HTWG KONSTANZ - WISE 2016/17

Prof. Dr. Matthias Franz, Theresa Kocher

BILDVERARBEITUNG ÜBUNGEN

ORGANISATORISCHES

WANN	Di. 15:45 - 17:15 evt. 2. Gruppe 17:30 - 19:00
WO	F130
WER	2-er Teams
WIE OFT	ein Übungsblatt pro Woche
ABGABE	bis zur nächsten Stunde: per Mail an tkocher@htwg-konstanz.de und mündlich kurz vorstellen

ÜBUNGSAUFGABEN

SPRACHE	Python und C/C++
LIBRARY	Numpy, Pillow
IDE	PyCharm

PYTHON EINFÜHRUNG

- Interpretersprache
- ermöglicht objektorientierte und funktionale Programmierung
- einfach zu lernende, aber mächtige Programmiersprache

Anwendungsbereiche:

- Webentwicklung, Wissenschaftliche Anwendungen, Datei-Management, Prozess-Management, TCP-/UDP-Kommunikation, MIME-Nachrichten, E-Mails, Datenbanken

PYTHON IM VERGLEICH

- deutlich kürzerer Code

- Programmstruktur:
  Einrücken statt Klammern;
  Zeilenende statt Semikolon

INTERPRETER

- analysiert und führt Code aus

➡ langsamer zur Ausführungszeit

➡ auf jeder Rechnerarchitektur lauffähig

COMPILER

- übersetzt in Maschinencode im voraus, der dann vom Prozessor direkt ausgeführt wird

➡ schneller zur Ausführungszeit

➡ bei jeder Änderung erneut kompilieren

DATENTYPEN

BEISPIEL	DATENTYP	BESCHREIBUNG
i = 1	Integer	Ganze Zahl
f = 0.1	Float	Gleitkommazahl
s = “hello“	String	Zeichenkette
l = [1, 2, 3]	Liste	Veränderbare Sequenzen
t = (1, 2, 3)	Tupel	Unveränderbare Sequenzen
d = {1: “eins“, 2: “zwei“, 3: “drei“}	Dictionary	Wörterbuch, auch Hash oder assoziatives Array

OPERATOREN

MATHEMATISCHE OPERATOREN	FUNKTION
+, -	Addition, Subtraktion bzw. Negation
*, /	Multiplikation, Division
**	Potenz
%	Modulo (ganzzahliger Rest)
divmod(x,y)	Funktion, die sowohl x/y als auch x%y zurückgibt

OPERATOREN

BOOLSCHE OPERATOREN	FUNKTION
<,>	kleiner als, größer als
<=, >=	kleiner als oder gleich, größer gleich
==, !=	gleich, ungleich
=	weist einen Wert zu

OPERATOREN

BOOLSCHE OPERATOREN	FUNKTION
a in b	Listen-Operator, testet ob Element a in Liste b ist.
not	Kehrt den Wahrheitswert eines Ausdrucks um.
a is b	Gleichheit
a is not b	Ungleichheit
a and b	Ist wahr, wenn a und b wahr sind.
a or b	Ist wahr, wenn a oder b wahr ist.

PYTHON VERSIONEN

Python 2.7.x

```
print 'Hello, World!'

print('Hello, World!')
```

```
3 / 2       # 1

3 // 2      # 1

3 / 2.0     # 1.5

3 // 2.0    # 1.0
```

Python 3.x

```
print('Hello, World!')
```

```
3 / 2           # 1.5

3 // 2          # 1

3 / 2.0         # 1.5

3 // 2.0        # 1.0
```

Manche Libraries wurden noch nicht zu Python 3 übertragen (z.B. eine for Web Development)

LINKS

- Beschreibung der Standardobjekte und -module:
  https://docs.python.org/3.3/library/index.html#library-index
- formellere Definition der Sprache:
  https://docs.python.org/3.3/reference/index.html#reference-index
- **PEP - Python Style Guide:**
  https://www.python.org/dev/peps/pep-0008/

Python lernen - Online-Kurse:

- http://www.programiz.com/python-programming#learn-python-tutorial
- https://www.codecademy.com/learn/python
- https://s3.amazonaws.com/learntocodewith.me/BONUS+-+My+Favorite+Python+Resources.pdf

THE ZEN OF PYTHON

Philosophie: Programmlesbarkeit

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one-- and preferably only one --obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea -- let's do more of those!

https://www.python.org/dev/peps/pep-0020/

Beautiful is Better Than Ugly.

The Zen of Python

```
if is_valid(a) and b == 0 or s == ,yes':
```

- leserlich, verständlich

```
if (is_valid(a) && b == 0 || s == 'yes') {
```

- kryptisch

> Explicit is better than implicit
>
> Namespaces are one honking great idea -- let's do more of those!
>
> The Zen of Python

```
import os

print os.getcwd()
```

- Herkunft der Funktion ist klar

```
from os import *

print getcwd()
```

- Herkunft ist unklar

Sparse is better than dense.

(= „Don't try to stick too much code on one line.“)

The Zen of Python

```
if i > 0:
    return sqrt(i)
elif i == 0:
    return 0
else:
    return 1j * sqrt(-i)
```

- klar

```
if i>0: return sqrt(i)
elif i==0: return 0
else: return 1j * sqrt(-i)
```

- viel Code in einer Zeile wird schnell unverständlich

Readability Counts.

The Zen of Python

```
print "Hello world!"
```

- Python - Hello World Skript

```
#include <stdio.h>

int main(void)

{

    printf("Hello, world!\n");

    return(0);

}
```

- C - Hello World Programm

In the face of ambiguity, refuse the temptation to guess.

The Zen of Python

```
if not a and b:

    do something
```

- könnte fälschlicherweise als:
  if not (a and b)
  verstanden werden

```
if b and not a:

    do something

# oder:

if (not a) and b:

    do something
```

- keine Mehrdeutigkeit

PYTHON-PROGRAMM

```
#!/usr/bin/env python3.5
# -*- coding: utf-8 -*-


class Hello(object):
     def __init__(self, word):
          self.name = word
     def get(self):
          return self.name
     def say(self):
          print(self.name)
def main():
  ausruf = Hello('Mustermann')
        print(ausruf.get())
        ausruf.say()
if __name__ == ,__main__':
       main()
```

shebang line

Encoding (Standard: ASCII)

Definition von Klasse

Definition von Methode

__name__ Attribut ermöglicht Ausführung

als **Standalone** Programm oder als **Modul**

PILLOW

- Lesen eines Bildes
- Anzeigen eines Bildes

```
from PIL import Image


# read image from path
img = Image.open(„monkey.jpg")


# show image
img.show()
```

NUMPY

- Bild als Numpy Array
- nützliche Funktionen:
  img.size
  img.shape
  img.mean()
  img.mean()
  img.max()
- Konvertieren von Pillow Image zu Numpy Array und andersrum

```
from PIL import Image
import numpy as np


image = Image.open(„monkey.jpg")
np_array = np.array(image)
print(np_array.size)
print(np_array.shape)
print(np_array.mean())
print(np_array.max())
```

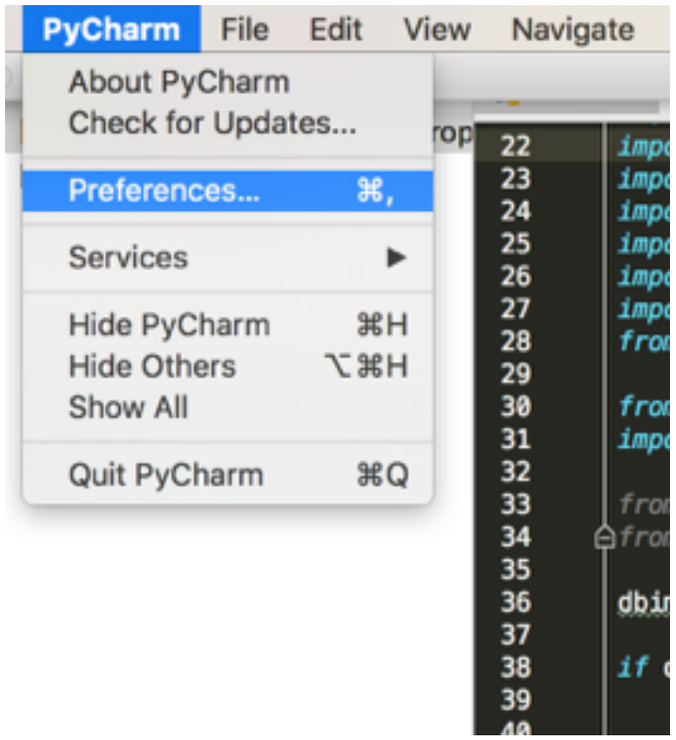
```
from PIL import Image

import numpy as np


# read image from path
img = Image.open(„monkey.jpg")

# convert to numpy array
img = np.array(im)
# do anything
processed_img = do_stuff(img)

# convert to pillow image
img = Image.fromarray(processed_img, ‚RGB')

# show image
img.show()
```

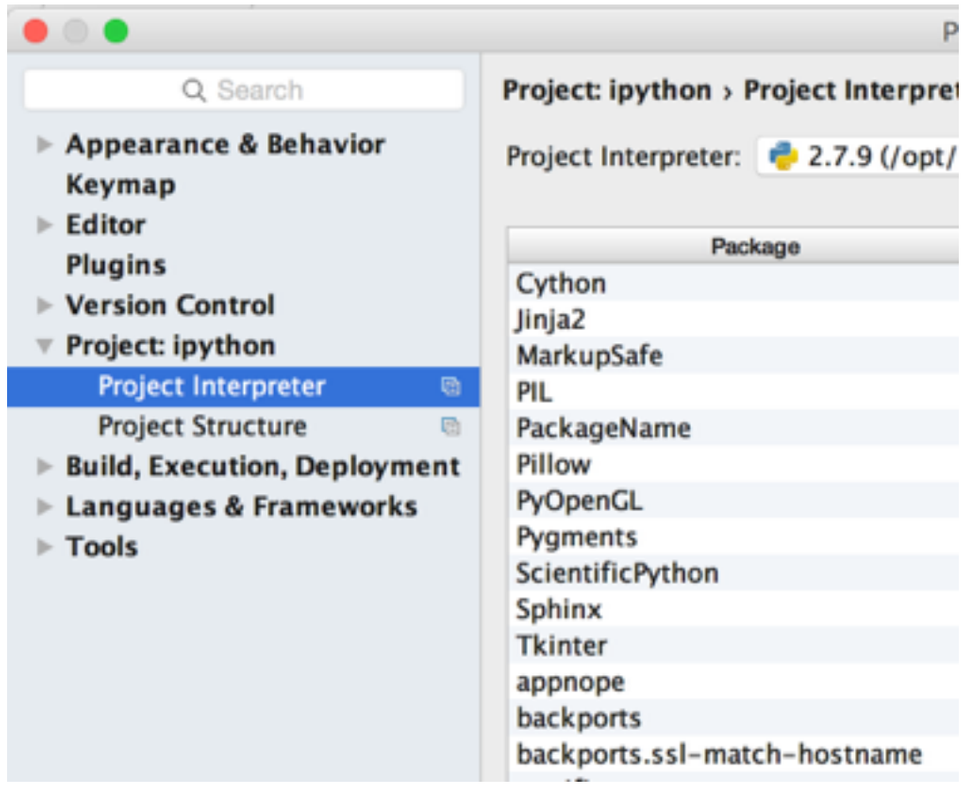
GRUPPEN

- 2-er Teams
- evtl. an 2 Termine:
  Gruppe A: Di. 15:45 - 17:15
  Gruppe B: Di. 17:30 - 19:00
- Teams eintragen und welcher Termin möglich

PYCHARM

- Einrichten
- Interpreter Path zu C:\Anaconda3\python.exe





FARBKANÄLE UND SPIEGELUNG

- Bild einlesen mit Pillow
- Bild zu Numpy Array konvertieren
- Funktion zum Aufteilen von Bildern in einzelne Farbkanäle
- Funktion um Bild zu Spielgen (an x- und y-Achse)
- Bild zurück konvertieren zu Pillow Image
- Bild anzeigen