Praktikum Programmieren - #1

Programmieren

Kleine Übungen (ruhig alle in eigenen .py Dateien)

1. Zwei Listen verbinden

Wir haben zwei Listen mit Zahlen gegeben (integer und/oder floats). Diese wollen wir zu einer Liste kombinieren und die Zahlen darin der Größe nach sortieren.

Aufgabe 1.1:

Kombiniere die beiden Listen `list_1 = [50, 40, 30, 100]` und `list_2 = [10, 20, 1, 51]` mit Hilfe von Python und sortiere der Größe nach.

Aufgabe 1.2:

Entwickele eine Python Funktion `list_combiner(list_1, list_2)` die diese Aufgabe für beliebge Eingaben von list_1 und list_2 erledigt.
Startpunkt:

```
def list_combiner(list_1, list_2):
    # den eigenen Code hier einfügen
    return combined_list
```

Anwendungsbeispiel:

```
combined_list = list_combiner([1, 2, 3], [2.5, 0.5, 1.5])
print(combined_list)  # -> soll ausgeben: [0.5, 1, 1.5, 2, 2.5, 3]
```

2. while loops / Schleife

Wir starten mit zwei Variablen (`credit` und `delta`) die jeweils Zahlenwerte enthalten. `credit` muss einen positiven Wert enthalten.

Aufgabe 2.1

Erstelle mit Python eine while-Schleife die in jedem Durchgang die Zahl `delta` von `credit` abzieht. Die Schleife soll solange laufen bis kein Wert `delta` mehr aus `credit` entnommen werden kann (`credit` darf nicht negativ werden!).

Erinnerung while Schleife mit Python:

```
while True:
    # do something
```

Fragen:

- Wenn wir mit `credit = 1000` und `delta = 0.9999` starten, welchen Wert hat `credit` nach beenden der while-Schleife?
- Bei welcher Art von Werten für `credit` und `delta` bekommen wir Probleme mit der Ausführung des Programmes? (und: Warum?)
- Wie könnten wir die while-Schleife ergänzen, so dass wir zum Schluss auch ausgeben können wie oft `delta` von `credit` entnommen wurde?

3. for-Schleife

Wir starten mit zwei Variablen. Zum einen `credit` wie bei Aufgabe 2, zum anderen `debits`, eine Liste die Zahlenwerte enthält.

Aufgabe 3.1.

Erstelle mit Python eine for-Schleife die über alle Werte in `debits` läuft. Jeder dieser Werte soll von `credit` abgezogen werden.

Fragen:

- Wenn wir mit `credit = 1000` und `debits = [1.1, 22.22, 333.333]` starten, welchen Wert hat `credit` nach beenden der for-Schleife?
- Wie können wir den Code verändern, so dass der Wert `credit` niemals kleiner Null wird?

4. Textadventure erstellen

Hier möchten wir mit Python ein erstes, kleines Textadventure programmieren!

Der Ablauf sollte in etwa folgendermaßen aussehen:

```
Du befindest dich in einem unbekannten, dunklen Raum...
Was machst du? a) Rufen b) Lichtschalter suchen c) Warten
>> b

Die Wände fühlen sind an wie aus Fels.
Nirgendwo ein Schalter!
Doch halt! Da ragt etwas aus der Wand heraus... ein metallischer Hebel.
a) Ich betätige den Hebel b) Ich suche lieber noch weiter
>>
```

Und so weiter...

Um dafür ein Programm zu erstellen, benötigen wir zumindest drei Elemente:

- Nutzereingaben. Das können wir hier einfach mit `input()` umsetzten, z.B. `action = input("Was machst du? a) Rufen b) Lichtschalter suchen c) Warten ")`
- Verzweigungen.
- Schleifen (while).

Tipp zur Abfrage der Nutzereingabe: Bei Sequenzen können wir in Python nach enthaltenen Elementen fragen mit `x in sequenz` was dann mit True/False beantwortet wird.

Aufgabe 4.1

Erstelle ein solches Textadventure mit mindestens 4 aufeinander folgenden "Räumen" oder Handlungsabfragen. Jede Handlungsabfrage muss 2 oder mehr Optionen haben.

Aufgabe 4.2

An einer Stelle soll ein dreistelliger Zahlencode eingegeben werden und es geht nur weiter wenn der korrekte Code eingegeben wird. Im Text sollte es darum einen klaren Hinweis geben. Wenn der Code falsch eingegeben wird soll "Nichts geschieht" erscheinen, aber erst nach einer kleinen Pause.
Die Pause können wir einbauen mit

```
import time

time.sleep(1)  # pause of 1 second
```

Level 2 (optional, falls Level 1 zu einfach war)

Statt den gesamten Code als einen großen Block zu schreiben, können wir an dieser Stelle auch sehr gut mit Funktionen arbeiten! Dazu z.B. jeden Raum (oder jeden Abenteuerteil mit Handlungsoption) in eine eigene Funktion umwandeln.

- Was für Vorteile bietet das?

Debugging

Beim Programmieren ist es völlig normal, dass das Schreiben von Code nur einen kleinen Teil der Zeit einnimmt. Oft geht mehr Zeit ins Land um den geschriebenen Code auch wirklich zum Laufen zu bringen. Damit wären wir beim Thema: Debugging!

Hier eine unglaublich gute Funktion, die aber leider irgendwie so nicht funktioniert.
Bitte einmal "reparieren".

```
def framed_print(input_str, frame_size, frame_symbol="#"):
    """Put your print in the right frame.
    """
     string_length = len(input_str)
    for _ in range(frame_size):
        print((string_length + 2 + 2 * frame_size) * frame_symbol)
    # print actual content
    print(f"{frame_size * frame_symbol} {input_str} {grame_size * frame_symbol")
    for _ in range(frame_size):
        print((string_length + 2 + 2 * frame_size) * frame_symbol)

framed_print("yes this look awesome!", 3, ">")
```