TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät für Elektro- und Informationstechnik, Professur für Grundlagen der Elektrotechnik und Elektronik

FliK Modul 2020

Deep Learning mit Keras

Steffen Seitz, Marvin Arnold & Markus Fritsche

Prof. Ronald Tetzlaff

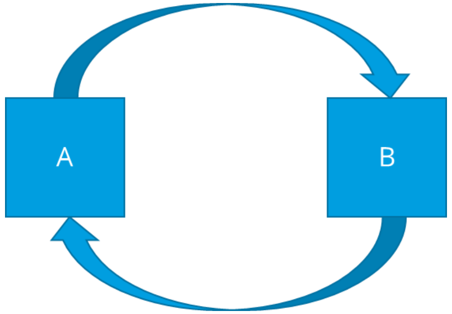
Dresden, 19-23.10.

Disclaimer

This course is for **you**. Don't hesitate to **ask** us!
Many of you will have questions and **we can't answer all** of them instantaneously, please respect that.

The scope of the course is to give you an **overview** to modern Machine Learning/Deep Learning and that you are able to build some of the most important architectures on your own. We will try to **skip heavy math/coding** but we can't really **leave out everything**. Some things will be **hard to explain** since you need some domain knowledge from Computer Science, but take your time and think about it!

Sometimes you require knowledge of A to understand B and vice versa! → Try to keep going!



To help you to get over this obstacle, course will **consist of lectures and exercises** which will also serve as a kind of proof.

Schedule

Monday

Introduction
HPC and GIT
Jupyter Notebooks
Python Basics
Neural Networks with Keras
Keras Basics
MLP

Tuesday

Convolutional Neuronal Networks
BatchNorm
Activation Maps
CAM/GradCam

Wednesday

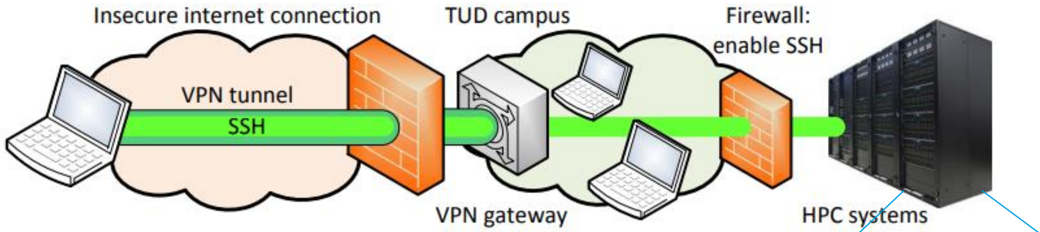
Recurrent Neural Networks
GAN
Autoencoder/VAE

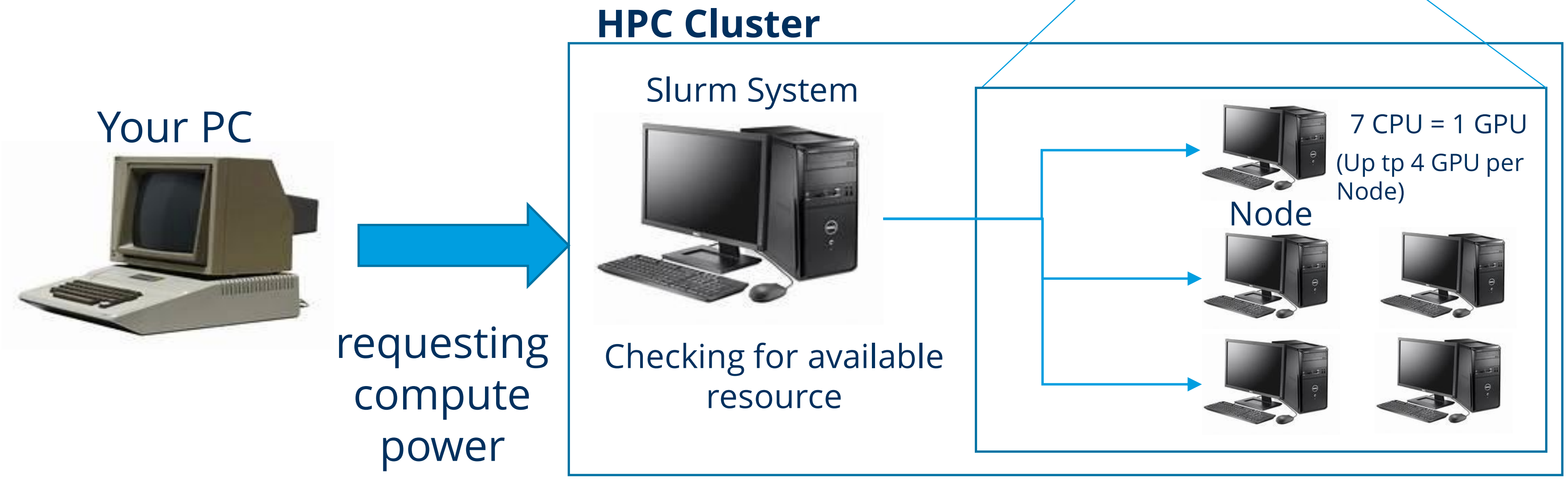
Thursday

Do it yourself! Hackday!

High Performance Computing

Through this course we will **need compute power** that could exceed the limitations of your PC/Laptop at home.
To run everything smoothly we are lucky to be supported by the TU Dresden Supercomputer **TAURUS**. This so called **H**ight **P**erformance **C**omputing (HPC) System is not accessible to public use and can only be used from the inside of TU Dresden Network





HPC Cluster

Your PC

requesting compute power

Slurm System

Checking for available resource

7 CPU = 1 GPU

(Up tp 4 GPU per Node)

Node

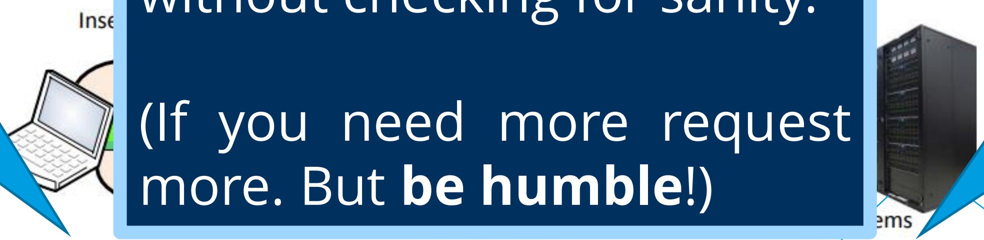
🟩 Free 🟥 Occupied

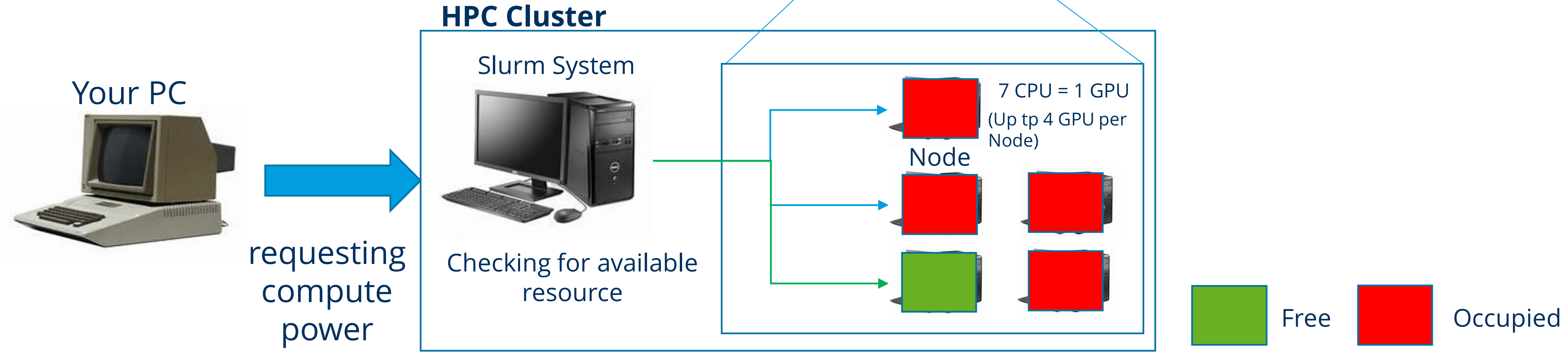
High Performance Computing

Through this co se we will nee mitations of yo PC/Laptop at home.
To run eve moothly we n Supercomp RUS. This so called **H**ight **P**erformance ting (HPC) an only be u the inside of TU Dresden Network

Slurm will allocate the resources requested by you without checking for sanity.

(If you need more request more. But **be humble**!)

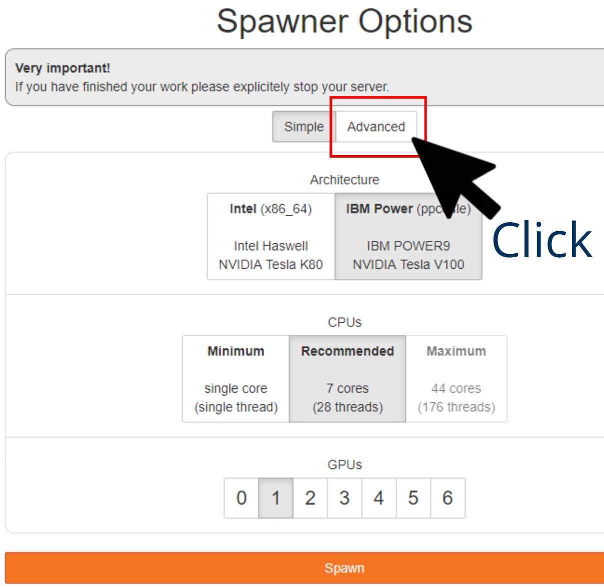




How to connect to HPC through the course

Make sure your you are connecting from the secure network **within** TU Dresden!
Go to your browser and visit: https://taurus.hrsk.tu-dresden.de/jupyter

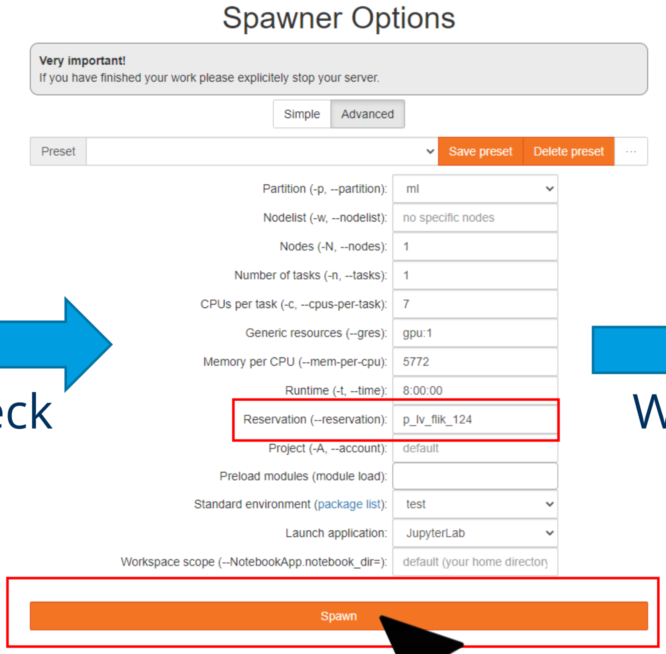
If you are within the TU network you can click on the button Start My Server to enter this screen



Spawner Options

Very important!
If you have finished your work please explicitely stop your server.

Check



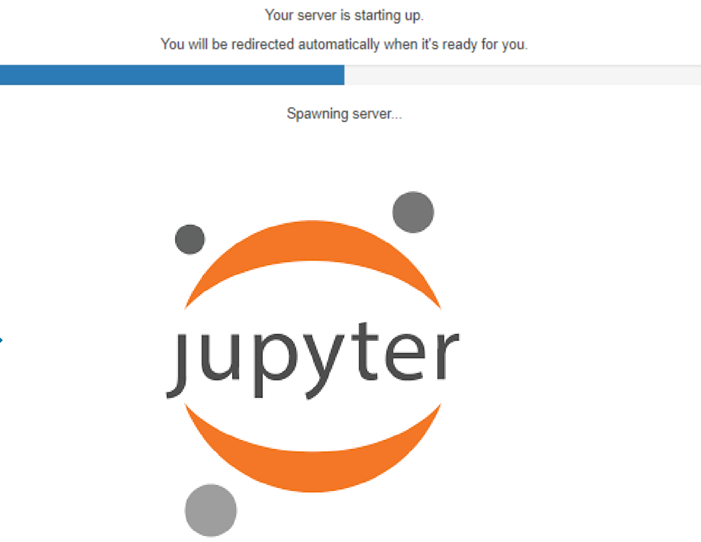
Spawner Options

Very important!
If you have finished your work please explicitely stop your server.

Partition (-p, --partition): ml
Nodelist (-w, --nodelist): no specific nodes
Nodes (-N, --nodes): 1
Number of tasks (-n, --tasks): 1
CPUs per task (-c, --cpus-per-task): 7
Generic resources (--gres): gpu:1
Memory per CPU (--mem-per-cpu): 5772
Runtime (-t, --time): 8:00:00
Reservation (--reservation): p_lv_flik_124
Project (-A, --account): default
Preload modules (module load):
Standard environment (package list): test
Launch application: JupyterLab
Workspace scope (--NotebookApp.notebook_dir=): default (your home directory

Click

Wait

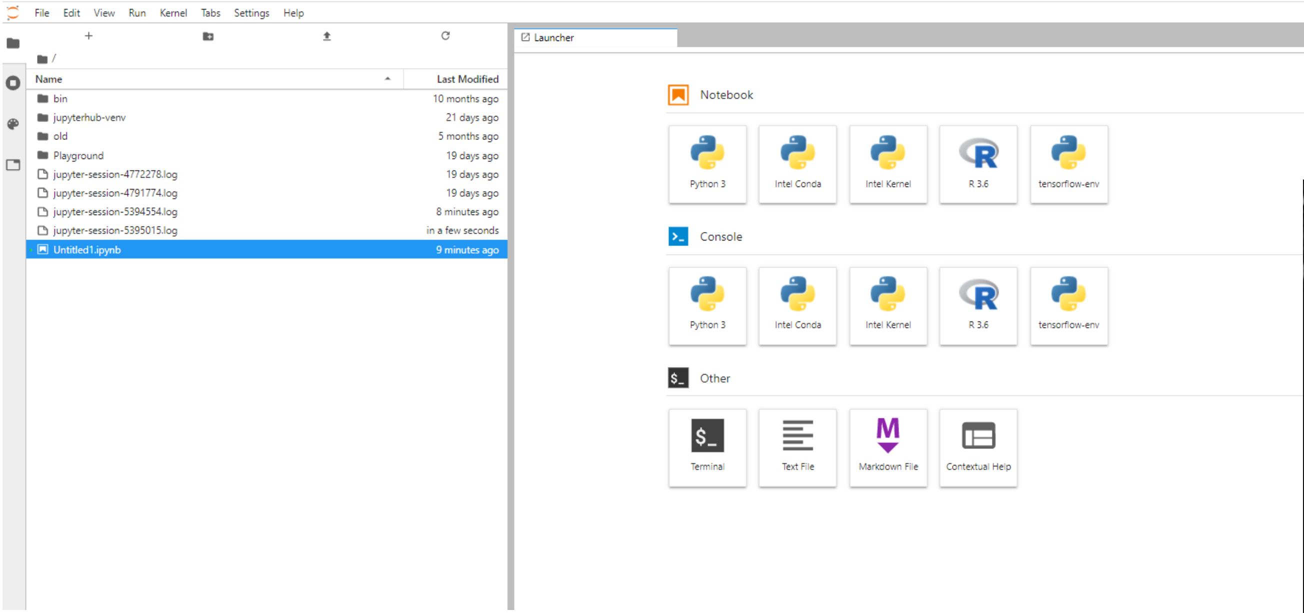


Your server is starting up.
You will be redirected automatically when it's ready for you.

Spawning server...

How to connect to HPC through the course

This is your „home“ screen for Jupyter on the HPC



Name	Last Modified
bin	10 months ago
jupyterhub-venv	21 days ago
old	5 months ago
Playground	19 days ago
jupyter-session-4772278.log	19 days ago
jupyter-session-4791774.log	19 days ago
jupyter-session-5394554.log	8 minutes ago
jupyter-session-5395015.log	in a few seconds
Untitled1.ipynb	9 minutes ago

Your screen will look a little different but don‘t worry!

Yep! ☺

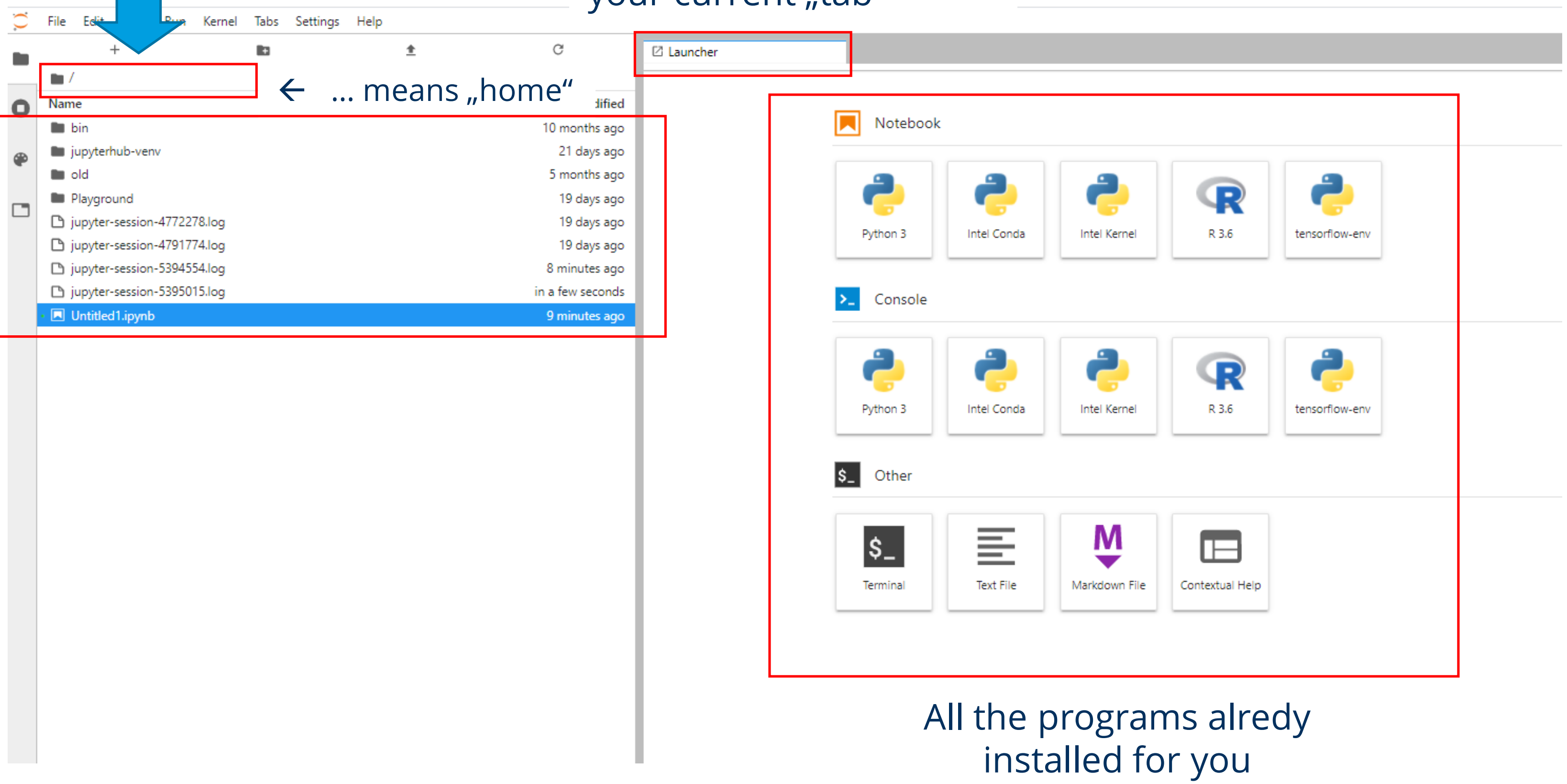
Jupyter Hub

your current folder name

your current „tab“

… means „home“

All your folders
(this will probably be empty)



Name	dified
bin	10 months ago
jupyterhub-venv	21 days ago
old	5 months ago
Playground	19 days ago
jupyter-session-4772278.log	19 days ago
jupyter-session-4791774.log	19 days ago
jupyter-session-5394554.log	8 minutes ago
jupyter-session-5395015.log	in a few seconds
Untitled1.ipynb	9 minutes ago

All the programs alredy installed for you

The Jupyter Hub is **GUI** on top of the Linux running on the HPC, which is accessible by the command line. (don‘t worry we will only touch this in a single case)

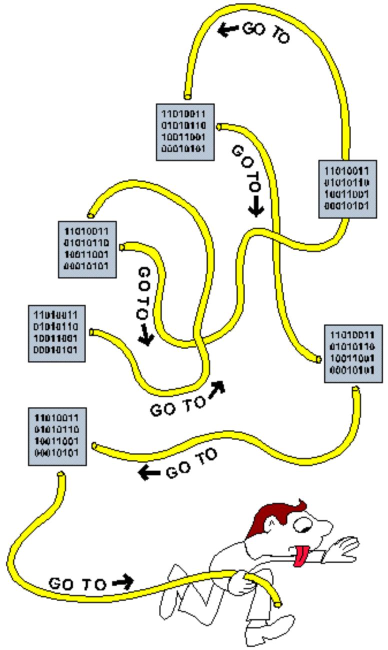
GitHub

We prepared your materials to be downloaded via **GitHub** (Microsoft)

Github is an online code sharing platform which is very popular in Computer Science (especially in ML) Using a „Git“ is a **very helpful tool** in programing alone and with colaborateurs because of his version controll. (Coding a giant project with many collegues can end in a **codemess**)

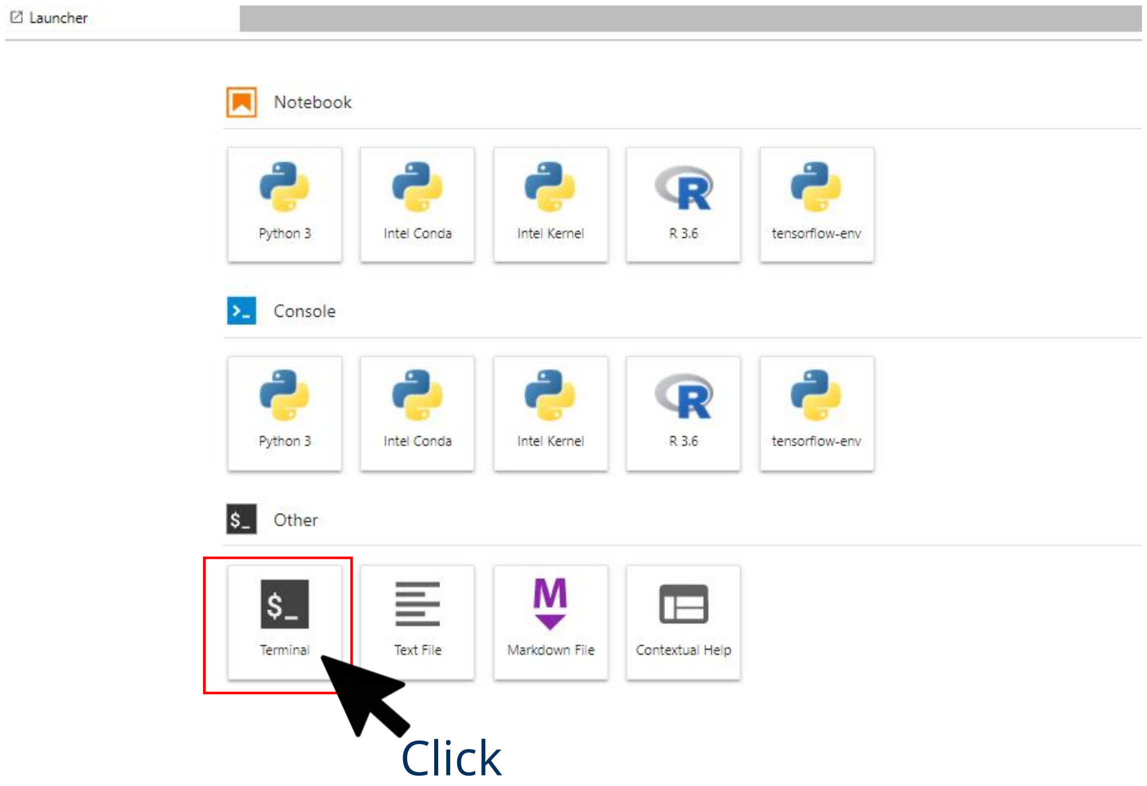
Unfortunatly we don‘t have the time to introduce you to git in this course. But **we encourage you to learn using it** from one of the many very good resources already out on the internet.

For the sake of simplicity we just show you how to download an existing project (our material) from the github server to your HPC local folder.



Git is **already installed** for you in the Taurus! ☺

How to get the course material with Git?



Click

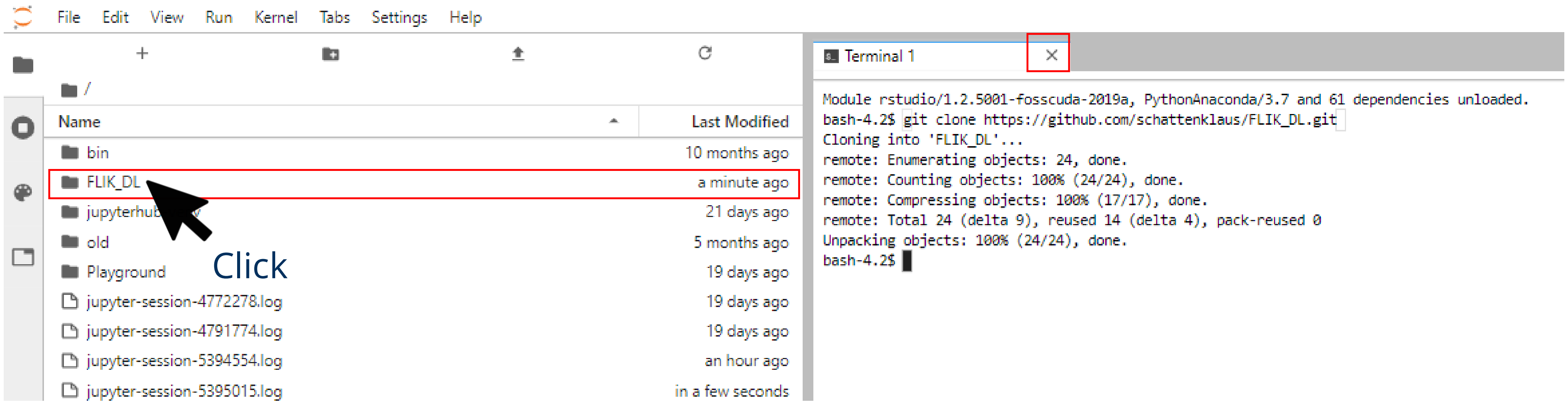
In the new terminal window write:

```
$ git clone https://github.com/schattenklaus/FLIK_DL.git
```

Press Enter

How to get the course material with Git?

You should see the **new „FLIK_DL“ folder** in your directory → Click on it to excess all the course materials (our slides and notebooks on a daily basis)



Name	Last Modified
bin	10 months ago
FLIK_DL	a minute ago
jupyterhub	21 days ago
old	5 months ago
Playground	19 days ago
jupyter-session-4772278.log	19 days ago
jupyter-session-4791774.log	19 days ago
jupyter-session-5394554.log	an hour ago
jupyter-session-5395015.log	in a few seconds

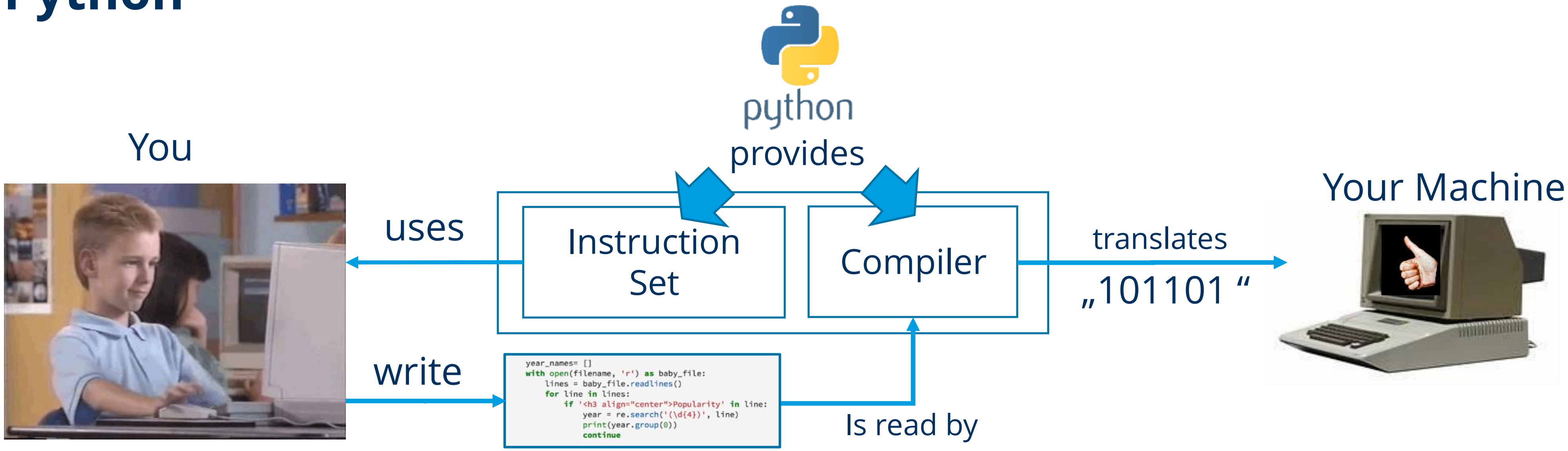
```
Module rstudio/1.2.5001-fosscuda-2019a, PythonAnaconda/3.7 and 61 dependencies unloaded.
bash-4.2$ git clone https://github.com/schattenklaus/FLIK_DL.git
Cloning into 'FLIK_DL'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 24 (delta 9), reused 14 (delta 4), pack-reused 0
Unpacking objects: 100% (24/24), done.
bash-4.2$
```

This is a **local copy** of our materials on your Taurus. If you unintentionally deleted something **just redo** the whole cloning process. You can now close the terminal window.

1. Exercise

Setup your HPC resources and download the course material to your home folder via GIT!

Python



You

uses

write

python

provides

Instruction Set

Compiler

Is read by

translates

„101101 “

Your Machine

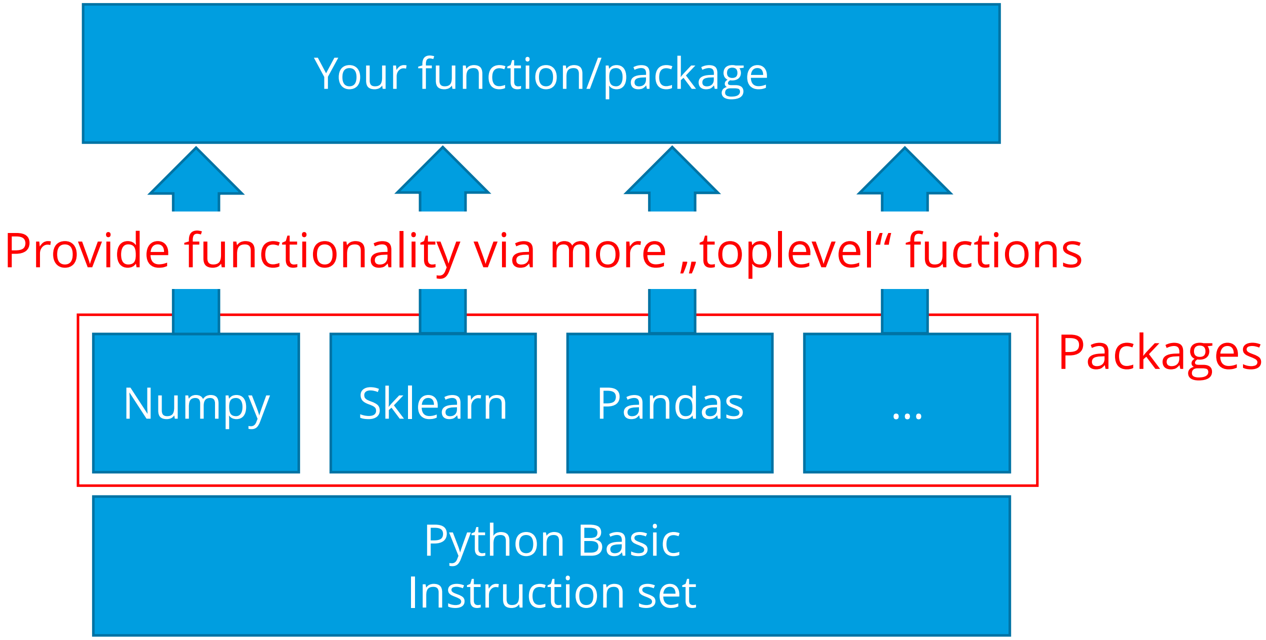
```
year_names= []
with open(filename, 'r') as baby_file:
    lines = baby_file.readlines()
    for line in lines:
        if '<h3 align="center">Popularity' in line:
            year = re.search('(\d{4})', line)
            print(year.group(0))
            continue
```

Very popular programming language in the scientific comunity!

Reason: **easy** understandable code, **free** of charge and **modular**

Python Modules

Packages/Modules are nothing else than **classes** with **functions** written by other users free of charge!



Your function/package

Provide functionality via more „toplevel“ fuctions

Numpy

Sklearn

Pandas

…

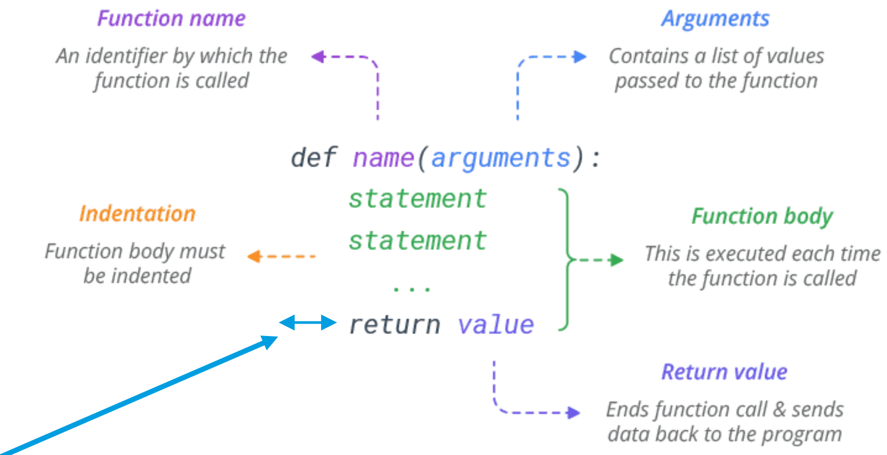
Packages

Python Basic
Instruction set

Python Code Explained

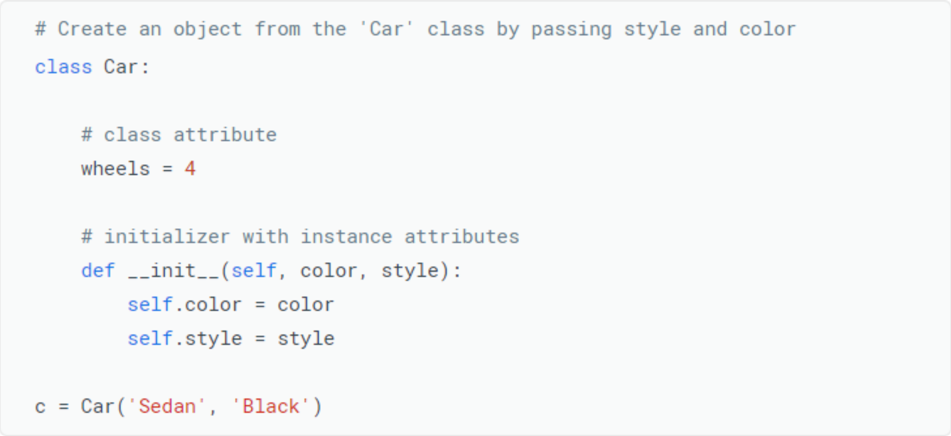
classes and functions

Class
Object
Learnbyexample.org


Function name
An identifier by which the function is called
Arguments
Contains a list of values passed to the function
Indentation
Function body must be indented
Function body
This is executed each time the function is called
Return value
Ends function call & sends data back to the program

```
def name(arguments):
    statement
    statement
       ...
    return value
```

Python **groups code** blocks by looking at the **intent**.

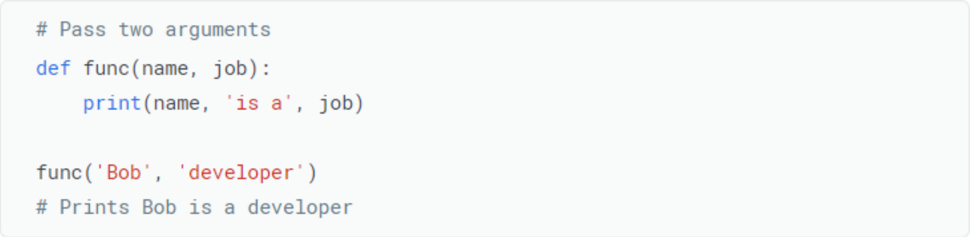


```
# Create an object from the 'Car' class by passing style and color
class Car:

    # class attribute
    wheels = 4

    # initializer with instance attributes
    def __init__(self, color, style):
        self.color = color
        self.style = style

c = Car('Sedan', 'Black')
```

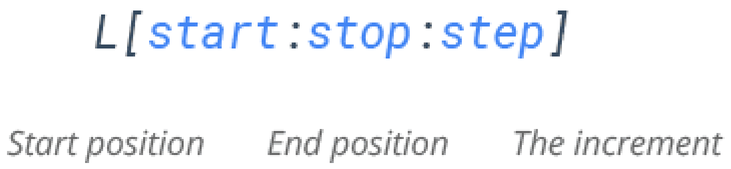


```
# Pass two arguments
def func(name, job):
    print(name, 'is a', job)

func('Bob', 'developer')
# Prints Bob is a developer
```

The **def** statement only creates a **function** but does not call it. After the def has run, you can can call (run) the function by adding parentheses after the function’s name.

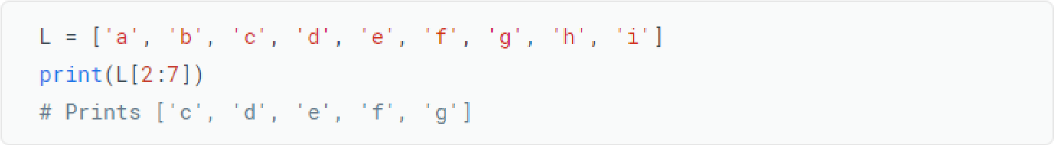
Slicing



```
L[start:stop:step]
```

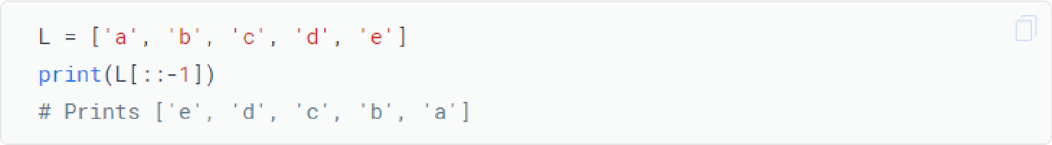
Start position *End position* *The increment*

To access a **range** of items in a list or array, you need to **slice** a list. One way to do this is to use the simple slicing operator "**:**"



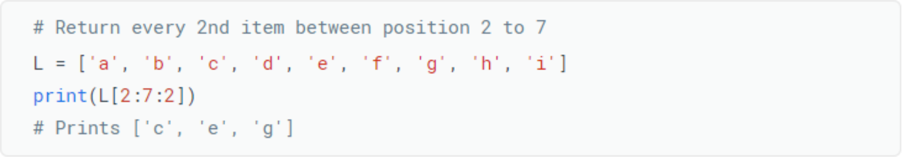
```
L = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
print(L[2:7])
# Prints ['c', 'd', 'e', 'f', 'g']
```

Reverse a list



```
L = ['a', 'b', 'c', 'd', 'e']
print(L[::-1])
# Prints ['e', 'd', 'c', 'b', 'a']
```

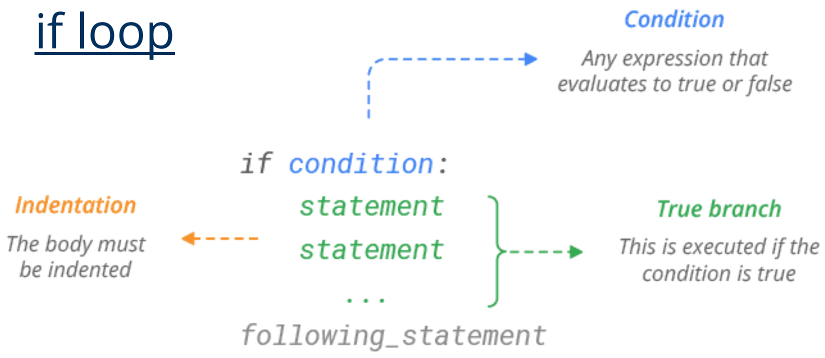
Only every 2nd value



```
# Return every 2nd item between position 2 to 7
L = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
print(L[2:7:2])
# Prints ['c', 'e', 'g']
```

Python Code Explained

if loop



Condition

Any expression that evaluates to true or false

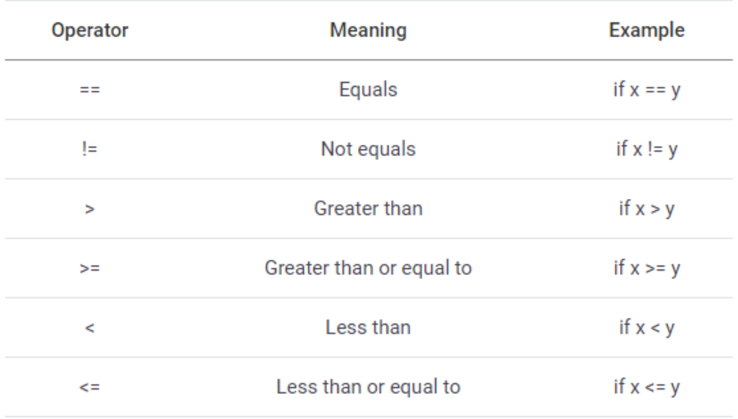
```
if condition:
    statement
    statement
    ...
following_statement
```

Indentation

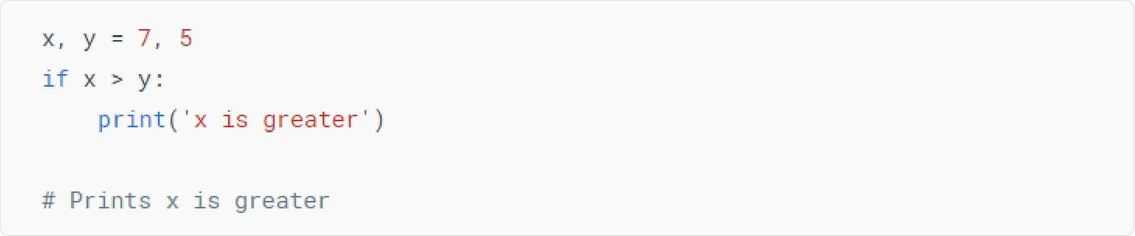
The body must be indented

True branch

This is executed if the condition is true



Operator	Meaning	Example
==	Equals	if x == y
!=	Not equals	if x != y
>	Greater than	if x > y
>=	Greater than or equal to	if x >= y
<	Less than	if x < y
<=	Less than or equal to	if x <= y



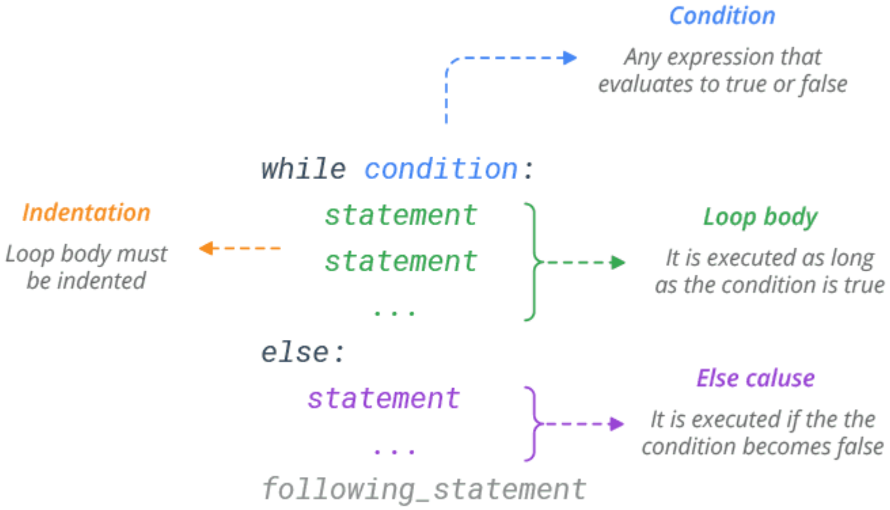
```
x, y = 7, 5
if x > y:
    print('x is greater')

# Prints x is greater
```

Python Code Explained

while loop

Any **non-zero** value or **nonempty** container is considered **TRUE**; whereas Zero, None, and empty container is considered **FALSE**.



```
while condition:
    statement
    statement
    ...
else:
    statement
    ...
following_statement
```

Condition

Any expression that evaluates to true or false

Indentation

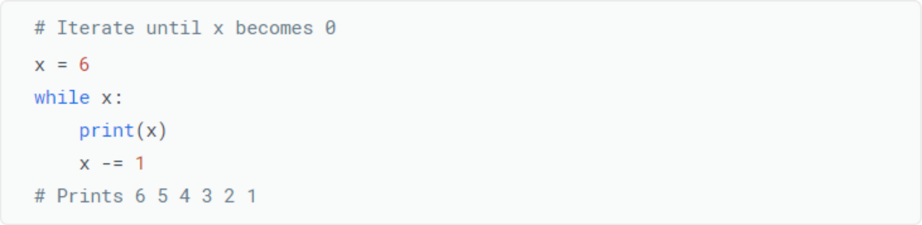
Loop body must be indented

Loop body

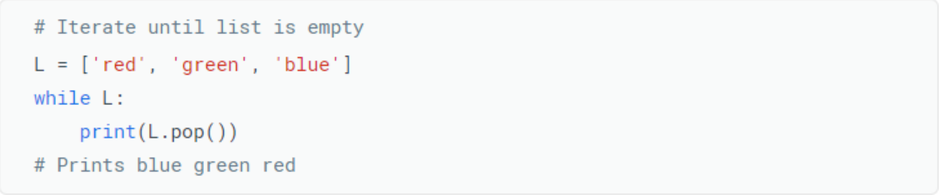
It is executed as long as the condition is true

Else caluse

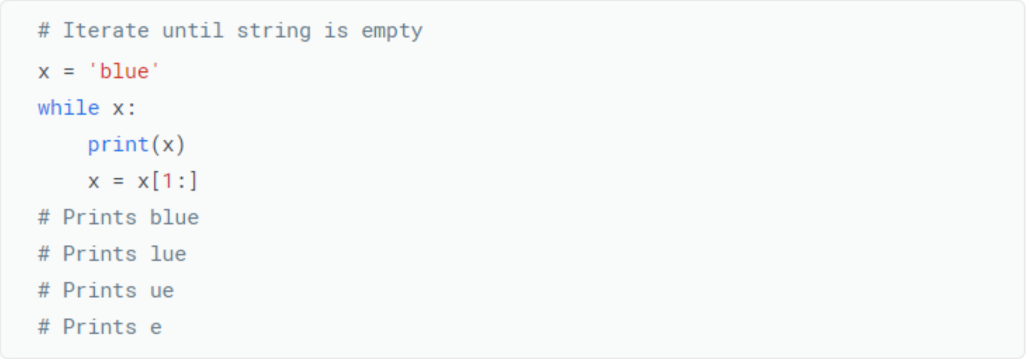
It is executed if the the condition becomes false



```
# Iterate until x becomes 0
x = 6
while x:
    print(x)
    x -= 1
# Prints 6 5 4 3 2 1
```



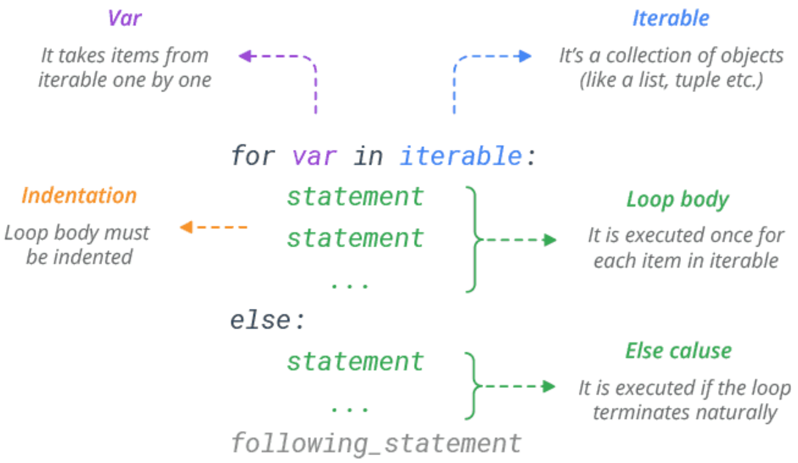
```
# Iterate until list is empty
L = ['red', 'green', 'blue']
while L:
    print(L.pop())
# Prints blue green red
```



```
# Iterate until string is empty
x = 'blue'
while x:
    print(x)
    x = x[1:]
# Prints blue
# Prints lue
# Prints ue
# Prints e
```

Python Code Explained

for loop



Var

It takes items from iterable one by one

Iterable

It's a collection of objects (like a list, tuple etc.)

```
for var in iterable:
    statement
    statement
    ...
else:
    statement
    ...
following_statement
```

Indentation

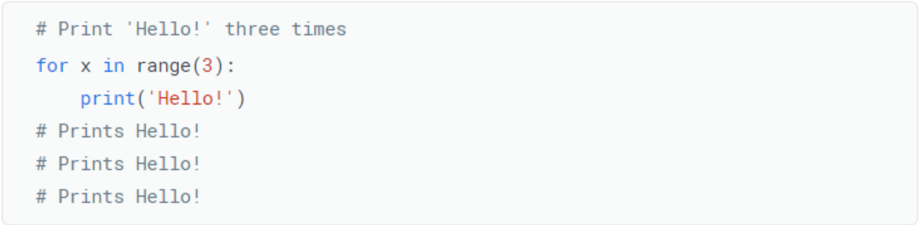
Loop body must be indented

Loop body

It is executed once for each item in iterable

Else caluse

It is executed if the loop terminates naturally



```
# Print 'Hello!' three times
for x in range(3):
    print('Hello!')
# Prints Hello!
# Prints Hello!
# Prints Hello!
```

2. Exercise

Open the notebook „Python_Intro“ and follow the instructions