Aufgabenblatt 6

In diesem Aufgabenblatt sind die Aufgaben 26 bis 30 gelöst, welche in der Schulwoche 7 erledigt werden.

# Aufgabe 26

Für eine Liste an Studenten soll nach einer Klausur eine Punkteliste angelegt und Student für Student mit Werten gefüllt werden. Dabei sollen keine String-Eingaben erlaubt sein. Schreiben Sie Code, der mit einem try… except… else… abfängt, ob jemand einen Integer oder eine Floating Point Zahl eingegeben hat.

Lösung:

Ich bin generell kein Freund von Programmunterbrüchen, da ich aus Java komme und habe deswegen eine zweite Version programmiert (siehe Skript Aufgabe26\_OhneExceptions.py). Die Lösung zur Aufgabe, wenn ein Error im «except» behandelt werden soll, kann so aussehen:

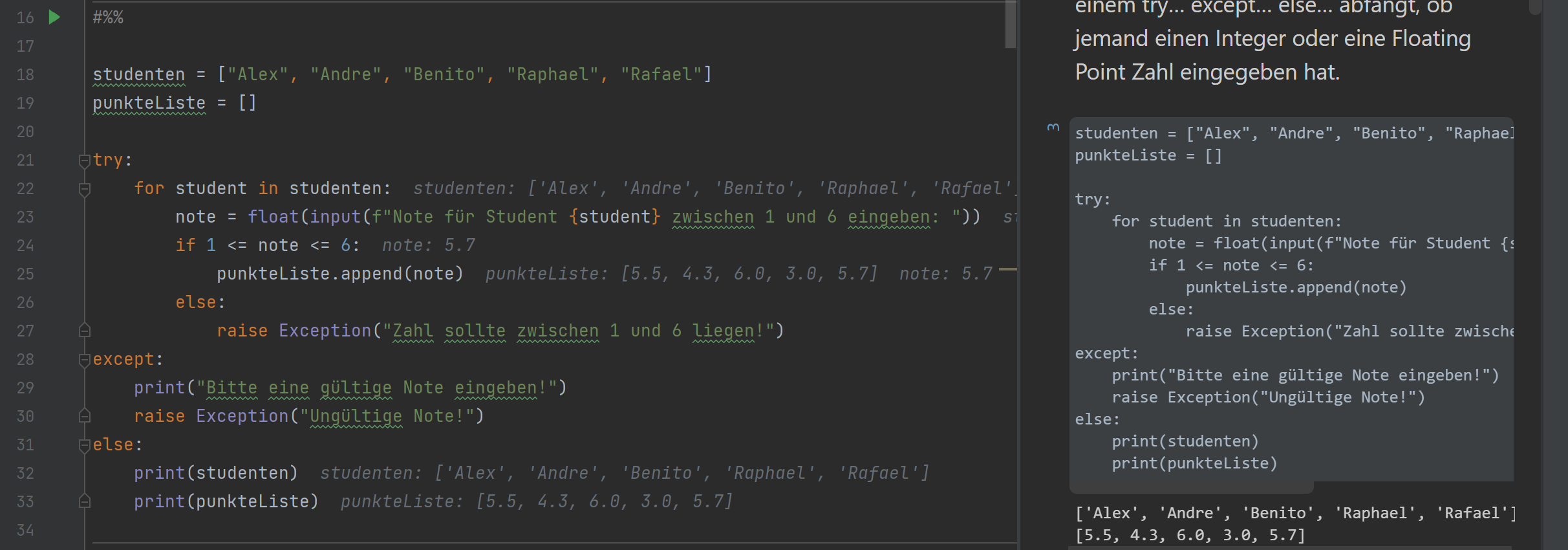


Abbildung - Aufgabe 26

Wenn nun die Expection nicht fallen sollte, dann muss der eingegebene String überprüft werden, was so aussehen kann. Ich versuche meine Probleme immer so zu lösen:

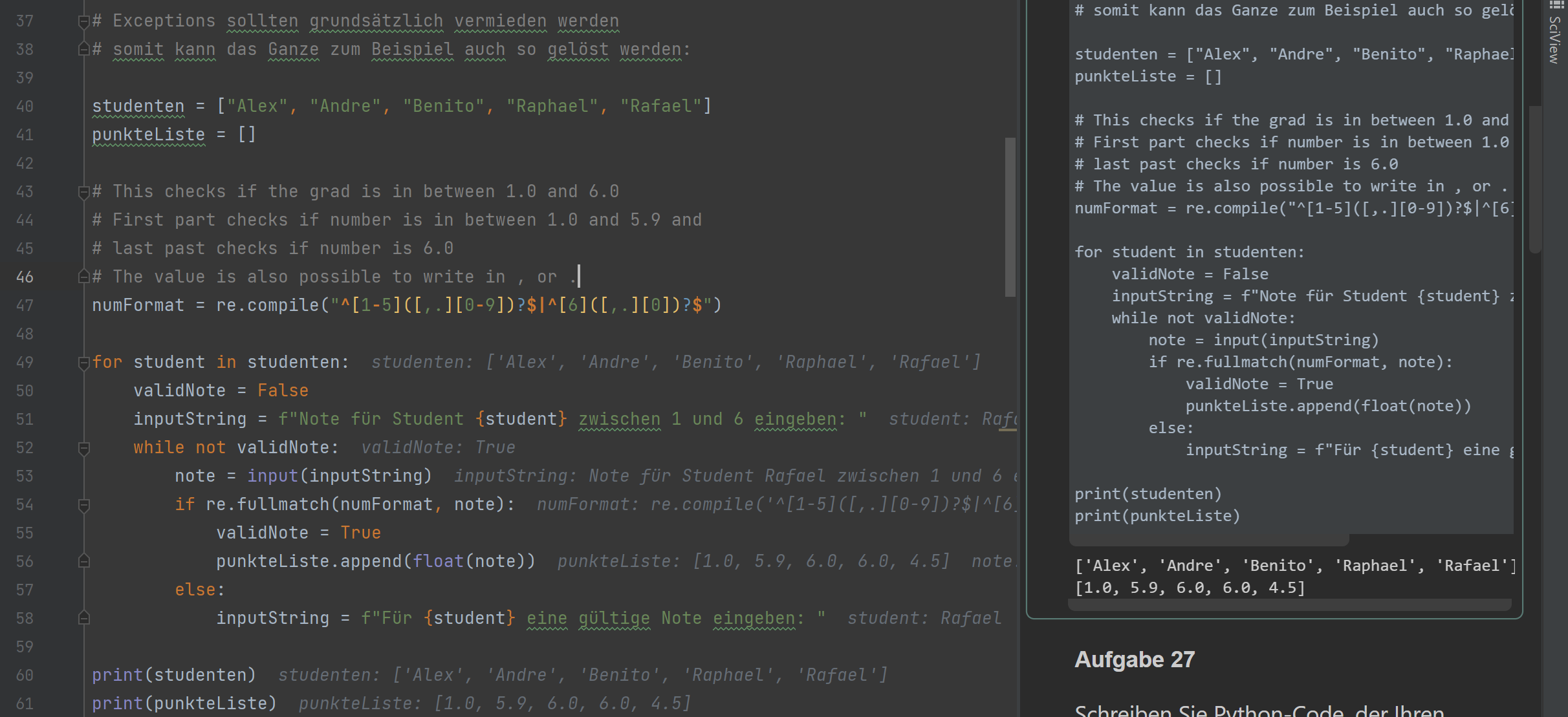


Abbildung - Aufgabe 26 ohne Exception

Wichtig ist nur der Regex Ausdruck. Bis dieser Funktioniert hat habe ich etwa 2 Stunden benötigt.

# Aufgabe 27

Schreiben Sie Python-Code, der Ihren Namen in eine Zahl verschlüsselt. Hierbei soll jedes Zeichen aus Ihrem Namen mit der ord()-Funktion in einen Dezimalwert aus der ASCII-Tabelle übersetzt werden. Multiplizieren Sie alle diese Dezimalwerte und dividieren Sie das Produkt durch die Summe aller Dezimalwerte. Die Zahl vor dem Komma sei die Verschlüsselungszahl. Geben Sie diese aus.

Lösung:

Ich habe eine for Schleife benutzt und in dieser einfach jeden einzelnen Buchstaben im String berechnet:

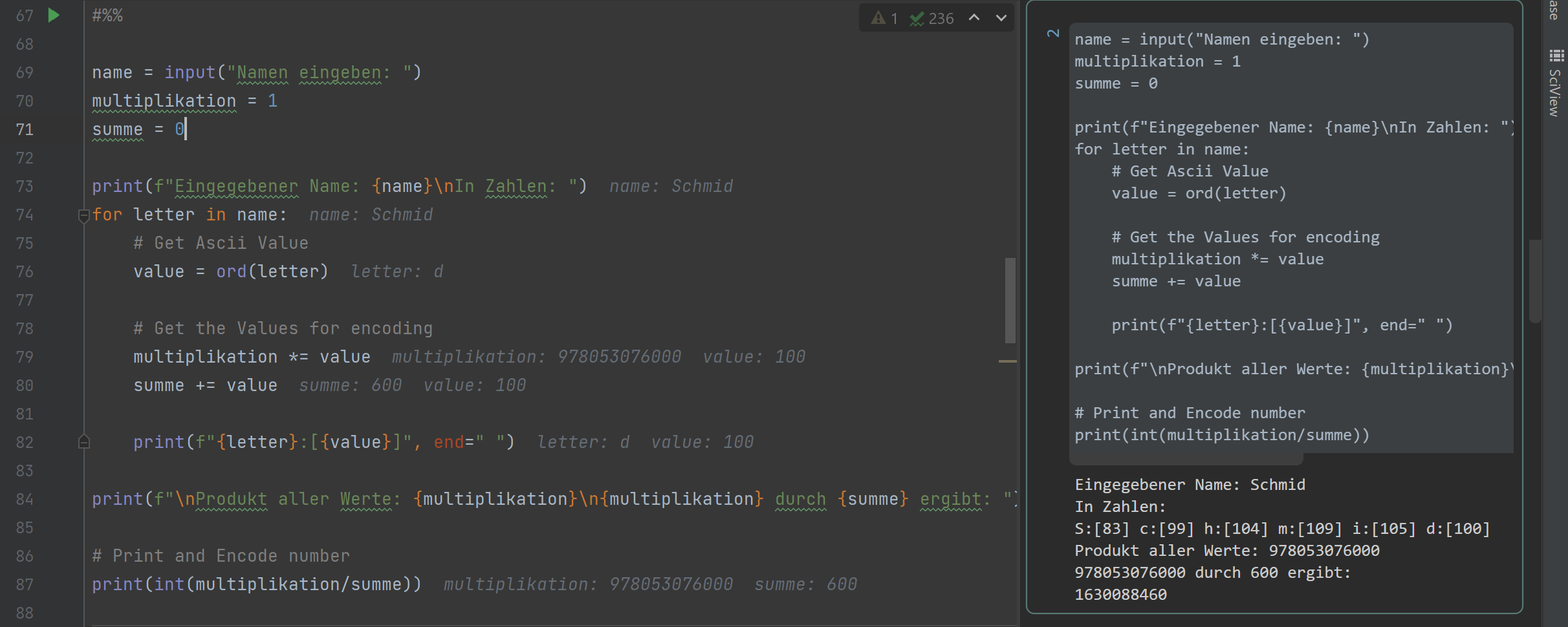


Abbildung - Aufgabe 27

Hier gibt es keine zusätzliche Version, das Programm kann auch als Skript verwendet werden (Aufgabe27.py).

# Aufgabe 28

Schreiben Sie Python-Code, der mit einem regulären Ausdruck überprüft, ob eine eingegebene User ID von der Form.

(a) 1 bis 3 Grossbuchstaben gefolgt von genau 5 Ziffern ist

(b) 2 Grossbuchstaben gefolgt von genau 1 Ziffer ist.

Lösung:

Mit zwei Validationsstrings, welche die beiden unterschiede mit Regex definieren ist die Aufgabe auch schon gelöst:

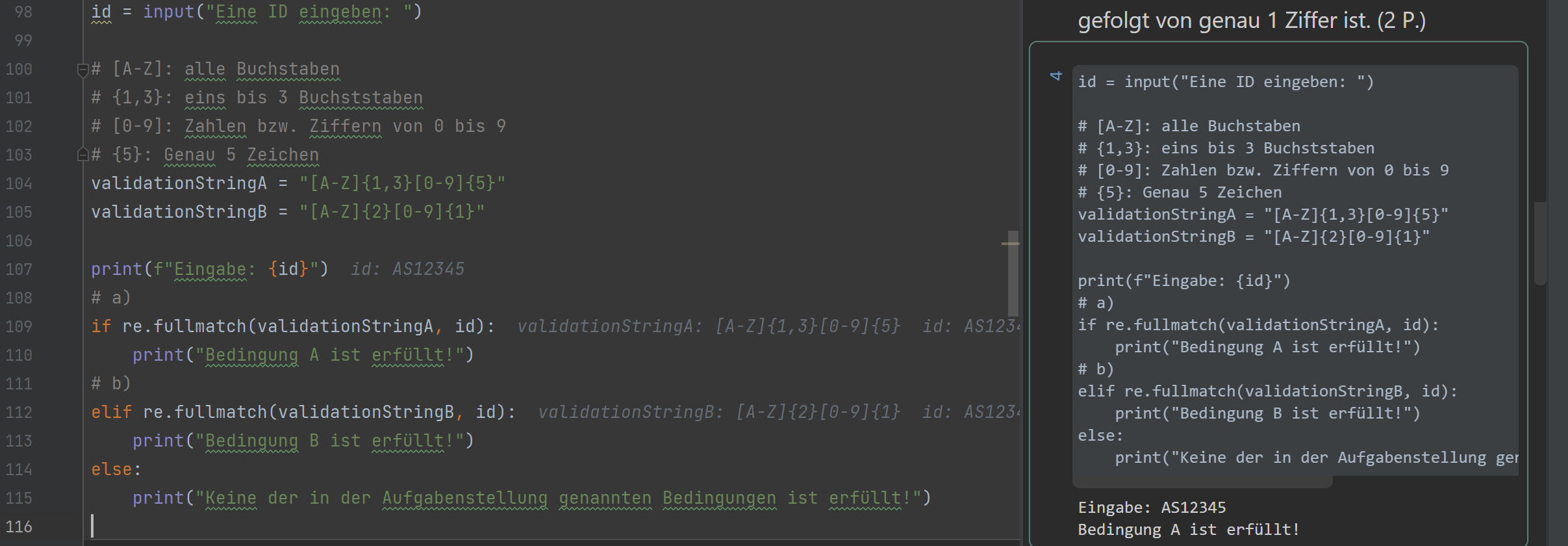


Abbildung - Aufgabe 28

Der Code kann im Skript «Aufgabe28.py» im .zip nachvollzogen werden.

# Aufgabe 29

Gegeben seien 2 Dictionaries D1 und D2. Schreiben Sie Python-Code, der überprüft

(a) Ob die beiden Dictionaries D1 und D2 gleich sind**.**

(b) Ob sie sich unterscheiden und eine Liste U mit Unterscheidungsindizes ausgeben.

Lösung:

Die Aufgabe a) ist sehr einfach, da der „=“-Operator verwendet werden kann:

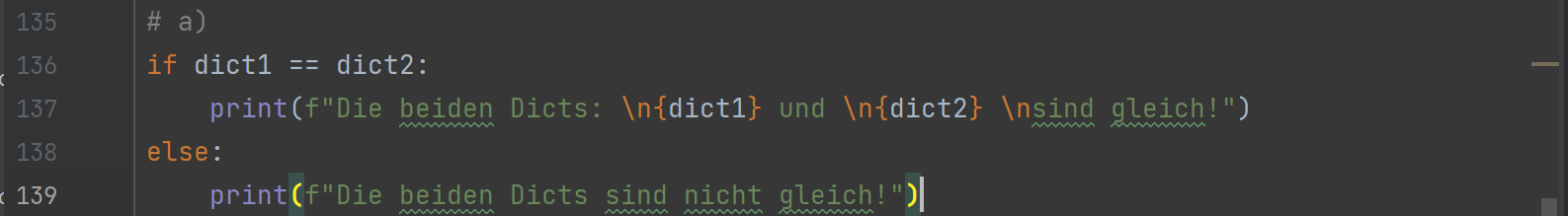


Abbildung - Aufgabe 29 a)

Die Aufgabe b) allerdings habe ich einmal mit einem One-Liner gelöst und einmal auf mehreren Zeilen.

Ich werde hier nur den One-Liner besprechen, da dieser interessanter ist. 😊

print([(key in list(d1.keys())) and (key in list(d2.keys())) and (d1[key] == d2[key]) for key in list(dict.fromkeys(list(d1.keys()) + list(d2.keys())))])

Dieser Code ist in einer Zeile geschrieben und passt nicht auf eine Word Zeile.

Der Ausdruck beginnt mit «[« und endet mit «]» was eine Liste zurück gibt. Man kann auch mit «{« und «}» einen Dict zurückgeben.

Im Ausdruck habe ich zwei Teile: den Rückgabewert und die Wertbeschaffung. Der Rückgabewert sollte «True» oder «False» ergeben und es werden die Schlüssel, Werte und allfällige nicht enthaltenen Werte geprüft (also falls im einen Dictionary mehr Werte als im anderen enthalten sind).

Der zweite Teil ist das Erschaffen des Schlüssels, mit welchem ich vergleiche ob die Werte in den Dictionarys gleich sind. Da der Dict nicht iterierbar ist, merge ich die Schlüssel in eine Liste und iteriere über die Liste. (Das mergen der Schlüssel ist wichtig, da es sein kann, dass im einen Dict mehr Werte sind, als im anderen)

Der Einzeiler ist nicht zu empfehlen, kann aber im Skript «Aufgabe29\_OneLiner.py» angeschaut werden, während dem der andere Ansatz (viel Länger) im Skript «Aufgabe29.py» angeschaut werden kann.

# Aufgabe 30

Gegeben sei eine Liste L mit Floating Point Zahlen. Schreiben Sie Python-Code, um die Liste L in aufsteigender Reihenfolge in eine sortierte Liste S zu überführen. Beide Listen haben die Länge N. Berechnen Sie nach der Sortierung den sogenannten **Mean-Squared Error (MSE)** zwischen der Originalliste L und der sortierten Liste S. Die Elemente der Liste L seien hierbei mit y bezeichnet und die der sortierten Liste S mit y-Dach.

Lösung:

Meiner Meinung nach war der Einzeiler in Aufgabe 29 wesentlich schwieriger als die Aufgabe 30, aber ich denke dafür bin ich selbst Schuld. Hier noch meine Lösungen.

Einmal habe ich die Bibliothek sklearn verwendet und die Funktion mean\_sqaured\_error() und als zweites habe ich die Mathematische Formel in den Code umgesetzt:

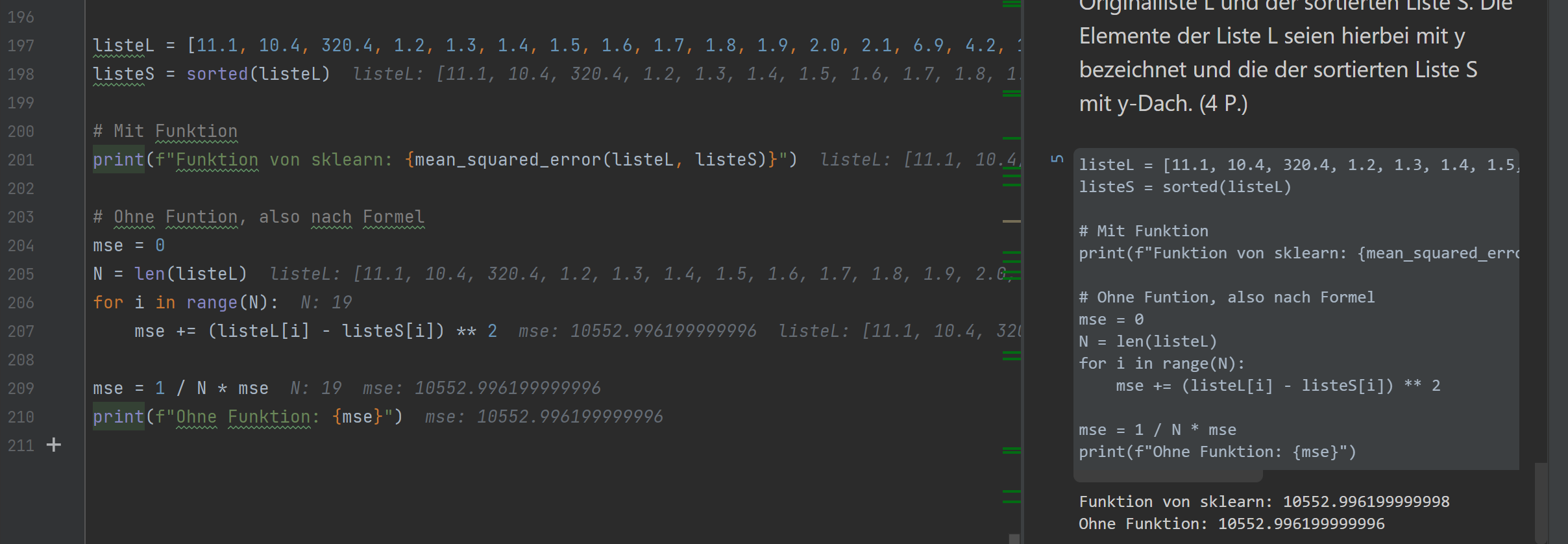


Abbildung - Aufgabe 30

Die Formel ist noch hier zu sehen:



Abbildung - Formel Aufgabe 30

Die Lösungen sind auch im Skript „Aufgabe30.py“ zu finden.