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:

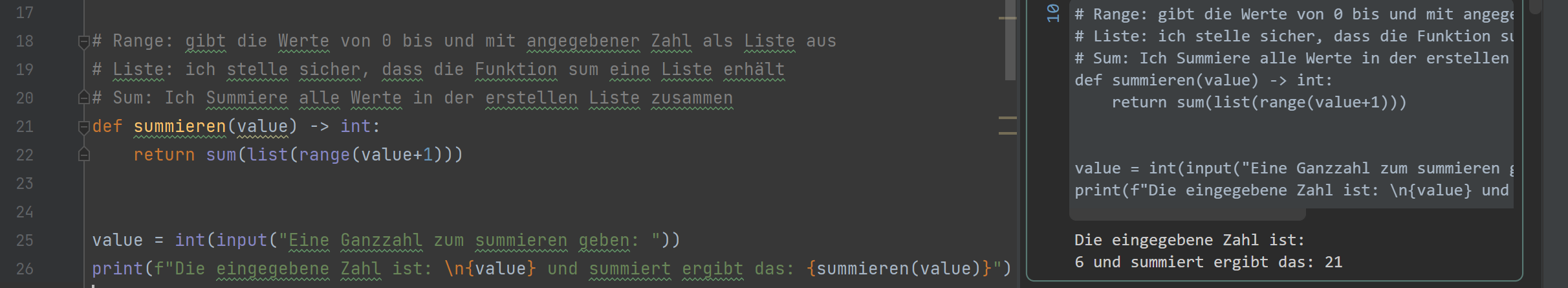


Abbildung 1 - Aufgabe 31

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:

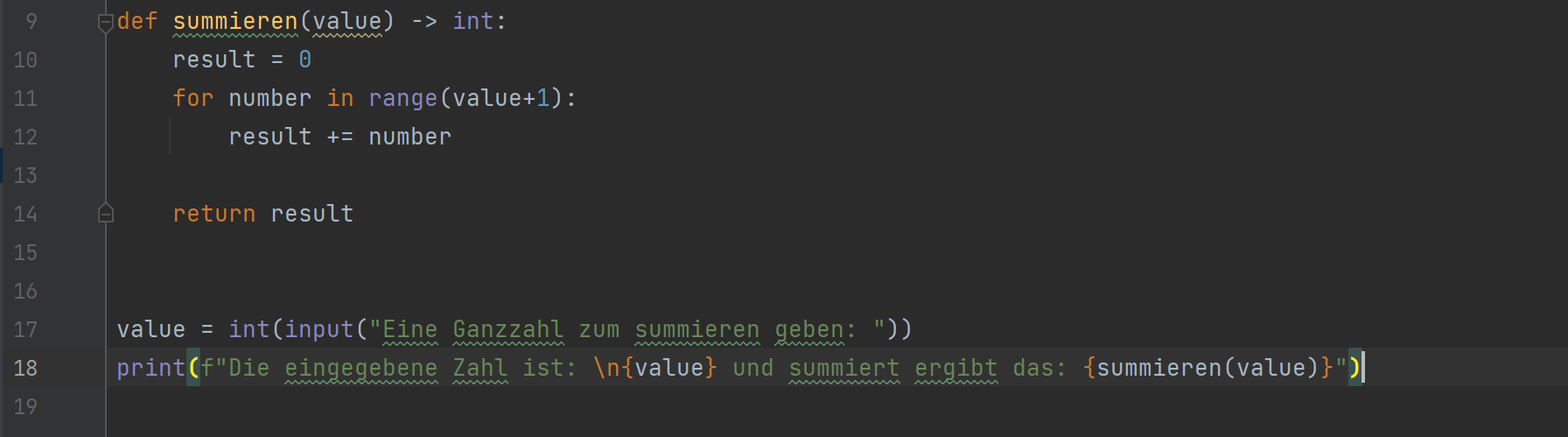


Abbildung 2 - Aufgabe 31 Kein 1 Zeiler

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 32

1. Schreiben Sie eine Python-Funktion, die für eine positive ganze Zahl n die Fakultät von n berechnet (mathematisch ausgedrückt n!), also das Produkt der Zahlen von k=1 bis n.
2. Schreiben Sie eine weitere Funktion für die Fakultät von n, die die sogenannte Stirling-Formel

verwendet. Die Konstante e ist hierbei die Eulersche Zahl, die Zahl pi sollte allen als Kreiszahl bekannt sein. Diese Konstanten kann als Python-Konstanten im Internet finden.

1. Vergleichen Sie die beiden Funktionen für die Werte zwischen 1 und 100, also bilden Sie die Differenz dieser Funktionswerte. Was passiert mit der Differenz für grösser werdende n?

Lösung:

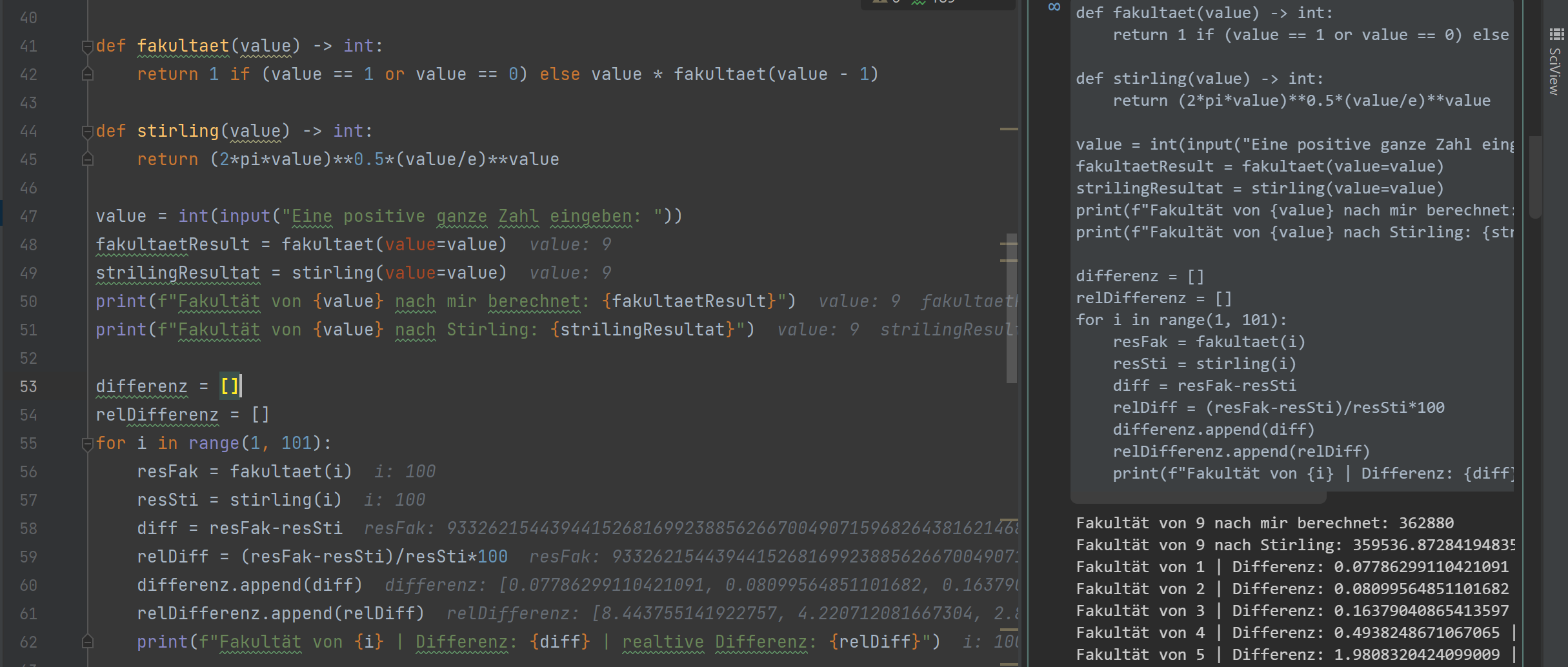


Abbildung 3 - Aufgabe 32

Ich habe alles wieder als Einzeiler versucht (und geschafft) und im File „Aufgabe32.py“ normal, mit for Schleifen programmiert.

Wichtig zu erwähnen ist nur, dass im File „Aufgabe32.py“ sowohl ein Rekursiver als auch ein Iterativer Ansatz/Lösung zur Fakultätsberechnung zu finden ist.

Der Einzeiler bei der Funktion „fakultaet(value)“ erwartet einen integer und funktioniert folgendermassen:

Wenn die Zahl 1 oder 0 ist, dann wird 0 zurückgegeben, ansonsten wird die aktuelle Zahl mit dem Resultat der Methode „fakultaet(value)“ multipliziert. Damit nun auch durchiteriert wird, wird beim rekursiven Aufruf einfach die Zahl aber um 1 kleiner mitgegeben.

Die Differenz habe ich als relative und nicht relative (also nicht prozentuale) Differenz ausgegeben. Wie ich erkennen konnte ging diese relative Differenz in Richtung 0.

# 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:

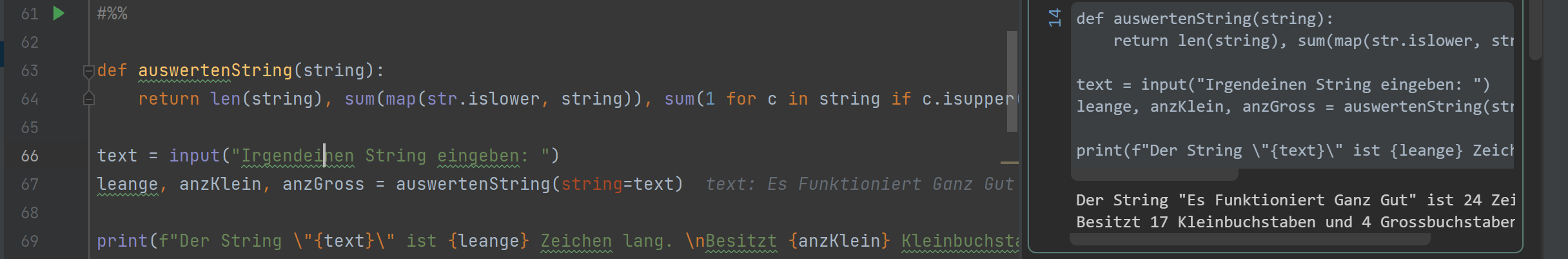


Abbildung 4 - Aufgabe 33

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:

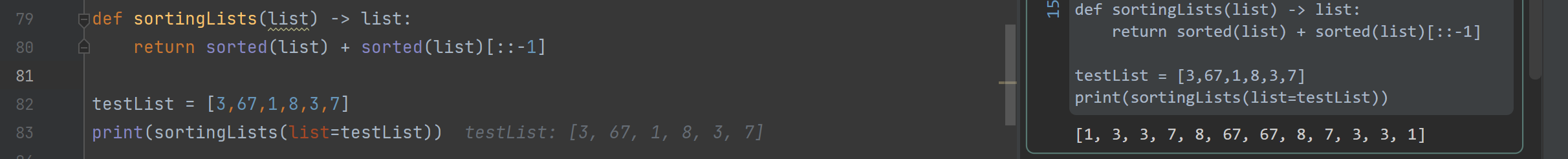


Abbildung 5 - Aufgabe 34

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).

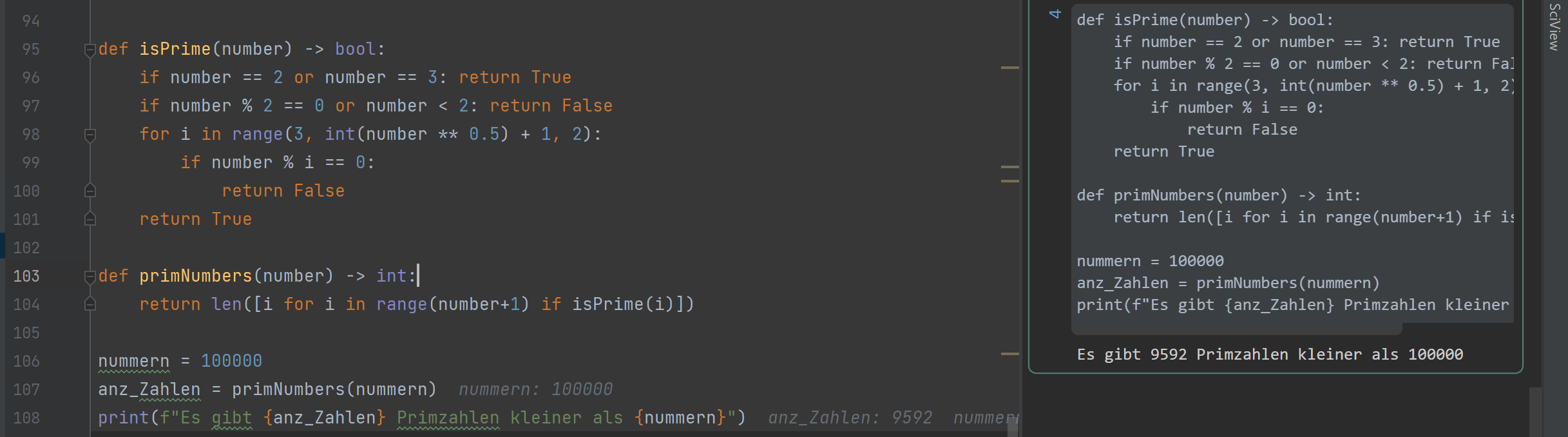


Abbildung 6 - Aufgabe 35

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 «Aufagbe35.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.