esade

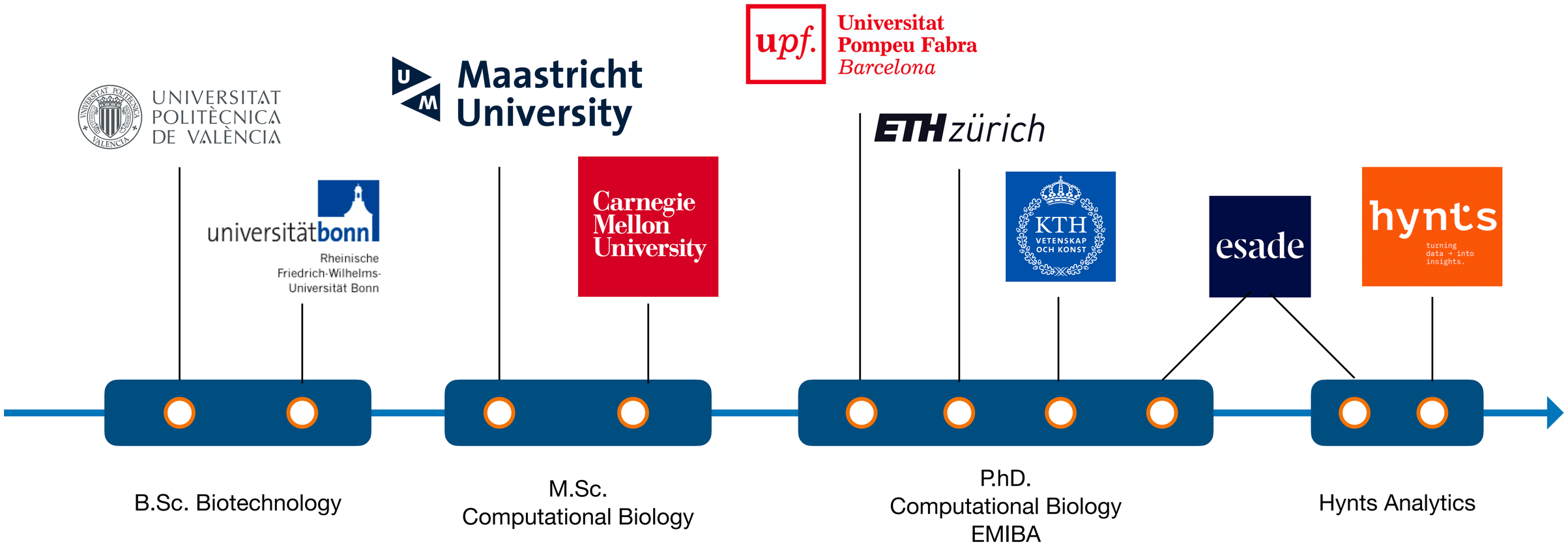
# Introduction to Python

Jose Bonet Giner

21st May 2022

# Index

- Topic 1: What is python and why is it important?
- Topic 2: Python Basics
- Topic 3: Pandas
- Topic 4: Numpy
- Topic 5: What’s next?


UNIVERSITAT POLITÈCNICA DE VALÈNCIA
universitätbonn
Rheinische Friedrich-Wilhelms-Universität Bonn
Maastricht University
Carnegie Mellon University
upf. Universitat Pompeu Fabra Barcelona
ETH zürich
KTH VETENSKAP OCH KONST
esade
hynts
turning data → into insights.
B.Sc. Biotechnology
M.Sc.
Computational Biology
P.hD.
Computational Biology
EMIBA
Hynts Analytics

esade

# Topic 1: What is Python and why is it important?

# What is Python?

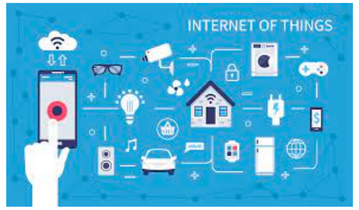
- Versatile, and powerful **programming language**
- Concise, and **easy to learn**, use and read
- **Large** python **community** (Widely used)
- Large number of libraries (Big data, Machine Learning)
- **High demand** in the job market

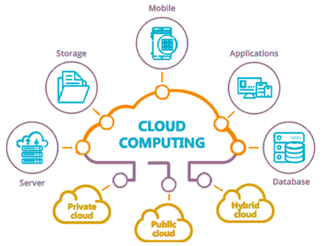
**PYTHON**

```python
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.applications.inception_v3 import InceptionV3


class ConvBlock(Model):

    def __init__(self, filters, kernel_size):
        super(ConvBlock, self).__init__()
        self.conv = Conv1D(filters, kernel_size, padding="same",
                           kernel_initializer="lecun_normal")
        self.bn = BatchNormalization()
        self.relu = ReLU()

    def call(self, inputs):
        x = self.conv(inputs)
        x = self.bn(x)
        x = self.relu(x)
        return x
```

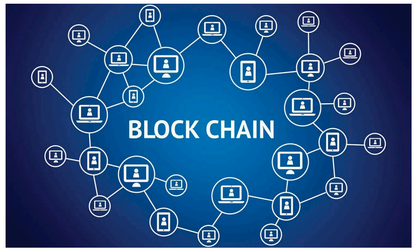
# Future trends & Challenges

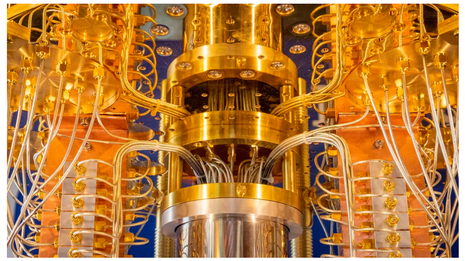
- Artificial Intelligence (AI)
- Internet Of Things (IoT)
- Cloud Computing
- Big Data
- Blockchain
- Quantum Computing
- Metaverse
- Smart Cities
- Autonomous Vehicles
- Cybersecurity

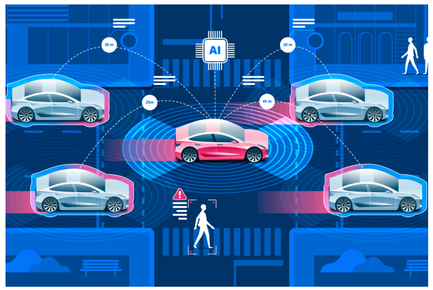


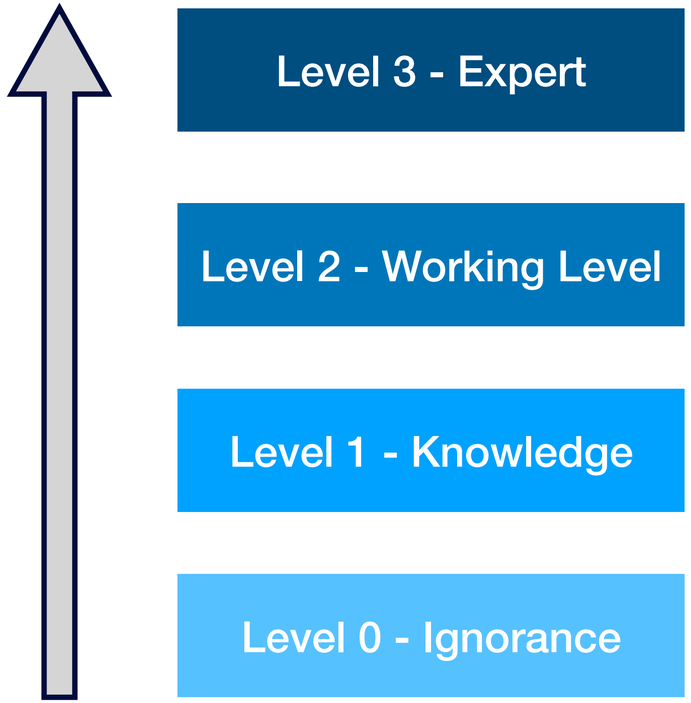




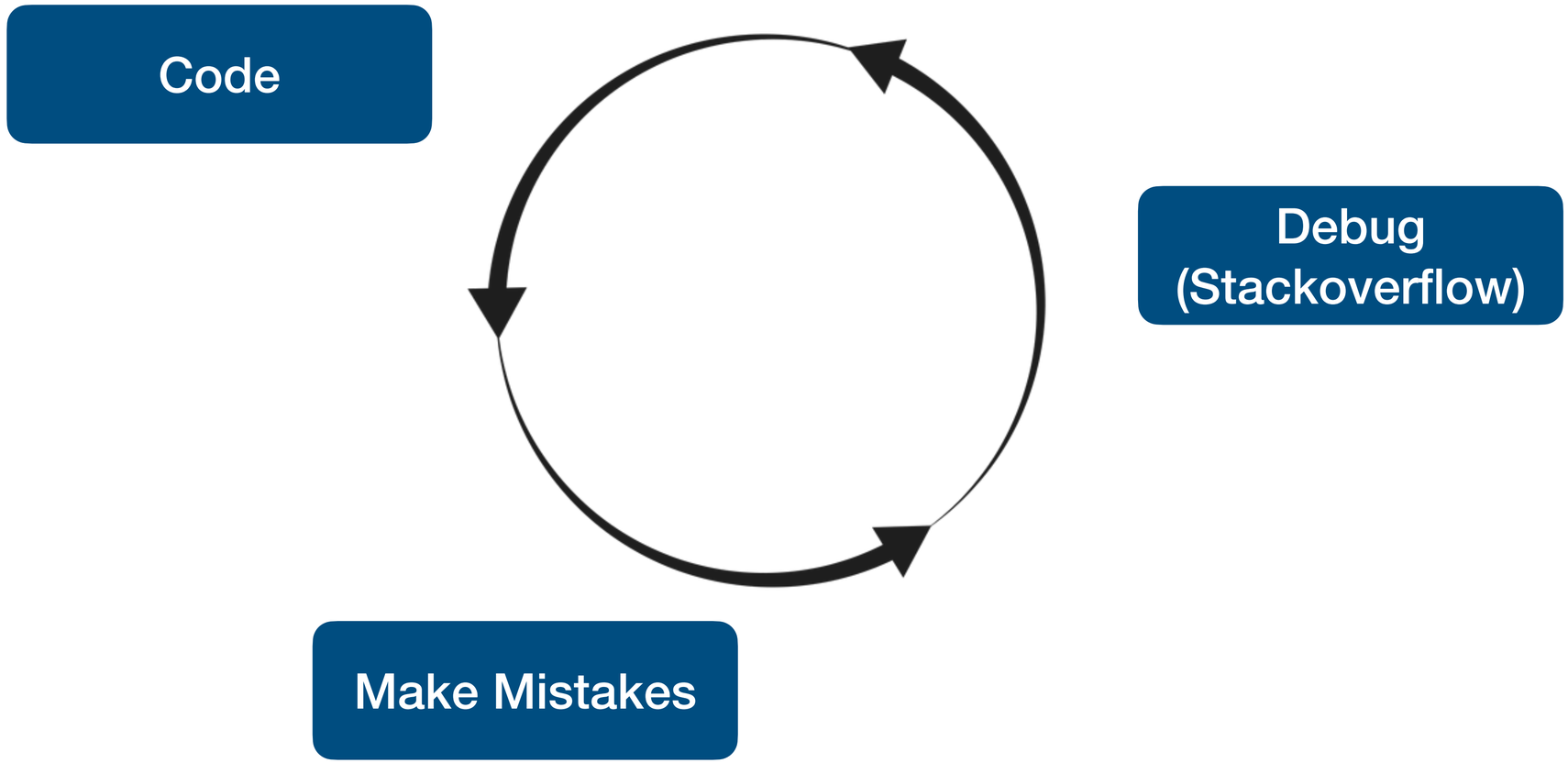


# Towards achieving (at least) level 1

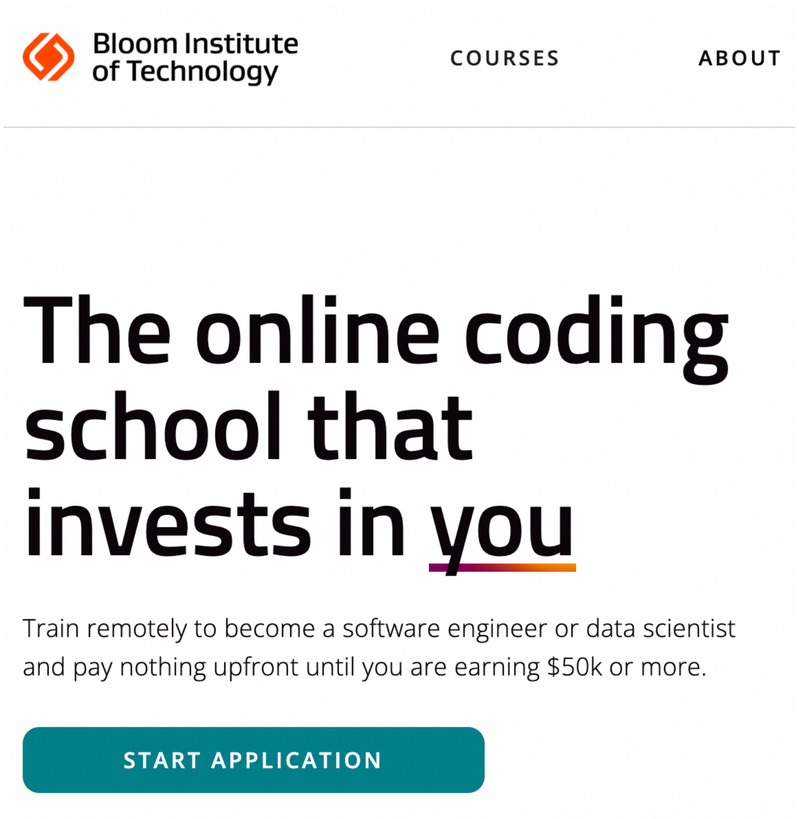


# How do we learn it?

1. **Learn by doing** (The coding cycle)



# How do we learn it?




# My Take

- Python (+ SQL) + Github + (maybe) Cloud Computing can completely change your work
- Do not stress over python. You will be perfectly fine without it.
- But it should be a must to (Harvard HBS):
    - Know about it
    - Understand what it is useful for
    - How it applies to companies now
    - As you are doing the EMIBA: Get an intuition on how to use it

esade

# Topic 2: Python Basics

# Python Backbone

- Variables
- Programming mind (indentation)
- Lists, Dictionaries
- Loops: for, if, elif, else, while
- Functions

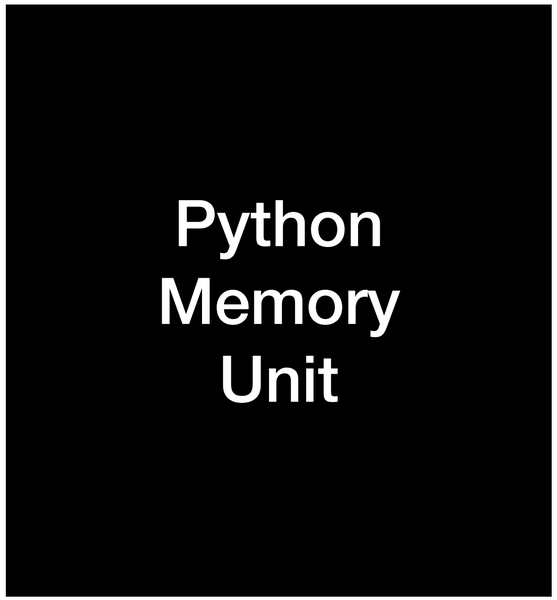
```
In [7]: my_list = [1, 5, 12, 91, 102]
        my_list_length = len(my_list)
        for i in range(0,my_list_length):
            print(i, my_list[i] * my_list[i])

        0 1
        1 25
        2 144
        3 8281
        4 10404
```

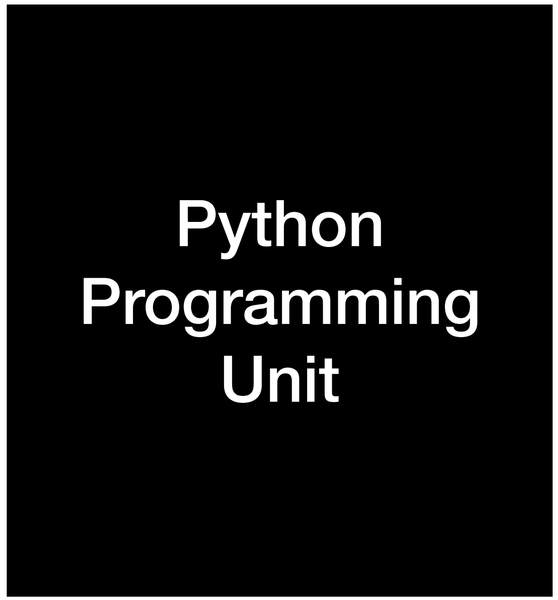
# Python Libraries

- Pandas
- Numpy
- Scikit Learn
- Tensorflow
- Keras

# Variables

- What is a variable? A place for storing data values (Numbers, text, DataFrames, Lists, dictionaries…)
- Types of variables: float, int, string/text, boolean, tuple, dict



# Tips for Variables

- Name your variables properly
  - Not long names but expressive (meaningful names related to the variable)
  - Allows you to recognize what the variable stores
- Example: **df** is a convention for naming a data frame. Why not: **my_data**

Instead, you can assign variables a value like below and then this will be stored:

```
In [4]: a = 20

In [5]: b = 10

In [6]: print(a, b)
20 10
```

```
In [8]: type(b)
Out[8]: int

In [9]: e = 'Hello'

In [10]: type(e)
Out[10]: str

In [11]: f = 1.0

In [12]: type(f)
Out[12]: float
```

# Operations with variables

- Python can essentially be a calculator
- Regular operators: +, -, /, *, ^, **, %
- The results from operations can be saved as new variables

```
In [16]: a * b
Out[16]: 200

In [17]: a + b
Out[17]: 30
```

The result from operations can be saved as a new variable.

```
In [18]: c = a + b
         print(c)
         30

In [19]: g = c * 10 * a

In [20]: print(g)
         6000

In [21]: g / 10
Out[21]: 600.0

In [22]: g % 10
Out[22]: 0
```

# Lists

- One of the building blocks of Python for storing information
- A way of storing multiple items in a single variable

  Sports = [‘Football’, ‘Basketball’, ‘Hockey’]

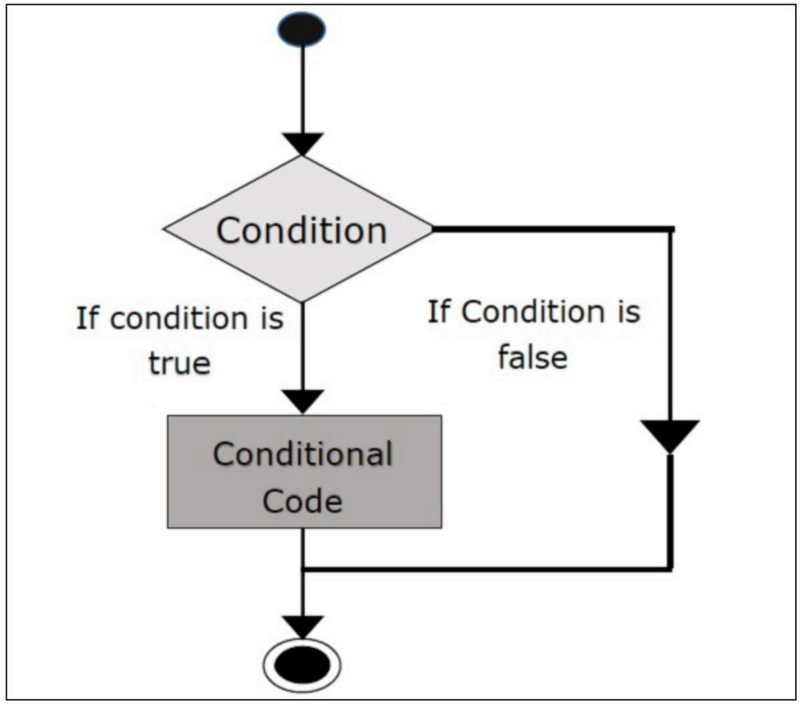
  Students = [‘Iñigo’, ‘Tomas’, ‘Margarita’, ‘Antonella’]
- List indexing: Students[0]
- List operations: append, remove, count…

# Exercise

- Create a list that you like, append an element and output its second element

# If-Else statements

- Evaluate test expressions and execute the code only when the test condition is true.
- Suppose two variables **a** and **b**. We can assess:
  - a == b, a != b, a > b, a < b, a <= b, a >= b




```
In [26]: if a == 10:
             print('The variable a is equal to 10')
         else:
             print('The variable a is different than 10')

The variable a is different than 10
```

# Exercise

- Evaluate whether your previous list contains more than 3 items.
  - Help: len(my_list) > 3
- If it is: then print yes. Otherwise (else) print no.

# For loops

- For loops are used to iterate over sequences like lists, arrays, dictionaries, or others.

In [63]:
```
students = ['Margarita', 'Antonella', 'Tomas', 'Iñigo']
for el in students:
    print(el)
```

```
Margarita
Antonella
Tomas
Iñigo
```

In [61]:
```
for el in students:
    if el == 'Iñigo':
        print('This student is Iñigo')
    else:
        print('This is other student')
```

```
This is other student
This is other student
This student is Iñigo
This is other student
This is other student
```

# Exercise

- Create a list with numbers in it. my_list = [0, 1, 2, 3]
- Iterate over it with a for loop, subtract 2 to every number, and print every number at every iteration.
- Plus points if: Start a new list and save the new numbers on it and print it after the for loop

# While Loops

- While loops are used to execute a set of statements as long as a condition is true.
- They are not widely used

```
In [70]: aa = 5
while aa != 0:
    print(aa)
    aa -= 1
print(f'The new value of aa is {aa} as it should be')
```

```
5
4
3
2
1
The new value of aa is 0 as it should be
```

# Introduction to functions

- A function is a block of code that only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

```
In [78]: def square_root(x):
             return x ** 0.5

In [79]: square_root(36)
Out[79]: 6.0

In [80]: square_root(16)
Out[80]: 4.0
```

```
In [71]: def sum_two(my_list):
             new_list = []
             for el in my_list:
                 new_list.append(el + 2)
             return new_list

In [73]: list1 = [0, 1, 2, 3]
         list2 = [0, -1, -2, -3]

In [76]: sum_two(list1)
Out[76]: [2, 3, 4, 5]

In [77]: sum_two(list2)
Out[77]: [2, 1, 0, -1]
```

# Exercise

- Create a function that computes and returns the square of any number that is passed to the functions.

esade

# Topic 3: Pandas

# What is Pandas?


- Python package that provides:
  - Fast
  - Flexible
  - Expressive data structures
- To make working with "relational" or "labeled" data both easy and intuitive

Step 0
Load the package

```
In [2]: import pandas as pd
```

**Notebook lessons**

**1.- Load a csv/tsv/excel file and explore it**

**2.- Basic Data Exploration**

**3.- Column & row operations**

**4.- Groupby, apply & Lambda function**

# Step 1: Load a csv / tsv / excel file

- To work with a dataset in Jupyter Notebook we need to load it.
- We can load any file type: csv, tsv, excel…
- This allows us to start exploring the data

**Step 1**
**Load dataset in Jupyter notebook**

```
In [2]: df = pd.read_csv('../datasets/Dataset_credit_scoring.tsv', sep='\t')

In [3]: df_excel = pd.read_excel('../datasets/Dataset_credit_scoring.xlsx')

In [12]: df_excel.head()

Out[12]:
```

| | Occupation | Age | Loan_Salary_ratio | Outcome |
|---|---|---|---|---|
| 0 | Industrial | 34 | 2.96 | Repay |
| 1 | Professional | 41 | 4.64 | Default |
| 2 | Professional | 36 | 3.22 | Default |
| 3 | Professional | 41 | 3.11 | Default |
| 4 | Industrial | 48 | 3.80 | Default |

**Rows**

**Columns**

# Step 2: Data frame exploration

- One the most important things is to learn how to explore the data before taking actions
- To do so there are a set of simple functions that can be employed:
  - columns, index, describe(), info(), dtypes, head(), tail()

```
In [28]: df.columns
Out[28]: Index(['Occupation', 'Age', 'Loan_Salary_ratio', 'Outcome'], dtype='object')

In [25]: df.index
Out[25]: RangeIndex(start=0, stop=10, step=1)

In [29]: df.describe()
Out[29]:
```

| | Age | Loan_Salary_ratio |
|---|---|---|
| **count** | 10.000000 | 10.00000 |
| **mean** | 40.300000 | 3.30900 |
| **std** | 8.692909 | 1.17557 |
| **min** | 32.000000 | 1.50000 |
| **25%** | 34.500000 | 2.63750 |
| **50%** | 38.500000 | 3.16500 |
| **75%** | 41.000000 | 4.06250 |
| **max** | 61.000000 | 5.25000 |

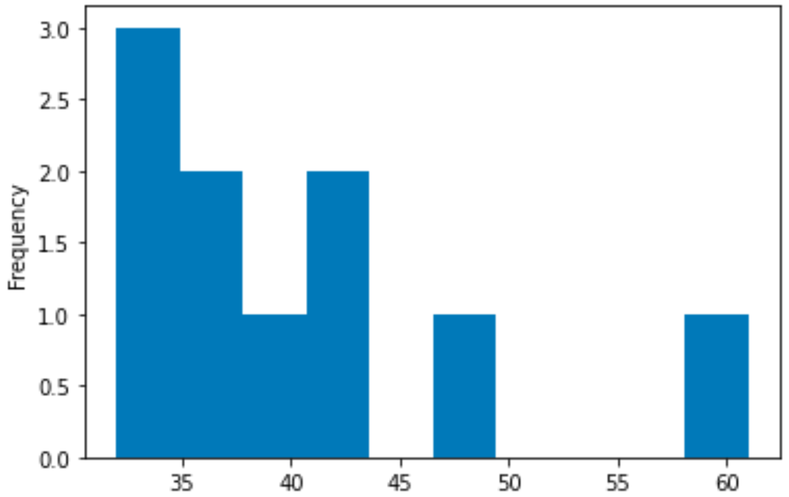
# Exercise

- Which is the oldest? Youngest? Largest Loan_Salary_ratio?

# Step 3: Operations with Columns & Rows

- Knowing how to handle the data and easily access the information is essential.
- Exploration will set our next steps to clean the data and build a model

```
In [64]: df['Outcome'].value_counts()

Out[64]: Default    6
         Repay      4
         Name: Outcome, dtype: int64

In [65]: df['Outcome'].unique()

Out[65]: array(['Repay', 'Default'], dtype=object)

In [66]: df['Outcome'].nunique()

Out[66]: 2

In [67]: print(df['Age'].min(), df['Age'].max())

         32 61
```




# Step 3: Operations with Columns & Rows

- Separate data frames
- Filter Data by selecting a certain number of columns or rows
- Create new columns through an operation

```
In [171]: df_age_40 = df[df['Age'] > 40]
          df_age_40.head()
```

Out[171]:

| | Occupation | Age | Loan_Salary_ratio | Outcome |
|---|---|---|---|---|
| **1** | Professional | 41 | 4.64 | Default |
| **3** | Professional | 41 | 3.11 | Default |
| **4** | Industrial | 48 | 3.80 | Default |
| **5** | Industrial | 61 | 2.53 | Repay |

```
In [172]: new_df = df[['Age', 'Loan_Salary_ratio', 'Outcome']]
          new_df.head()
```

Out[172]:

| | Age | Loan_Salary_ratio | Outcome |
|---|---|---|---|
| **0** | 34 | 2.96 | Repay |
| **1** | 41 | 4.64 | Default |
| **2** | 36 | 3.22 | Default |
| **3** | 41 | 3.11 | Default |
| **4** | 48 | 3.80 | Default |

```
In [77]: df['Age_Loan_Salary_ratio'] = df['Age'] / df['Loan_Salary_ratio']
```

```
In [78]: df.head()
```

Out[78]:

| | Occupation | Age | Loan_Salary_ratio | Outcome | Age_Loan_Salary_ratio |
|---|---|---|---|---|---|
| **0** | Industrial | 34 | 2.96 | Repay | 11.486486 |
| **1** | Professional | 41 | 4.64 | Default | 8.836207 |
| **2** | Professional | 36 | 3.22 | Default | 11.180124 |
| **3** | Professional | 41 | 3.11 | Default | 13.183280 |
| **4** | Industrial | 48 | 3.80 | Default | 12.631579 |

# Exercise

- There was a mistake when introducing the age of our clients. We need to correct them. They ask us to create a new column named *corrected_age* that contains the original age minus 2. Can you do that?

# Groupby, apply and Lambda Function

- The groupby operation can be used to group large amounts of data and compute operations on these groups separately.
- "split-apply-combine" operation:
  - Splitting the data into groups based on some criteria
  - Applying a function to each group independently
  - Combining the results into a data structure

```
In [188]: df[df['Outcome'] == 'Repay']['Age'].mean()
Out[188]: 43.0

In [189]: df[df['Outcome'] == 'Default']['Age'].mean()
Out[189]: 38.5
```

```
In [194]: df.groupby('Outcome')['Age'].mean()
Out[194]: Outcome
          Default    38.5
          Repay      43.0
          Name: Age, dtype: float64

In [193]: df.groupby('Outcome')['Age'].apply(lambda x: x.sum()/x.count())
Out[193]: Outcome
          Default    38.5
          Repay      43.0
          Name: Age, dtype: float64
```

# Exercise

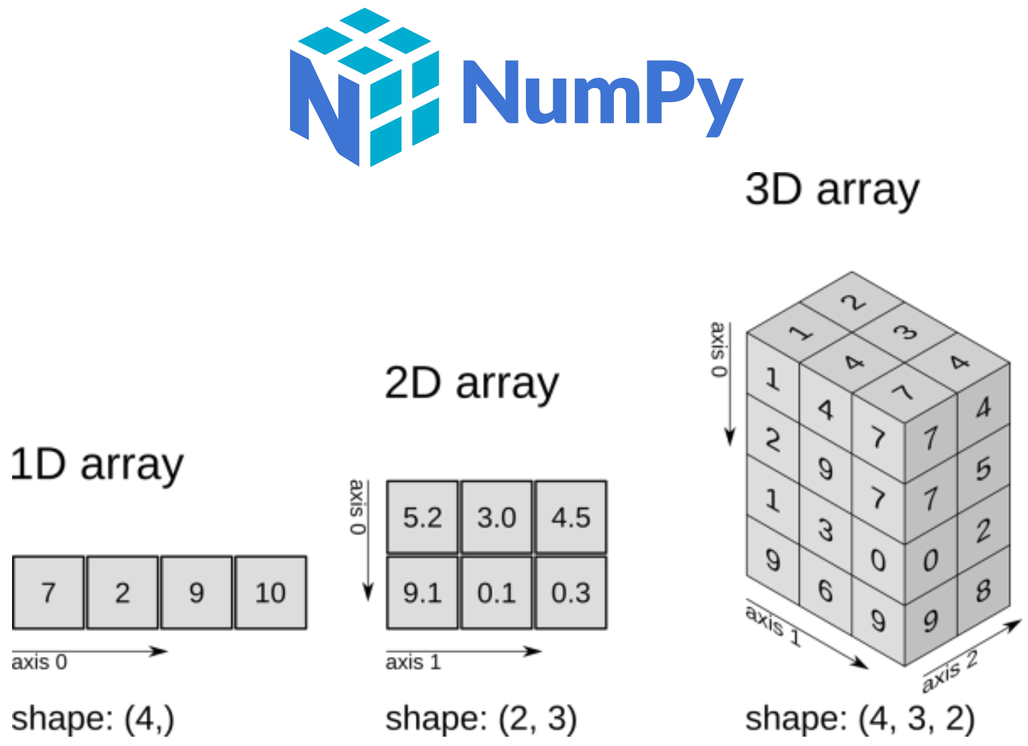
- We have the feeling that the loan to salary ratio determines whether the loan will be repaid or not. Can we get an idea of the mean of the loan to salary ratio for each of these groups?

esade

# Topic 4: Numpy

# What is Numpy?

- NumPy is a Python library used for working with arrays
- Why use it? NumPy aims to provide an array object that is up to 50x faster than traditional Python lists
- Arrays are very frequently used in data science, where speed and resources are very important
- Large support for statistics, distributions, etc…



# Exercise

- Build different arrays of different shapes.
- Get the first element from the array
- Get the mean and standard deviation

esade

# Topic 5: What’s Next

# Datacamp Courses

## Python Fundamentals

Are you ready to gain the foundational skills you need to become a Python programmer? In this track, you'll learn the Python basics you need to start on your programming journey, including how to clean real-world data ready for analysis, use data visualization libraries, and even how to write your own Python functions.

Your instructor Hugo will introduce you to how companies worldwide use Python to gain a competitive edge. Through hands-on coding exercises you'll then learn how to store, manipulate, and explore data using NumPy. Then it's time to level-up as you learn how to visualize your data using Matplotlib, manipulate DataFrames and dictionaries using pandas, and write your own functions and list comprehension. Start this track to add these essential Python skills to your data science toolbox.

Switch Track

Python 15 hours 4 Courses 1 Skill Assessment

## Data Scientist with Python ENROLLED

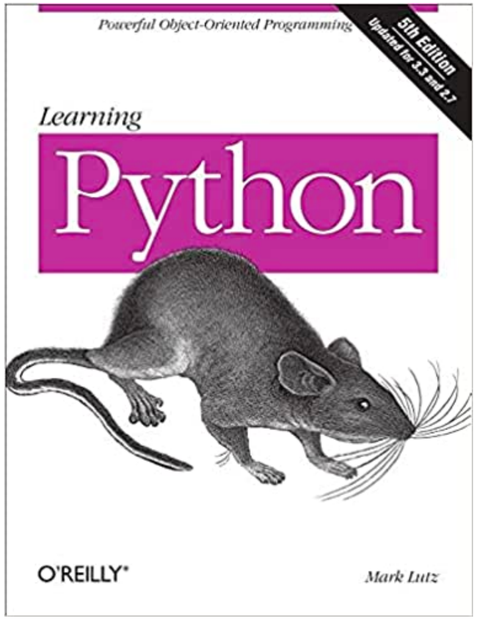
Gain the career-building Python skills you need to succeed as a data scientist. No prior coding experience required.

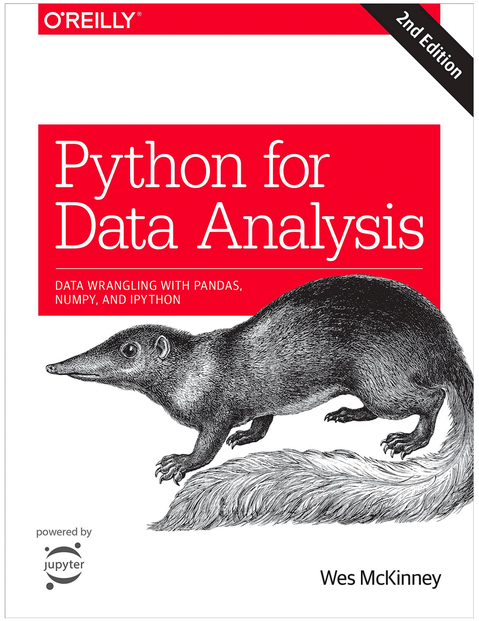
In this track, you'll learn how this versatile language allows you to import, clean, manipulate, and visualize data—all integral skills for any aspiring data professional or researcher. Through interactive exercises, you'll get hands-on with some of the most popular Python libraries, including pandas, NumPy, Matplotlib, and many more. You'll then work with real-world datasets to learn the statistical and machine learning techniques you need to train decision trees and use natural language processing (NLP). Start this track, grow your Python skills, and begin your journey to becoming a confident data scientist.

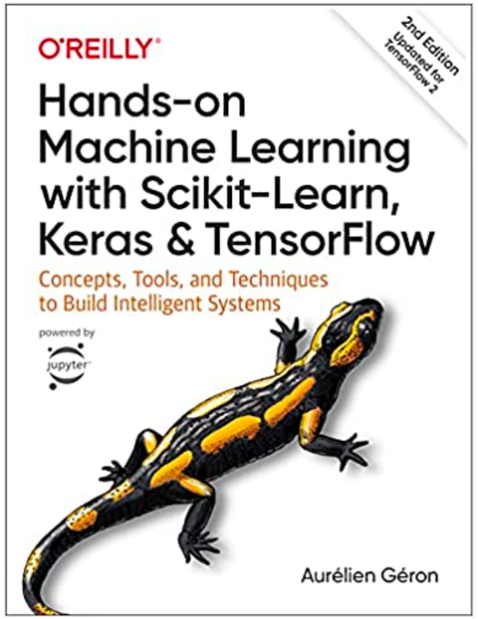
Resume Track

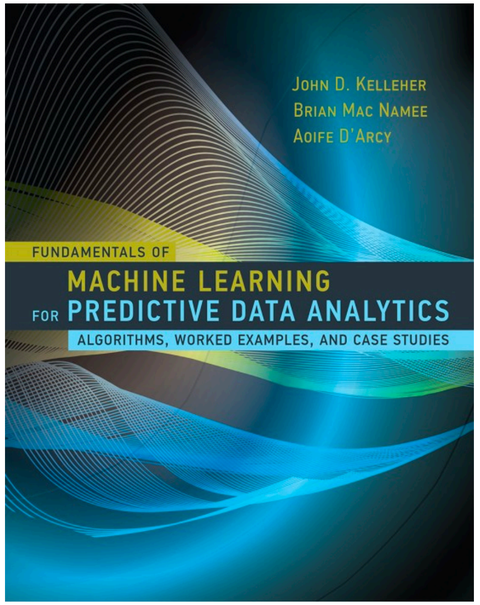
Python 88 hours 23 Courses 6 Projects 3 Skill Assessments

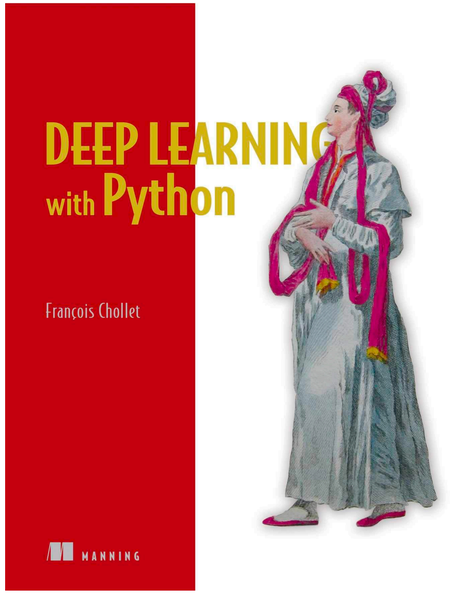
# Free Books to learn Python & data analytics



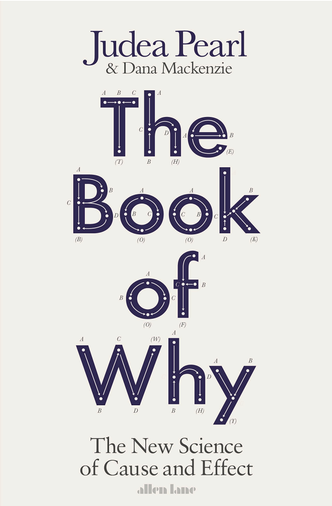



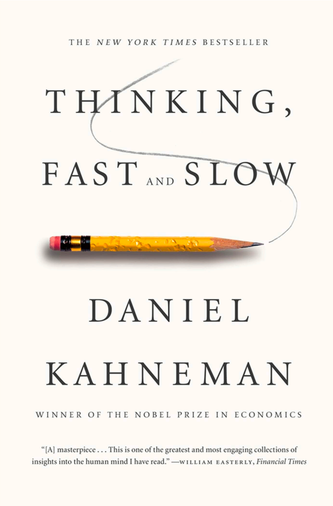
# My recommendations for the Master

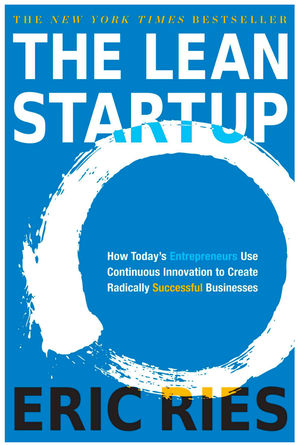
- Talk to your colleagues. Networking, networking, networking!
- Read the materials and participate
- Ask questions
- Work on your own on the things that you believe can be better for yourself
- Enjoy & make friends
- READ!

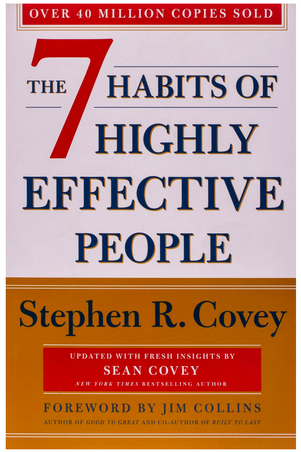
# Read as many books as possible!

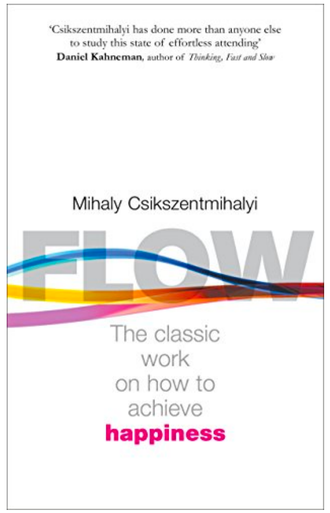
## Gaining a wide perspective is a long process



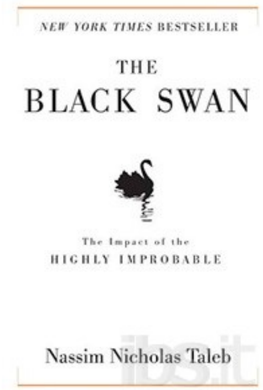



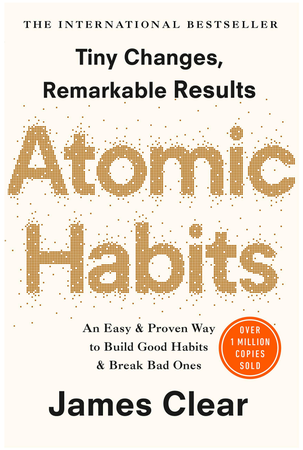



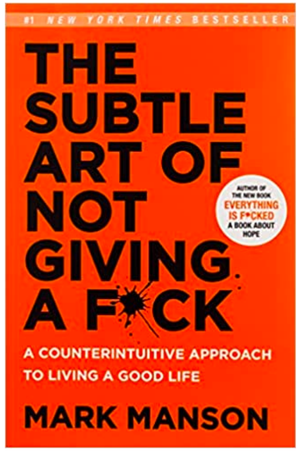



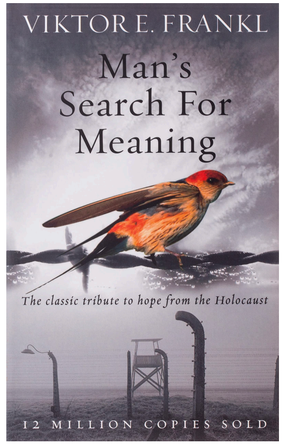



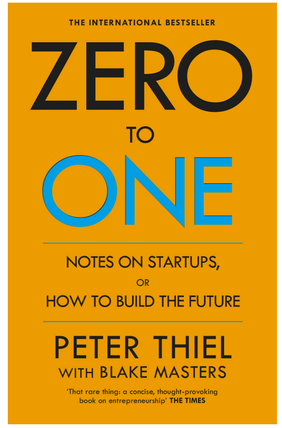


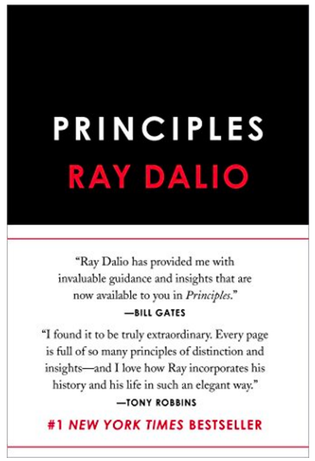


# Feel Free to contact me whenever!

- Linkedin: Jose Bonet Giner
- Personal mail: pepebogi5@gmail.com
- Blog: pepesjourney.com
- Twitter: pepeb_5

**Thank you!**
**&**
**Please fill in the feedback**