IWGS1_WS1920_Zettel2

July 7, 2020

1 Informatische Werkzeuge der digitalen Geistes- und Sozialwissenschaften I

If you are learning programming (or anything else) what you do whenyou get "stuck" will be the sole determinant of your success. (Sarah Mount)

1.1 Hausaufgabe 2 (Einführung in Python, 100 Punkte)

Erschienen: 26.10.2019
Abgabe bis: 03.11.2019

Bitte laden Sie Ihre Notebooks bis 23:59 Uhr am Abgabetag in Ihrer Übungsgruppe bei StudOn hoch.

Wenn Ihnen einige der hier verwendeten Konzepte unbekannt sind oder Sie nicht wissen, wie Sie fortfahren sollen, können Sie die Vorlesungsunterlagen zu Rate ziehen oder jederzeit Fragen im Forum stellen, im Tutorium nachfragen, oder Ihrem Tutor eine Mail schreiben.

1.1.1 Aufgabe 2.1 ("Hallo Welt!", 10 Punkte)

Programmieren ist die Kunst/Wissenschaft, einen Computer dazu zu bringen, für uns Aktionen zu tätigen und Berechnungen durchzuführen. Eine Ansammlung von Instruktionen nennen wir ein Programm. Wir werden hier mit den Einstieg einem der einfachsten und typischsten Programme machen: Ausgabe von Text.

Aufgabe: Schreiben Sie in der nächsten Zelle ein Python-Programm, dass den String (= die Zeichenkette) `"Hallo Welt!"` ausgibt.

[1]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.2 Aufgabe 2.2 (Benutzung von Variablen, 15 Punkte)

Einer der großen Vorteile von Computern ist, dass wir sie leicht Berechnungen durchführen lassen (und meistens auch ihrem Ergebnis trauen) können, die für uns mit Stift und Papier viel Zeit in Anspruch nehmen würden und fehleranfällig wären. Die Kunst besteht "lediglich" darin, dem Computer genau klar zu machen, welche Berechnung wir denn jetzt genau haben wollen.

Aufgabe: Schreiben Sie in der nächsten Zelle ein Python-Programm, dass die Anzahl der Sekunden in einem Schaltjahr berechnet, in einer Variable speichert und mit `print` ausgibt.

[2]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.3 Aufgabe 2.3 (Wiederverwendung von Variablen, 15 Punkte)

Wenn wir Programmieren ist oft *Effizienz* eines unserer Ziele. Warum sollten wir uns sonst die Mühe machen, dem Computer eine Aufgabe zu geben, wenn wir selbst schneller wären? Ein Werkzeug in diesem Prozess ist die Wiederverwendung von Variablen.

Aufgabe: Schreiben Sie in der nächsten Zelle ein Python-Programm, dass den String `"satanarchäolügenialkohöllischer Wunschpunsch"` fünf Mal ausgibt.
Sie sollen dabei die Phrase nicht selbst fünf mal tippen (oder copy/pasten) müssen.

[3]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.4 Aufgabe 2.4 (Formatierung von Text, 25 Punkte)

Beim Programmieren passiert es oft, dass Sie Ihr Ergebnis (wie auch immer Sie dran gekommen sind) einem Menschen vor dem Bildschirm präsentieren müssen. Dabei muss bedacht werden, wie diese die Information am besten aufnehmen können.
Für diese Zwecke gibt es in den meisten Programmiersprachen spezielle Symbole, die für Formatierung benutzt werden können. Wenn Sie zum Beispiel `\n` in einem String schreiben, werden von `print` nicht diese Symbole selbst gedruckt, sondern eine neue Zeile begonnen.

Aufgabe: Schreiben Sie in der nächsten Zelle einen Python-Programm, das Ihr Lieblings-Haiku in drei Zeilen ausgibt. Benutzen Sie dafür nur *einen* `print`-Befehl.

Anmerkung: Falls Sie selbst kein Lieblingshaiku haben und sich keins ausdenken wollen, können Sie dieses benutzen:

My cow gives less milk
now that it has been eaten
by a fierce dragon.

[4]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.5 Aufgabe 2.5 (Verständnisaufgabe 1, 35 Punkte)

Um erfolgreich Programmieren zu lernen ist es mindestens genauso wichtig, den Code anderer Leute (z.B. in Aufgaben, Beispielen, bestehenden Projekten...) zu *lesen*, wie eigenen zu schreiben. Beides spielt eine kritische Rolle im Programmieralltag. Deswegen werden wir immer mal (wie hier)

Aufgaben stellen, in denen wir Sie darum bitten werden, sich mit gegebenem Code auseinander zu setzen.

Im Alltag und in der Wissenschaft begegnen uns viele Zahlen (Messungen, Daten, Mengen, ...), weswegen der Umgang mit denselben auch in der Programmierung eine wichtige Rolle spielt. Auch Python ist für Arithmetik geeignet. Die meisten Rechenoperationen funktionieren wie erwartet (Addition mit `+`, Subtraktion mit `-` und Multiplikation mit `*`). Siehe die folgenden Zellen mit Beispielen (an denen Sie gerne herum probieren können):

[5]:
```
3 + 7
```

[5]:
```
10
```

[6]:
```
9 - 5
```

[6]:
```
4
```

[7]:
```
4 * 4
```

[7]:
```
16
```

Für Division gibt es jedoch zwei Möglichkeiten: Division *ohne* Rest und Division *mit* Rest. Für die "normale" Division, bei der auch Kommazahlen als Ergebnis möglich sind, benutzen wir in Python `/`.

[8]:
```
17 / 5
```

[8]:
```
3.4
```

Für *ganzzahlige Division* benutzen wir `//`. 17 geteilt durch 5 ergibt 3 (plus einen Rest, der hier fallen gelassen wird).

[9]:
```
17 // 5
```

[9]:
```
3
```

Was aber, wenn uns der Rest der ganzzahligen Division interessiert? Dafür ist der `%`-Operator (lies: *modulo*) in Python nützlich. 17 geteilt durch 5 ergibt einen Rest von (17 modulo 5 =) 2:

[10]:
```
17 % 5
```

[10]:
```
2
```

Ein sicherer Umgang mit diesen und anderen Operatoren wird für IWGS nötig sein.

Aufgabe: Machen Sie sich mit dem Code in der nächsten Zelle vertraut. Was passiert, wenn Sie andere Zahlen statt `1234567` einsetzen? Was repräsentiert das Ergebnis, was am Ende ausgegeben wird? Wie wird dieses Ergebnis berechnet? Geben Sie Ihre Antwort in der übernächsten Zelle.

[1]:
```
sekunden = 1234567

ta = sekunden // (60*60*24)
sekunden = sekunden % (60*60*24)

st = sekunden // (60 * 60)
sekunden = sekunden % (60*60)

mi = sekunden // 60
sekunden = sekunden %  60

print("Ergebnis:", ta, st, mi, sekunden)
```

```
Ergebnis: 14 6 56 7
```

Ihre Beschreibung und Erläuterungen hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle auszuführen)

[ ]:
```

```