Aufgabensammlung Python

Dipl.-Ing. Christoph Schreiber

November 11, 2019

Inhaltsverzeichnis

1 Erste Schritte

1.1. Erstelle eine Datei `hello_user.py`, kopiere folgenden Text die Datei und führe die Datei mit `python hello_user.py` aus.

```
name = input("Name: ")
print("Hello " + name)
```

1.2. Erstelle eine Datei `quadrat.py`, kopiere folgenden Text die Datei und führe die Datei mit `python quadrat.py` aus. Ändere das Programm so ab, dass es auch mit `float`-Zahlen funktioniert und gib weiters den Umfang aus.

```
import math

print("Programm zur Berechnung von Umfang, "
      "Fläche und Diagonale eines Quadrats")
a = input("Seitenlänge a = ")

# A = a * a      Fehler: a ist ein String, wir wollen aber
   mit Zahlen rechnen
a = int(a)
A = a * a
A = a ** 2       # ** => hoch
print("Fläche =", A)

d = math.sqrt(2 * a ** 2)       # 2a^2
print("Diagonale =", d)
```

1.3. Schreibe ein Programm, welches zwei `Strings` einliest und diese zusammengehängt ausgibt. Beispielablauf:

```
Erster String: Man kann Strings
Zweiter String: addieren und multiplizieren.
-------
Zusammen: Man kann Strings addieren und multiplizieren
```

1.4. Schreibe ein Programm, welches einen zu zahlenden Betrag und einen erhaltener Betrag einliest. Das Programm berechnet das Rückgeld. Tests:

Zu zahlen	Erhalten	Rückgeld
80	100	20
15,60	20	4,4

1.5. Schreibe ein Programm, welches einen Nettopreis und einen Steuersatz einliest und den Bruttopreis ausgibt. Tests:

Netto	Steuer	Brutto
100	20	120
80	20	96
60,7	12	67,984

1.6. Schreibe ein Programm, welches den Radius eines Kreises einliest und Fläche und Umfang des Kreises berechnet.

Hinweis: $A = r^2\pi, u = 2r\pi$. Auf π wird mit `import math` und `math.pi` zugegriffen. Tests:

r	A	u
4	50.265	25.133
1.5	7.069	9.425

1.7. Schreibe ein Programm, welches die drei Seitenlängen eines Dreiecks einliest und mit Hilfe der Heron'schen Flächenformel die Fläche berechnet und diese ausgibt.

$$s = \frac{a+b+c}{2} \text{ und } A = \sqrt{s \cdot (s-a)(s-b)(s-c)}$$

Tests:

a	b	c	A
3	5	7	6,495
1	2	3	0

1.8. In der Wissenschaft und der Programmierung werden Winkel üblicherweise in Radiant angegeben; Schreibe ein Programm, welches einen Winkel in Grad(engl. *degree*) einliest und diesen in Radiant konvertiert.

$$rad(deg) = \frac{deg}{180} \cdot \pi$$

Tests:

Input	Output
180	3.141593
1	0.017453

1.9. Die Firma `SmartCar` vermietet Autos und verrechnet pro km 0.23€ und zusätzlich einen Pauschalbetrag von 3€[1] unabhängig von der Gefahrenen Kilometerzahl. Schreibe ein Programm, welches vom User eine Kilometeranzahl einliest und den entsprechenden Preis ausgibt; gib auch den Preis ohne 20% Mehrwertsteuer an. Beispielablauf:

```
Distanz: 120
-------
Preis:   25,5
Steuer:   5,1
Gesamt:  30,6
```

1.10. Schreibe ein Programm, welches den durchschnittlichen Benzinverbrauch und die Kosten pro km zwischen zwei Tankstopps berechnen soll. Beispielablauf:

```
Kilometerstand beim letzten Tankstopp[km]: 94328
Kilometerstand beim heutigen Tankstopp[km]: 94870
Benzinverbrauch[L]: 39.5
Benzinpreis pro Liter[€]: 1.312
-------
Durchschnittlicher Benzinverbrauch[L/km]: 0,0729
Durchschnittliche Benzinkosten[€/km]: 0,096
```

[1] Preise inklusive Mehrwertsteuer

1.11. Erstelle eine Datei `format_strings.py`, kopiere folgenden Text die Datei und führe die Datei aus.

```
n = 7
print("{:3d}".format(n))          # d == Digit

x = 0.3
print("{:.2f}".format(x))         # f == Float

string = "Formatstrings. Yay!"
print("{:s}".format(string))      # s == String

print("{:5.1f} {:.1f}".format(x, n))
```

Ändere den Output **nur** durch Änderung der `print`-Anweisungen, aber belasse jeweils {} als erstes Zeichen im Formatierungsstring. Erwartete Ausgabe:

```
00007 HTL
0.300
Formatstrings. Yay!
0.3 7.0
```

1.12. Schreibe ein Programm, welches eine Zapfenrechnung nach folgendem Schema durchführt:

```
Dreistellige Zahl: 685
    685 * 2
   1370 * 3
   4110 * 4
  16440 * 5
  82200 / 5
  16440 / 4
   4110 / 3
   1370 / 2
    685
```

1.13. Schreibe ein Programm, welches vom User eine dreistellige Zahl einliest und die Zerlegung der Zahl in Zehnerpotenzen aufzeigt. Beispielablauf:

```
Zahl: 642
642 = 6 * 100 + 4 * 10 + 2 * 1
```

1.14. Postleitzahlen in Wien haben das Format 1BBF, wobei

- B der Bezirk ist, von 01 bis 23
- F die Filiale ist

Lies vom Benutzer eine Postleitzahl ein und gib den entsprechenden Bezirk aus. Beispielablauf:

```
Postleitzahl: 1043
-------
4. Bezirk
```

1.15. Erstelle eine Datei `random_number.py`, kopiere folgenden Text die Datei und führe die Datei aus. `randrange(a, b)` liefert eine zufällige Ganzzahl im Intervall $[a, b[$[2].

```
from random import randrange

n = randrange(11, 21)
print(n)
```

Ändere den Code so ab, dass 10 zufällige Ganzzahlen im Intervall $[42, 47[$ erzeugt werden und gib alle aus.

1.16. Schreibe ein Programm, welches vom User eine Sekundenzahl nach Mitternacht einliest und die entsprechende Uhrzeit ausgibt. Tests:

Input	Output
0	00:00:00
62	00:01:02
12345	03:25:45

1.17. Schreibe ein Programm, welches vom User eine vierstellige Binärzahl einliest, diese in eine Dezimalzahl konvertiert und sie ausgibt. Tests:

Input	Output
1	1
10	2
1101	13

Zur Erklärung:

$$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$$

2 if

2.1. Zeichne ein Struktogramm und dann schreibe ein Programm, welches überprüft, ob sich jemand mit Usernamen `"IF"` und Passwort `"If4y0u"` einloggt. Stimmen Passwort und Username, so begrüße den User; andernfalls erzeuge eine Exception. Beispielablauf:

```
Username: IF
Passwort: If4y0u
-------
Hallo IF

Username: lolwtf
Passwort: dunnolol
-------
Ungültige Username/Passwort-Kombination
```

2.2. Fortsetzung von Beispiel 1.4: Schreibe ein Programm, bei ein zu zahlender Betrag und ein erhaltener Betrag eingegeben wird. Das Programm berechnet das Rückgeld oder gibt eine Fehlermeldung aus, falls eine Eingabe fehlerhaft ist bzw. der erhaltene Betrag zu klein ist. Tests:

[2] a inklusive, b exklusive

Zu zahlen	Erhalten	Rückgeld
80	100	20
15,60	20	4,4
100	80	Fehler
-3	Fehler	

2.3. Fortsetzung von Beispiel 1.14: Postleitzahlen in Wien haben das Format 1BBF, wobei

- B der Bezirk ist, von 01 bis 23
- F die Filiale ist

Lies vom Benutzer eine Postleitzahl ein und gib den entsprechenden Bezirk aus. Bei einer falschen PLZ wird eine Fehlermeldung ausgegeben geworfen. Tests:

PLZ	Output
1041	4. Bezirk
1234	23. Bezirk
1111	11. Bezirk
1241	Fehler
7041	Fehler
1001	Fehler

2.4. Zeichne ein Struktogramm und dann schreibe ein Programm, welches 2 zufällige Zahlen im Intervall $[100, 400]$ erzeugt. Gib beide Zahlen aus und kennzeichne klar erkennbar die größere. z.B: `243 < 328`

2.5. Ergänze den Code, sodass ermittelt wird ob die Zahl gerade oder ungerade ist und ob die Zahl positiv oder negativ ist.

```
n = input("Zahl: ")
try:
    n = int(n)
except ValueError:
    print("Fehler. Erwartet war eine Zahl; Eingabe: " + n)
    exit()
```

Beispielablauf:

```
Zahl: 17
Die Zahl ist positiv und ungerade.
```

2.6. Zeichne ein Struktogramm und dann schreibe ein Programm, welches einen Temperaturwert in Celsius einliest und diesen in Kelvin und Fahrenheit umrechnet. Die minimale Temperatur sind 0 Kelvin.

$$\frac{C^\circ}{5} = \frac{F^\circ - 32}{9}$$
$$C^\circ + 273,15 = K$$

Tests:

C°	F°	K
0	32	273.15
7	44.6	280.15
35	95.0	308.15
-4.2	24.44	268.95
-273.16	Fehler	
Text	Fehler	

2.7. Zeichne ein Struktogramm und dann schreibe ein Programm, welches eine Jahreszahl einliest und überprüft, ob diese ein Schaltjahr ist, oder nicht. Ein Schaltjahr tritt dann auf, wenn die Jahreszahl durch 400 teilbar ist **oder** durch 4, aber nicht durch 100.
Tests:

Input	Output
2019	kein Schaltjahr
1900	kein Schaltjahr
2004	Schaltjahr
2000	Schaltjahr
Text	Fehler

2.8. Zeichne ein Struktogramm und dann schreibe ein Programm, welches zwei Zahlen und eine Mathematische Operation(+-*/) einliest und das Ergebnis ausgibt. Programmablauf:

```
a = 3
b = 4
Operation: -
-------
3.000 - 4.000 = -1.000
```

Tests:

a	b	Operation	Ergebnis
3	4	-	-1
4	3	/	1.333
2.3	4.2	*	9.66
0	0	*	0
0	2	/	0
2	0	/	Fehler

2.9. Schreibe ein Programm, welches ein Jahr und einen Monat einliest und die Anzahl der Tage in diesem Monat zurückgibt. Siehe auch Beispiel 2.7. Programmablauf:

```
Jahr: 2000
Monat: 2
Der Monat hat 29 Tage.
```

2.10. Fortsetzung von Beispiel 2.2: Schreibe ein Programm, bei ein zu zahlender Betrag und ein erhaltener Betrag eingegeben wird. Das Programm gibt exakt aus, was für ein Rückgeld zu geben ist. Wir nehmen hier an, dass unser Programm nur für ganze Eurobeträge verwendet wird(wegen des Aufwands). Programmablauf:

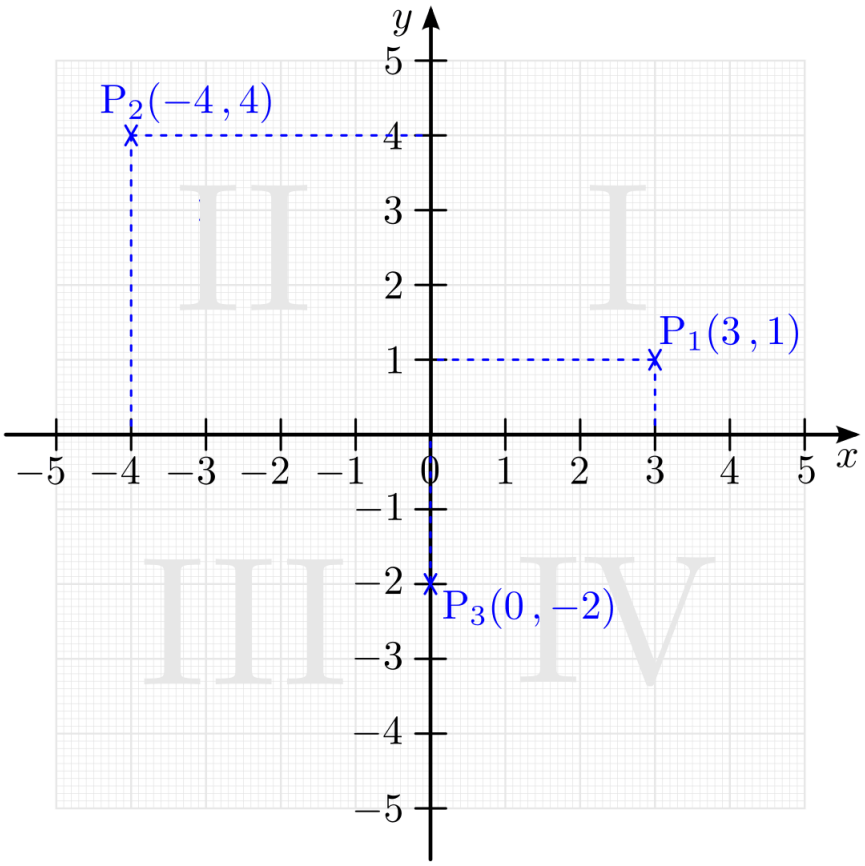
```
Zu Zahlen: 52
Erhalten: 100
Retour:
2x 20
1x  5
1x  2
1x  1
```

2.11. Schreibe ein Programm, welches eine lineare Gleichung $ax + b = c$ vom Benutzer einliest und die Lösung dieser Gleichung ($x = \dots$) ausgibt. Falls die Gleichung **unlösbar** ist, so ist dies dem User auch mitzuteilen![3] Programmablauf:

[3] Der mathematisch auch mögliche Fall, dass jede Zahl Lösung ist, kann ignoriert werden; Bsp: $0x + 3 = 3$

```
a: 3
b: -4
c: 6
Die Gleichung 3.0x - 4.0 = 6.0 hat die Lösung x = 3.333
```

2.12. Schreibe ein Programm, welches x- und y-Koordinate eines Punktes einliest und die Position des Punktes ausgibt.



Mögliche Ausgaben:

- x-Achse
- y-Achse
- Ursprung $O(0|0)$
- Quadrant I-IV

Die angegebene Reihenfolge ist nicht unbedingt die cleverste für Überprüfungen.

2.13. Schreibe ein Programm, welches einen zufälligen Zahlenwert im Intervall $[0, 99]$ erzeugt[4] und diesen ausgibt. Dies seien die Punkte, die ein Schüler auf einen Test hat. Gib die entsprechende Note des US-Amerikanischen Notensystems dazu aus:

Punkte p	**Note**
$[0, 59]$	F
$[60, 69]$	D
$[70, 79]$	C
$[80, 89]$	B
$[90, 99]$	A

[4] Für Testzwecke empfiehlt es sich, direkt die zu testende Punktezahl zu speichern

Endet eine Note auf 8 oder 9, so wird der Note ein `+` hinzugefügt, bei einer Endziffer 0 oder 1 ein `-`. Bei `F` soll dies nur an den Intervallgrenzen erfolgen. Tests:

Punkte	**Note**
0	F-
29	F
90	A-
89	B+
71	C-
60	D-
63	D
58	F+

2.14. Schreibe ein Programm, welches drei Zahlen einliest. Das Programm sortiert diese Zahlen aufsteigend und gibt sie aus. Die beste Version dieses Algorithmus kommt mit 3 `if` aus. Programmablauf:

```
a: 3
b: 1
c: 7
1, 3, 7
```

3 Schleifen

3.1. Schreibe ein Programm, welches die ungeraden Zahlen von 1 bis 20 ausgibt

(a) mit einer `while` - Schleife

(b) mit einer `for` - Schleife

3.2. Zeichne zuerst ein Struktogramm. Schreibe ein Programm, welches vom User zwei ganze Zahlen einliest und alle Zahlen dazwischen ausgibt. Wie in der Informatik üblich ist dabei der Beginn inklusive, das Ende exklusive. Programmablauf:

```
Beginn: 3
Ende: 7
-------
3 4 5 6

Beginn: 8
Ende: 3
-------
8 7 6 5 4
```

3.3. Schreibe eine Programm, welches eine natürliche Zahl n einliest und die Fakultät[5] $n!$ der Zahl berechnet. Tests:

[5] $n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \dots \cdot 1$

n	n!
4	24
0	1
1	1
13	6227020800
-3	Fehler

3.4. Zeichne zuerst ein Struktogramm. Schreibe ein Programm, welches eine natürliche Zahl einliest und alle Teiler dieser Zahl ausgibt. Programmablauf:

```
Zahl: 42
-------
Teiler: 1 2 3 6 7 14 21 42
```

3.5. Schreibe ein Programm, welches die Zahlen von 1 bis 32 ausgibt. Ist die Zahl durch 3 teilbar, ist `Fizz` auszugeben, bei 5 `Buzz`; ist die Zahl sowohl durch 3 als auch durch 5 teilbar, so ist `FizzBuzz` auszugeben. Programmablauf:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
...
14
FizzBuzz
...
```

3.6. Schreibe ein Programm, welches den User nach einer ganzen Zahl fragt, bis dieser eine eingibt. Beispielablauf:

```
Ganzzahl: nein
Keine Zahl: nein
Ganzzahl: ???
Keine Zahl: ???
Ganzzahl: 3.1415
Keine Zahl: 3.1415
Ganzzahl: 0.0
Keine Zahl: 0.0
Ganzzahl: 7
Erfolg
```

Dieser Code ist ab jetzt bei jedem Einlesen zu verwenden.

3.7. Schreibe ein Programm, welches ein simples Zahlenratespiel darstellt. Es wird eine geheime Zufallszahl im Intervall $[0, 99]$ erzeugt und dann muss der Spieler versuchen, diese zu erraten, wobei das Programm Feedback gibt.
Der Spieler hat maximal 6 Versuche, wobei immer klar sein muss, beim wievielten er ist. Zu Debugzwecken und bei der Abgabe ist die Zufallszahl anzuzeigen. Programmablauf:

```
Geheimzahl: 25
Versuch 1: meh
Ungültig: meh
Versuch 1: 80
zu groß
Versuch 2: 17
zu klein
Versuch 3: 25
richtig
```

3.8. Löse Beispiel 1.12 mit Hilfe von Schleifen.

3.9. Schreibe ein Programm, welches folgende Ausgabe erzeugt:

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

3.10. Schreibe ein Programm, welches folgende Ausgabe erzeugt:

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

3.11. Schreibe ein Programm, welches alle dreistelligen Zahlen mit einer Ziffernsumme größer als 25 ausgibt. Erwartete Ausgabe: `899 989 998 999`

3.12. Schreibe ein Programm, das eine positive Ganzzahl einliest und die Primfaktorzerlegung dieser Zahl ausgibt.
Beginnt man mit 2 und bewegt sich aufsteigend weiter, ist es egal, ob eine Zahl prim ist oder nicht. Beispielsweise kann eine Zahl nicht durch 6 dividierbar sein, wenn man bereits durch 2 und 3 dividiert hat.
Tests:

Input	Output
60	2 * 2 * 3 * 5
37	37
884	2 * 2 * 13 * 17
-3	Fehler

3.13. Zeichne zuerst ein Struktogramm. Jemand hat die Ziffernkombination für sein vierstelliges Ziffernschloss vergessen. Er erinnert sich jedoch daran, dass alle Ziffern gerade sind

und genau zwei Ziffern 0 sind. Gib alle möglichen Ziffernkombinationen und ihre Anzahl aus. Ausgabe:

```
0022
0024
0026
0028
0042
0044
...
8200
8400
8600
8800

Mögliche Kombinationen: 96
```

3.14. Korrigiere den Fehler in folgendem Programm.

```
d = 0
print("Die Antwort auf die Frage nach dem Leben, "
      "dem Universum und dem ganzen Rest ist ", end='')
while d <= 1:
    if d == 0.3:
        print(42)
        exit()
    d += 0.1

raise ValueError("Unerreichbar")
```

4 Listen, Tupel

4.1. Schreibe ein Programm, welches eine Sozialversicherungsnummer einliest und ermittelt, ob diese gültig ist. Eine SZVN besitzt das Format $N_1N_2N_3PT_1T_2M_1M_2J_1J_2$, beispielsweise 1234010180. Diese Zahl besteht aus:

- NNN - einer laufenden Zahl
- P - der Prüfziffer
- TT - der Tag der Geburt
- MM - das Monat der Geburt
- JJ - das Jahr der Geburt

$$s = 3N_1 + 7N_2 + 9N_3 + 5T_1 + 8T_2 + 4M_1 + 2M_2 + J_1 + 6J_2$$

Eine SZVN ist dann gültig, wenn s durch 11 geteilt den Rest P ergibt. Tests:

SZVN	gültig
4908060650	true
1234010203	false
deine eigene	true

4.2. Schreibe ein Programm, welches eine Ganzzahl im Intervall $[1, 7]$ einliest und den entsprechenden Wochentag ausgibt, wobei Montag 1 ist. Tests:

Input	Output
1	Montag
6	Samstag
0	Fehler
8	Fehler
w	Fehler

4.3. In Python werden Variablen üblicherweise im Stil `so_machts_python` bezeichnet; in Java/C# wird der Stil `camelCase` verwendet. Schreibe ein Programm, das vom User einen Python-Variablennamen einliest und in camelCase verwandelt. Tests:

Input	Output
gelesene_zeile	geleseneZeile
eine_kleine_liste_aus_strings	eineKleineListeAusStrings

4.4. Die Fibonacci-Folge ist eine unendliche Folge von Zahlen, bei der sich die jeweils folgende Zahl durch Addition ihrer beiden vorherigen Zahlen ergibt, wobei die ersten beiden mit 1 gegeben sind.
Schreibe ein Programm, das die ersten 10 Fibonacci-Zahlen in eine Liste speichert und diese Liste dann ausgibt.
Erwartete Ausgabe: `[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]`

4.5. Schreibe ein Programm, das vier Listen mit Lottotips für 6-aus-45 erstellt. Ein Lottotip muss sortiert sein und darf keine Zahl doppelt enthalten!
Beispielausgabe:

```
[2, 6, 18, 19, 22, 35]
[5, 6, 11, 25, 31, 45]
[9, 18, 24, 29, 34, 35]
[23, 24, 33, 35, 43, 44]
```

4.6. Schreibe ein Programm, welches vom Benutzer eine Ganzzahl einliest und eine Liste ihrer Ziffern zurückgibt. z.B.: 1834 → `[1,8,3,4]`

4.7. Schreibe ein Programm, welches feststellt, ob ein vom Benutzer eingelesener String ein Palindrom[6] ist.
Tests:

Input	Output
otto	Palindrom
Otto	kein Palindrom
reger	Palindrom
Regal	kein Palindrom

4.8. Schreibe ein Programm, welches einen String zufälliger Länge im Intervall [5,10] aus zufälligen Groß- und Kleinbuchstaben des englischen Alphabets erstellt und diesen ausgibt.

4.9. Schreibe ein Programm, welches einen String einliest und alle Vorkommen des ersten Buchstabens(ungeachtet der Groß-/Kleinschreibung) außer dem ersten durch `$` ersetzt.

[6] Von vorne und hinten gelesen das gleiche Wort

Tests:

Input	Output
`CamelCase.cc`	`Camel$ase.$$`
`restart`	`resta$t`
`Anaconda`	`An$cond$`

4.10. Schreibe ein Programm, welches zwei Uhrzeiten im Format `hh:mm:ss` einliest und im selben Format die Zeit dazwischen ausgibt. Tests:

Uhrzeit 1	Uhrzeit 2	Dazwischen
12:58:20	14:01:18	01:02:58
14:01:18	12:58:20	-01:02:58
14:23:12	18:32:21	04:09:09
18:32:21	14:23:12	-04:09:09
01:00:00	22:00:00	21:00:00
22:00:00	01:00:00	-21:00:00
1:2:3	6:5:4	05:03:01

4.11. Schreibe eine Programm, welches **einen** Satz einliest und den Satz in umgedrehter Reihenfolge wieder ausgibt. Satzzeichen müssen nicht berücksichtigt werden. Tests:

Input	Output
`Programmieren macht Spaß!`	`Spaß macht Programmieren!`
`Ab hier fies?`	`fies hier Ab?`
`ab bc de fg hi jk.`	`jk hi fg de bc ab.`
`wtf?  lol!  zomg!`	`Fehler`

4.12. Fortsetzung von Beispiel 2.1: Schreibe ein Programm, welches überprüft, ob sich jemand mit der richtigen Username/Kennwort-Kombination einloggt. Erstelle ein Tupel von Usernamen und eine Liste von Kennwörtern. User müssen das jeweils richtige Kennwort eingeben, um sich einzuloggen. Nach dem Einloggen wird der User begrüßt und kann sein Passwort ändern. Tut er dies, so kommt er erneut zum Login-Dialog.

4.13. Schreibe ein Programm, das einen String einliest und ermittelt, ob es sich um eine gültige IP-Adresse handelt. Eine gültige IPv4-Adresse genügt der Form xxx.xxx.xxx.xxx, wobei jeder Ziffernblock eine Zahl im Intervall [0, 255] beinhaltet.[7] Tests:

Input	Output
0.0.0.1	gültig
255.0.255.0	gültig
255.0.255	ungültig
256.0.0.0	ungültig
lol	ungültig

4.14. Schreibe ein Programm, welches Run-length encoding auf Strings realisiert. Zu schreiben ist sowohl eine Funktion `encode` als auch eine Funktion `decode`. Tests:

decoded	encoded
aaabbaaccc	a3b2a2c3
abc	a1b1c1

4.15. Schreibe ein Programm, welches einen String einliest, der ausschließlich aus Klammern besteht. Das Programm ermittelt, ob alle Klammern in der richtigen Reihenfolge korrekt geschlossen werden. Tests:

[7] Unter gewissen Voraussetzungen sind manche IP-Adressen reserviert; dies wird hier nicht überprüft.

Input	Output
`()[]{}`	`korrekt`
`)(`	`falsch`
`([)]`	`falsch`
`([{([{}])}])`	`korrekt`
`([{([{()[]([])}])}])`	`korrekt`
`abc.=`	`falsch`

4.16. Schreibe ein Programm, welches vom User einen IBAN einliest und überprüft, ob dieser korrekt ist. Die Eingabe soll dabei sowohl in der elektronisch übermittelten durchgehenden Form als auch in der lesbareren Form mit 4er Blöcken möglich sein.
Vorgangsweise am Beispiel AT61 1904 3002 3457 3201:

- Leerzeichen eliminieren. Resultat: AT611904300234573201
- Die ersten 4 Zeichen werden ans Ende des des Strings getauscht.
  Resultat: 1904300234573201AT61
- Jeder Buchstabe wird durch eine Zahl ersetzt. $A \to 10, B \to 11, C \to 12, \ldots, Z \to 35$
  Resultat: 1904300234573201**1029**61
- Die entstehende Zahl wird durch 97 dividiert. Ist der Rest der Division 1, so ist der IBAN gültig.

Tests:

Input	Output
`AT611904300234573201`	`gültig`
`AT61 1904 3002 3457 3201`	`gültig`
`AT61 1904 3002 3457 2301`	`ungültig`
`BA39 1290 0794 0102 8494`	`gültig`
`GB82 WEST 1234 5698 7654 32`	`ungültig`[8]

4.17. Schreibe ein Hangman-Spiel für zwei Spieler: Ein Spieler gibt ein Wort ein, dass der andere Spieler erraten muss. Zur versteckten Eingabe verwende die Funktion `getpass.getpass("Wort: ")`. Programmablauf:

```
Wort:
> p
P_____
Bisher getippt: ['P']
> y
PY____
Bisher getippt: ['P', 'Y']
```

5 Funktionen

5.1. Schreibe eine Funktion `inputInt(text)`, welche den User so lange mit der Aufforderung `text` zur Eingabe eines `int`s auffordert, bis dieser eine eingibt.

5.2. Schreibe Beispiel 4.12 mit Funktionen neu. Schreibe mindestens:

- `login(username, password)`

[8] Man müsste das Beispiel erweitern und für jedes Land die entsprechende Länge der IBAN abfragen

- `changePassword(newPassword)`

5.3. Schreibe ein Funktion `median`, welche eine Liste übergeben bekommt, die Liste sortiert und das mittlere Element retourniert; ist die Anzahl der Listenelemente gerade, so ist das arithmetische Mittel $\frac{a+b}{2}$ der beiden mittleren zu retournieren. Tests:

Input	Output
`[3, 1, 2]`	`2`
`[1, 2]`	`1.5`
`[7]`	`7`
`[3, 1, 4, 1, 5, 9, 2, 6]`	`3.5`

5.4. Überlege, welchen Inhalt die Variable `lst` zu welchem Zeitpunkt hat. Dann debugge das Programm und beobachte die Änderungen.

```
def assign(lst):
    lst = [0, 1]
    return None


def append(lst):
    lst.append(0)
    return None


def mutate(lst):
    lst[0] = 1
    return None


lst = [0]
assign(lst)
append(lst)
mutate(lst)
exit()
```

5.5. Bei der Entwicklung von neuronalen Netzen ist es erforderlich, dass alle Daten als Zahlen vorliegen. Daher müssen nicht-numerische Daten in Zahlen konvertiert werden.
Schreibe eine Funktion, welche eine Liste von Strings in eine Liste von Zahlen konvertiert, wobei stets demselben String dieselbe Zahl zugeordnet werden soll. Dabei wird dem ersten eindeutigen String `0` zugeordnet, dem zweiten `1` und so weiter. Tests:

Input	Output
`["G", "A", "J", "A", "J", "G"]`	`[0, 1, 2, 1, 2, 0]`
`["GER", "USA", "AUT", "GER", "AUT"]`	`[0, 1, 2, 0, 2]`

5.6. Schreibe eine Funktion `filter_duplikate`, welche eine Liste übergeben bekommt und alle mehrfach vorkommenden Elemente aus der Liste entfernt, sodass in der retournierten Liste jeder Eintrag der Originalliste exakt ein mal vorkommt. Die retournierte Liste ist zu sortieren. Tests:

Input	Output
`[2, 1, 2, 1]`	`[1, 2]`
`[3, 1, 2, 1, 2, 7, 5]`	`[1, 2, 3, 5, 7]`
`["Nein", "1AHIF", "1AHIF"]`	`['1AHIF', 'Nein']`

5.7. Schreibe eine Funktion `merge_lists`, welche zwei Listen übergeben bekommt und eine sortierte Liste aller Elemente, welche in beiden mindestens einmal vorkommen retourniert. Die Originallisten sollen unverändert bleiben. Tests:

Liste	Liste	Output
`[1, 2, 1]`	`[1, 3, 2]`	`[1, 2]`
`[True, False]`	`[False]`	`[False]`
`["Nein", "Ja"]`	`["Vielleicht"]`	`[]`
`"strings"`	`"listen"`	`['i', 'n', 's', 't']`

5.8. (a) Schreibe eine Funktion, welche das n-te Glied der Harmonischen ReiheH_n berechnet, wobei

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

Rufe diese Funktion auf, um folgende Tabelle zu erzeugen:

```
       n     |   H_n
-------------------------
           1 |  1.000000
          10 |  2.928968
         100 |  5.187378
        1000 |  7.485471
       10000 |  9.787606
      100000 | 12.090146
     1000000 | 14.392727
    10000000 | 16.695311
   100000000 | 18.997896
```

(b) Optimiere dein Programm, sodass jeder Quotient nur einmal berechnet wird.

5.9. Ergänze den Code.

```
import matplotlib.pyplot as plt
from random import randrange


def get_noten(prozent: list):
    """Bekommt eine Liste von Prozentwerten([0, 100])
    übergeben und retourniert eine Liste von Noten. """


def get_prozent():
    """Retourniert eine Liste von 36 zufälligen
       Prozentwerten([0,100]). """


def plot(prozent, noten):
    plt.rcParams.update({'font.size': 24})
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(6, 4))
```

```
        fig.dpi = 400
        ax1.hist(prozent, bins=10)
        ax1.set_title("Punkteverteilung")
        ax1.set_ylabel("Anzahl an Schülern")
        ax1.set_xlabel("Erreichte Punkte")
        ax2.bar(range(1, 6), height=noten)
        ax2.set_title("Notenverteilung")
        ax2.set_ylabel("Anzahl an Schülern")
        ax2.set_xlabel("Note")
        plt.show()

prozent = get_prozent()
noten = get_noten(prozent)
plot(prozent, noten)
```

5.10. (a) Schreibe eine Funktion `ist_datum_gültig(tag, monat, jahr)` und verwende diese, um den User so lange zur Eingabe eines Datums zu zwingen, bis es stimmt. Weise den User mit Fehlermeldungen auf Fehler hin. Schreibe mindestens:

- `gib_maximale_tage(monat, jahr)` - liefert zu jedem Monat den letzten Tag. z.B. `gib_maximale_tage(3, 2000) = 31`
- `ist_schaltjahr(jahr)` - liefert `True`, wenn es sich um ein Schaltjahr[9] handelt.

Beispielablauf:

```
Datum(TT.MM.JJJJ): nein
Datum ungültig
Datum(TT.MM.JJJJ): 2019-06-08
Datum ungültig
Datum(TT.MM.JJJJ): 1.13.2000
Datum ungültig
Datum(TT.MM.JJJJ): 29.2.1900
Datum ungültig
Datum(TT.MM.JJJJ): 31.12.2018
```

(b) Nach Eingabe eines gültigen Datums wird der nächste Tag ausgegeben, wobei die Ausgabe im Format ISO 8601[10] erfolgt. Tests:

Datum	Nächster Tag
5.4.2000	2000-04-06
29.2.2000	2000-03-01
31.12.2000	2001-01-01

5.11. Schreibe eine Funktion, welche ein Minimum, ein Maximum und eine Anzahl übergeben bekommt und eine Liste so vieler Zufallszahlen in dem Bereich erzeugt. Alle Parameter sollen Optional sein. Mögliche Ausgabe für das Intervall [3,5] und Anzahl 10:

```
[4, 3, 5, 3, 4, 4, 3, 5, 5, 3]
```

5.12. Erweitere Beispiel 4.11, sodass Satzzeichen an der ursprünglichen Position[11] bleiben. Tests:

[9] siehe Beispiel 2.7
[10] JJJJ-MM-TT
[11] also nach dem i-ten Wort

Input	Output
`To be, or not to be?`	`be to, not or be To?`
`lol, jk; wtf?`	`wtf, jk; lol?`
`wtf? lol! zomg!`	`Fehler`

5.13. Schreibe eine Funktion `ist_prim`, welche eine Zahl übergeben bekommt und ermittelt, ob es sich um eine Primzahl handelt oder nicht. Benutze diese Funktion um eine Liste aller Primzahlen im Bereich [0,50] auszugeben. Erwartete Ausgabe:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

5.14. Schreibe eine Funktion, welche ein Tupel mit der Anzahl der geraden und ungeraden Ziffern in einer Zahl retourniert. Tests:

Input	Output
12345	(2, 3)
13579	(0, 5)
0	(1, 0)

5.15. Schreibe Funktionen `encrypt` und `decrypt`, welche eine Cäsar-Verschlüsselung realisieren. Um wie viele Buchstaben gedreht werden soll und welcher Text zu verschlüsseln ist, sei einzulesen. Tests:

Klartext	Rotation	Output
Caesar	0	CAESAR
caesar	1	DBFTBS
caesar	3	FDHVDU
abcxyz	3	DEFABC
python	11	AJESZY

5.16. Schreibe eine Funktion, welches eine Liste übergeben bekommt, deren Elemente wieder Listen von Zahlen sind. Bestimme die Summe der Gesamtliste. Tests:

Input	Output
`[[1], [2], [3]]`	6
`[[0, 1, 2, 3], [4, 5, 6, 7]]`	28
`[]`	0

5.17. Schreibe eine Funktion, welches eine Liste übergeben bekommt, deren Elemente wieder Listen von Zahlen sind, einen Zeilenindex `z` und einen Spaltenindex `s`. Die Funktion berechnet die Summe aller Nachbarelemente des Elements $A_{z,s}$.
Verwende diese Funktion um mit der linken Matrix die rechte zu erhalten.

$$A = \begin{matrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{matrix} \longrightarrow \Sigma = \begin{matrix} 8 & 14 & 10 \\ 18 & 32 & 22 \\ 14 & 26 & 16 \end{matrix}$$

$$\Sigma_{1,0} = 0 + 1 + 4 + 6 + 7 = 18$$

6 Dictionaries

6.1. Schreibe ein Programm, das vom User einen String einliest und ein Dictionary ausgibt, dass jedem Buchstaben die Anzahl seiner Vorkommen zuordnet. Tests:

Input	Output
aaabbc	{'a': 3, 'b': 2, 'c': 1}
parallel	{'p': 1, 'a': 2, 'r': 1, 'l': 3, 'e': 1}
""	{}

6.2. Schreibe ein Programm, das vom User eine Zahl > 2 einliest und eine Primfaktorzerlegung in einem Dictionary ausgibt, wobei jedem Primfaktor die entsprechende Potenz zugeordnet ist. z.B.:

$$756 = 2^2 \cdot 3^3 \cdot 7^1 = \{2: 2, 3: 3, 7: 1\}$$

Tests:

Input	Output
756	{2: 2, 3: 3, 7: 1}
497097	{3: 4, 17: 1, 19: 2}
31	{31: 1}
1	Fehler

6.3. Schreibe ein Programm, dass vom User eine Zahl einliest und das Pascal'sche Dreieck bis zu dieser Zahl erzeugt. Beispielausgabe:

```
(a+b)^n bis n = 5
1
a + b
a^2  + 2ab + b^2
a^3  + 3a^2 b + 3ab^2  + b^3
a^4  + 4a^3 b + 6a^2 b^2  + 4ab^3  + b^4
a^5  + 5a^4 b + 10a^3 b^2  + 10a^2 b^3  + 5ab^4  + b^5
```

7 Filezugriffe

7.1. Ergänze den Code so, dass nur jene Zeilen der Datei ausgegeben werden, die den Suchstring enthalten.

```
def filter_lines(lines, pattern):
    """Bekommt eine Liste von Strings übergeben und
    retourniert nur jene Strings, welche das übergebene
       pattern enthalten. """

file = open("Nationalratswahl2017.csv", "r")
lines = [line.rstrip() for line in file.readlines()]
file.close()
filtered = filter_lines(lines, "Pölten")
for line in filtered:
    print(line)
```

7.2. JSON[12] ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Schreibe ein Programm, welches eine `csv`-Datei öffnet und ein JSON-Array aller Einträge der Datei ausgibt. Benutze für einen Test die Datei `table.csv`. Erwartetes Ergebnis (die Zeilenumbrüche sind nur der Lesbarkeit halber angeführt und sollten **nicht** enthalten sein):

[12] JavaScript Object Notation

```
[{'country': 'Austria', 'short': 'AUT', 'capital': 'Vienna'},
{'country': 'Germany', 'short': 'DE', 'capital': 'Berlin'},
{'country': 'France', 'short': 'FR', 'capital': 'Paris'},
{'country': 'Italy', 'short': 'IT', 'capital': 'Rome'},
{'country': 'Australia', 'short': 'AUS', 'capital': 'Canberra'}]
```

Versuche auch, einen Browser deiner Wahl zu öffnen, F12 zu drücken und in den Reiter Konsole zu wechseln. Kopiere deine Ausgabe dort hinein, bestätige die Eingabe und beobachte, was geschieht. Wir haben erfolgreich Daten mit dem Browser ausgetauscht![13]

7.3. Schreibe eine Funktion, welche eine Liste übergeben bekommt und die Länge des längsten Eintrags retourniert. Ermittle mit Hilfe dieser Funktion die Länge der längsten Zeile der Datei `Nationalratswahl2017.csv`. Tests:

Input	Output
`["Java", "Python", "C"]`	6
`[(), [42, 123, 81], {4: 5, "ja": "nein"}]`	3
`Nationalratswahl2017.csv`	141

7.4. Schreibe ein Programm, welches aus der Datei `Nationalratswahl2017.csv` für jede angeführte Partei diejenigen Bezirke, bei denen die Partei das beste relative Ergebnis eingefahren hat, also bei denen der Anteil der Partei an den abgegebenen Stimmen maximal ist. Gib das Ergebnis in Tabellenform aus. Erwartetes Ergebnis:

```
SPÖ   | 67.57 | Tschanigraben
ÖVP   | 83.33 | Hinterhornbach
FPÖ   | 52.85 | Deutsch-Griffen
GRÜNE | 16.67 | Wahlkarten - Vorarlberg Nord
NEOS  | 19.35 | Dalaas
PILZ  | 13.51 | Wahlkarten - Burgenland Nord
```

7.5. Ergänze den Code so, dass ein Säulendiagramm entsteht, bei dem jeder Partei der Anteil aller abgegebenen Stimmen zugewiesen wird.

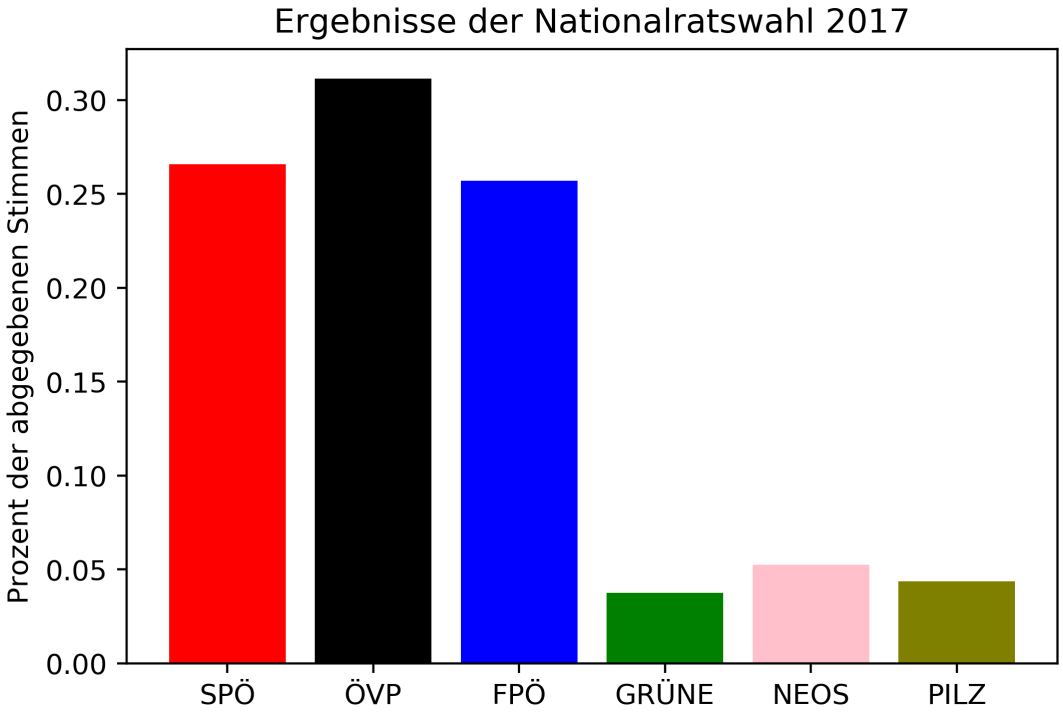
```
import matplotlib.pyplot as plt
import numpy as np


def get_data():
    """Liefert im Tupel(Abgegebene Stimmen, [% der Stimmen],
    [Parteien] die Ergebnisse der Nationalratswahl 2017."""

(total_votes, percent, parties) = get_data()
x_axis = np.arange(6)
plt.figure(dpi=400, figsize=(6, 4))
plot = plt.bar(x_axis, height=percent, color=[
    "red", "black", "blue", "green", "pink", "olive"])
plt.xticks(x_axis, parties)
plt.title("Ergebnisse der Nationalratswahl 2017")
plt.ylabel("Prozent der abgegebenen Stimmen")
plt.show()
```

[13] Unsere Lösung ist zwar nah dran, aber nicht immer richtig. z.B. sollte aus `True true` werden.

Erwartetes Ergebnis:



7.6. Schreibe ein Programm, dass eine `csv`-Datei einliest und den Inhalt der csv-Datei in einer Tabelle ausgibt.

country	short	capital
Austria	AUT	Vienna
Germany	DE	Berlin
France	FR	Paris
Italy	IT	Rome
Australia	AUS	Canberra