

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:

```

16 #%%
17
18 studenten = ["Alex", "Andre", "Benito", "Raphael", "Rafael"]
19 punkteliste = []
20
21 try:
22     for student in studenten:
23         note = float(input(f"Note für Student {student} zwischen 1 und 6 eingeben: "))
24         if 1 <= note <= 6:
25             punkteliste.append(note)
26         else:
27             raise Exception("Zahl sollte zwischen 1 und 6 liegen!")
28 except:
29     print("Bitte eine gültige Note eingeben!")
30     raise Exception("Ungültige Note!")
31 else:
32     print(studenten)
33     print(punkteliste)
34

```

einem try... except... else... abfangt, ob jemand einen Integer oder eine Floating Point Zahl eingegeben hat.

```

studenten = ["Alex", "Andre", "Benito", "Raphael", "Rafael"]
punkteliste = []

try:
    for student in studenten:
        note = float(input(f"Note für Student {student} zwischen 1 und 6 eingeben: "))
        if 1 <= note <= 6:
            punkteliste.append(note)
        else:
            raise Exception("Zahl sollte zwischen 1 und 6 liegen!")
except:
    print("Bitte eine gültige Note eingeben!")
    raise Exception("Ungültige Note!")
else:
    print(studenten)
    print(punkteliste)

['Alex', 'Andre', 'Benito', 'Raphael', 'Rafael']
[5.5, 4.3, 6.0, 3.0, 5.7]

```

Abbildung 1 - Aufgabe 26

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

```

37 # Exceptions sollten grundsätzlich vermieden werden
38 # somit kann das Ganze zum Beispiel auch so gelöst werden:
39
40 studenten = ["Alex", "Andre", "Benito", "Raphael", "Rafael"]
41 punkteliste = []
42
43 # This checks if the grad is in between 1.0 and 6.0
44 # First part checks if number is in between 1.0 and 5.9 and
45 # last part checks if number is 6.0
46 # The value is also possible to write in , or .
47 numFormat = re.compile("^[1-5]([.][0-9])?|[6]([.][0-9])?$")
48
49 for student in studenten:
50     validNote = False
51     inputString = f"Note für Student {student} zwischen 1 und 6 eingeben: "
52     while not validNote:
53         note = input(inputString)
54         if re.fullmatch(numFormat, note):
55             validNote = True
56             punkteliste.append(float(note))
57         else:
58             inputString = f"Für {student} eine gültige Note eingeben: "
59
60 print(studenten)
61 print(punkteliste)

```

somit kann das Ganze zum Beispiel auch so gelöst werden:

```

studenten = ["Alex", "Andre", "Benito", "Raphael", "Rafael"]
punkteliste = []

# This checks if the grad is in between 1.0 and 6.0
# First part checks if number is in between 1.0 and 5.9 and
# last part checks if number is 6.0
# The value is also possible to write in , or .
numFormat = re.compile("^[1-5]([.][0-9])?|[6]([.][0-9])?$")

for student in studenten:
    validNote = False
    inputString = f"Note für Student {student} zwischen 1 und 6 eingeben: "
    while not validNote:
        note = input(inputString)
        if re.fullmatch(numFormat, note):
            validNote = True
            punkteliste.append(float(note))
        else:
            inputString = f"Für {student} eine gültige Note eingeben: "

print(studenten)
print(punkteliste)

['Alex', 'Andre', 'Benito', 'Raphael', 'Rafael']
[1.0, 5.9, 6.0, 6.0, 4.5]

```

Abbildung 2 - Aufgabe 26 ohne Exception

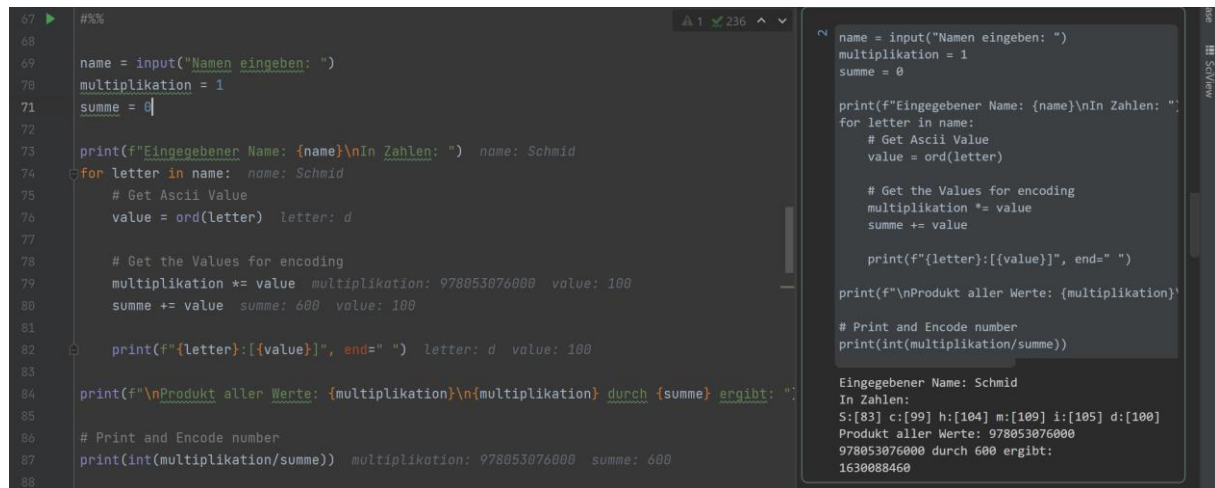
Wichtig ist nur der Regex Ausdruck. Bis dieser funktioniert hat habe ich etwa 2 Stunden benötigt.

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:



```
67 #70%
68
69 name = input("Namen eingeben: ")
70 multiplikation = 1
71 summe = 0
72
73 print(f"Eingegebener Name: {name}\nIn Zahlen: ")
74 for letter in name:
75     # Get Ascii Value
76     value = ord(letter)
77
78     # Get the Values for encoding
79     multiplikation *= value
80     summe += value
81
82     print(f"{letter}:{value}", end=" ")
83
84 print(f"\nProdukt aller Werte: {multiplikation}\n{multiplikation} durch {summe} ergibt: ")
85
86 # Print and Encode number
87 print(int(multiplikation/summe))
88
```

```
name = input("Namen eingeben: ")
multiplikation = 1
summe = 0

print(f"Eingegebener Name: {name}\nIn Zahlen: ")
for letter in name:
    # Get Ascii Value
    value = ord(letter)

    # Get the Values for encoding
    multiplikation *= value
    summe += value

    print(f"{letter}:{value}", end=" ")

print(f"\nProdukt aller Werte: {multiplikation}")

# Print and Encode number
print(int(multiplikation/summe))

Eingegebener Name: Schmid
In Zahlen:
S:[83] c:[99] h:[104] m:[109] i:[105] d:[100]
Produkt aller Werte: 978053076000
978053076000 durch 600 ergibt:
1630083460
```

Abbildung 3 - 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:

```
98 id = input("Eine ID eingeben: ")
99
100 # [A-Z]: alle Buchstaben
101 # {1,3}: eins bis 3 Buchstaben
102 # [0-9]: Zahlen bzw. Ziffern von 0 bis 9
103 # {5}: Genau 5 Zeichen
104 validationStringA = "[A-Z]{1,3}[0-9]{5}"
105 validationStringB = "[A-Z]{2}[0-9]{1}"
106
107 print(f"Eingabe: {id}") id: AS12345
108 # a)
109 if re.fullmatch(validationStringA, id): validationStringA: [A-Z]{1,3}[0-9]{5} id: AS12345
110     print("Bedingung A ist erfüllt!")
111 # b)
112 elif re.fullmatch(validationStringB, id): validationStringB: [A-Z]{2}[0-9]{1} id: AS12345
113     print("Bedingung B ist erfüllt!")
114 else:
115     print("Keine der in der Aufgabenstellung genannten Bedingungen ist erfüllt!")
116
```

gefolgt von genau 1 Ziffer ist. (2 P.)

```
id = input("Eine ID eingeben: ")
# [A-Z]: alle Buchstaben
# {1,3}: eins bis 3 Buchstaben
# [0-9]: Zahlen bzw. Ziffern von 0 bis 9
# {5}: Genau 5 Zeichen
validationStringA = "[A-Z]{1,3}[0-9]{5}"
validationStringB = "[A-Z]{2}[0-9]{1}"
print(f"Eingabe: {id}")
# a)
if re.fullmatch(validationStringA, id):
    print("Bedingung A ist erfüllt!")
# b)
elif re.fullmatch(validationStringB, id):
    print("Bedingung B ist erfüllt!")
else:
    print("Keine der in der Aufgabenstellung genannten Bedingungen ist erfüllt!")
Eingabe: AS12345
Bedingung A ist erfüllt!
```

Abbildung 4 - 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:

```
135 # a)
136 if dict1 == dict2:
137     print(f"Die beiden Dicts: \n{dict1} und \n{dict2} \nsind gleich!")
138 else:
139     print(f"Die beiden Dicts sind nicht gleich!")
```

Abbildung 5 - 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 \hat{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_squared_error()` und als zweites habe ich die Mathematische Formel in den Code umgesetzt:

```

196
197 listel = [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
198 listeS = sorted(listel) listel: [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
199
200 # Mit Funktion
201 print(f"Funktion von sklearn: {mean_squared_error(listel, listeS)}") listel: [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
202
203 # Ohne Funktion, also nach Formel
204 mse = 0
205 N = len(listel) listel: [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
206 for i in range(N): N: 19
207     mse += (listel[i] - listeS[i]) ** 2 mse: 10552.996199999996 listel: [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
208
209 mse = 1 / N * mse N: 19 mse: 10552.996199999996
210 print(f"Ohne Funktion: {mse}") mse: 10552.996199999996
211 +

```

Originalliste L und der sortierten Liste S. Die Elemente der Liste L seien hierbei mit y bezeichnet und die der sortierten Liste S mit \hat{y} -Dach. (4 P.)

```

listel = [11.1, 10.4, 320.4, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 6.9, 4.2, 1.1]
listeS = sorted(listel)

# Mit Funktion
print(f"Funktion von sklearn: {mean_squared_error(listel, listeS)}")

# Ohne Funktion, also nach Formel
mse = 0
N = len(listel)
for i in range(N):
    mse += (listel[i] - listeS[i]) ** 2

mse = 1 / N * mse
print(f"Ohne Funktion: {mse}")

Funktion von sklearn: 10552.996199999998
Ohne Funktion: 10552.996199999996

```

Abbildung 6 - Aufgabe 30

Die Formel ist noch hier zu sehen:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Abbildung 7 - Formel Aufgabe 30

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