

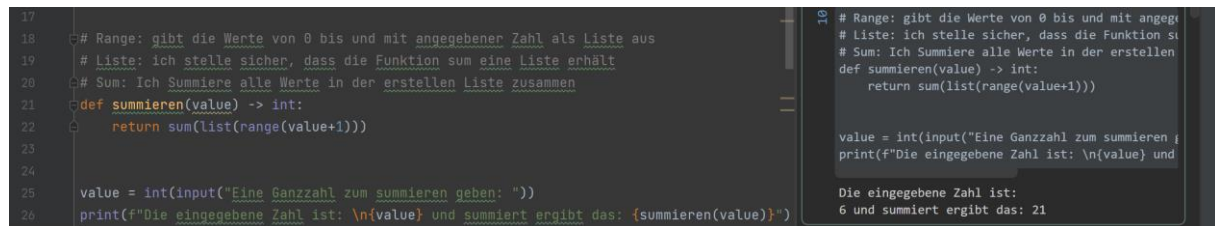
Aufgabenblatt 7

In diesem Aufgabenblatt werden die Aufgaben bezüglich Funktionen der Schulwoche 8 gezeigt.

Aufgabe 31

Für eine eingegebene positive ganze Zahl n (Integer) soll die Summe von $i=1$ bis n berechnet werden und zurückgegeben werden. Schreiben Sie eine Python-Funktion dafür.

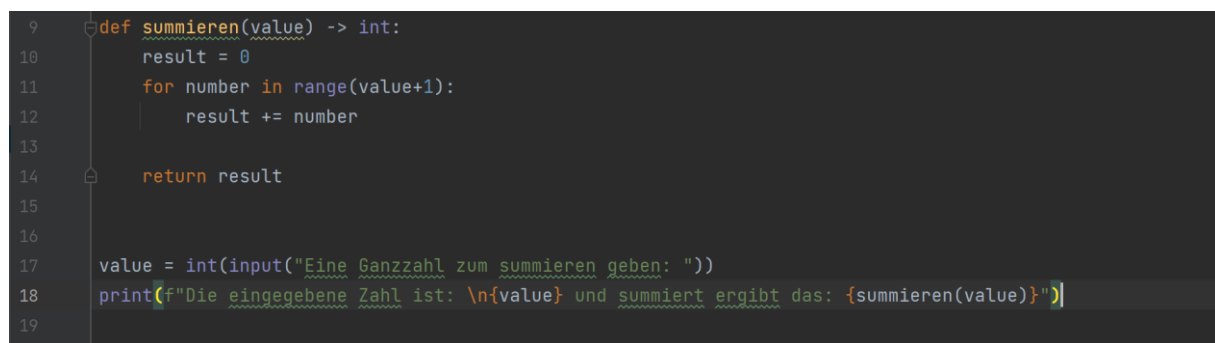
Lösung:



```
17 # Range: gibt die Werte von 0 bis und mit angegebener Zahl als Liste aus
18 # Liste: ich stelle sicher, dass die Funktion sum eine Liste erhält
19 # Sum: Ich Summiere alle Werte in der erstellen Liste zusammen
20 def summieren(value) -> int:
21     return sum(list(range(value+1)))
22
23
24
25 value = int(input("Eine Ganzzahl zum summieren geben: "))
26 print(f"Die eingegebene Zahl ist: \n{value} und summiert ergibt das: {summieren(value)}")
```

Abbildung 1 - Aufgabe 31

Ich habe das als Einzeiler implementiert, weil ich die Herausforderung mag. Um aber nicht nur unwartbaren und unerwünschten Code zu produzieren, kann der nicht Einzeilige Versuch im Python File „Aufgabe31.py“ zu finden sein:



```
9 def summieren(value) -> int:
10     result = 0
11     for number in range(value+1):
12         result += number
13
14     return result
15
16
17 value = int(input("Eine Ganzzahl zum summieren geben: "))
18 print(f"Die eingegebene Zahl ist: \n{value} und summiert ergibt das: {summieren(value)}")
19
```

Abbildung 2 - Aufgabe 31 Kein 1 Zeiler

Das Prinzip ist das gleiche wie beim Einzeiler. Mit einer For-Schleife iteriere ich durch alle Zahlen und addiere diese zusammen. Der Unterschied zum Einzeiler liegt, darin, dass ich die Zahlen nicht in eine Liste speichere, sondern direkt addiere, dafür ist die „sum“ Funktion beim Einzeiler zuständig.

Aufgabe 33

Schreiben Sie eine Python-Funktion, die als Parameter einen String bekommt und sowohl die Länge des Strings, als auch die Anzahl der Kleinbuchstaben, als auch die Anzahl der Grossbuchstaben darin zurückliefert.

Lösung:



```
61 #%%
62
63 def auswertenString(string):
64     return len(string), sum(map(str.islower, string)), sum(1 for c in string if c.isupper)
65
66 text = input("Irgendeinen String eingeben: ")
67 leange, anzKlein, anzGross = auswertenString(string=text)  text: Es funktioniert ganz gut
68
69 print(f"Der String \"{text}\" ist {leange} Zeichen lang. \nBesitzt {anzKlein} Kleinbuchstaben und {anzGross} Grossbuchstaben")
```


Abbildung 4 - Aufgabe 33

Auch hier habe ich einen Einzeiler hingekriegt, aber dieser ist echt unschön und habe deswegen in der Datei «Aufgabe33.py» die einzelnen Resultate in Variablen abgespeichert.

Aufgabe 34

Schreiben Sie eine Funktion, die eine gegebene Liste (Integers, Floats oder Strings) in aufsteigender und absteigender Reihenfolge sortiert, die beiden Listen zusammenfügt und die neue Liste zurückliefert.

Lösung:



```
79 def sortingLists(list) -> list:
80     return sorted(list) + sorted(list)[::-1]
81
82 testList = [3,67,1,8,3,7]
83 print(sortingLists(list=testList))  testList: [3, 67, 1, 8, 3, 7]
```

Abbildung 5 - Aufgabe 34

Diese Aufgabe mit einer Zeile zu lösen ist, meiner Meinung sogar schlau, denn es ist klar was gemacht wird und spart somit Speicherplatz.

Im File „Aufgabe34.py“ habe ich die Funktion noch in mehreren Zeilen programmiert, ich aber nur gemacht habe um meine Ergebnisse zu kontrollieren.

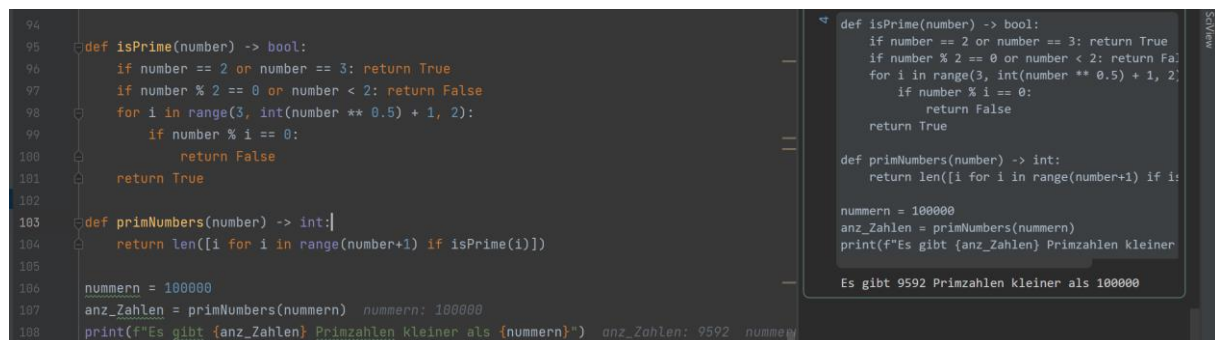
Aufgabe 35

Schreiben Sie eine Python-Funktion, die von einer gegebenen ganzen positiven Zahl n testet, ob diese eine Primzahl ist. Ist dies der Fall, soll die Funktion True zurückliefern, ansonsten False.

Schreiben Sie eine weitere Funktion **primNumbers**, die für eine eingegebene ganze positive Zahl m die Anzahl der Primzahlen kleiner m zählt und diese Anzahl zurückliefert. Wieviele Primzahlen kleiner 10000 gibt es?

Lösung:

Ich habe 2 Funktionen programmiert. Einmal das Überprüfen, ob die eingegebene Zahl eine Primzahl ist und die Anzahlberechnung der Primzahlen (diese Funktion in einer Zeile).



```
94
95 def isPrime(number) -> bool:
96     if number == 2 or number == 3: return True
97     if number % 2 == 0 or number < 2: return False
98     for i in range(3, int(number ** 0.5) + 1, 2):
99         if number % i == 0:
100             return False
101     return True
102
103 def primNumbers(number) -> int:
104     return len([i for i in range(number+1) if isPrime(i)])
105
106 nummern = 100000
107 anz_Zahlen = primNumbers(nummern)    nummern: 100000
108 print(f"Es gibt {anz_Zahlen} Primzahlen kleiner als {nummern}")    anz_Zahlen: 9592    nummern: 100000
```

```
def isPrime(number) -> bool:
    if number == 2 or number == 3: return True
    if number % 2 == 0 or number < 2: return False
    for i in range(3, int(number ** 0.5) + 1, 2):
        if number % i == 0:
            return False
    return True

def primNumbers(number) -> int:
    return len([i for i in range(number+1) if isPrime(i)])

nummern = 100000
anz_Zahlen = primNumbers(nummern)
print(f"Es gibt {anz_Zahlen} Primzahlen kleiner als {nummern}")

Es gibt 9592 Primzahlen kleiner als 100000
```

Abbildung 6 - Aufgabe 35

Die Funktion `isPrime(number)` habe ich aus verständnis und wartungsgründen nicht in einer Zeile programmiert.

Ich versuche so schnell und optimiert wie möglich zu überprüfen ob eine Zahl eine Primzahl ist oder nicht und das als boolean zurückzugeben. (Stichwort: Sieb des Eratosthenes)

Die zweite Funktion, welche mir die Anzahl an Primzahlen unterhalb der angegebenen Zahl geben sollte, habe ich in einer Zeile implementiert und in der Datei «Aufgabe35.py» mit einem Iterator.

Der Einzeiler funktioniert, indem alle Primzahlen bis und mit zur angegebenen Zahl in eine Liste gespeichert werden. Danach wird die Länge der Liste bestimmt und zurückgegeben.