

Alexander Jenke, Theodor Straube

# Übung1: Einführung

Programmierkurs Python // Mittwoch, 30. Oktober 2019

# ToC

- Organisatorisches
- IDEs
- Hello World
  - Interpreter
  - print()
  - Funktionen
- Simple Calculator
  - Import
  - Main-Boilerplate
  - input()
  - int()

# Organisatorisches

## Tutoren

### Alexander Jenke

alexander.jenke@tu-dresden.de

- Master Informatik, 3. Sem
- Seit 2015 an der TU
- Schwerpunkte:
  - Rechnernetze
  - Künstliche Intelligenz (Deep Learning)
- Folien & Übungsaufgaben unter <https://github.com/AlexanderJenke/python-lessons>



### Theodor Straube

theodor.straube@mailbox.tu-dresden.de

- Diplom Informatik, 9. Sem
- Seit 2015 an der TU
- Schwerpunkte:
  - Nicht festgelegt ;)

Hier bitte ein  
Bild einfügen

# Organisatorisches

## Termine

- Zeitraum: 28.10.2019 – 09.02.2020
- Weihnachtsferien: 22.12.2019 – 05.01.2020
- Buß- und Bettag: 20.11.2019 ⇒ Ersatz-Termin
- 2 Wochen pro Themenblock
  - Erste Woche Einführung und Aufgaben
  - Zweite Woche Fragen und Lösungen
- PyTorch-Einführung gegen Ende des Semesters (Deep Learning Framework)

# IDEs - PyCharm

- IDE  $\triangleq$  integrated development environment
- Integrierter Python Interpreter
- Debugger, Coverage Measurement, Profiler, uvm.
- Autovervollständigung
- Gratis Lizenz als Informatikstudent der TU Dresden

<https://www.jetbrains.com/shop/eform/students>

```
1 import torch
2 import torch.nn as nn
3
4 device = torch.device('cpu')
5
6 class InvertibleBlock(nn.Module):
7     def __init__(self, S1, T1, S2, T2, numChannels):
8         super(InvertibleBlock, self).__init__()
9         self.T1 = T1
10        self.S1 = S1
11        self.T2 = T2
12        self.S2 = S2
13
14        self.perm = torch.randperm(numChannels)
15        self.permInverse = torch.argsort(self.perm)
16
17        #self.extreme = torch.FloatTensor([10]).to(device)
18        self.extreme = torch.FloatTensor([1e2]).to(device)
19
20    def forward(self, u):
21        bs = u.shape[0]
22        c = u.shape[1]//2
23        u = u[:,self.perm]
24        u1,u2 = torch.split(u, c, dim=1)
25        v1 = u1*torch.exp(torch.min(self.S2(u2), self.extreme)) + self.T2(u2)
26        v2 = u2*torch.exp(torch.min(self.S1(v1), self.extreme)) + self.T1(v1)
27        v = torch.cat((v1,v2), dim=1)
28        v = v[:,self.permInverse]
29        return v
30
31    def inverse(self, v):
32        bs = v.shape[0]
```

Run: trainGaussModes x

/usr/local/bin/python3.7 /Users/alexanderjenke/Documents/NCT/INN/InvertibleNetworkTest/trainGaussModes.py

Output directory: out

Created dataset with 5 modes (3 unique labels)

Created dataset with 5 modes (3 unique labels)

Traceback (most recent call last):

File "/Users/alexanderjenke/Documents/NCT/INN/InvertibleNetworkTest/trainGaussModes.py", line 34, in <module>

datasetTest.toImage( os.path.join( outDir, "test\_dataset.png" ) )

File "/Users/alexanderjenke/Documents/NCT/INN/InvertibleNetworkTest/GaussModes.py", line 89, in toImage

self.samplesToImage( samples, filename )

File "/Users/alexanderjenke/Documents/NCT/INN/InvertibleNetworkTest/GaussModes.py", line 75, in samplesToImage

lblID = torch.argmax( lbl )

KeyboardInterrupt

Process finished with exit code 1

# Hello World

- Der Python Interpreter wird direkt in der Konsole mittels `python3` gestartet.
- Python-Skripte haben die Endung: `.py`
- Python-Skripte können mittels `python3 script.py` ausgeführt werden
- Python-Skripte können mittels `python3 -i script.py` interaktiv ausgeführt werden
- Ausgaben auf die Konsole erfolgen mit der `print()` -Funktion
- Eigene Funktionen definieren:

```
def name_der_funktion():  
    do_some_stuff()
```

```
alexanderjenke@MacBook-Pro ~ % python3  
Python 3.7.0 (default, Oct 2 2018, 09:18:58)  
[Clang 10.0.0 (clang-1000.11.45.2)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

# Simple Calculator

- Funktionen einer Datei im selben Verzeichnis können mittels `import skript.py` aufrufbar gemacht werden
- Code innerhalb der main-boilerplate wird nur ausgeführt, wenn das Skript ausgeführt wird.  

```
if __name__ == '__main__':  
    do_some_stuff()
```
- Eingaben auf der Konsole können mittels `input()` gelesen werden.
- Mit `var = input()` können sie in eine Variable namens var gespeichert werden.
- Strings können mit `+` zusammengefügt werden.
- Strings können mit `int()` zu Integern umgewandelt werden.

# Fibonacci Reihe

- Beim Aufruf von Funktionen können Variablen übergeben werden

```
def foo(bar):  
    print(bar)  
  
foo("Hello World!")
```

- Das If-Statement führt abhängig von Bedingungen unterschiedlichen Code aus

```
if bar == "Hello World":  
    print("Hallo Welt")  
  
elif bar > 0:  
    print(bar - 1)  
  
else:  
    foo(bar)
```

- While-Schleifen wiederholen Code, solange eine Bedingung erfüllt ist

```
i = 0  
while i < 5:  
    print(i)  
    i += 1
```

- For-Schleifen führen Code n-mal aus

```
for i in range(5):  
    print(i)
```

- Logische Ausdrücke:

==	gleich
!=	ungleich
>	größer
<	kleiner



# nächste Woche

- PEP8
- Facts about Python
- String Operationen
- mehr zu Funktionen
- list comprehensions