The deadline for this exercise sheet is **Monday, April 20, 2020 at 23:59 CEST.**

About the Exercise Sheets

Normally all tasks on each exercise sheet will give you points. Since this is the first week, your submission wont be graded (free point, but you need to complete them in order to get everything set up). You can usually estimate how much time you should spend on each task by the points indicated in parentheses. To pass a sheet your solution should cover at least 50% of the whole sheet.

There will be 12 exercise sheets in total for this course. Each sheet you pass will earn you a homework point. You need at least 10 homework points to pass the course.

Please solve the sheets together as a group and hand in only one solution for each sheet. First save the solutions for each task in a separate file, then pack everything into a .zip archive. Finally upload your .zip file into Vips on StudIP before the deadline.

1   Installing Python

To get started, we of course need to download and install the Python language. There are many, many flavours of Python, but for our purposes it mostly will not matter which one we choose. We will provide instructions on how to install Anaconda, which has some additional features that will come in handy if you want to pursue programming in Python further after this course. If for some reason you want an even lighter Python installation (e.g. less than 5GB or shitty internet connection), replace *Anaconda* by *Miniconda* in the following instructions.

1.1   Opening a Terminal

Windows: Press ⊞ + R and type `cmd` into the window that opens. Press ↵ .

macOS: Press ⌘ + ⎵ to open spotlight and type `terminal`. Then press ↵ .

Ubuntu: Press Ctrl + Alt + T .

1.2   Downloading Anaconda

First, open your favourite browser, go to the Anaconda website `https://www.anaconda.com/products/individual` (for Miniconda: `https://docs.conda.io/en/latest/miniconda.html`) and download the appropriate installer for your operating system (you probably have a 64-bit system). On MacOS choose the Command Line Installer, On Linux the one just called Installer. Save the file someplace where you can find it again.

1.3   Installing Anaconda

For Windows Users:

Find the installer you downloaded in 1.2 and double click it. You can accept the default options **except for the section "Advanced Options".** Make sure both the box about the `PATH` variable and the box about the system's default Python are checked! Click on Install and let the installer do its thing.

For Linux Users:

Open a terminal, navigate to the installer you downloaded in 1.2, type `bash Anaconda3-2020.11-Linux-x86_64.sh` and press ↵ . Once again press the enter key and hold it until the User License Agreement has finished.

If you agree, type `yes` and press enter. Once again press enter to confirm the location of the installation. After the installation has finished, once more type `yes` and press enter to confirm necessary additions to your `.bashrc` file.

For MacOS Users:

Open a terminal, navigate to the installer you downloaded in 1.2, type `bash Anaconda3-2020.11-MacOSX-x86_64.sh` and press ⏎. Once again press the enter key and hold it until the User License Agreement has finished. If you agree, type `yes` and press enter. Once again press enter to confirm the location of the installation. After the installation has finished, once more type `yes`. For the changes to take effect, you have to close this terminal and open a new one.

1.4   Testing your Installation

To check whether everything is set up correctly, open a terminal.

In the terminal, type `python` and press ⏎. You should now see something like this:

```
soeren:~$ python
Python 3.7.1 (default, Dec 14 2018, 19:28:38)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

This is the Python shell. If for some reason it says `Python 2.x` instead of `Python 3.7`, you need to use the command `python3` instead of `python` for the remainder of this course.

Type `import this` and see what happens!

1.5   Installing Atom

You have learned in the lecture that Python scripts are no more than simple text files. As such, you can use many different programs to edit them (but perhaps not *all* the ones you think - Microsoft Word for instance is not a pure text editor!). We suggest you use **Atom** because it is simple to use and can be adapted to more complex needs as you progress in your programming life.

You can download Atom from this website: `https://atom.io/`. The installation process should be straight-forward. You can test whether it works by typing `atom` into the terminal.
For Windows users: In order to use Atom from the command prompt easily, you should add it to your PATH system variable (in MacOS and Linux this should happen automatically):

- In the control panel navigate to *System->Advanced System Settings->Environment Variables*

- Under System Variables select PATH and click on edit

- At the end add ";C:\Users*Username*\anaconda3\condabin", where *Username* is the username under which conda was installed. If you installed miniconda replace *anaconda3*. If you installed anaconda in a different folder than the default one, you'll have to change what you add to PATH accordingly.

2   Getting Used to the Terminal

Have a look at *Programming with Python->Using the Terminal* in courseware.

2.1   Baby Steps

First, open a terminal like you did in 1.3. Try to navigate to your desktop (the desktop is just another directory). There is a list of basic terminal commands on slide 22 of this week's lecture.

It is probably a good idea to have a dedicated directory for this course. On your desktop (or in any other place that you can find again), create a new directory called `BasicPython` using the terminal. What goes wrong if you try to name it `Basic Python` (with a space)?

Now create a directory called `Week01` **inside** your `BasicPython` directory.

2.2   Some New Commands

Inside the `Week01` directory, run the command `atom my_script.py`. This will create a new file `my_script.py` and open it with Atom. Save the file and go back to the terminal.

The next command you will use is `cp <file_to_copy> <destination_of_copy>`. Use it to create a copy of `my_script.py`. Use `ls` or `dir` to show both files.

Now use `rm <file_to_delete>` to delete the file `my_script.py`. Did you notice there wasn't any warning or confirmation? The file also did not go to the trash folder, but was deleted permanently. You can do a lot of damage to your system with the terminal, so always double check what you type in.

Lastly, there is `mv <old_file> <new_file>`. It moves a file from one location to another location. This can also be used to rename a file. Use it now to rename the copy you created before to `logo.py`.

2.3   Using Auto-Complete

You may have gotten the impression that using the terminal to create directories, copy files etc. looks very cool, but is slow compared to just doing it in a file explorer. However, once you practise a bit, you will actually be much quicker this way - and a big part of that is using auto-completion. Whenever you are typing a *filename*, you can press ⇆ after the first three letters or so. The terminal will then try to guess which file you meant, based on which files exist in the current working directory. Play around with this feature by for instance typing `cp dra` and then hitting ⇆.

3   Basic Operations in the Python Shell

Have a look at *Programming with Python->The Python Shell* in courseware.

For this task you have to open a Python shell by opening a terminal as described in 1.1. and then executing the command `python`. When you are done with the task, use atom to create a file named `operators.txt` and save your results in it (on line for each result).

3.1   Operations on Numbers

In the lecture you learned how the Python shell can be used like a text-based calculator. In this task we want you to make use of it. Using the Python shell and the operators `+`, `-`, `*`, `/` and `**`, calculate the results of the following formulas:

1. $5 + 2 - 7$

2. $\frac{7}{5}$

3. $12 \cdot 274.3$

---

4. $2^{\frac{24}{7}}$

5. $\frac{3674+280398}{(3678+2321)*43}$

3.2   Operations on Text

We also learned how the operators can be applied to text in quotation marks. Which commands would you use to get the following outputs in the terminal?

1. How are you?

2. Hello! Hello! Hello! Hello!

3.3   Bonus Questions

Why are the outputs of `print("7" + "9")` and `print(7 + 9)` different?

What is the difference between `print(7 / 9)` and `print(7 // 9)`?

4   Turtle!

For this task open a Python shell once again and execute the following commands:

```python
from turtle import forward, right, left, penup, pendown, reset
forward(0)
```

You should see a turtle now! Or at the very least a moving arrow. Try moving it with the command `forward(100)`. With `reset()`, you can set it back to its initial position. What happens if you do just `forward(50)` instead of `forward(100)`? `right(90)` and `left(90)` will make the turtle turn around. `penup()` and `pendown()` influence whether the turtle will draw a line when walking or not. Try experimenting with the other commands as well!

In the lecture you have learned that Python code can not only be executed in a Python terminal step by step, but also from a `.py` file - a so called script. Your task is to write a script that will open the turtle window and make it draw a logo for your group. This logo can be anything, depending on your creativity and motivation. It could be a word, an animal, a plant - whatever you like.

Use atom to create this script and save it as `logo.py`. Then try executing it from the terminal with the command `python logo.py`. This should be handed in as well, so your solution for this exercise sheet should consist of a .zip file containing the file `operators.txt` from task 3 as well as `logo.py`.
Below you can find an example of how a script to write the letter "M" could look like:

```python
from turtle import forward, right, left, penup, pendown, reset

penup()
right(180)
forward(100)
pendown()
right(90)
forward(150)
right(150)
forward(100)
left(120)
forward(100)
```

```
13   right(150)
14   forward(150)
15
16   input()
```

You can ignore the `input()` for now, it is just there to make the script wait for you to press a key until the window closes. Now go on and train your own turtle!

Afterword

We always try to structure the exercise sheets so that all concepts you need will already have been explained in courseware. If despite that we ask for something that you have no idea how to do, try the following steps in this order:

1. Check the relevant courseware content or slides (including the grey notes) and the whole exercise sheet.

2. Use your favourite search engine on relevant keywords in conjunction with "python 3".

3. If you did not find anything on the internet, come to the coding support on Monday or Thursday.

4. If you can't make it to the coding support, send us a mail explaining your problem and how you tried to solve it so far.

We wish you all the best for the first exercise sheet!

Julia (jwippermann@uos.de), Robin (rhorn@uos.de), Kamran (kvatankhahba@uos.de) and Leonard (lfrommelt@uos.de)