

FliK Modul 2020

Deep Learning mit Keras

Steffen Seitz & Marvin Arnold

Prof. Ronald Tetzlaff

Dresden, 19-23.10.

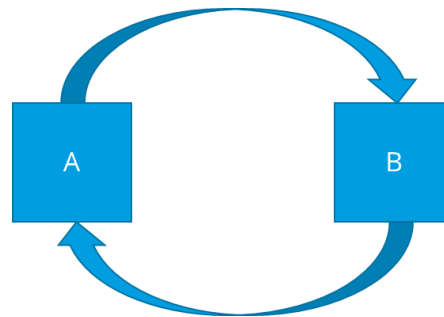
Disclaimer

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 in 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 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

1. Introduction
 - 1.1 HPC and GIT
 - 1.2 Jupyter Notebooks
- 1. Exercise ---
2. Python Basics
- 2. Exercise: Python Basics ---
3. Neural Networks with Keras
 - 3.1 Keras Basics

Tuesday

1. Convolutional Neuronal Networks
 - 1.1 HPC & Setup
- 1. Exercise: Setup ---

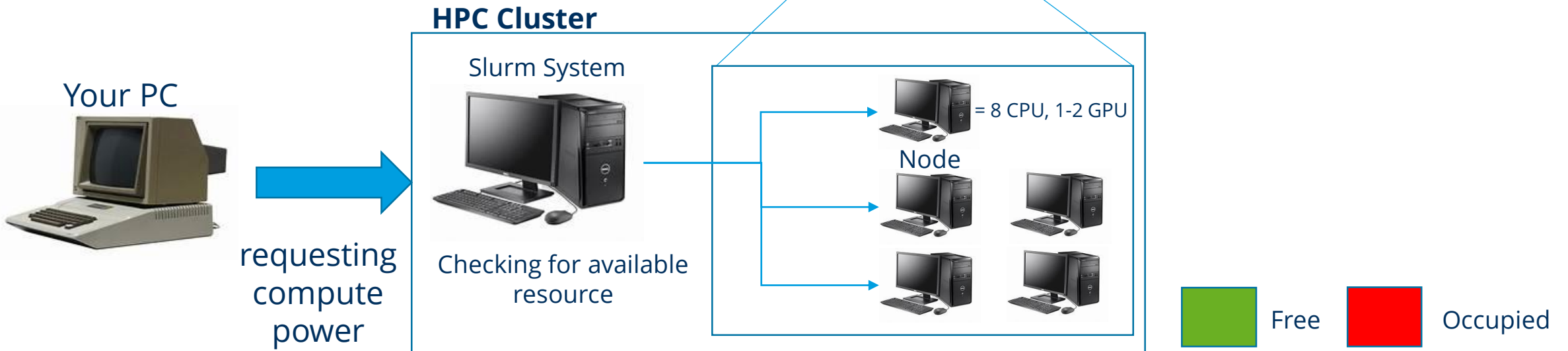
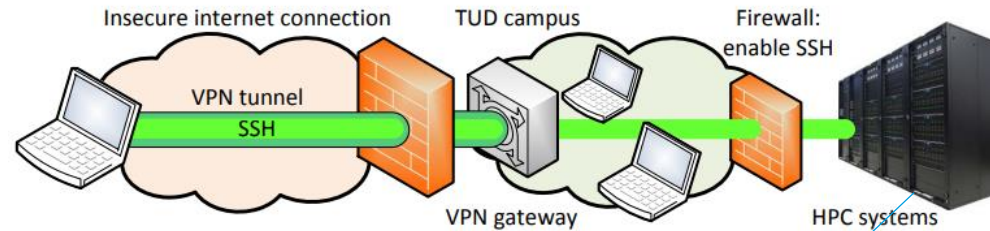
Wednesday

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 **Hight Performance Computing (HPC) System** is not accessible to public use and can only be used from the inside of TU Dresden Network



High Performance Computing

Through this course we will need
To run everything smoothly we
Performance Computing (HPC)
Network

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

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

limitations of your PC/Laptop at home.
in Supercomputing **URUS**. This so called **Hight**
can only be used in the inside of TU Dresden



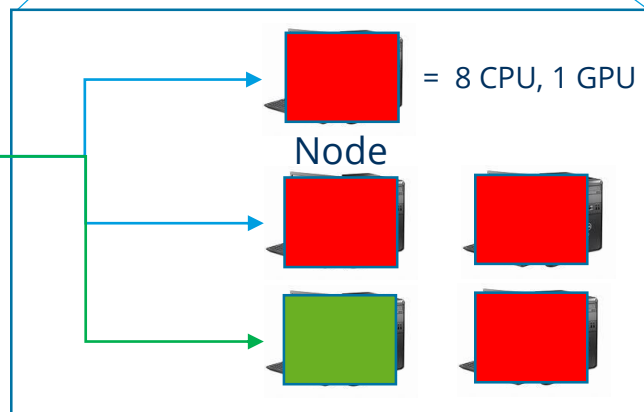
requesting
compute
power

HPC Cluster

Slurm System



Checking for available
resource



Free



Occupied

How to connect to HPC through the course

Make sure 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  to enter this screen

Spawner Options

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

Simple **Advanced**

Architecture

Intel (x86_64)	IBM Power (ppc64le)
Intel Haswell NVIDIA Tesla K80	IBM POWER9 NVIDIA Tesla V100

Click

CPU

Minimum	Recommended	Maximum
single core (single thread)	28 cores (112 threads)	44 cores (176 threads)

GPU

0	1	2	3	4	5	6
---	---	---	---	---	---	---

Spawn

Check

Spawner Options

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

Simple **Advanced**

Preset Save preset Delete preset

Partition (-p, --partition): ml

Node list (-w, --nodelist): no specific nodes

Nodes (-N, --nodes): 1

Number of tasks (-n, --tasks): 1

CPU per task (-c, --cpu-per-task): 8

Generic resources (--gres): gpu:1

Memory per CPU (--mem-per-cpu): 5772

Runtime (-t, --time): 8:00:00

Reservation (--reservation): „Reservation Code Name“

Project (-A, --account): default

Preload modules (module load):

Standard environment (package list): production

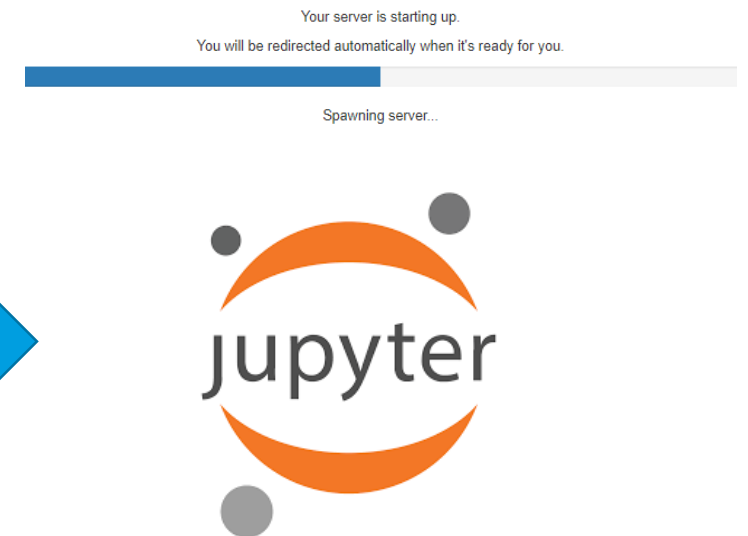
Launch application: JupyterLab

Workspace scope (--NotebookApp.notebook_dir): default (your home director

Spawn

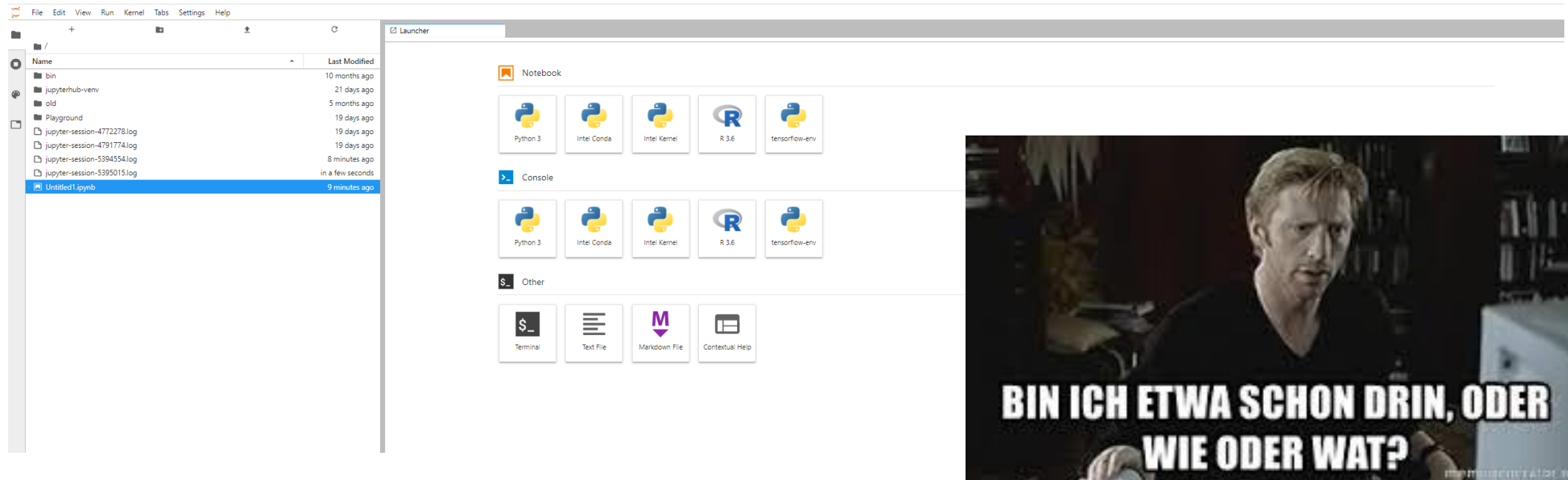
Click

Wait



How to connect to HPC through the course

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



Your screen will look a little different but don't worry!

Yep! 😊

Jupyter Hub

All your folders
(this will probably be empty)



your current folder name

your current „tab“

← ... means „home“

Launcher

Notebook

Python 3 Intel Conda Intel Kernel R 3.6 tensorflow-env

Console

Python 3 Intel Conda Intel Kernel R 3.6 tensorflow-env

Other

Terminal Text File Markdown File Contextual Help

All the programs already 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



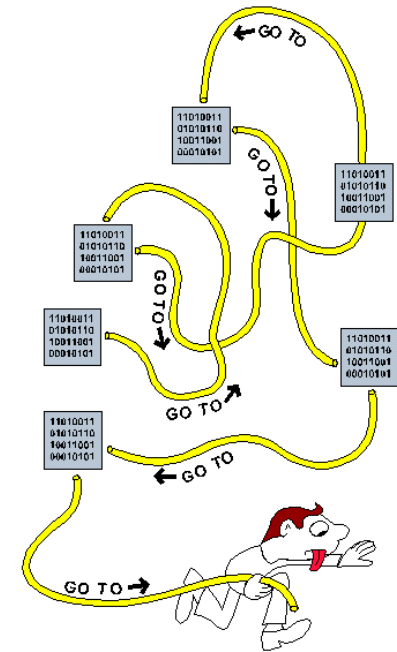
We did prepare your materials to be downloaded via **GitHub** (Microsoft)

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 programming alone and with collaborators because of its version control. (Coding a giant project with many colleagues can end in a **codemess**)

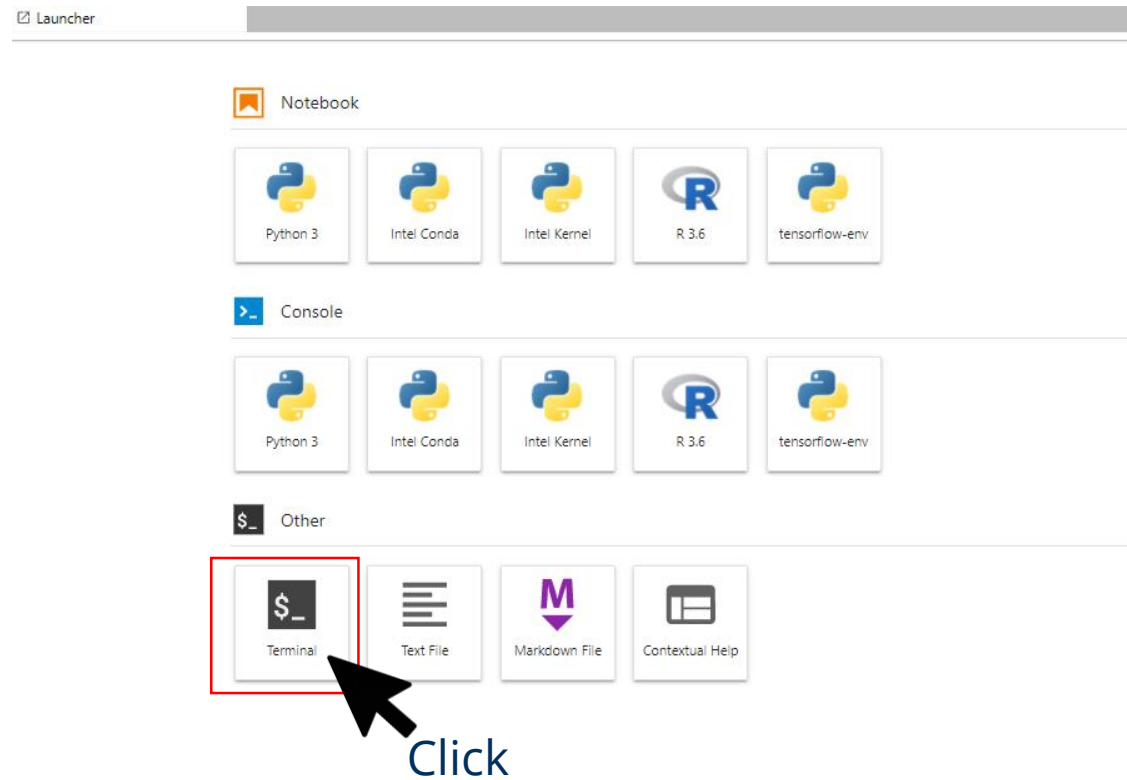
Unfortunately we don't have the time to introduce you to git in this course. But **we encourage you to learn using it** from one of the many very good resources already out in the internet.

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! 😊

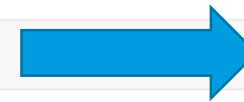


How to material with Git?



In the new terminal window write:

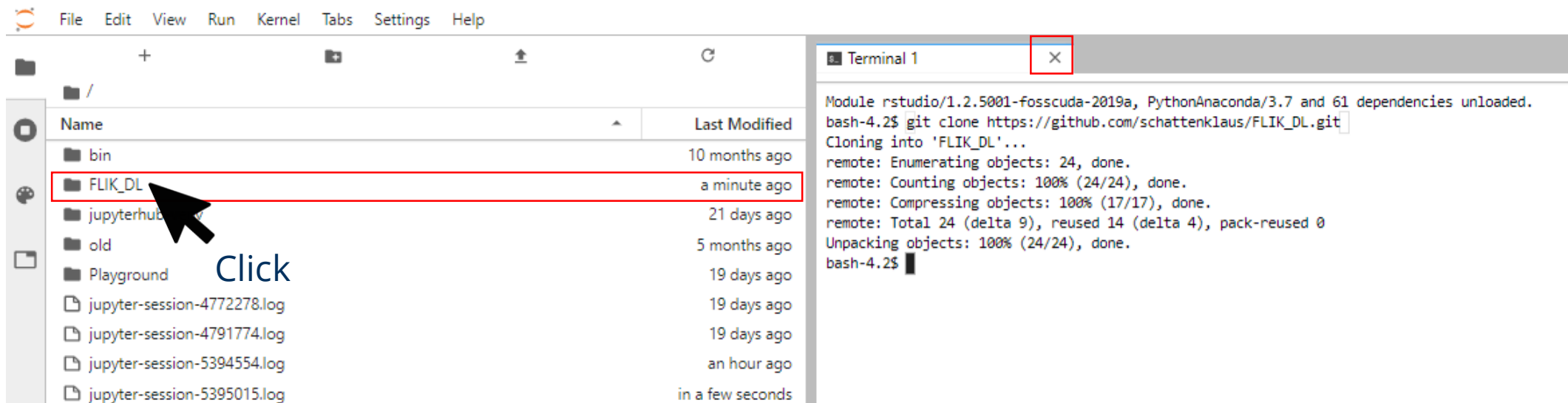
```
$ git clone https://github.com/schattenklaus/FLIK_DL.git
```



Press Enter

How to 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)

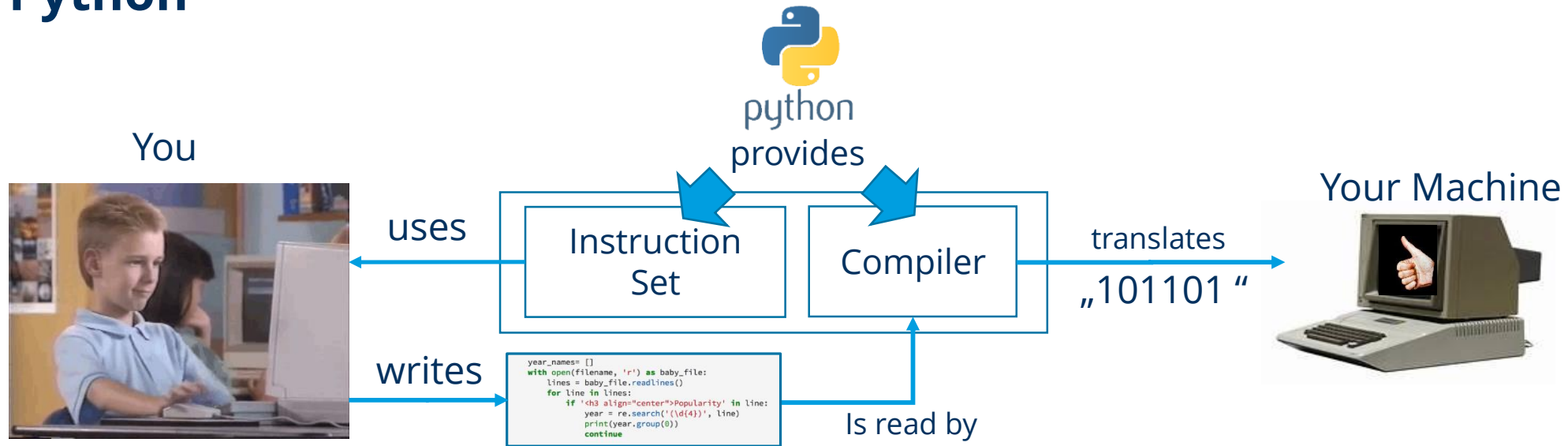


This is a **local copy** of our materials on your Taurus. If you unintentionally deleted something **just redo** the whole cloning process. You can now close the terminal window.

1. Exercise

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

Python

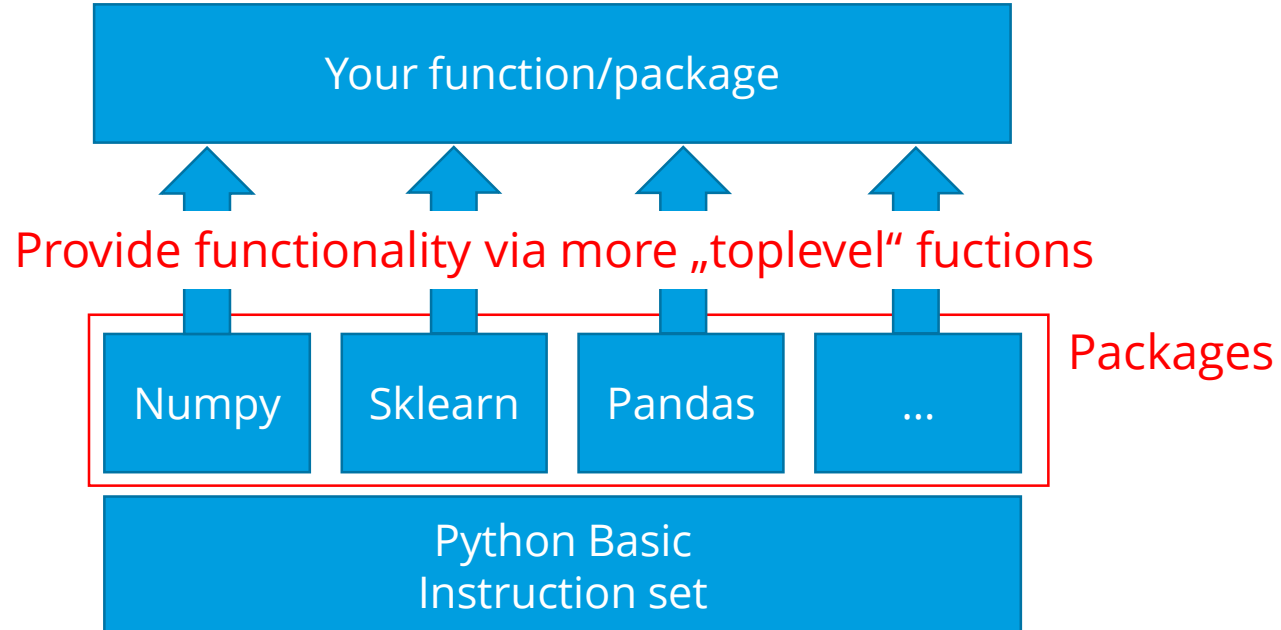


Very popular programming language in the scientific community!

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

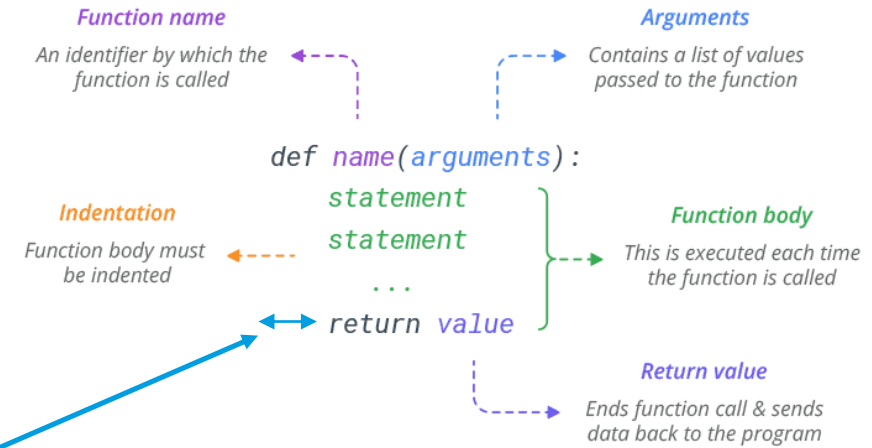
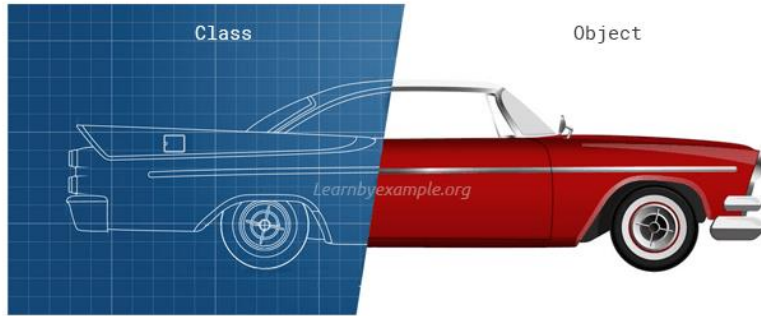
Python Modules

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



Python Code Explained

classes and functions



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 call (run) the function by adding parentheses after the function's name.

Slicing

$L[\textit{start}:\textit{stop}:\textit{step}]$

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

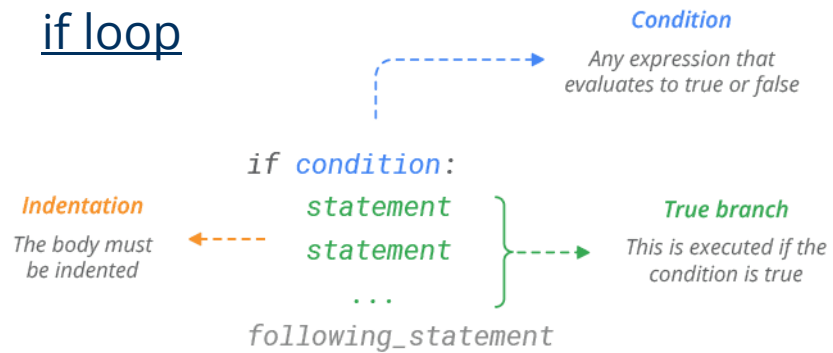
```
L = ['a', 'b', 'c', 'd', 'e']
print(L[::-1])
# Prints ['e', 'd', 'c', 'b', 'a']
```

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



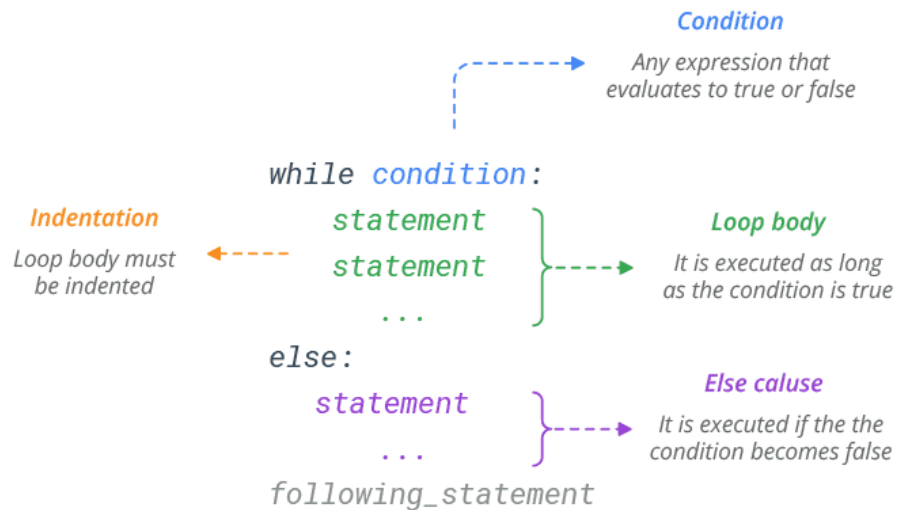
```
x, y = 7, 5  
if x > y:  
    print('x is greater')  
  
# Prints x is greater
```

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

Python Code Explained

while loop

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



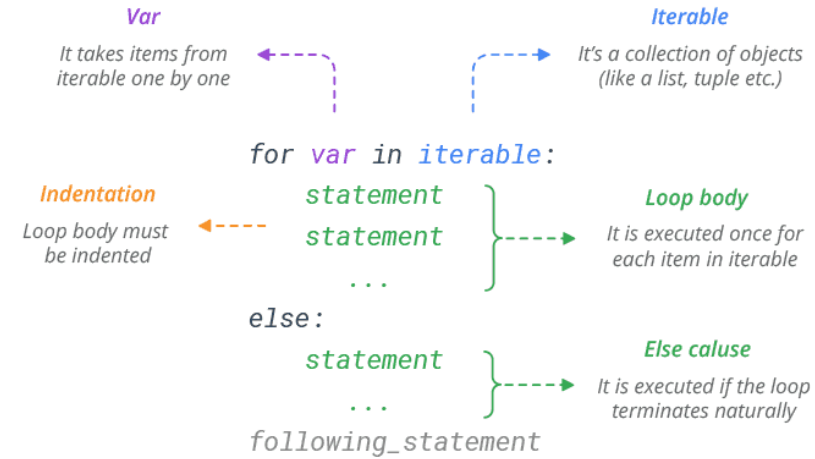
```
# Iterate until x becomes 0
x = 6
while x:
    print(x)
    x -= 1
# Prints 6 5 4 3 2 1
```

```
# 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



```
# 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