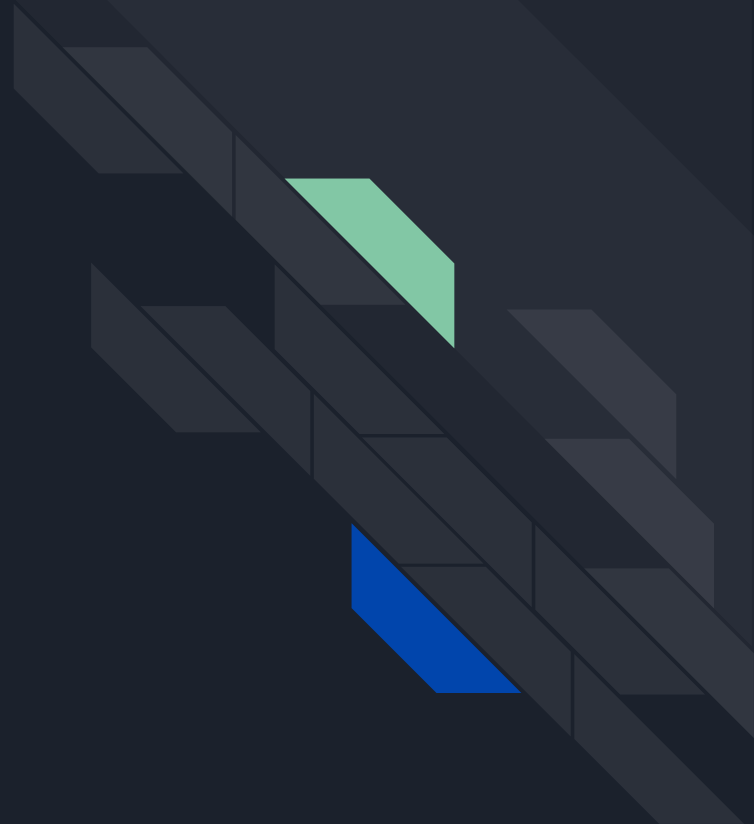
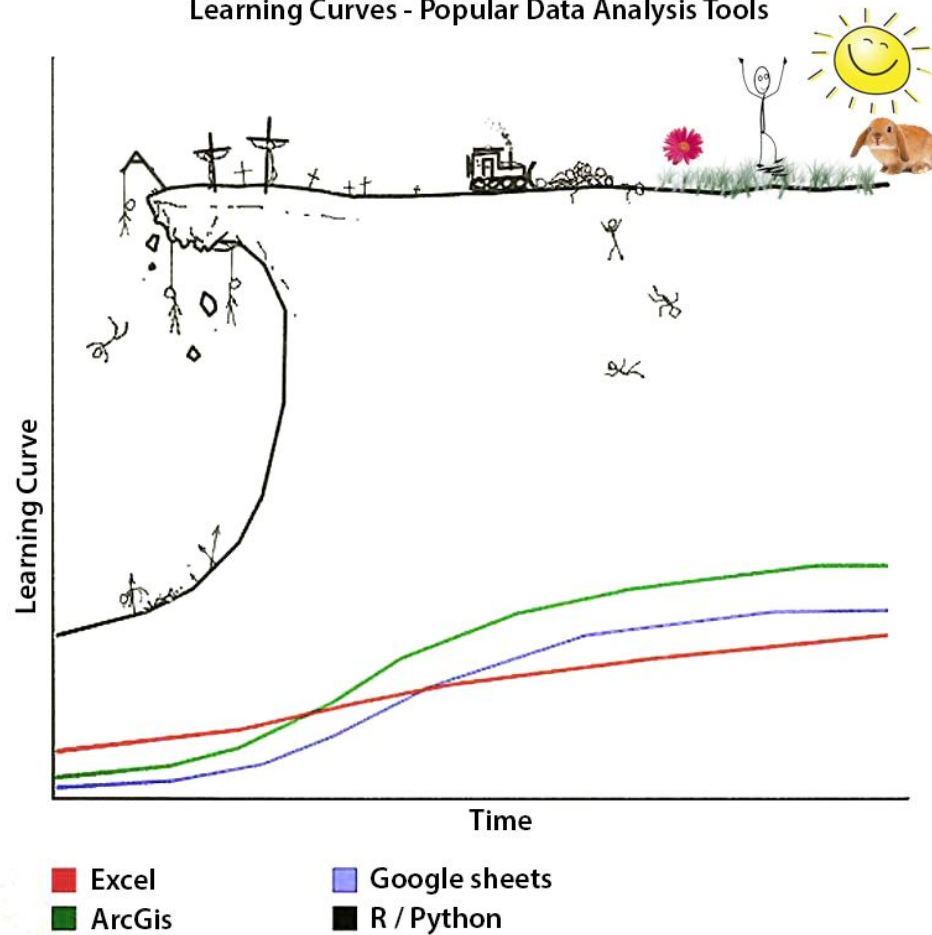


Introduction to Data Science with Python

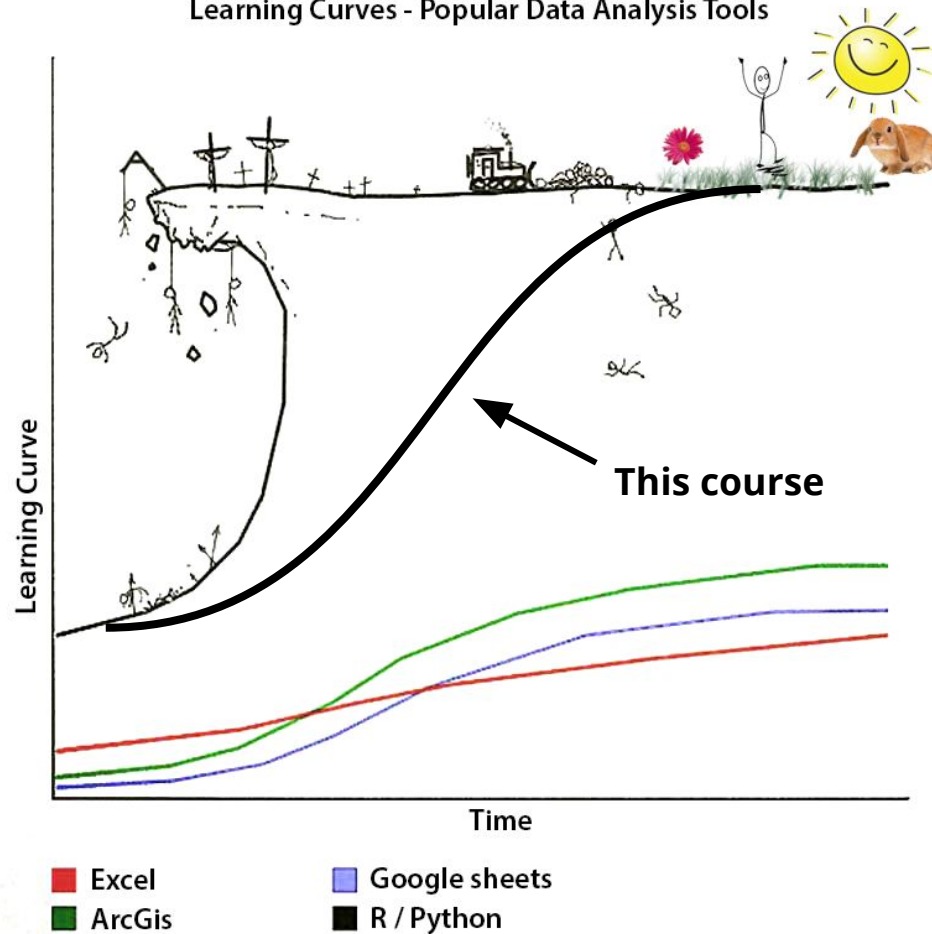
Lecture 1: December 6, 2022



Learning Curves - Popular Data Analysis Tools



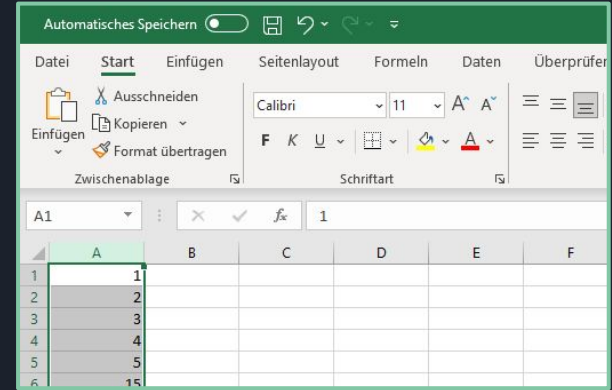
Learning Curves - Popular Data Analysis Tools



Excel vs Python



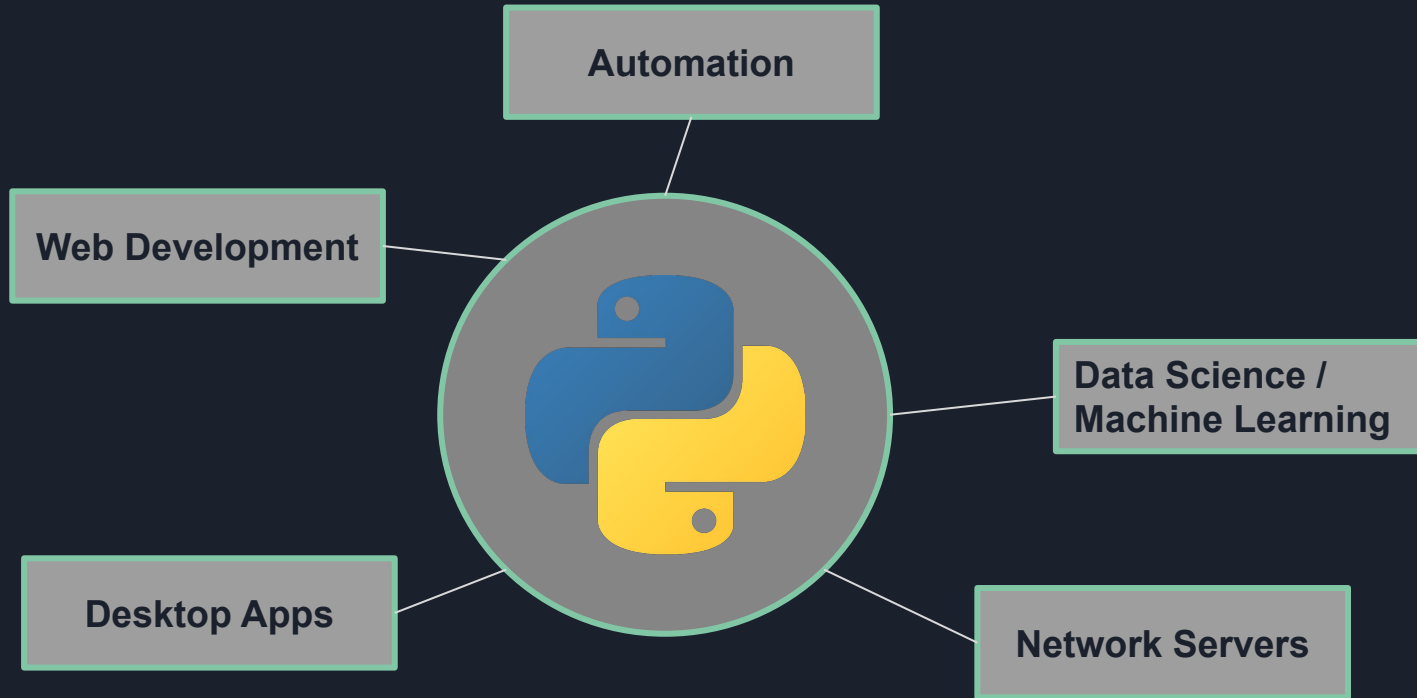
1. Select data
2. Click on buttons :)



1. Write code in text editor
2. Execute code with Python
3. Result will be returned

```
data = pd.read_csv(file)
mean = data.mean()
print(mean)
```

A General Purpose Coding Language



Structure of the course

Lectures (3 times)

- Introduce new concepts on slides
- Your turn! - Small exercises (~5 minutes)
- Live coding in Google Colab
- Slides and notebooks will be uploaded



Homework (3 times)

- each week a set of exercises about the topics from the lecture (~ 1h)
- solutions will be provided, no hand-in



Questions?

felix.schneider@tech-academy.io

Topics of the course

1.Lecture

2.Lecture

3.Lecture

Python Fundamentals

Basic concepts,
Variables, basic data
structures, functions

Data Wrangling & Simple visualizations

How to process data with
pandas and visualize it
with matplotlib

Visualizations & Modelling

More plots with matplotlib
and seaborn and an
introduction to modelling



Google Colab

- Environment to write and execute code
- Accessed via Browser (runs on Google Servers)
- No pre-configurations necessary
- Independent from your local machine
- Jupyter Notebook format heavily used in data science community



colab

Fundamentals - Data Types



How is data stored and processed ?

- Values are stored in **variables**
- The four most important data types in Python:

```
integer = 10  
float = 2.8  
string = "This is a string"  
boolean = True
```



How is data stored and processed ?

- We can compute with these variables



```
a = 10  
b = 5  
c = a + b  
print(c)
```



Output: 15



QUIZ

What kind of data type is this: "27-03-2021" ?

- a) integer b) float c) string d) date



QUIZ

What kind of data type is this: "27-03-2021" ?

- a) integer b) float c) string d) date

Fundamentals - Data Structures



Data Structures - Lists

We can combine values in lists

```
a = [5, 3, 9, 7, 4, 10, 3]  
b = ["Justus", "Peter", "Bob"]
```





Data Structures - Lists

| | | | | | | | |
|--------------|----------|----------|----------|----------|----------|-----------|----------|
| Value | 5 | 3 | 9 | 7 | 4 | 10 | 3 |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

Data Structures - Lists

Access the data with an index



```
a = [5, 3, 9, 7, 4, 10, 3]
b = ["Justus", "Peter", "Bob"]
a[0] → 5
```

Data Structures - Lists

Access the data with an index



```
a = [5, 3, 9, 7, 4, 10, 3]
b = ["Justus", "Peter", "Bob"]
a[0] → 5
b[1] → "Peter"
```



Data Structures - Lists



```
a[start:stop:step_size]
```

Data Structures - Lists

| | | | | | | | |
|-------|---|---|---|---|---|----|---|
| Value | 5 | 3 | 9 | 7 | 4 | 10 | 3 |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

`a[1:4]`



Data Structures - Lists

| | | | | | | | |
|-------|---|---|---|---|---|----|---|
| Value | 5 | 3 | 9 | 7 | 4 | 10 | 3 |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

```
a[1:4:2]
```



Data Structures - Dictionaries



```
translate = {"Eins": "One",  
             "Zwei": "Two",  
             "Ja": "Yes"}  
translate["Eins"] → "One"
```



Quick - Summary

Data types

integer 2

float 2.32

string "Text"

boolean True/False

Data structures

lists: a = [1, 2, 3]

dictionary: b={ "a": 1 }

Exercise 1

Data structures - Hints

lists:

create: `a = [1, 2, 3]`

access: `a[0]`

dictionary:

create: `b={"a":1}`

access: `b["a"]`



Relational operators

- Compare variables

`a == b` → is a equal to b?

returns **True / False**

- Combine operators with “and” / “or”

“and”: `(a >= b) & (a <= c)`

“or”: `(a >= b) | (a <= c)`

| | |
|--------------------|-----------------------|
| <code>==</code> | is equal |
| <code><</code> | smaller than |
| <code>></code> | greater than |
| <code><=</code> | smaller or equal than |
| <code>>=</code> | greater or equal than |
| <code>!=</code> | not equal to |

QUIZ



```
a = 1
```

```
b = 2
```

```
c = 2
```

```
(a>b) or (a<=c)
```

QUIZ



```
a = 1
```

```
b = 2
```

```
c = 2
```

```
(a>b) or (a<=c)
```

```
False or True
```

QUIZ



```
a = 1
```

```
b = 2
```

```
c = 2
```

```
(a > b) or (a <= c)
```

```
False or True → True
```



Very important for filtering

| Name | Gender | Age |
|--------|--------|-----|
| "Tim" | "M" | 20 |
| "Nina" | "F" | 24 |
| "John" | "M" | 26 |

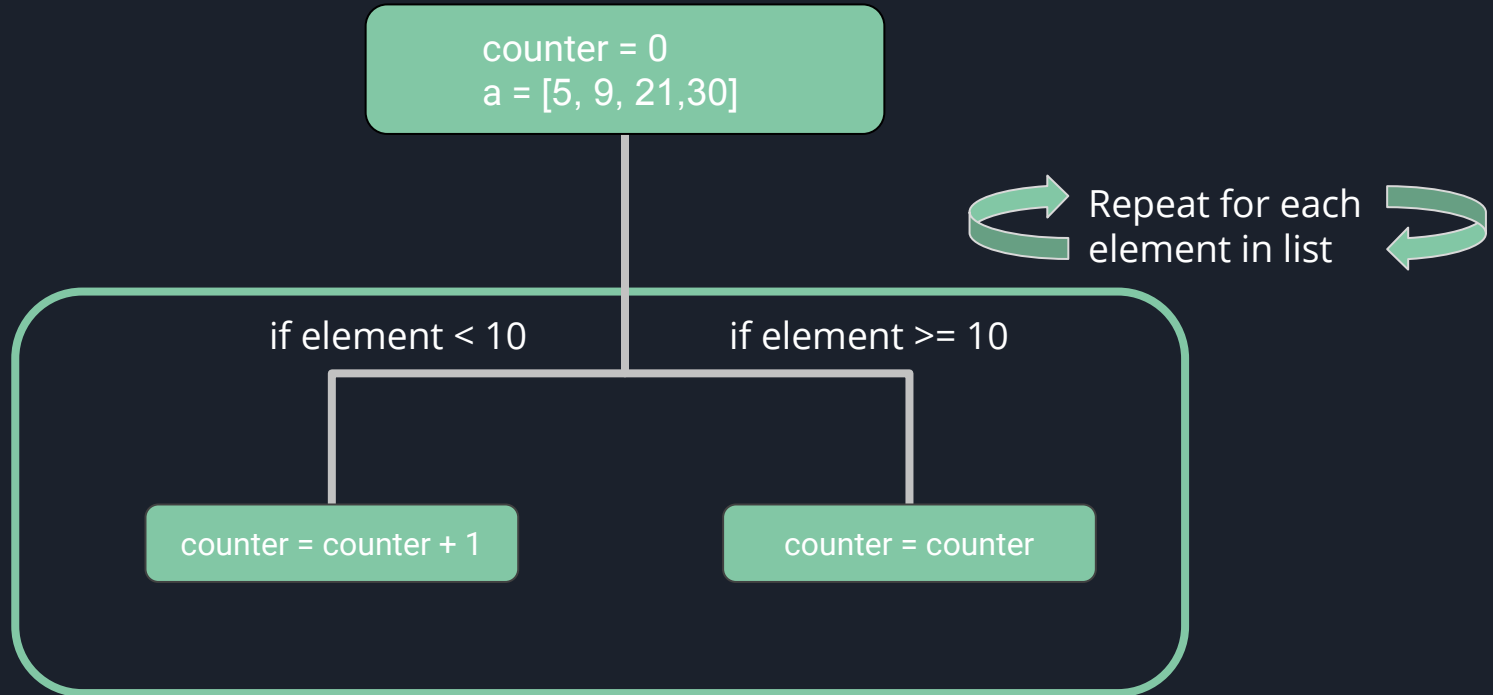
Select all Names with following condition:

(Gender == "F") & (Age > 20)

Fundamentals - Control Flow



Count numbers smaller than 10 in a list





Control Flow - if / else

- Control which block of code will be executed
- Blocks defined by indentation

```
if BOOLEAN-CONDITION:  
    print("A")  
else:  
    print("B")
```





Control Flow - if / else

- Control which block of code will be executed
- Blocks defined by indentation

```
if a>2:  
    print("A")  
else:  
    print("B")
```





Control Flow - for-loop

- Repeat blocks of your code
- Use different values in each loop

```
for element in [1,2,3,4]:  
    print(element)
```





Exercise 2

Count amount of numbers in a list which are smaller than 5

```
for element in [1,2,3,4]:  
    print(element)
```

```
if a>2:  
    print("A")  
else:  
    print("B")
```

Fundamentals - Functions & Libraries

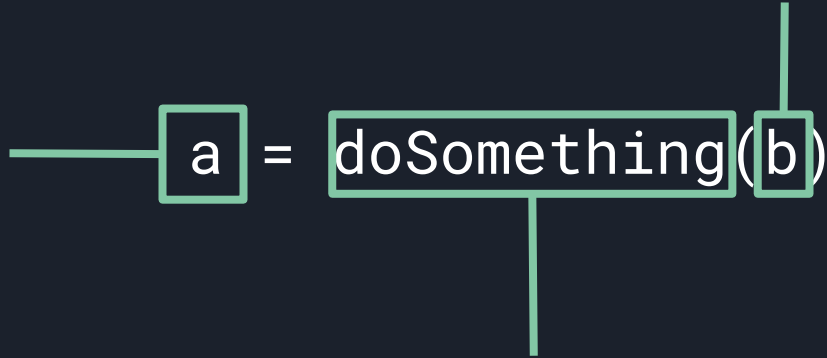




functions

Parameter of function
(can be a value, variable, list, dict,...)

Variable for
return value



Name of function



functions - round function



```
b = 5.2
```

```
a = round(b)
```

```
→ a = 5.0
```



Built-in Functions



```
print()    sum()  
round()    abs()  
min()      range()  
max()      sorted()
```

QUIZ



`round(2.34) == ?`

`abs(-2) == ?`

`a = [0, 4, 1, 3, 2]`

`max(a) == ?`

`sum(a) == ?`

`len(a) == ?`

`sorted(a) == ?`

QUIZ



```
round(2.34) == 2.0
```

```
abs(-2) == 2
```

```
a = [0, 4, 1, 3, 2]
```

```
max(a) == 4
```

```
sum(a) == 10
```


```
len(a) == 5
```

```
sorted(a) == [0, 1, 2, 3, 4]
```



Create own functions

- Define own functions for repeating tasks
- reduce amount of code lines




```
def my_function(a,b):  
    c = ...  
    return c
```



Create own functions

- Define own functions for repeating tasks
- reduce amount of code lines



```
def my_function(a,b):  
    return a + b  
my_function(1,2) # 3
```



Exercise 3

Convert your code which counts amount of numbers smaller 10 into a function



```
def smaller_than(numbers, value):  
    # your code here  
    return counter
```

Libraries

- A collection of functions is bundled in a **library**
- we import these libraries and can use the defined functions
- Some libraries come with a Python installation, some need to be installed

matplotlib



... for plotting and visualization

pandas

... for working with tabular data (Excel-files, csv-files,...)



... creating machine learning models