“Basic Programming”

Homework Assignment 1

October 17, 2018

1 Tasks

1.1 Mini Tasks (5p - 0.25 per Task)

Download the assignment files[1] to your local system. The ZIP archive chontains two notebooks (German/English) and three files that you need for the mini tasks **Via jupyter notebook.** Open the notebook

with the tasks in your preferred language and fill in the missing cells. Please make sure you do not modify the cells that already contain code. (**Hint:** You will still need to execute those cells to use the code within them!).

Before you submit your work make sure the filename of your notebook is correct (`assigment_1.ipynb`)!

Via native Python. You can also submit the requested functions in a plain python file. Here copy the content of requested cells (that already contain code) into your program. You can find a printcopy of the assignment notebook at the very end of this document.

2 Restrictions

- Do not add any bloat to your submission
- Use built-in functions where possible
- Do NOT load extra packages (no imports)
- Do NOT use any command line arguments!

[1] https://github.com/pkasper/info1-bm/raw/master/2018/assignments/assignment_1_files.zip

3 File Headers

All Python source files have to contain a comment header with the following information:

- Author: – Your name
- MatNr: – Your matriculate number
- Description: – Purpose of the file
- Comments: – Any comments as to why if you deviate from the task or restrictions

Please copy and paste the example and change its contents to your data. (Depending on your PDF-viewer you may have to fix the whitespaces!)

Example Header:

```
########################################################################
# Author:      [FIRST NAME] [LAST NAME]
# MatNr:       [MAT NR]
# Description: [SHORTDESCRIPTION]
# Comments:    [ANY RELEVANT COMMENTS.
#              CAN BE MULTILINE]
########################################################################
```

4 Coding Standard

This lecture follows the official PEP 8 Standard[2]. it defines basic formalities as to how your code should look like. In particular, please look at the following aspects:

Language. Programming is done in English. This is to ensure someone else at the other end of the world can read your code. Please make sure all sources you submit are thus written in English. This covers both variable names and comments!

Spaces, not tabs. Indent your files with 4 spaces instead of tabs. PEP 8 forces this in Python 3 (whilst allowing more freedom in Python 2). Most editors have an option to indent with 4 spaces when you press the tab button!

Descriptive names. Use descriptive names for variables where possible! Whilst a simple i can be enough for a simple single loop code can become very messy really fast. If a variable has no purpose at all, use a single underscore (_) for its name. Should you be uncertain if a name is descriptive enough, simply as a comment describing the variable.

120 **characters line-limit** PEP 8 suggests a maximum line length of 79 characters. A different established limit proposes 120 characters. In this lecture, we thus use the wider limit to allow for more freedom. The maximum number of characters is 120 including indentations! Note that this is also applies to comments!

[2] https://www.python.org/dev/peps/pep-0008/

5 Automated Tests

Your code will be tested by an automated program.Please make sure your submission follows all the restrictions defined in this document. In particular concentrate on the predefined names for functions and variables!
If your submission fails any of the automated tests, we will look into your code to ensure the reason was not on our end and to evaluate which parts of your code are correct and awards points accordingly.

6 Submission

6.1 Deadline

13^{th} of November 2018 uatm 23:59:59.

Any submission handed in too late will be ignored with the obvious exception of emergencies. In case the submission system is globally unreachable at the deadline it will be pushed back for 24 hours. If for whatever reason you are unable to submit your work, contact your tutor *PRIOR* to the deadline.

6.2 Uploading your submission

Assignment submissions in this course will always be archives. You are allowed to use .zip or .tar.gz formats.
In addition to your Python source files, please also pack a *readme.txt*. The file is mandatory but its contents are optional. In this file please write down how much time you spent on the assignment and where you ran into issues. Your feedback allows for immediate adjustments to the lecture and tutor sessions.
Upload your work to the Palme website. Before you do please double-check the following aspects:

- File names and folder structure (see below)
- Comment headers in every source code file
- Coding standard upheld

6.3 Your submission file

```
assignment_1.zip (or assignment_1.tar.gz)
    assignment_1.ipynb (or assignment_1.py)
    readme.txt
```

In [ ]:

```
#########################################################################
# Author:      [VORNAME] [NACHNAME]
# MatNr:       [MATRIKELNUMMER]
# Description: [KURZBESCHREIBUNG DER DATEI]
# Comments:    [KOMMENTARE ZUR ABGABE.
#              KANN MEHRZEILIG SEIN]
#########################################################################
```

Assignment 1

Informatik 1 für Biomedical Engineering 2018/2019

Lösen Sie diese einzelnen Aufgaben. Schreiben Sie die Lösung für jede Aufgabe als einzelne Funktion (in die leeren Codezellen unterhalb der Aufgabenbeschreibungen). Achten Sie darauf, die Funktionen genau wie gefragt zu benennen!

Beispiel

Mit Rückgabewert und Funktionsparameter:
Schreiben Sie eine Funktion *add(a, b)*, die die Summe zweier gegebener Zahlen a und b berechnet und zurückgibt.

```
def add(a, b):
    result = a + b
    return result

#for testing:
x = 2
print(add(x, 5)) # -> 7
```

Ohne Rückgabewert und Funktionsparameter:
Schreibe eine Funktion *print_hello*, die den String "Hello!" Ausgibt.

```
def print_hello():
    print("Hello!")

#for testing:
print_hello() # -> Hello!
```

1 Input/Output

1.1 I/O kombiniert

Schreiben Sie eine Funktion *io_1()*. Verwenden Sie *input* um nach dem Namen und dem Alter eines Benutzers zu fragen. Als Output soll die Funktion folgenden Satz (mit den eingegebenen Daten) ausgeben:
Hi, my name is *name* and I'm *age* years old.

In [ ]:

1.2 Datei I/O

Schreiben Sie eine Funktion *io_2()*. Die Funktion soll die Datei "query.txt" einlesen. Verwenden Sie die Datei als Abfrage für Benutzereingaben und schreiben Sie die eingegebene Antwort in eine Datei namens "result.txt".

In [ ]:

2 Funktionen

Erstellen Sie eine Funktion *solve_quadratic*, um die quadratische Formel zu berechnen. Die Funktion soll drei Argumente annehmen, die die Koeffizienten a, b und c sind, und die Ergebnisse als Tupel zurückgeben. Sie müssen sich keine Gedanken über komplexe Zahlen machen, Python weiß was zu tun ist. Testen Sie die Funktion mit den folgenden Gleichungen:
$x^2 + 2x - 15 = 0$
$2x^2 - 6x - 36 = 0$
(Die Ergebnisse sollten $(3, -5)$ und $(24, -12)$ sein)

In [ ]:

3 Datentypen

3.1 Listen

3.1.1

Schreibe eine Funktion *lists_1()*. Erstellen Sie eine Liste mit allen Vielfachen von 3, beginnend bei 0 und endend bei 21. Berechnen Sie die Quadratwurzel jedes Elements und speichern Sie es in einer neuen Liste. Geben Sie die Summe des ersten und letzten Elements der neuen Liste aus.

In [ ]:

```
```

3.1.2

Schreiben Sie eine Funktion *task__31_2(given_numbers)*. Erstellen Sie eine Liste mit den Quadraten der Zahlen 1 bis 10. Fügen Sie sie mit einem an die Funktion übergebenen Tupel zusammen (Sie können *given_example* zum Testen verwenden) und entfernen Sie jede Zahl, die durch 5 teilbar ist. Berechnen Sie die Summe über die restlichen Zahlen und lassen Sie die Summe anschließend ausgeben.

In [ ]:

```
# do not edit this cell
given_example = (4, 1, 5, 23, 5)
```

In [ ]:

```
```

3.1.3

Schreiben Sie eine Funktion *lists_3()*.
Erstelle eine Liste von Listen, die das folgende Tic-Tac-Toe-Spiel repräsentieren (verwenden Sie Strings für X und O):
XOO
OXO
XXO
Anschließend Drehen Sie das Raster 4-mal um 90 ° im Uhrzeigersinn, wobei Sie das Ergebniss inklusieve einer leeren Zeile nach jedem Schritt wie folgt ausgeben:
XOX
XXO
OOO

OXX
OXO
OOX ...

In [ ]:

```
```

3.2 Strings

3.2.1

Warum verwenden die Deutschen immer Kommas anstatt Dezimalpunkten? Schreiben Sie eine Funktion *strings_1()*. Konvertieren Sie die in dem folgenden String angegebenen Zahlen in eine Liste von Gleitkommazahlen. Summiere Sie daraufhin alle Zahlen und geben Sie die Berechnung und ihr Ergebnis im folgenden Stil aus (Begrenze die Anzahl der Dezimalstellen für das Ergebnis auf zwei!):
$1.22 + 3.56 + 3.21 = 7.9$

In [ ]:

```
# do not edit this cell
numbers = "8,78; 2,89; 9,5; 0,71; 4,14; 1,27; 9,7; 1,65; 7,31; 6,21; "
numbers += "6,2; 9,02; 0,49; 1,73; 0,54; 0,9; 7,63; 9,69; 1,81; 8,07"
```

In [ ]:

```
```

3.2.2

Schreibe eine Funktion *strings_2()*. Nehmen Sie für jedes Wort in der gegebenen Zeichenfolge nur jeden zweiten Buchstaben und kehren Sie das Wort um. Geben Sie den resultierenden Satz aus!

In [ ]:

```
# do not edit this cell
encrypted = "MeAhGT bkDcxizuDq vnxwNoPrsb ixfoCf ysHpMmsuOj orbeJvgo bejhBt yywzZ
azl A.ugaond"
```

In [ ]:

```

```

3.2.3

Ein Text wurde in dem String *mystery* codiert und benötigt Ihre Hilfe um entschlüsselt zu werden. Schreiben Sie eine Funktion *strings_3()*, die den folgenden Schritten folgt, um den Text zu ändern und den richtigen Text zu erhalten. Geben Sie das Ergebnis aus.

1. Jeder Großbuchstabe wurde kleingeschrieben, links und rechts davon steht "big"
2. Jeder Buchstabe a hat ein x auf beiden Seiten
3. Jedes Komma wurde in einen Hashtag umgewandelt
4. Jedes Wort endet mit 1b und beginnt mit .8
5. Jedes r wurde zu einer "rat" erweitert und t wurde in eine "turtle" verwandelt
6. Jedes Leerzeichen wurde durch einen Unterstrich ersetzt

In [ ]:

```
# do not edit this cell
mystery = ".8biglbigxaxnd1b_.8derat1b_.8bigbbigeratge1b#.8biglbigxaxnd1b_.8xaxm1
b.8bigsbigturtleratome1b#"
mystery += ".8biglbigxaxnd1b.8derat1b_.8xbigabigxckerat1b#.8biglbigxaxnd1b.8derat
1b_.8bigdbigome1b#""
mystery += ".8biglbigxaxnd1b.8derat1b_.8bighbigxäxmmerat1b#_.8zukunfturtlesrateic
h1b"
```

In [ ]:

```

```

3.2.4

Schreiben Sie eine Funktion *strings_4()*. Konvertiere den gegebenen String *num* in eine Liste von zweistelligen Zahlen. Entfernen Sie die Ganzzahl 23 aus der Liste, und fügen Sie dann die Ganzzahl 96 ein. Sortieren Sie anschließend die Liste in aufsteigender Reihenfolge mit ganzen Zahlen und geben Sie den Mittelwert, Minimum und Maximum, sowie die Entfernung dazwischen aus (z. B. "Der Mittelwert ist 28").

In [ ]:

```
# do not edit this cell
num = "28561823817394"
```

In [ ]:

```

```

3.3 Dictionaries

3.3.1

Schreiben Sie eine Funktion *dicts_1()*. Bitten Sie den Benutzer, Name, Menge und Preis pro Artikel für 3 verschiedene Artikel einzugeben (verwenden Sie *input*!). Speichern Sie die eingegebenen Informationen in einem dictionary (Produktnamen als Schlüssel, die anderen beiden Werte als Tupel (Preis, Betrag); z.B. {"Milch": (1.00, 2), ...}). Verwenden Sie geeignete Datentypen! Fügen Sie das übergebene dictionary mit weiteren Produkten (*old_products*) demselben dictionary hinzu. Ändern Sie den Käsepreis auf 3,65 (aber behalten Sie die Menge). Geben Sie alle Produkte zusammen mit ihren Preisen im folgenden Stil aus:

cheese: 4.65€
flour: 0.49€
...

Berechnen Sie, was alle Produkte mit ihren angegebenen Mengen kosten würden und geben Sie das Ergebnis wie folgt aus: "The total price is 123.45€"

In [ ]:

```
# do not edit this cell
old_products = {"cheese": (4.65, 1), "flour": (0.49, 2)}
```

In [ ]:

```
```

3.3.2

Schreiben Sie eine Funktion *dicts_2()*. Lesen Sie die Datei "data.csv" ein, um ein verschachteltes dictionary der Gesundheitsprofile von vier Patienten zu erstellen. Jeder Patient hat sowohl persönliche als auch medizinische Daten. Verwenden Sie in der ersten Ebene des dictionary die Namen als Schlüssel, die Werte sollen dictionary mit key-value Paaren für Geburtsstatus, Geschlecht, Herzfrequenz und Anzahl der roten Blutkörperchen sein. Das dictionary für jeden Patient soll das gleiche Format haben. Achten Sie darauf korrekte Datentypen zu verwenden. Einheiten können ignoriert werden!
Das fertige dictionary für eine Person soll folgendermaßen aussehen:
health_profile = {"Max": {"Birth state": "Bayern", "Gender": "Male", "Resting Heart Rate": 74, "Red blood cell count": 4.33}, "Omar": {...}, ...}

In [ ]:

```
```

3.4 Tuples

3.4.1

Schreibe eine Funktion *tuples_1()*. Erstelle zwei Tupel mit den Koordinaten zweier Punkte:

- $P_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$
- $P_2 = \begin{pmatrix} 7 \\ -4 \end{pmatrix}$

Berechnen und printen Sie den Abstand zwischen den beiden Punkten
$d = \sqrt{(P_{1_x} - P_{2_x})^2 + (P_{1_y} - P_{2_y})^2}$.
Ändern Sie die x-Koordinate von P_1 auf 4. Berechne den Abstand erneut und gebe ihn aus!

In [ ]:

```
```

3.4.2

Schreiben Sie eine Funktion *tuples_2()*. Erstelle ein Tupel mit den Zahlen von 1 bis 10 (inklusive). Verwenden Sie Mehrfachzuweisung/tuple unpacking um den Variablen a bis c (Ganzzahlen) die Werte 1 bis 3 zuzuweisen, speichern Sie die restlichen Zahlen in d (eine Liste). Returnen Sie alle 4 Variablen.

In [ ]:

```
```

3.5 Sets

Schreibe eine Funktion *sets_1()*. Erstellen Sie drei sets:

- even_numbers: Enthält alle geraden Zahlen von 2 bis 100 (inklusive)
- multiples_of_3: Enthält alle Vielfachen von 3 bis 100
- multiples_of_7: Enthält alle Vielfachen von 7 bis 100

Gib nun alle Nummern aus (in dieser Reihenfolge),

- die durch 2 und 7 teilbar sind
- die ungerade und durch 7 teilbar sind

Addiere die kleinste und größte Zahlen, die entweder durch 7 oder 3 teilbar ist (aber nicht beides!) Und gebe das Ergebnis aus!

In [ ]:

```
```

4 Control Flow Statements

4.1 If/Else

4.1.1

Schreiben Sie eine Funktion *ifelse_1()*. Iteriere über die Zahlen von 1 bis 100 (beide inklusive). Für Zahlen, die durch 5 teilbar sind, geben Sie "Fizz" aus. Bei Zahlen mit einer Ziffernsumme von 7 geben Sie "Buzz" aus. Wenn beide Bedingungen zutreffen soll "FizzBuzz" ausgegeben werden. Wenn beides nicht zutrifft, printen Sie einfach die Nummer.

In [ ]:

```
```

4.1.2

Schreibe eine Funktion *ifelse_2()*, die dem Benutzer die Note in Abhängigkeit von den eingegebenen Punkten (mit *input*) mitteilt. Es sollte folgendermaßen funktionieren:

Enter your points: 77
You got a C!

Berechnen Sie die Noten nach folgender Tabelle:

Grade	Range
A	90 <= P
B	80 <= P < 90
C	70 <= P < 80
D	60 <= P < 70
E	50 <= P < 60
F	P < 50

In [ ]:

```
```

4.2 For Schleifen

4.2.1

Schreibe eine Funktion *for_loops_1()*. Erstellen Sie eine leere Liste. Fügen Sie in einer for-Schleife dieser Liste insgesammt 1000 zufälligen Ganzzahlen von 0 - 100 hinzu. Gibt eine Liste mit nur den geraden Zahlen zurück.

In [ ]:

```

```

4.2.2

Schreibe eine Funktion *for_loops_2()*. Öffnen Sie dafür die Datei "for_loops.txt". Lesen Sie Zeile für Zeile und zeigen Sie den Inhalt jeder Zeile im folgenden Stil an

Line No. 1: "Whatever line 1 says"
Line No. 2: "Whatever line 2 says"
... containing the word "end".

Stoppen Sie die Ausgabe nach der ersten Zeile mit dem Wort "end".

In [ ]:

```

```

4.3 While Schleifen

Schreibe eine Funktion *while_loops_1()*. Erstellen Sie eine leere Liste. Vordern Sie nun mit *input* den Benutzer auf eine Nummer einzugeben. Überprüfen Sie, ob das, was eingegeben wurde, tatsächlich eine ganze Zahl ist, andernfalls geben Sie eine Warnmeldung aus. Ist es eine Zahl, so hängen Sie diese an die Liste an (verwenden Sie den entsprechenden Datentyp!). Wiederholen Sie diesen Vorgang, bis die Zeichenfolge "stop" eingegeben wird. Geben Sie dann alle Nummern in der Liste in umgekehrter Reihenfolge aus.

In [ ]:

```

```