LU10.A02 - Einleser als Modul

Ausgangslage

In vielen der Aufgaben im Modul 319 lesen Sie mit `input()` Eingaben eines Benutzers ein. Ist die Eingabe eine Zahl, so müssen Sie die Eingabe mit `int(input('Zahl eingeben'))` in eine Ganzzahl oder mit `float(input('Zahl eingeben'))` in eine Fliesskommazahl umwandeln. Gibt der Benutzer nun aber anstatt einer `Zahl` einen `Buchstaben` ein, erhalten Sie einen `ValueError` vom Interpreter. Wir möchten uns nun ein Modul mit Funktionen erstellen, welche uns diese Eingaben validieren und im Falle einer falschen Eingabe eine Fehlermeldung ausgeben und den User erneut zur Eingabe auffordern.

Der einfachste Weg dieses Problem zu lösen sehen Sie im folgenden abschnitt Pseudocode:

```
Funktion read_float(text_to_display)
  Start Endlosschleife:
    num =
Einlesen_von_konsole(text_to_display)
    Versuche:
      num = in_float_umwandeln(num)
    Ein ValueError wurde ausgelöst:
      Fehlermeldung ausgeben
      Endlosschleife nochmals durchlaufen
    Kein Error:
      Die Variable num zurückgeben
```

Wir können mit unserem jetzigen Wissen aus dem Modul 319 leider noch nicht den ganzen Pseudocode in Python umsetzen. Wir wissen noch nicht wie wir Python etwas versuchen lassen. Um dieses Problem in Python zu lösen muss etwas Wissen aus dem Modul 320 vorgeholt werden. In Python gibt es die Möglichkeit das Programm etwas zu versuchen zu lassen und wenn es Schief geht, dann stürzt das Programm nicht ab sondern läuft weiter. Um dies zu verwirklichen gibt es den Befehl `try-except`. Ein `try-except`-Block sieht in etwa so aus:

```
try:
    #Code to try
except <Name of Exception>:
    #Code if it fails
else:
    #Code if it works
```

Im `try`-Block steht der Code, den Python auszuführen versuchen soll, im `except`-Block steht der Code der ausgeführt wird, wenn es nicht geklappt hat. Im `else`-Block steht der Code der ausgeführt

wird, wenn es geklappt hat.

Aufgabe

Da Sie jetzt den `try-except`-Block kennen können wir den Pseudocode jetzt komplett in Python umsetzen.

Teilaufgabe 1

Nehmen Sie die Github-Classroom Aufgabe an und clonen Sie das Repository in ihre Entwicklungsumgebung.

main.py

```
def main():
    # do something
    float = read_float('Please enter a
real number> ')
    int = read_int('Please enter a whole
number> ')

    print(float)
    print(int)


if __name__ == '__main__':
    main()
```

input_reader.py

```
"""
input_reader module with input
validation
"""
```

Erstellen Sie im File `input_reader.py` die Funktionsdefinition der Funktionen `read_float()` und `read_int()`.

Teilauftrag 2

Übersetzen Sie den Pseudocode aus der Ausgangslage in Python code für die Funktion `read_float`.

```
def read_float(text_to_display):
  """
  TODO: Erstellen Sie den DocString
  """
  Start Endlosschleife:
    num =
Einlesen_von_konsole(text_to_display)
    Versuche:
      num = in_float_umwandeln(num)
    Ein ValueError wurde ausgelöst:
      Fehlermeldung "Please, enter a real
number!" ausgeben
      Endlosschleife nochmals durchlaufen
    Kein Error:
      Die Variable num zurückgeben
```

Erstellen Sie einen passenden Docstring wie in LU09 erklärt.

Teilauftrag 3

Überlegen Sie sich, was Sie alles ändern müssen, damit die Funktion `read_int` sinnvoll funktionieren würde. Ergänzen Sie ihr Modul `input_reader.py` mit dieser Funktion.

input_reader.py

```
    def read_float(text_to_display):
      ...

    def read_int(text_to_display): #
    Fehlermeldung: Please, enter a valid
    whole number!
      ...
```

Teilauftrag 4

Importieren Sie ihr `input_reader` Modul in ihrem `main.py` damit das Programm funktioniert.

```python
# Import the input_reader module here

def main():
    # do something
    float = read_float('Please enter a real number> ')
    int = read_int('Please enter a whole number> ')

    print(float)
    print(int)

if __name__ == '__main__':
    main()
```

⇒ *GitHub Repo für externe Besucher*

GitHub Repository https://github.com/templates-python/m319-lu10-a02-reader-module

Lernende am BZZ müssen den Link zum GitHub Classroom Assignment verwenden

M319-LU10