

Introduction to Python

May 2018

What is Programming?

- All computers are stupid.
- All computers are deterministic.
- You have to tell the computer what to do.
- You can tell the computer in any (programming) language) you like.

What is Python?

- Python is a very powerful, but easy language.
- Why easy? You can learn it easily. It is like writing down logic in pseudocode, only that the pseudocode is the real code (Unlike many other programming languages).
- Why powerful? There is nearly nothing you cannot do: Calculate, Draw, Plot, Read data (from anywhere), Simulate, Analyse data, Web applications, Make real executable programs (GUI), interactive programs, ...
- The best thing: Open Source: Probably somebody in the world has already done what you want to do.

Workflow

- Write your code/program.
- No compiling step necessary
- Execute your program.

Other languages: C,C++, Fortran. You have to compile, after writing your code, before executing.

Executing python online

- www.trinket.io
- Sign up.
- Write your code.
- Run it.

Our first line of code

Variable definition

CODE:

- `name = value`

Console Output

CODE:

- `print name`

Variable types

CODE:

- `print type(name)`

Variable types

- `string`: sequence of characters ('Hallo')
- `integer`: positive or negative natural number, inclusive zero (`..,-2, -1, 0, 1, 2,..`)
- `float`: floating point number (1,2345)
- `bool`: 'True' or 'False'

Math expressions

CODE:

- `a + b`
- `a - b`
- `a * b`
- `a / b`
- `a ** b`

Examples

CODE:

- `i = 2`
- `j = 5`
- `print i, j, i+j`
- `k = 2.5`
- `print k*i`
- `print i*k`

Logic operations

CODE:

```
• if a==1:  
    print "a ist 1"  
elif a==2:  
    print "a ist 2"  
else:  
    print "a ist irgendwas."
```

Logic operators

- `==` equal
- `<` smaller
- `>` larger
- `<=` smaller or equal
- `>=` larger or equal
- `!=` not equal

CODE:

- ```
if a <= 5 and a >2:
 print "a ist 3, 4 oder 5"
```

## CODE:

- `list = [1,2,3,4,5]`
- `list = ["ha", "he", "hi", "ho"]`
- `print "erstes Element: ", list[0]`
- `print "drittes Element: ", list[2]`

## CODE:

- ```
for element in list:  
    print "Listenelement ist ", element
```
- ```
for element in range(10):
 print "Listenelement ist ", element
```
- ```
while a <= 5:  
    print a  
    a = a+1
```

Functions

CODE:

- ```
def func_name(input1, input2):
 print input1, input2
 print input1*input2
 return input1**input2
```



# Libraries

CODE:

- `import libraryname as othername`

common libraries:

`numpy, matplotlib.pyplot, ...`

# Using libraries

CODE:

- `import libraryname as othername`

common libraries:

`numpy`, `matplotlib.pyplot`, ...

USAGE e.g.:

- `import numpy as np`
- `print np.exp(1)`
- `array = np.array([1,2,3])`
- `array_auto = np.linspace(1,10,10)`

# Pseudozufallszahlen

Pseudozufallszahlen sind **Zahlenfolgen** die durch einen **deterministischen Algorithmus** (Pseudozufallszahlengenerator) berechnet werden (und somit nicht zufällig sind). Für hinreichend kurze Sequenzen sehen sie jedoch zufällig aus. Bei jedem Start der Zufallszahlen-Berechnung mit gleichem Startwert (seed) wird die gleiche Zahlenfolge erzeugt.

# Zufallszahlen in Python

Modul laden:

- `import random`

Zufallszahlengenerator mit seed [x] initialisieren:

- `random.seed([x])`

Zufallszahl zwischen 0.0 und 1.0 aufrufen:

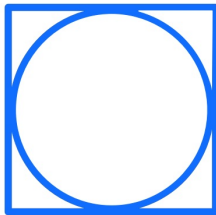
- `random.random()`

Zufallszahl zwischen a und b aufrufen:

- `random.uniform(a, b)`

*Ohne `random.seed([x])` verwendet Python die Systemzeit um einen Seed zu generieren.*

## Aufgabe: Berechnen Sie $\pi$ .



- Wählen sie ein geeignetes Koordinatensystem.
- Rechnen sie zufällige Punkte aus und bestimmen ob diese im Kreis liegen.
- Das Verhältnis der Flächen von Kreis und Quadrat entspricht dem Verhältnis der Zufallswerte in Kreis und Quadrat.