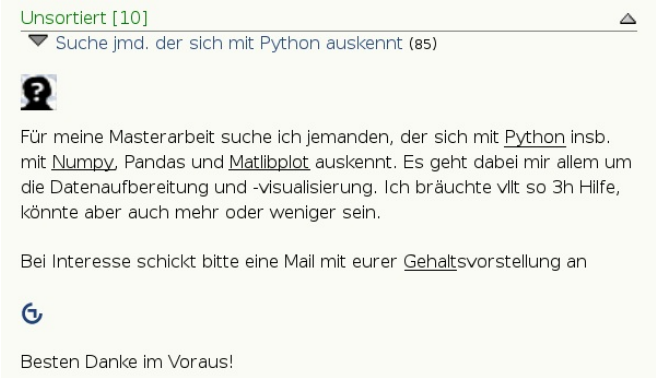
As easy as pi.py

Calculations and loops in python

Enable native scrolling

Motivation



for python

`for`-loops in python operate on so-called iterables, this makes them very versatile
Python provides a lot of iterables. For now we will focus on `lists`, `ranges` and `strings`

for python

```
>>> for elem in range(4):
>>>     print(elem)
0
1
2
3
```

Ranges start at `0` and do not include the end

for python

```
>>> fruits= ['apples', 'bananas', 'strawberries']
>>> for fruit in fruits:
>>>     print(fruit)
apples
bananas
strawberries
```

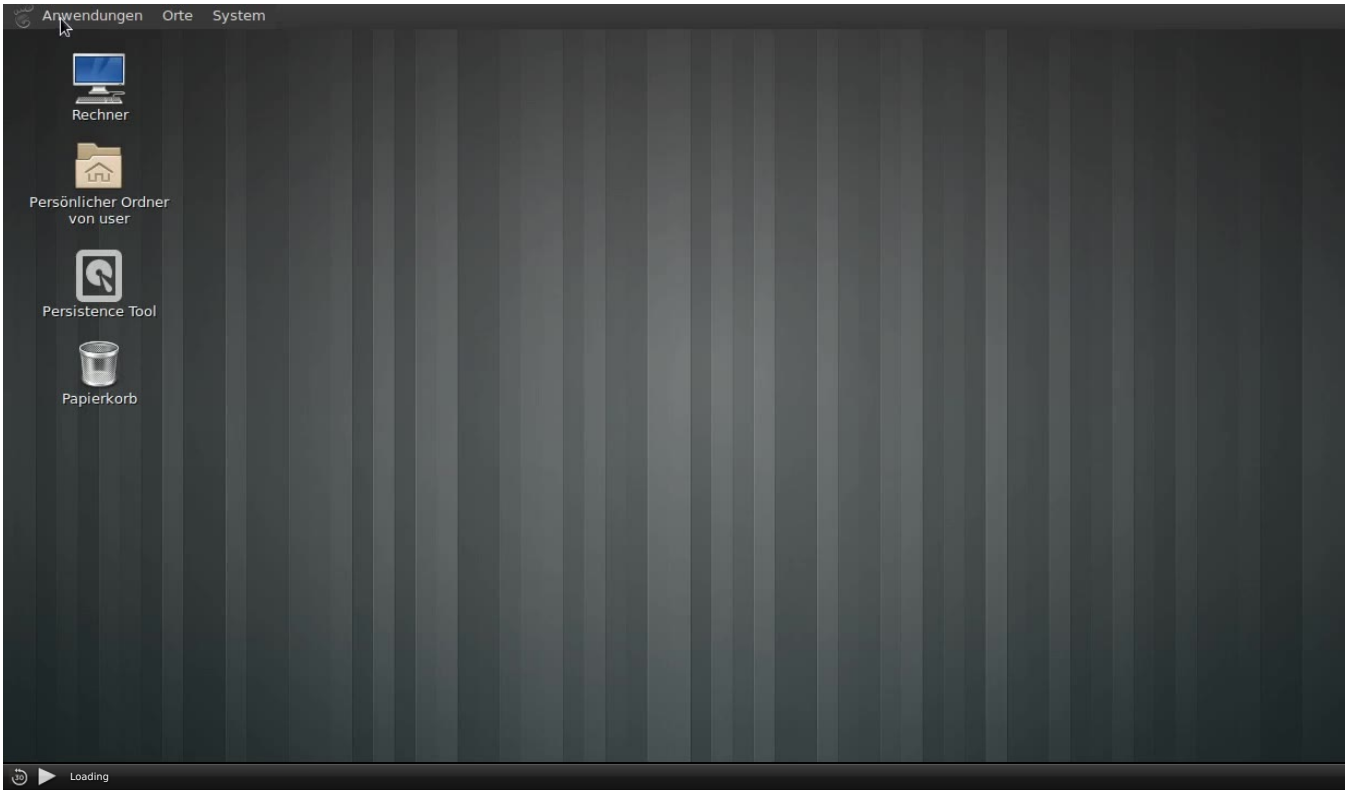
`for` can also iterate over the elements of a list

for python

```
>>> for letter in 'café':
>>>     print(letter)
c
a
f
é
```

Strings in python 3 can handle special unicode characters
Strings are also iterables

notepad.exe



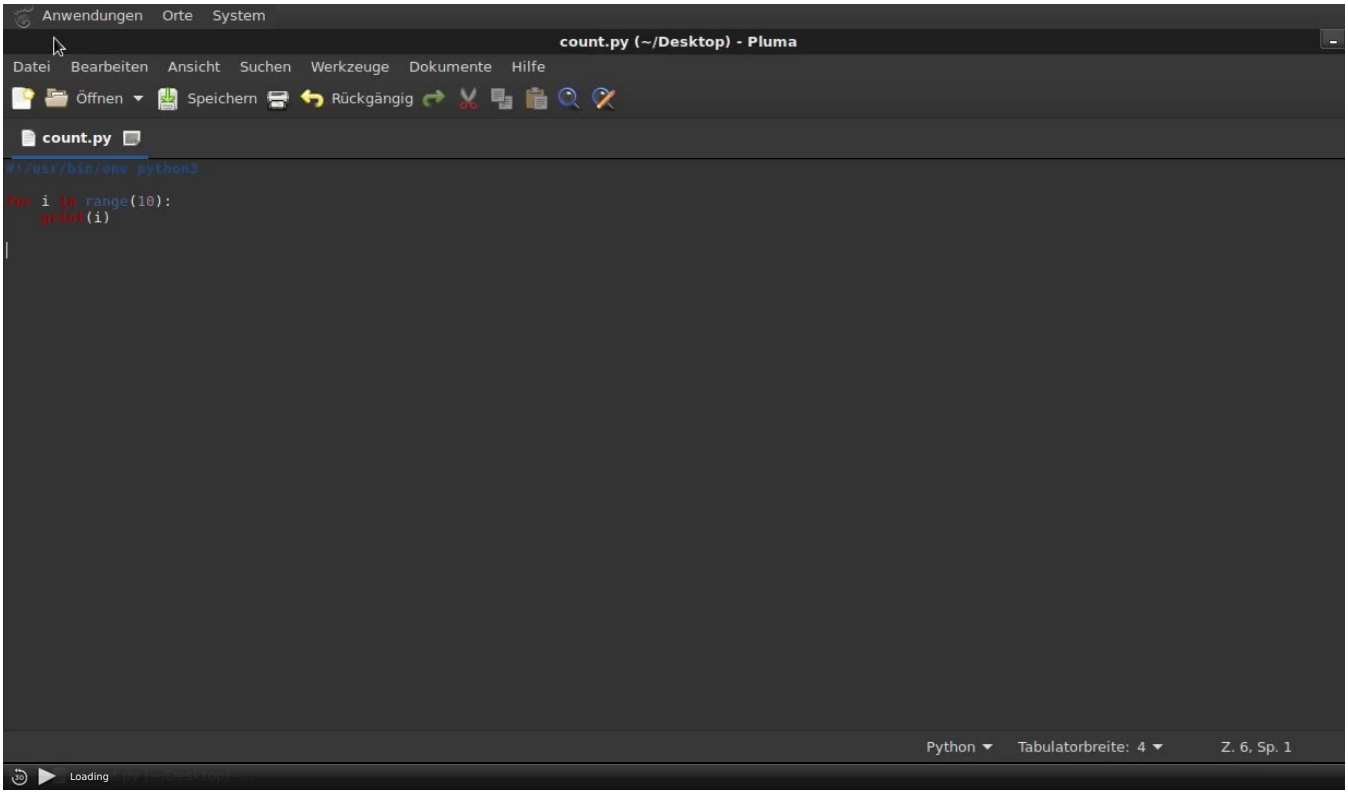
Open a text editor and save the content of the next slide in a file called `count.py`
Make sure that the editor does not add a filename-extension like `.txt`

count.py

```
#!/usr/bin/env python3

for i in range(10):
    print(i)
```

cmd.exe



Open a terminal window and use `cd` to navigate to the directory where you stored `count.py`

Type `python count.py` (on Windows) / `python3 count.py` (on Linux) and hit *enter*⏎ to execute `count.py`

Hint: Use the *Tab*↹ key to automatically complete directory/file/program names

9/14

Division 10

Edit `count.py` so that it outputs the numbers from `0.0` to `0.9` in steps of `0.1`.

```
0.0
0.1
0.2
0.3
0.4
…
```

10/14

for for

Edit `count.py` so that its output looks like this example

```
x   y
0.0 0.4
0.0 0.5
0.0 0.6
0.0 0.7
0.0 0.8
0.0 0.9
0.1 0.0
0.1 0.1
```

11/14

thagoras.py

Python has an exponentiation operator `b**e`

A square root is an exponentiation by ½

Thus, in python, the distance `d` between two points in a cartesian coordinate system can be calculated as `d= (x**2 + y**2)**0.5`

12/14

pi.py

Add a variable `inside=0` to `count.py` that is incremented by one whenever `(x**2 + y**2)` is smaller than `1`
At the end of the program `print(inside*4/100)`

pi.py

Decrease the step size from `0.1` to `0.001`
At the end of the program `print(inside*4/(1000**2))`
Try to recognize the number and find out why it is the result of the calculation

...