Übungsblatt 11 zur Vorlesung Programmierung 1

Aufgabe 1 (teilweise Wiederholung aus der Vorlesung)

Welche Ausgabe erzeugen die folgenden Funktionen bei den gegebenen Eingaben? Erläutern Sie bitte die Ausgabe. Nicht behandelte Fehler einfach als „Error“ benennen

- `fancy_divide([0, 2, 4], 1)`
- `fancy_divide([0, 2, 4], 4)`
- `fancy_divide([0, 2, 4], 0)`

Tipp: Veranschaulichen Sie sich den Weg durch die Funktion mithilfe des Debuggers.

Funktion 1:

```
def fancy_divide(numbers, index):
   try:
      try:
         denom = numbers[index]
         for i in range(len(numbers)):
            numbers[i] /= denom
      except IndexError:
         fancy_divide(numbers, len(numbers) - 1)
      else:
         print("1")
      finally:
         print("0")
   except ZeroDivisionError:
      print("-2")
```

Funktion 2:

```
def fancy_divide(numbers, index):
   try:
      denom = numbers[index]
      for i in range(len(numbers)):
         numbers[i] /= denom
   except IndexError:
      fancy_divide(numbers, len(numbers) - 1)
   except ZeroDivisionError:
      print("-2")
   else:
      print("1")
   finally:
      print("0")
```

Für die folgenden Funktionen genügt die Prüfung des folgenden Aufrufs:

- `fancy_divide([0, 2, 4], 0)`

Funktion 3:

```
def fancy_divide(list_of_numbers, index):
   try:
      try:
         raise Exception("0")
      finally:
         denom = list_of_numbers[index]
         for i in range(len(list_of_numbers)):
            list_of_numbers[i] /= denom
   except Exception as ex:
      print(ex)
```

Funktion 4:

```
def fancy_divide(list_of_numbers, index):
   try:
      try:
         denom = list_of_numbers[index]
         for i in range(len(list_of_numbers)):
            list_of_numbers[i] /= denom
      finally:
         raise Exception("0")
   except Exception as ex:
      print(ex)
```

Aufgabe 2

Die folgende Funktion `fancy_divide([0, 2, 4], 0)` erzeugt einen ZeroDivisonError. Bitte korrigieren Sie die Funktion simple_divide! Bei Division mit 0 soll eine Liste mit dem Wert „0“ zurückgegeben werden.

```
def fancy_divide(list_of_numbers, index):
   denom = list_of_numbers[index]
   return [simple_divide(item, denom) for item in list_of_numbers]

def simple_divide(item, denom):
   return item / denom
```

Aufgabe 3

Gegeben ist ein einfacher Python-Code, der einen Benutzer dazu auffordert, zwei Zahlen einzugeben und dann eine Division durchzuführen. Der Code enthält jedoch keine Fehlerbehandlung. Ihre Aufgabe ist es, den gegebenen Code so zu erweitern, dass er Exceptions ordnungsgemäß behandelt.

```
numerator = int(input("Bitte geben Sie den Zähler ein: "))
denominator = int(input("Bitte geben Sie den Nenner ein: "))
result = numerator / denominator
print(f"Ergebnis: {result}")
```

a) Erweitern Sie den obigen Code, um Division durch Null und ungültige Eingaben (z. B. wenn der Benutzer Buchstaben statt Zahlen eingibt) zu behandeln. Geben Sie im Falle einer Exception eine entsprechende Fehlermeldung aus.

b) Ergänzen Sie den Code nun um einen Teil, der immer dann aufgerufen wird, wenn keine Exception aufgetreten wird. Bestätigen Sie in diesem Block die fehlerfreie Ausführung (ohne Exception) und geben das Ergebnis aus.

c) Ergänzen Sie den Code weiterhin um einen Teil, der immer zum Abschluss ausgeführt wird, unabhängig davon, ob vorher eine Exception geworfen wurde oder nicht. Geben Sie dem Benutzer eine kurze Information, dass das Programm nun beendet wird.