Philipp Schenk, 7093700
Übungsblatt ÜE-01
Aufgabe 1

a) Lauffähig: Ja
   Ausgabe: 5
   Fehlermeldung: Nein
   Fehlerklasse:
   Begründung:

b) Lauffähig: Nein
   Ausgabe:
   Fehlermeldung: Ja
   Fehlerklasse: TypeError
   Begründung: Zuweisung oben (variable a) ist ein tupel und wird weiter unten mit einem integer (4) verglichen.

c) Lauffähig: Nein
   Ausgabe:
   Fehlermeldung: Ja
   Fehlerklasse: NameError
   Begründung: b wird versucht als variable darzustellen, wurde bisher aber noch nicht definiert

d) Lauffähig: Ja
   Ausgabe: b gibt True
   Fehlermeldung: Nein
   Fehlerklasse:
   Begründung:

Aufgabe 2 & 3:
https://github.com/philippschenk2000/epr/tree/main/epr2_Schenk_7093700

epr2 Aufgabe 2 -- ANALYSE

Aufgabe ÜE-01 Berechnung von der Bonuspunkte für die Klausur mittels zweier Zahlen in python==3.0 oder neuer

Ein- und Ausgabeformat:

Ein: 3 Integer-Zahlen, die ersten beiden zwischen 0 & 110. Aus: Da Antwortsätze ausgegeben werden sollen in beiden Fällen, gebe ich es in diesem Fall als ein f-string aus, sodass dabei die entstehenden Bonus Punkte für die Klausur ausgegeben werden.

Annahmen:

So wie programmiert: keine, der Nutzer wird auf alle möglichen Fehler hingewiesen.

Entwurfsmuster:

Ich werde auch in Zukunft nach einem ähnlichen Muster vorgehen, nämlich dass die Überfunktion "def main" regelmäßig weitere Funktionen aufruft und somit als "oberste Funktion" gilt. Aus Gründen der Übersichtlichkeit füge ich mehrere ähnliche Aufgaben dann zu kleinen Unterfunktionen zusammen, sodass diese von "def main" aus gesteuert werden. In jedem Falle versuche ich den Code so darzustellen, dass kein Fehler letztendlich mehr auftritt bzw. der Nutzer über weiteres Vorgehen genau informiert wird. Hierbei wird nach dem EVA-Prinzip (Eingabe-Verarbeitung-Ausgabe) nachgegangen.

epr2 Aufgabe 2 -- TESTS

Code: siehe epr2_exc2_Schenk_7093700.py

Test1: In: 90, 40, 50 Out: The bonus points for the input would be 9.29 points

Test2: In: -10, 40, 10 Out: First number and second number have to be between 0 and 110

Test3: In: 0, 106.9, 10 Out: The bonus points for the input would be 2.5 points

Test4: In: 110, 110, 50 Out: The bonus points for the input would be 12.5 points

epr2 Aufgabe 2 -- DOKUMENTATION

Beschreibung des Programms:

Es ist erforderlich, dass ein Python-Interpreter in der Version 3.0 oder höher mit den Standardbibliotheken und unterstützenden Programmen, wie sie auf www.python.org verfügbar sind, installiert ist. Um das Programm auszuführen, können Sie dies auf die gewohnte Art und Weise für Ihr Betriebssystem tun, sei es aus der Interpreter-Shell oder in einer integrierten Entwicklungsumgebung (IDE) wie beispielsweise IDLE. Nach Input der Werte in der Konsole sichert das Programm die Funktion und den User wissend (durch die try-except funktion) ab, sodass nur Zahlen in die folgenden Berechnungen aufgenommen werden. Nach erfolgreicher eingegebener Zahlen springt das Programm zur Berechnung der maximalen Range (0 - 110) für die ersten beiden Werte und informiert den Nutzer erneut bei falschen Angaben. Bei richtiger Angabe der Werte erfolgt die Berechnung der zum Bestehen notwendigen Punkten (ZBNP). Dafür wird das Minimum bestimmt aus erstens, den ZBNP geteilt durch 4, und zweitens, der Summe der GPR und EPR Punkte, geteilt durch 14. Dabei wird der errechnete Wert (Bonus points) gerundet auf 2 Nachkommastellen wie in einem angegebenem Beispiel ausgegeben. https://github.com/philippschenk2000/epr/tree/main/epr2_Schenk_7093700

epr2 Aufgabe 3 -- ANALYSE

Aufgabe ÜE-01 Berechnung des Minimums von zwei Zahlen und anschließender Division durch die Werte 2, 4, 8 in python==3.0 oder neuer, um zu Prüfen, welcher Wert der Rest nach Division besitzt.

Ein- und Ausgabeformat:

Ein: 2 Integer-Zahlen. Aus: Da Antwortsätze ausgegeben werden sollen in beiden Fällen, gebe ich es in diesem Fall als ein f-string aus, sodass dabei sowohl das Minimum beider Werte als auch die zu teilende Zahl und der Rest nach Division an den Nutzer ausgegeben werden.

Annahmen:

So wie programmiert: keine, der Nutzer wird auf alle möglichen Fehler hingewiesen.

Entwurfsmuster:

Ich werde auch in Zukunft nach einem ähnlichen Muster vorgehen, nämlich dass
die Überfunktion "def main" regelmäßig weitere Funktionen aufruft und somit als "oberste Funktion" gilt. Aus Gründen der Übersichtlichkeit füge ich mehrere ähnliche Aufgaben dann zu kleinen Unterfunktionen zusammen (falls vorhanden), sodass diese von "def main" aus gesteuert werden. In jedem Falle versuche ich den Code so darzustellen, dass kein Fehler letztendlich mehr auftritt bzw. der Nutzer über weiteres Vorgehen genau informiert wird. Hierbei wird nicht nach dem EVA-Prinzip (Eingabe-Verarbeitung-Ausgabe) nachgegangen, da Ausgabe und Verarbeitung teilweise im Wechsel stehen.

epr2 Aufgabe 3 -- TESTS

Code: siehe epr2_exc3_Schenk_7093700.py

Test1: In: 4.0 Out: All input has to be integer.

Test2: In: 4, -9 Out: The minimum of both numbers (-9) is NOT dividable by 2 with a rest of 1, The minimum of both numbers (-9) is NOT dividable by 4 with a rest of 3, The minimum of both numbers (-9) is NOT dividable by 8 with a rest of 7

Test3: In: 4, 20 Out: The minimum of both numbers (4) is dividable by 2 with no rest, The minimum of both numbers (4) is dividable by 4 with no rest, The minimum of both numbers (4) is NOT dividable by 8 with a rest of 4

Test4: In: 42790, 2449 Out: The minimum of both numbers (2499) is NOT dividable by 2 with a rest of 1, The minimum of both numbers (2499) is NOT dividable by 4 with a rest of 3, The minimum of both numbers (2499) is NOT dividable by 8 with a rest of 3

epr2 Aufgabe 3 -- DOKUMENTATION

Beschreibung des Programms:

Es ist erforderlich, dass ein Python-Interpreter in der Version 3.0 oder höher mit den Standardbibliotheken und unterstützenden Programmen, wie sie auf www.python.org verfügbar sind, installiert ist. Um das Programm auszuführen, können Sie dies auf die gewohnte Art und Weise für Ihr Betriebssystem tun, sei es aus der Interpreter-Shell oder in einer integrierten Entwicklungsumgebung (IDE) wie beispielsweise IDLE. Nach Input der Werte in der Konsole sichert das Programm die Funktion und den User wissend durch die try-except funktion ab, sodass nur integer-Werte in die folgenden Berechnungen aufgenommen werden. Nach erfolgreicher integer-entsprechenden Zahlen springt das Programm zur Berechnung des Minimums für die ersten beiden Werte. Durch eine for-Schleife über die Liste "[2, 4, 8]" iteriert der anschließende Code über erstens, die Berechnung des Rests der Division von Minimum durch den aktuellen Wert der Liste, sowie zweitens, die resultierende Ausgabe in der Konsole für den Nutzer. Dabei werden das Minimum beider Input-Werte, der Wert aus der Liste (Nenner) und der resultierende Rest der Division davon ausgegeben. https://github.com/philippschenk2000/epr/tree/main/epr2_Schenk_7093700