round(
    bruttoeinkommen_diff / wage_male * 100,
    decimals=2
)
```

```python
# Filter by sex
male = df_filtered[df_filtered["Geschlecht"] == "männlich"]
["Verdienst"].dropna().astype(float)
female = df_filtered[df_filtered["Geschlecht"] == "weiblich"]
["Verdienst"].dropna().astype(float)

# T-Test 2 Sample
t_stat, p_value = stats.ttest_ind(male, female, equal_var=False)
```

Aufgabe – Unbereinigter Gender Pay Gap

6. Finde eine geeignete Darstellung für dein Ergebnis.

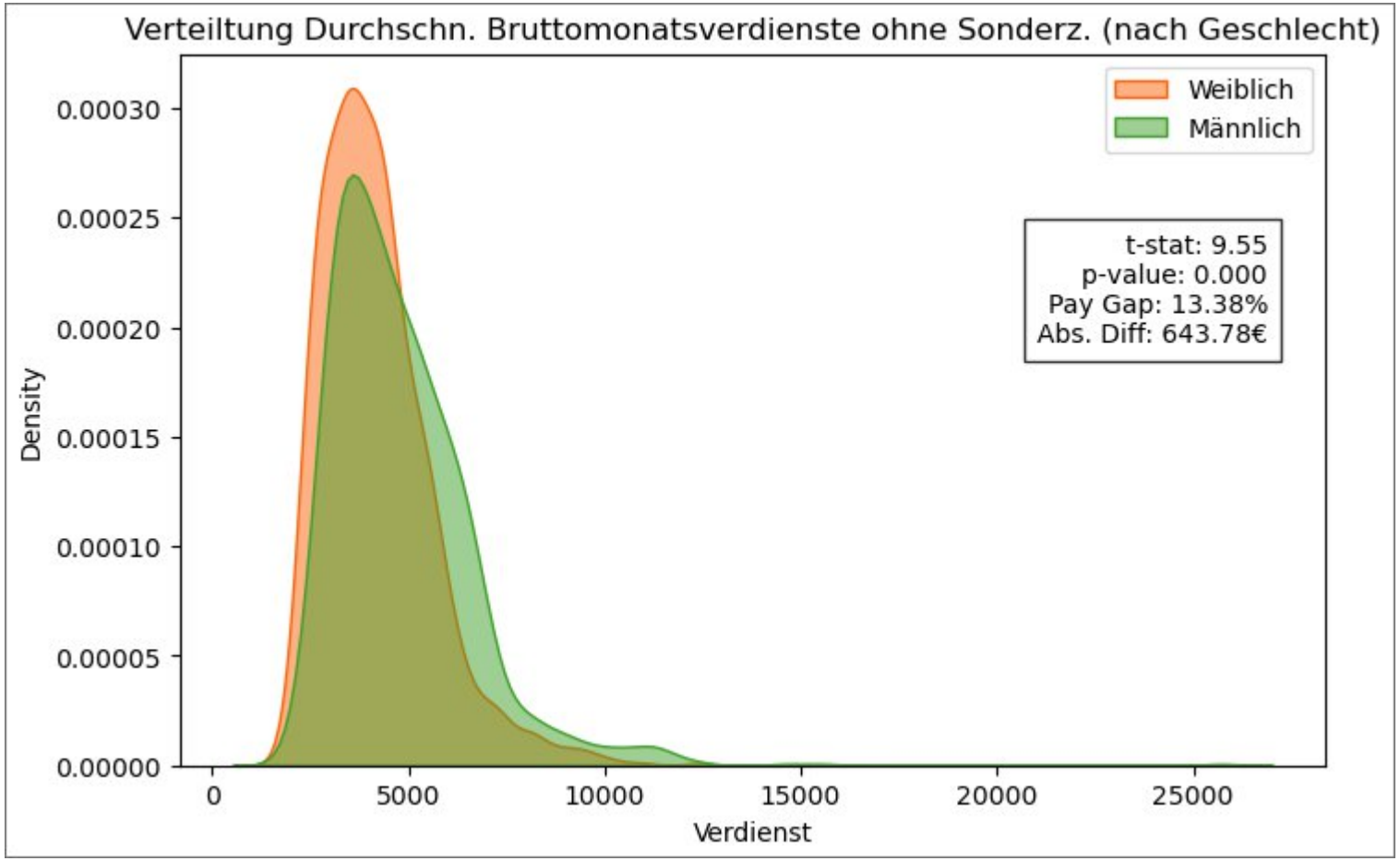
```
# Plot
fig, ax = plt.subplots(figsize=(8,5))
ax.set_title(
"Verteiltung Durchschn. Bruttomonatsverdienste ohne Sonderz. (nach Geschlecht)"
)
sb.kdeplot(
    female, label="Weiblich",
    color="#fe640b", fill=True,
    alpha=0.5, ax=ax
    )
sb.kdeplot(
    male, label="Männlich",
    color="#40a02b", fill=True,
    alpha=0.5, ax=ax
    )
ax.legend()
```

```
text = f'''t-stat: {t_stat:.2f}
p-value: {p_value:.3f}
Pay Gap: {unbereinigter_gpg:.2f}%
Abs. Diff: {bruttoeinkommen_diff:.2f}€'''

ax.text(0.95, 0.75, text,
        transform=ax.transAxes,
        fontsize=10,
        verticalalignment='top',
        horizontalalignment='right',
        bbox=dict(
            facecolor='white',
            edgecolor='black',
            pad=5, alpha=0.8
            )
        )
plt.show()
```

Aufgabe – Unbereinigter Gender Pay Gap

6. Finde eine geeignete Darstellung für dein Ergebnis.

H_0 widerlegt, da p-wert kleiner Signifikanzniveau 0.05.



Lösungen 9. Folium

21 mögliche Punkte

Aufgabe – Create a Map

4 Punkte

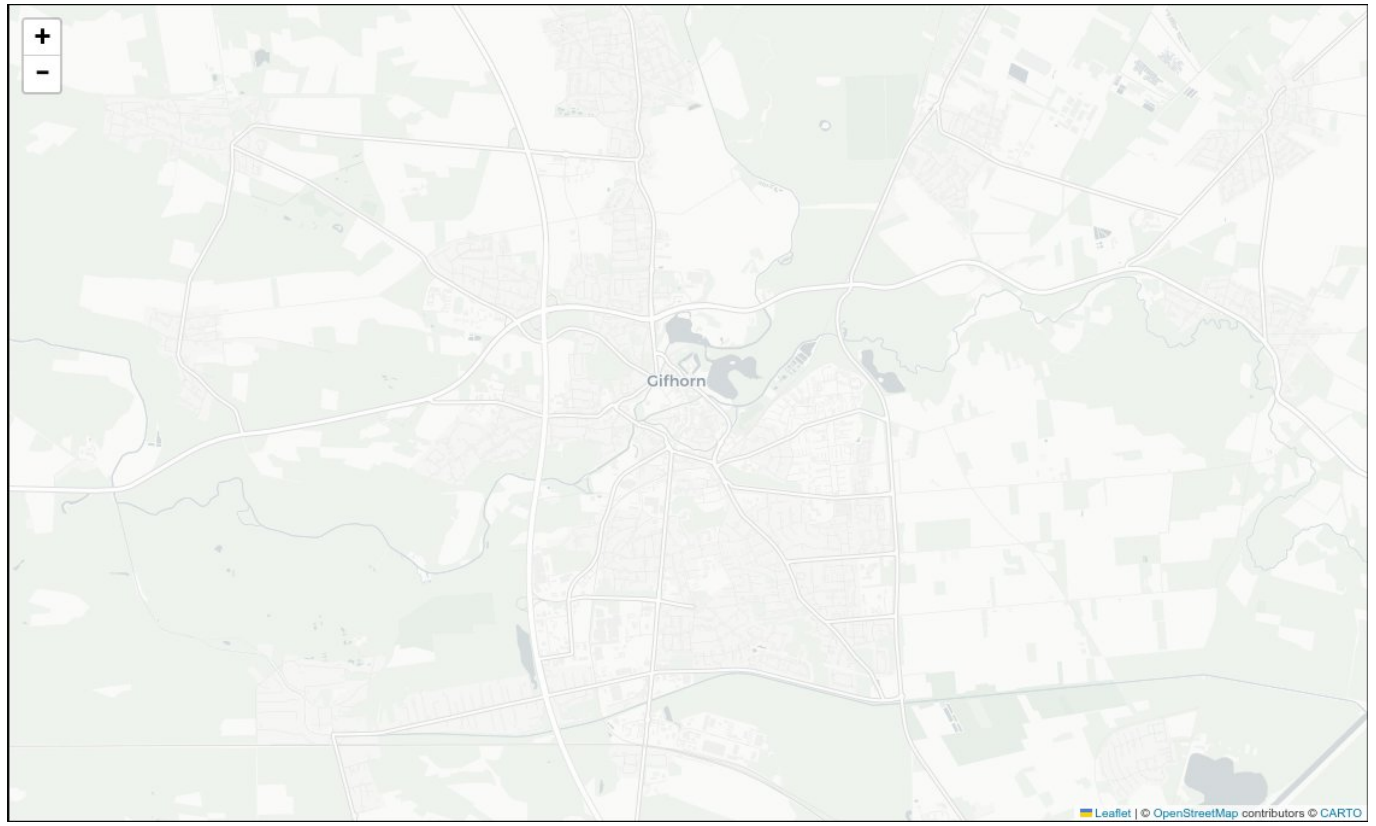
Create a map with the coordinates of your hometown, or your favorite city (Braunschweig does not count as a solution). Save your Solution in the variable *my_map*.

Adjust the *zoom_start* parameter so that your place is visible in the center. To find out which coordinates your place has you can use the online tool latlong.net.

- Use *CartoDB Positron* as tileset.
- Set the *prefer_canvas* parameter to *True*.
- Also please add your city as a comment.

Aufgabe – Create a Map

```
# Your City: Gifhorn
my_map = folium.Map(
        location=(52.485291, 10.547030),
        tiles='cartodb positron',
        zoom_start=13,
        prefer_canvas=True
    )
my_map
```



Aufgabe – Design our own Marker

7 Punkte

1. Define a reasonable *tooltip* with useful text.
2. Define a popup named *tu_popup_html*. Orient yourself to the given template. Design it freely as you like, but it must be an HTML string.
3. Define an icon *tu_icon*. Style it as you like; it should use an icon that is not *glyphicon-home*.
4. Combine the last three objects into one *tu_bs_marker* object.

Use *(52.273460, 10.529231)* as the location data.

Aufgabe – Design your own Marker

```python
tooltip = "More about TU Braunschweig"

tu_popup_html = folium.Popup(
    '''
    <p>
    <img

src="https://upload.wikimedia.org/wikipedia/commons/thumb/9/9d/Siegel_TU
_Braunschweig_transparent.svg/1200px-
Siegel_TU_Braunschweig_transparent.svg.png"
        alt="TU BS Logo"
        height="100"
        >
    </p>
    <p><strong>Universitätspl. 2</strong></p>
    <p><em>38106 Braunschweig</em></p>
    <p><small>Germany, DE</small></p>
    <p>Visit: <a href="https://www.tu-bs.de/">tu-bs.de</a> </p>
    ''',
    show=False
    )
```

```python
tu_icon = folium.Icon(
            color='red',
            icon_color='#eeeeee',
            prefix='glyphicon',
            icon='glyphicon-briefcase',
            angle=0
            )

tu_bs_marker = folium.Marker(
        location=(52.273460, 10.529231),
        popup=tu_popup_html,
        tooltip=tooltip,
        icon=tu_icon
    )
```

Aufgabe – Design your own Marker

Aufgabe – Mapping the Universties in Lower Saxony

10 Punkte

The dataset for this notebook is *unis_nd.csv*. (I recommend taking a look at the dataset.) The goal is to display every university in Lower Saxony on a map.

Take a look at the dataset and plot the locations according to the given data. Every marker should be designed and clickable. Also, distinguish between public and private universities. Style is up to you.

Map Attributes:
- The location should be the georaphic centre of Lower Saxony in *Wehrenberg 27318 Hoyerhagen* **(52.806390, 9.135110)**.
- Use a suitable tileset from the documentation
- Use a suitable zoom setting

Aufgabe – Mapping the Universties in Lower Saxony

```python
import pandas as pd
df = pd.read_csv('unis_nd.csv')

lower_saxony = folium.Map(
        # Georaphical centre Point of Lower Saxony
        # Wehrenberg 27318 Hoyerhagen
        location=(52.806390, 9.135110),
        tiles='OpenStreetMap',
        zoom_start=7
    )
```

```python
for index, row in df.iterrows():
    pp = popup_factory(
        adr=row['Address'],
        zipc=row['plz'],
        country='Germany, DE',
        pic=row['pic'],
        )
    location = (float(row['lat']), float(row['lon']))

    is_public = False
    if row['Sponsorship'] == 'public':
        is_public = True

    marker = marker_factory(location, pp, is_public)

    marker.add_to(lower_saxony)
```

Aufgabe – Mapping the Universties in Lower Saxony