Universität Potsdam
Institut für Informatik

GdP-Rechnerübung

Aufgabenblatt 7

Lernziele (zum Abhaken): Nach diesem Aufgabenblatt sollten Sie...

☐ Ausgaben mit `print` und dem `%`-Operator formatieren können

☐ Python-Skripte erstellen und ausführen können

☐ die Bedeutung von Wahrheitswerten (`bool`) verstehen und `if`-Verzweigungen einsetzen können

☐ `while`-Schleifen einsetzen können

☐ Nutzer-Eingaben mit `input()` einlesen können

☐ den Datentyp eines Wertes mittels explizitem Casting ändern können

☐ wissen, was die Funktion einer Shebang-Zeile ist

12 Ausgaben

Die `print`-Funktion haben Sie bereits in ihrer einfachsten Form kennen gelernt. Um innerhalb einer Ausgabe Werte von Variablen einzufügen, wird der `%`-Operator verwendet:

```
>>> name = 'Hannah'
>>> frage = "Wie geht's dir?"
>>> print("Hallo %s! %s" % (name, frage))
Hallo Hannah! Wie geht's dir?
```

Diese Notation ist der `sprintf`-Funktion aus der Programmiersprache C nachempfunden. Die Ausdrücke in den eckigen Klammern (`[ ]`) sind optionale Parameter und können bei Bedarf mit angegeben werden.[1]

```
%[flags][width][.precision]conversion
```
bzw.
```
%[schalter][breite][.genauigkeit]code
```

1. Bei der Darstellung von Zahlen kann man mit `width` angeben, wie viel Platz für die Zahl insgesamt eingeräumt werden soll:

   ```
   >>> print('Zahl: %5d' % 5)
   Zahl: ␣␣␣␣5
   ```
   (␣ = Leerzeichen)

[1] Eine Liste aller Flags und Conversions ist in der Python-Dokumentation zu finden:
`https://docs.python.org/3/library/stdtypes.html#old-string-formatting`

Hinweis: Hier wird die `conversion d` verwendet, um eine Ganzzahl (signed integer **d**ecimal) auszugeben.

Welche Ausgabe erzeugt `print('Zahl: %5d' % 125)`? Markieren Sie dabei die Leerzeichen wie im Beispiel!

Ausgabe: _____

2. Bei Gleitkommazahlen kann man mit `precision` angeben, auf wie viele Nachkommastellen gerundet ausgegeben werden soll:

```
>>> print('Zahl: %.2f' % 6.3774)
Zahl: 6.38
```

Hinweis: Hier wird die `conversion f` verwendet, um eine Gleitkommazahl (**f**loating point decimal) auszugeben.

Möchte man der Zahl auch vor dem Komma Platz einräumen, so muss bei `width` die komplette Länge der gewünschten Ausgabe angegeben werden. Möchte man also Platz für **drei** Stellen vor und **zwei** Stellen nach dem Komma einräumen, dann muss man folgendes eingeben:

```
>>> print('Zahl: %6.2f' % 6.3774)
Zahl: ␣␣6.38
```

Wie berechnet sich also die Länge `width` bei der Ausgabe von Gleitkommazahlen?

_____

_____

13 Skripting

Python-Skripte sind Textdateien mit Python-Quellcode. Ein Python-Skript kann dem Interpreter zur Ausführung übergeben werden.

1. Schreiben Sie Ihr erstes Python-Skript! Öffnen Sie eine neue Datei `hallo.py` in einem Texteditor Ihrer Wahl, z. B. `gedit`, `vi` oder `nano`:[2]

```
$ gedit hallo.py
```

Geben Sie ein:

```
1 print('Hallo Welt')
```

Speichern Sie die Datei ab. Führen Sie das Skript nun aus:

```
$ python hallo.py
```

[2] Das `$`-Zeichen signalisiert hier wieder eine Eingabe im Bash-Terminal, nicht im Python-Interpreter.

2. Schreiben Sie ein Skript (mit einem Editor Ihrer Wahl), welches in der Konsole zwölfmal untereinander den Spruch „`Die Rechnerübung ist toll!`“ ausgibt. Realisieren Sie diese Aufgabe mit einer *solange*-Schleife. Diese beginnen in Python mit einem `while <Bedingung>:` . Der darunter stehende, eingerückte Codeblock wird ausgeführt, solange die Bedingung zu `True` ausgewertet wird.

   Hinweis: Definieren Sie sich eine Variable, welche die Schleifendurchläufe „mitzählt“.

14 Kontrollstrukturen

1. Arbeiten Sie mit dem Datentyp `bool`. Welche Ausgabe erhalten Sie?

   ```
   >>> a = True; b = False
   ```

   `>>> not(a)` Ausgabe: _____

   `>>> a or b` Ausgabe: _____

   `>>> a and b` Ausgabe: _____

   `>>> 9 >= 7 and 5 < 3` Ausgabe: _____

   `>>> 2 != 2` Ausgabe: _____

   `>>> "a" == "a"` Ausgabe: _____

2. *Bedingte Anweisungen* beginnen in Python mit einem `if <Bedingung>:` . Der darunterstehende, eingerückte Codeblock wird nur dann ausgeführt, wenn die Bedingung zu `True` ausgewertet wird.
   Den folgenden Code[3] finden Sie in der Datei `if.py`, die Sie aus dem Verzeichnis `/home/gdplehre` kopieren oder im Moodle-Kurs herunterladen können:

   Skript 1: `if.py`

```
1  # change this code
2  number = 10
3  second_number = 10
4  first_list = []
5  second_list = [1,2,3]
6
7  #do not change the code below!
8  if number > 15:
9      print("1")
10
11 if first_list:
12     print("2")
13
```

[3] Entnommen von `http://www.learnpython.org/Conditions`

```
14 if len(second_list) == 2:
15     print("3")
16
17 if len(first_list) + len(second_list) == 5:
18     print("4")
19
20 if first_list and first_list[0] == 1:
21     print("5")
22
23 if not second_number:
24     print("6")
```

Ändern Sie im Code die Zeilen 2-5 so ab, dass jeder der `if`-Ausdrücke in den Zeilen 8-24 zu `True` ausgewertet wird.

3. Wann wird eine Zahl als `True` evaluiert, wann als `False`?

4. Wann wird eine Liste als `True` evaluiert, wann als `False`?

15 Casting

Durch explizites **Casting** kann eine Variable in einen bestimmten Typ konvertiert werden. Die am häufigsten verwendeten Funktionen dazu sind `int()`, `float()` und `str()`.

Skript 2: Beispiel: Casting

```
1 x = '100'
2 y = '-90'
3 print(x + y)  # Ausgabe: 100-90
4 print(int(x) + int(y))   # Ausgabe: 10
5 print(float(x) + float(y))  # Ausgabe: 10.0
```

5. Im folgenden Beispiel fehlen die Definitionen von `jahr` und `alter` mit korrekten Castings:

Skript 3: `casting.py`

```
1 input1 = input("Geben Sie Ihr Geburtsjahr an: ")
2 input2 = input("Geben Sie eine Jahreszahl an: ")
3
4 print("Im Jahr %d werden Sie %d Jahre alt." % (jahr, alter))
```

Sie finden die Datei `casting.py` im Verzeichnis `/home/gdplehre` oder im Moodle-Kurs.

a) Die Funktion `input()` liest eine Eingabe vom Benutzer ein. Von welchem Datentyp ist der Rückgabewert der Funktion?

_____

Hinweis: Wenden Sie die Funktion `type()` auf einen Rückgabewert von `input()` an! Am schnellsten geht dies im Python-Interpreter.

b) Schreiben Sie die beiden nötigen Zeilen auf, die `jahr` und `alter` definieren. Überprüfen Sie Ihren Code!

_____

_____

6. Verändern Sie das folgende Skript derart, dass ausgegeben wird, ob die Zahl `x` gerade oder ungerade ist.

```
1 #!/usr/bin/env python3.7
2 x = input("Bitte geben Sie eine ganze Zahl ein: ")
3 print(x)
```

a) Geben Sie an, wie Sie dafür den `print()`-Aufruf in Zeile 3 verändern müssen:

`print(` _____ `)`

Hinweis: Sie können eine `if`-Fallunterscheidung folgendermaßen in nur einer Zeile aufschreiben: `<True-Fall> if <Bedingung> else <False-Fall>`

b) Was passiert, wenn keine Zahl eingegeben wird?

_____

_____

_____

c) Was bewirkt die erste Zeile? Machen Sie dafür das Skript in der Shell ausführbar[4] und übergeben Sie es nicht dem Python-Interpreter, sondern rufen Sie es direkt auf!

---

*Hinweis: Bei dieser ersten Zeile handelt es sich um ein sogennates **Shebang** (kurz für „sharp bang“, wobei „sharp“ das Doppelkreuz („#“) und „bang“ das Ausrufezeichen bezeichnet). Sie gibt dem System an, mit welchem Programm die Datei geöffnet werden soll.*
Übernehmen Sie zukünftig diese Zeile in alle Ihre Python-Skripte!

7. Das folgende Skript `ratespiel.py` finden Sie im Verzeichnis `/home/gdplehre` oder im Moodle-Kurs.

Skript 4: `ratespiel.py`

```
1  #!/usr/bin/env python3.7
2  zahl = 23
3  weiter = True
4
5  while weiter:
6      geraten = input("Bitte Zahl eingeben: ")
7      if geraten == zahl:
8          print("Richtig geraten!")
9          weiter = False
10     else:
11         print("Falsch geraten")
```

Fügen Sie fehlende Casts ein und lassen Sie am Ende ausgeben, wie oft geraten wurde.

[4] D.h. Sie müssen das Recht, die Datei auszuführen, erteilen: `$ chmod +x skript.py`