Multimedia-Programmierung

Übung 1 – Einführung in Python

Overview

...Events, Animations, Physics Simulations, Sound...

Final Project: Erstes eigenes Spiel!

Heute

python™

What is Python?

- Programming language
- Supports object oriented as well as functional programming
- Fully dynamic type system
- Runs on all major operating systems

- Goal: create a **simple**, **efficient** and **easy-to-learn** programming language

"Wer hat's erfunden?"
"Die Holländer!"

Guido van Rossum. Programmer of Python.
Source: Doc Searls

For this lecture

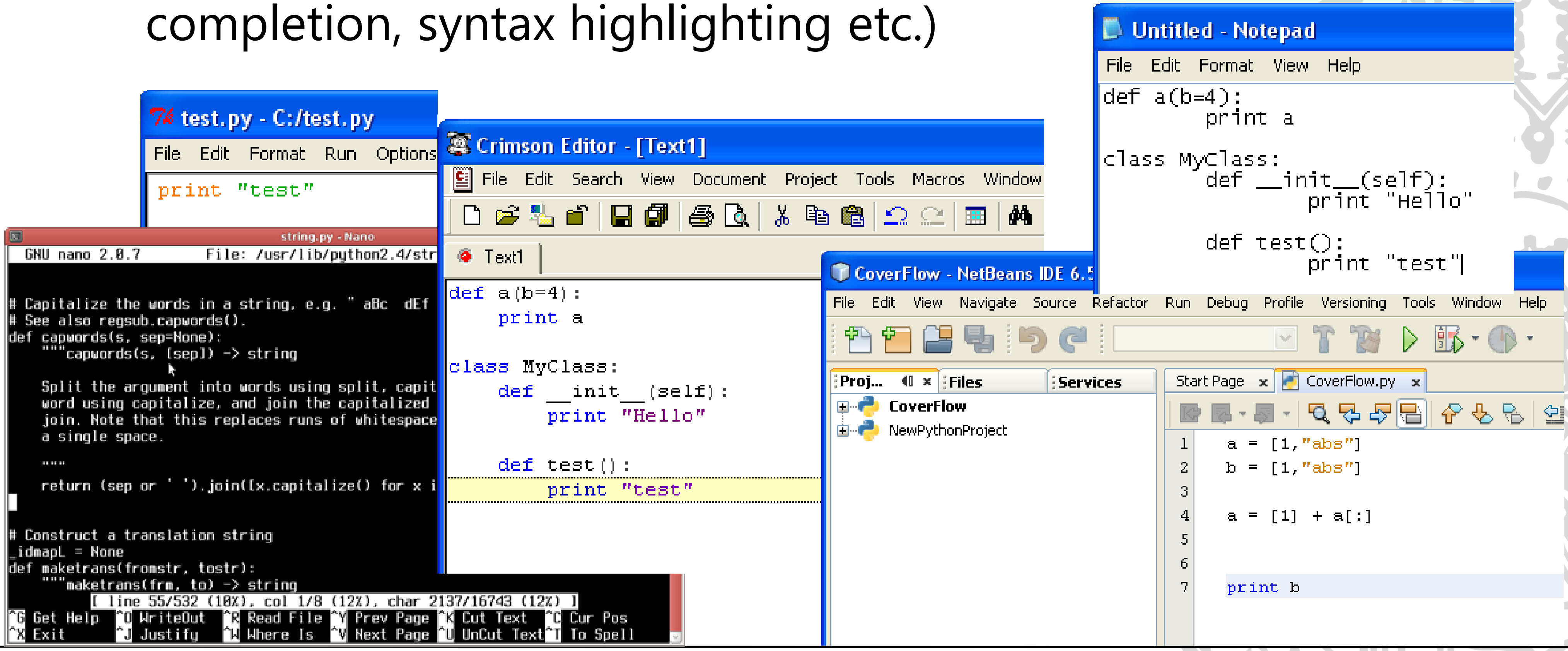
- Python **3**.10.x http://www.python.org/download/
- Command to install Pygame: python -m pip install –U pygame --user

- Recommended IDEs:
  - PyCharm (Professional version is free for students)
  - Netbeans 8.0 or higher (incl. JDK 8)
  - Eclipse 3.5 or higher
  - Atom

- Up-to-date installation recommendations: http://kidscancode.org/blog/2015/09/pygame_install/

Writing Python Code

- Python scripts are **text files**
- Thus they can be written using **any text editor**
- **IDEs** provide additional support (debugging, code completion, syntax highlighting etc.)



test.py - C:/test.py
File Edit Format Run Options
Crimson Editor - [Text1]
File Edit Search View Document Project Tools Macros Window
Text1
Untitled - Notepad
File Edit Format View Help
CoverFlow - NetBeans IDE 6.5
File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help
Proj... Files Services
CoverFlow
NewPythonProject
Start Page
CoverFlow.py
string.py - Nano

```
print "test"
```

```
  GNU nano 2.0.7          File: /usr/lib/python2.4/str

# Capitalize the words in a string, e.g. " aBc  dEf
# See also regsub.capwords().
def capwords(s, sep=None):
    """capwords(s, [sep]) -> string

    Split the argument into words using split, capit
    word using capitalize, and join the capitalized
    join. Note that this replaces runs of whitespace
    a single space.

    """
    return (sep or ' ').join([x.capitalize() for x i

# Construct a translation string
_idmapL = None
def maketrans(fromstr, tostr):
    """maketrans(frm, to) -> string
          [ line 55/532 (10%), col 1/8 (12%), char 2137/16743 (12%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
def a(b=4):
    print a

class MyClass:
    def __init__(self):
        print "Hello"

    def test():
        print "test"
```

```
def a(b=4):
        print a

class MyClass:
        def __init__(self):
                print "Hello"

        def test():
                print "test"
```

```
a = [1,"abs"]
b = [1,"abs"]

a = [1] + a[:]


print b
```

We use Python 3

Former Python 2 has different syntax and is still out there – Keep that in mind when googling for support

Aspect	Python 2	Python 3
Print function	**print** 'Hello, World!'	**print**('Hello, World!')
Integer division	3 / 2 = 1	3 / 2 = 1.5
Exceptions	**raise** IOError, "file error"	**raise** IOError("file error")
Error handling	**except** NameError, err:	**except** NameError **as** err:
Next function	next(my_generator) my_generator**.**next()	next(my_generator)

Python Code is compact

```
public class Hello {

    public static void main (String args[]) {
        System.out.println("Hello World!");
    }

}
```

```
print (“Hello World!“)
```

Python code is intuitive

Java

```
String[] a = [“test1“];
String[] b = [“test2“];

String[] c = ArrayUtils.addAll(a, b);

or

String[] a = [“test1“];
String[] b = [“test2“];
String[] c = new String[a.length+b.length];
System.arraycopy(a, 0, c, 0, a.length);
System.arraycopy(b, 0, c, a.length,
b.length);
```

python™

```
a = [“test1“]

b = [“test2“]


c = a + b
```

Python code is fun

Java

```
String a = “test“;

String b = ““;

for(int i = 0; i<5; i++) {
    b = b + a;
}
```

python™

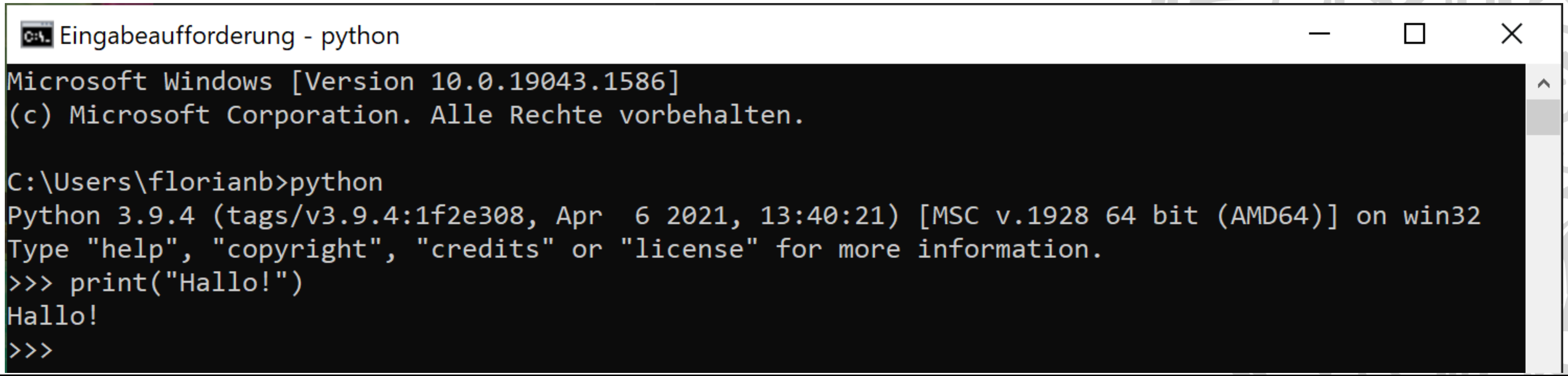
```
a = “test“

b = a * 5
```

Executing Python Code

Interactive Mode

- Lines of Python code can be directly interpreted by the Python interpreter
- Results are immediately visible
- Comes with all standard Python installations
- Mac OS X/Linux: type “python” in the command shell/Terminal
- Windows: e.g. start python.exe from your Python folder



```
Microsoft Windows [Version 10.0.19043.1586]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\florianb>python
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hallo!")
Hallo!
>>>
```

Executing Python Code

Python Scripts

- Python programs are usually called scripts
- Script files end on .py, sometimes .pyw in Windows
- To execute a script use the python interpreter followed by the location of the script
- For example:

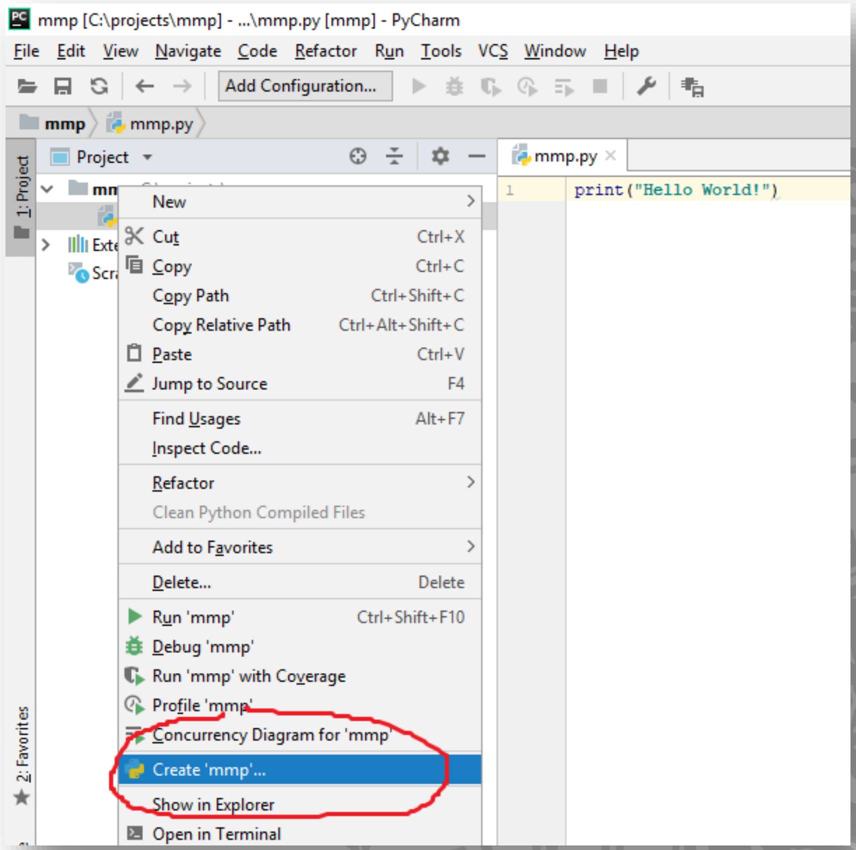
```
python helloworld.py
```

Executing Python Code

Python Scripts - PyCharm

Initially:

- Right-click your script file, e.g. `mmp.py`
- Select `create 'mmp'`…



mmp [C:\projects\mmp] - ...\mmp.py [mmp] - PyCharm

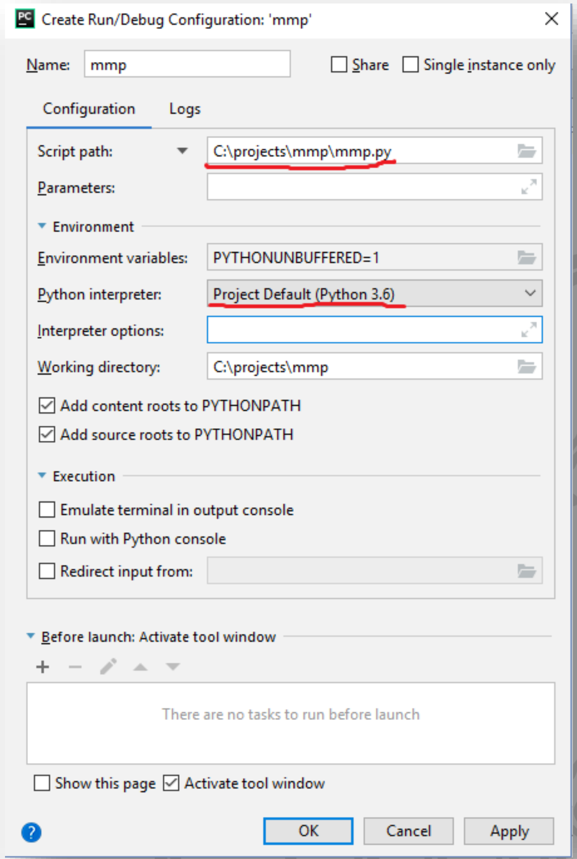
```
print("Hello World!")
```

Executing Python Code

Python Scripts - PyCharm

- Ensure that `Script path` and `Python interpreter` are set

- Finally click the “run” button


Create Run/Debug Configuration: 'mmp'

Name: mmp
☐ Share ☐ Single instance only

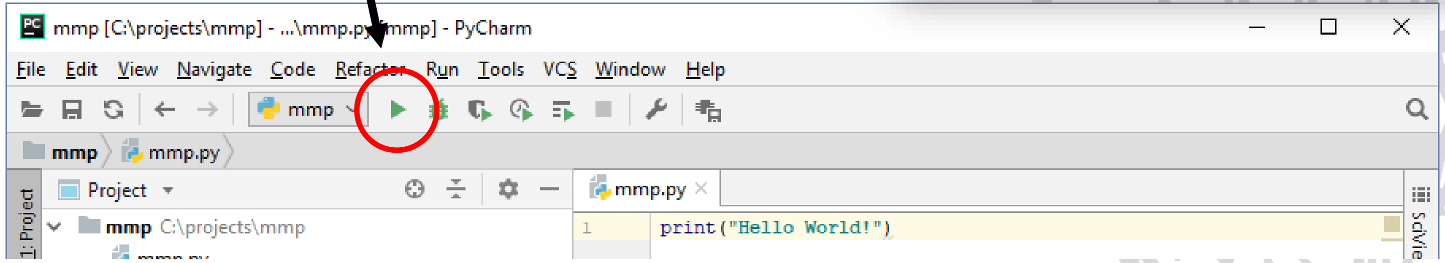
Configuration | Logs

Script path: C:\projects\mmp\mmp.py
Parameters:

Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Project Default (Python 3.6)
Interpreter options:
Working directory: C:\projects\mmp
☑ Add content roots to PYTHONPATH
☑ Add source roots to PYTHONPATH

Execution
☐ Emulate terminal in output console
☐ Run with Python console
☐ Redirect input from:

Before launch: Activate tool window
There are no tasks to run before launch
☐ Show this page ☑ Activate tool window



```
print("Hello World!")
```

Your first Python Script

Coding Challenge

- Write a Python scripts that prints your name on the console
- Run it via
    - The commandline
    - An IDE of your choice

Where the %$& are my delimiters?

- Python does not use special characters as delimiters (e.g. ‘{ ‘ and ‘}’ in Java)
- Blocks are delimited by indentations/whitespaces

```
a = 1
b = 2

if a > b:
    a = 10
    print(a)
else:
    a = 100
    print(a)
```

- editor support recommended
- forces the programmer to write clean and readable code
- a line of code cannot exceed several lines

allowed:

```
a = 1 + 2
```

forbidden:

```
a = 1
+ 2
```

allowed:

```
a = 1 \
+ 2
```

Everything’s an Object

with Consequences

Define:

```
def b():
    x = 0
    print(x)

b()
b = 4
b()
```

Output:

```
0
…
TypeError: 'int' object is not callable
```

“harharhar”

id() returns the identifier of the object

is can be used to check whether two objects are the same

Everything’s an Object

Types

Define:

```
def b():
    x = 0
    print(x)

print(type(b))
b = 4
print(type(b))

print(isinstance(b,int))
```

Output:

```
<type 'function'>
<type 'int'>
True
```

type() can be used to get the type of an object

isinstance() returns true if an object has a specific type

Types - Examples

- None
  - None
- Numbers
  - int (e.g. 2)
  - float (e.g. 2.0)
  - bool (True and False)
- Sequences
  - str (e.g. “zwei”)
  - tuple (e.g. (1,2))
  - List (e.g. [1,2])
- Callable types
  - functions
  - methods

Yes, capital letters!!

and many many more …

Comments

or: Being a Good Programmer

```
print("Who stole my Monkey?“) # weird but I‘ll let it in
a = 1
b = 2
print(a + b) # I hope it‘ll output 3

# print "bye“
```

NebeansTip:

str+shift+c comments the whole selection

Output:

```
Who stole my Monkey?
3
```

Documentation

or: Being a Good Programmer 2

```
def a():
    """This is function a"""
    return 1
print a.__doc__
```

Output:

```
This is function a
```

Functions

Define:

```
def a():
    print("I am function a“)

def b(text):
    return "I don‘t like "+text
```

Use:

```
a()
print(b("function a"))
```

Output:

```
I am function a
I don‘t like function a
```

Functions

Default Parameters

Define:

```
def test(a=1,b=2,c=3):
    print(a+b+c)

test(1)
test(2,2)
test(c=2)
```

Output:

```
6
7
5
```

Keyword arguments can be used to manipulate specific parameters only.

Namespaces

Local and Global Variables I

Define:

```
def b():
   x = 0
   print(x)

x = 2

b()
print(x)
```

Output:

```
0
2
```

Namespaces

Local and Global Variables II

Define:

```
def b():
   global x
   x = 0
   print(x)

x = 2

b()
print(x)
```

Output:

```
0
0
```

Namespaces

Local and Global Variables – Episode III

Define:

```
def b():
   x = 0
   print(locals())

b()
```

Output:

```
{'x': 0}
```

The functions locals() and globals() can help to get an overview.

Strings

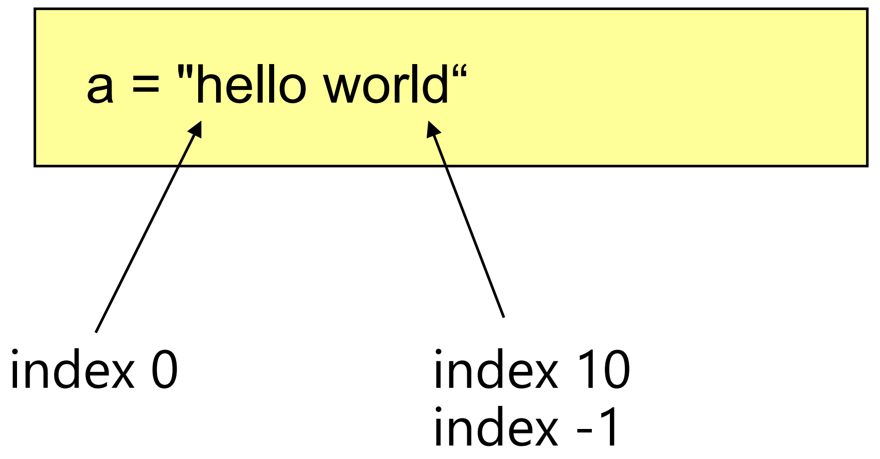
Range Slicing

The range slice notation can be used to access substrings.

string_name[x:y]
x: “from” index starting from 0 (included)

y: “to” index starting from 0 (excluded)

Define:



```
a = "hello world“
```

Strings

Examples

Define:

```
a = "hello"
print(a[0])
print(a[0:])
print(a[0:2])
print(a[0:len(a)])
print(a[2:])
print(a[:2])
print(a[2:4])
print(a[-1])
```

Output:

```
h
hello
he
hello
llo
he
ll
o
```

Attention: strings are immutable!

```
a[2] = "c"
```

```
...
TypeError: 'str' object does
not support item assignment
```

Strings

Formatted Text

Define:

```
print """lalala
test:
        aha"""
```

Output:

```
lalala
test:
        aha
```

Formatted strings are defined using """.

Strings

raw Strings

Define:

```
print("lalala\ntest“)
```

```
print(r"lalala\ntest“)
```

Output:

```
lalala
test
```

```
lalala\ntest
```

Adding an “r” to the string creates a raw string.

Lists a.k.a. Arrays

Define:

```
a = [1,3,"a","b"]
print(a)
print(a[0])

a[0] = 2
print(a)

print(2 * a)
```

Output:

```
[1, 3, 'a', 'b']
1
[2, 3, 'a', 'b']
[2, 3, 'a', 'b‘,2, 3, 'a', 'b']
```

Lists can contain any types (even mixed).

Dictionaries

Define:

```
priceDict = {'mehl': 99, 'butter': 78}

print(priceDict['mehl'])
print(priceDict.keys())

priceDict['oel'] = 112

print('oel' in priceDict)
```

Output:

```
99
['butter', 'mehl']
True
```

Dictionaries store key-value-pairs.

If-Statement

Define:

```
a = 0
if a > 0:
    print("a>0“)
elif a == 0:
    print("a=0“)
else:
    print("none“)
```

Output:

```
a=0
```

if…elif…else

Loops

Define:

```
a = [1,3,"a","b"]

for x in a:
    print(x)

while True:
    print("This will never end. :-s“)
```

Output:

```
1
3
a
b
This will never end. :-s
...
```

Don’t try this at home!

break stops a loop

continue skips to the next part of the loop

Classes

Constructor and Methods

Define:

```
class HelloWorld:
    def __init__(self):
        print("Hello World“)

    def test(self):
        print(“test“)
```

Use:

```
a = HelloWorld()
a.test()
```

Output:

```
Hello World
test
```

Modules

File test.py:

```
def a():
    print("there we are“)

def b():
    print("function b“)
```

Use:

```
import test

test.a()
```

Or:

```
from test import a

a()
```

Output:

```
there we are
```

Random Module

- The module random contains functions to create random numbers, lists etc.
- randint(a,b) creates a random number of the interval [a,b]
- random() creates a random float of the interval [0.0,1.0]
- shuffle(list) randomly shuffles a list
- Etc.
- Object Random() contains all those functions as well

```
import random

test = random.Random()
print(test.random())
print(random.randint(0,3))
```

Working with Files

Reading the Lines

example.txt:

```
line1
line2
cheese cake
cat
```

Open File:

```
file = open("example.txt", "r")
print(file.readline())
print(file.readline())
file.close()
```

Output:

```
line1
line2
```

open(filename,mode)

mode: ‘r’ for read, ‘w’ for write

‘a’ for append

Working with Files

Iterating all Lines

example.txt:

```
line1
line2
cheese cake
cat
```

Open File:

```python
file = open("example.txt", "r")
for line in file:
    print(line)
```

Output:

```
line1
line2
cheese cake
cat
```

Reading Input from the Command Line

Console:

```
a = raw_input("Name:")
```

Output:

```
Name:
```

Waits for user input. If necessary it waits forever. ;-)

input(prompt) is used to get input that is already converted to a type (e.g. an integer)

Exceptions

- Baseclass BaseException
- Own exceptions should be extended from class Exception
- Exceptions can be raised:

```
raise NameError("unknown name")
```

- try … except to handle exceptions

```
try:
    test = open("test.txt", "r")
except IOError:
    print("file doesn‘t exist“)
```

Endless Calculator

Coding Challenge

- Ask the user for a start number
- Then, endlessly…
  - Ask the user for a calculation method (e.g. „add")
  - And a next number
  - Print the result of the calculation (so far)

```
What is your first number?
➢  42
What do you want to do next?
➢  Add 50
Your result is 92. What do you want to do next?
➢  Subtract 9
Your result is 83. What do you want to do next?
…
```

Useful Links

- Python:
  - http://docs.python.org/
- Tutorials
  - http://www.learnpython.org
  - https://docs.python.org/3/tutorial/