esade

Advanced Python

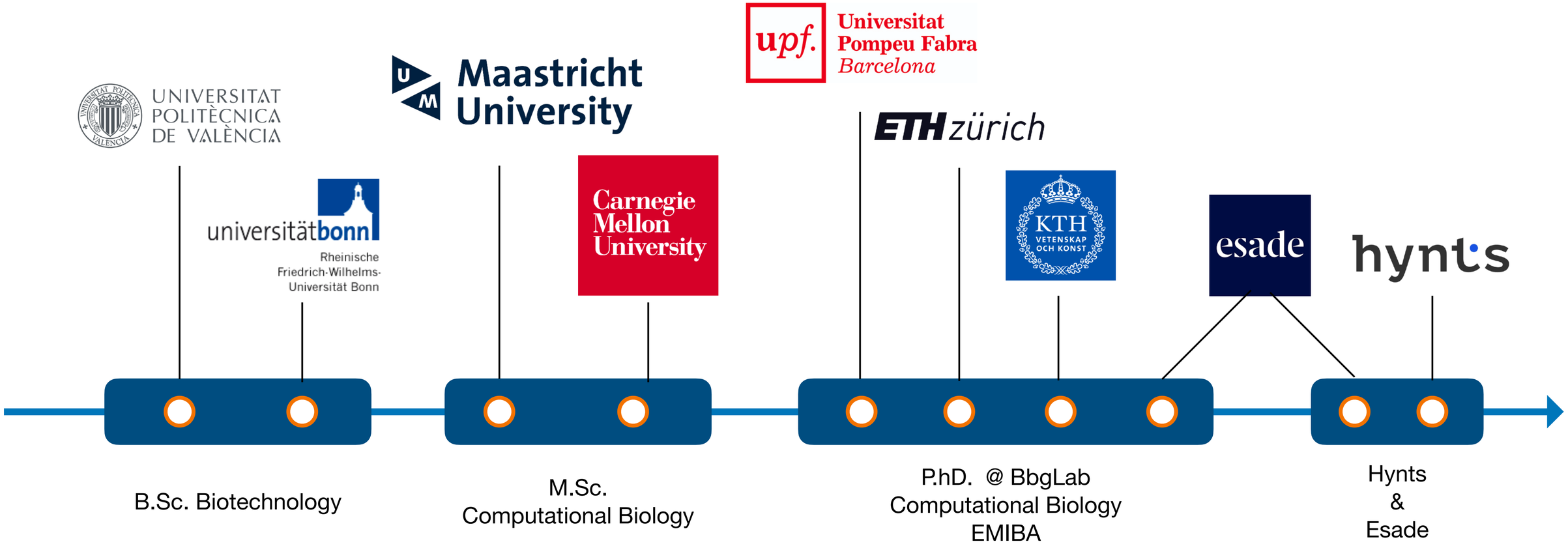
Class 1

Pepe Bonet Giner

8th Jan 2024

Index Class 1

Who am I?


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
B.Sc. Biotechnology
M.Sc.
Computational Biology
P.hD. @ BbgLab
Computational Biology
EMIBA
Hynts
&
Esade

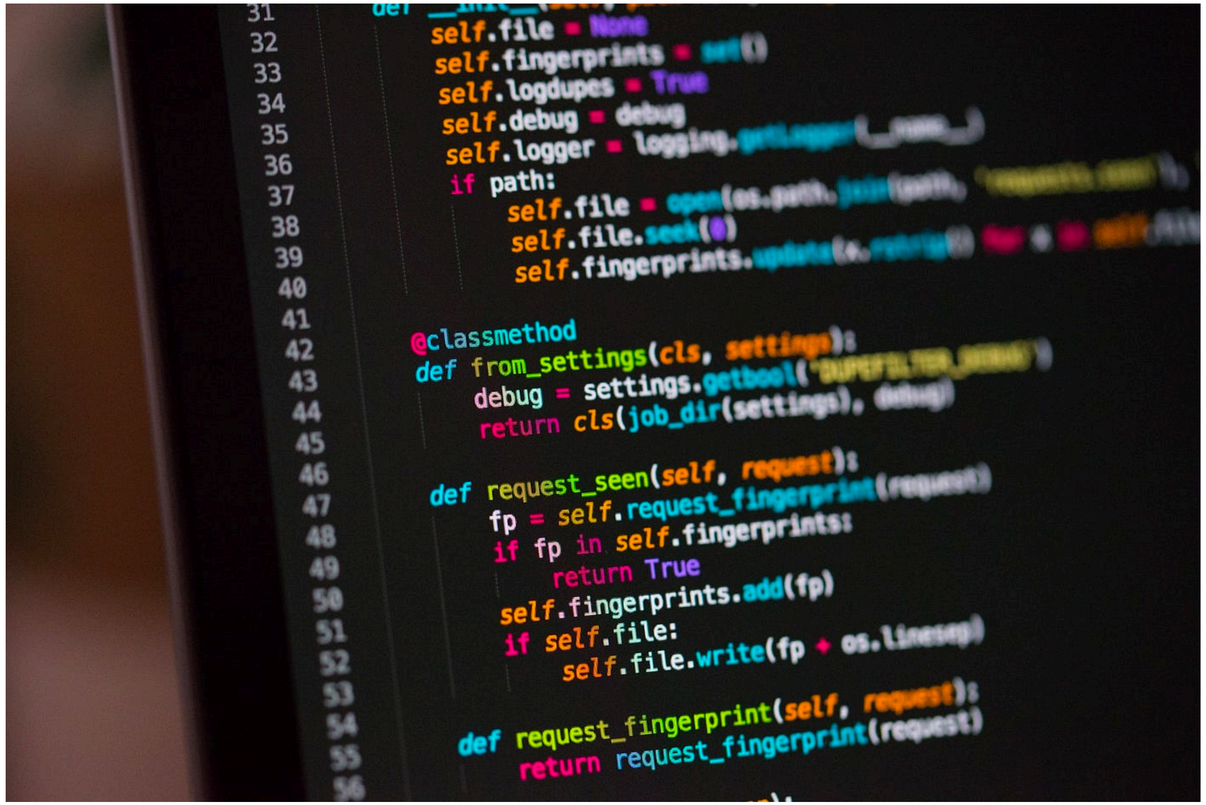
esade

Topic 1: What is this course about?

What are we going to learn?

- Scripting (VStudio)
- Classes
- Click
- Debugging
- GitHub 1 (Version Control)

- GitHub 2 (CD & CI)
- Project Structure
- Testing
- Linting + Clean Code
- Multiprocessing



How is it going to be graded?

- **60% Active participation and class attendance**:
  - 5 % Attendance
  - 55 % coding exercises (May involve after class work)
  - Exercises will be uploaded to the Moodle after every session.

How is it going to be graded?

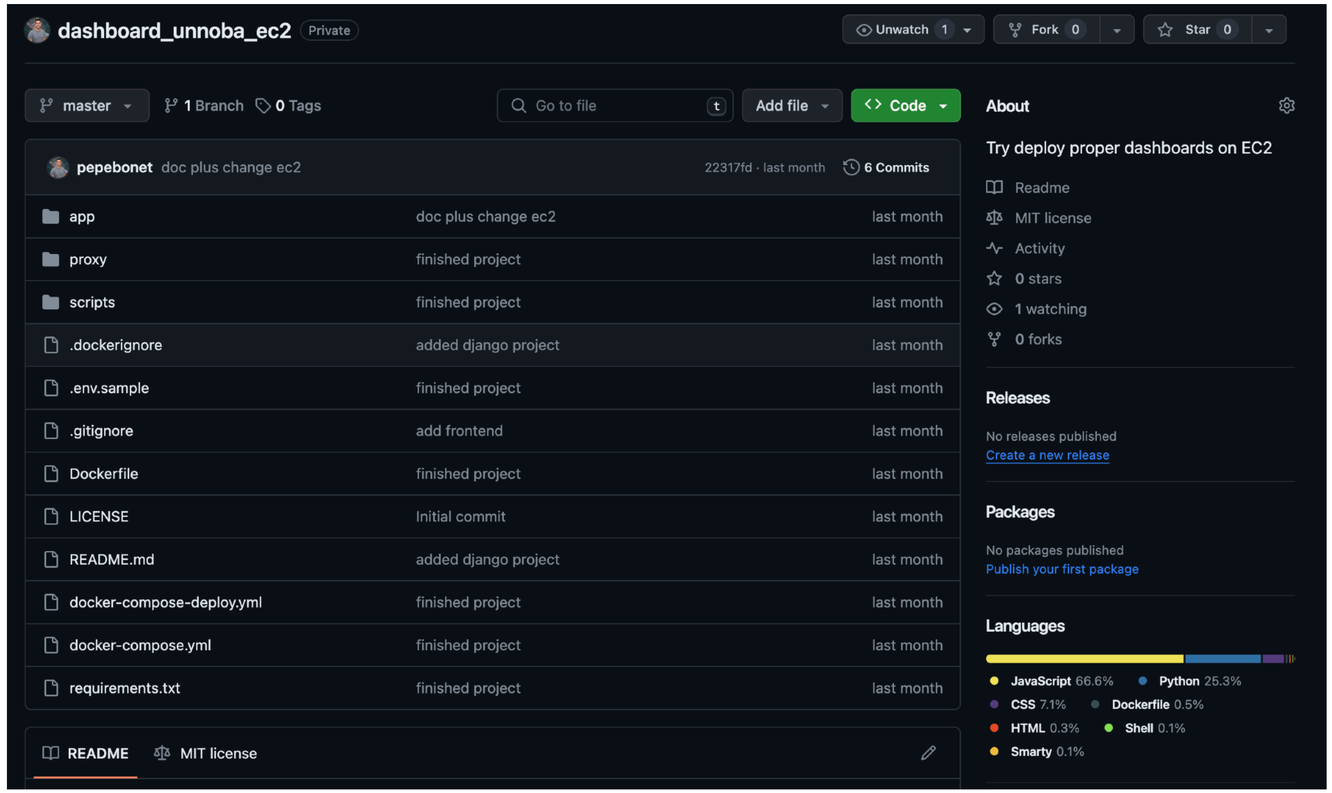
- **60% Active participation and class attendance**:
  - 5 % Attendance
  - 55 % coding exercises (May involve after class work)
  - Exercises will be uploaded to the Moodle after every session.

- **40% Final project**:
  - GitHub repository containing all your files and scripts applying the concepts learned in class on a dataset.
  - Upload both to the Moodle max two weeks after the last lecture.

Final Project

A GitHub repository containing all the things we see during the course for a given dataset:

- Scripting (VStudio)
- Classes
- Click
- Debugging
- GitHub 1 (Version Control)
- GitHub 2 (CD & CI)
- Project Structure
- Testing
- Multiprocessing
- Linting + Clean Code



Prerequisites of the Course

- Beginners Python Course
- Intermediate Python Course
- Pandas, Matplotlib, SkLearn
- Functions
- Data Analysis
- Comment and write clean code

Before we start

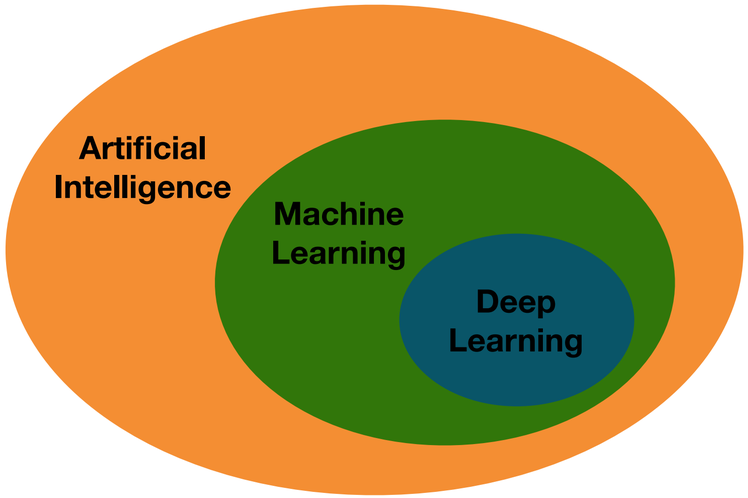
DISCLAIMER

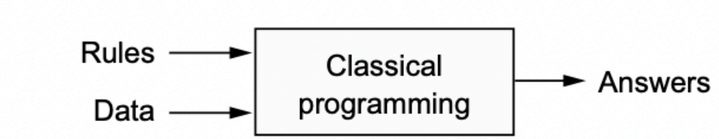
- **1st Time ever I am teaching the course**
  - Feedback is appreciated
  - We will see how fast we cover topics as we go
- **Some topics we will cover are not easy**
  - I struggle as well (at least to code them without help)
  - Not many people can produce the code we will work on without preparation or significant experience

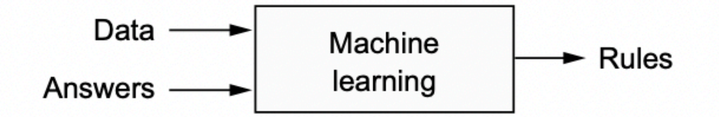
esade

Topic 2: AI revolution & Programming

A bit of AI

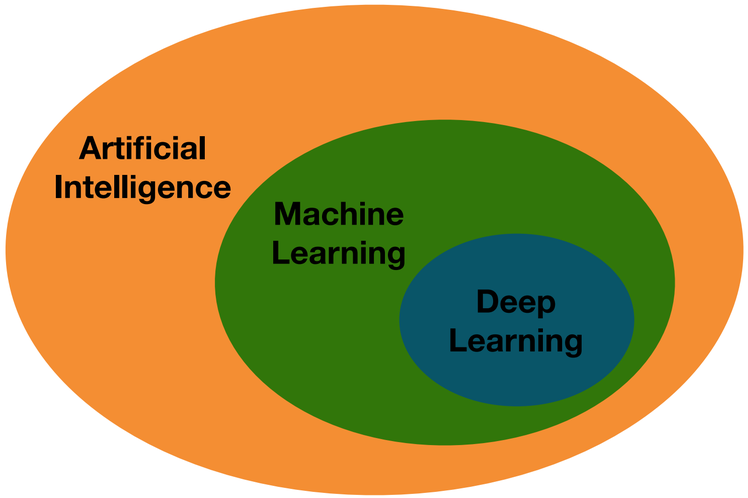


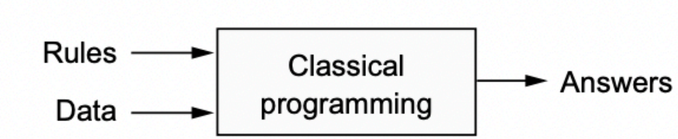


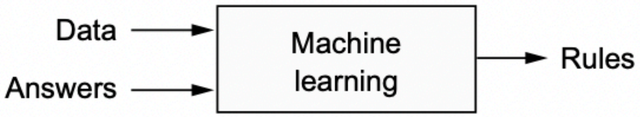


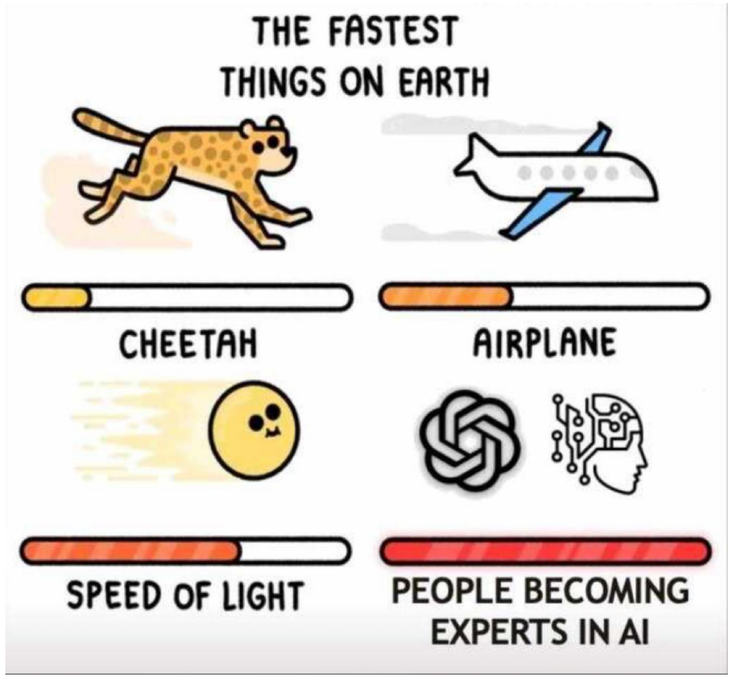
- AI: Effort to automate intellectual tasks normally performed by humans

A bit of AI


Artificial Intelligence
Machine Learning
Deep Learning


Rules
Data
Classical programming
Answers


Data
Answers
Machine learning
Rules


THE FASTEST THINGS ON EARTH
CHEETAH
AIRPLANE
SPEED OF LIGHT
PEOPLE BECOMING EXPERTS IN AI

The AI race has started

Programming languages are behind this revolution as well

- We need to tell the computers to do all of these things
- Programming languages allow us to **communicate with computers**
- A programming language is a **notation system** for writing **computer programs**

Why 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

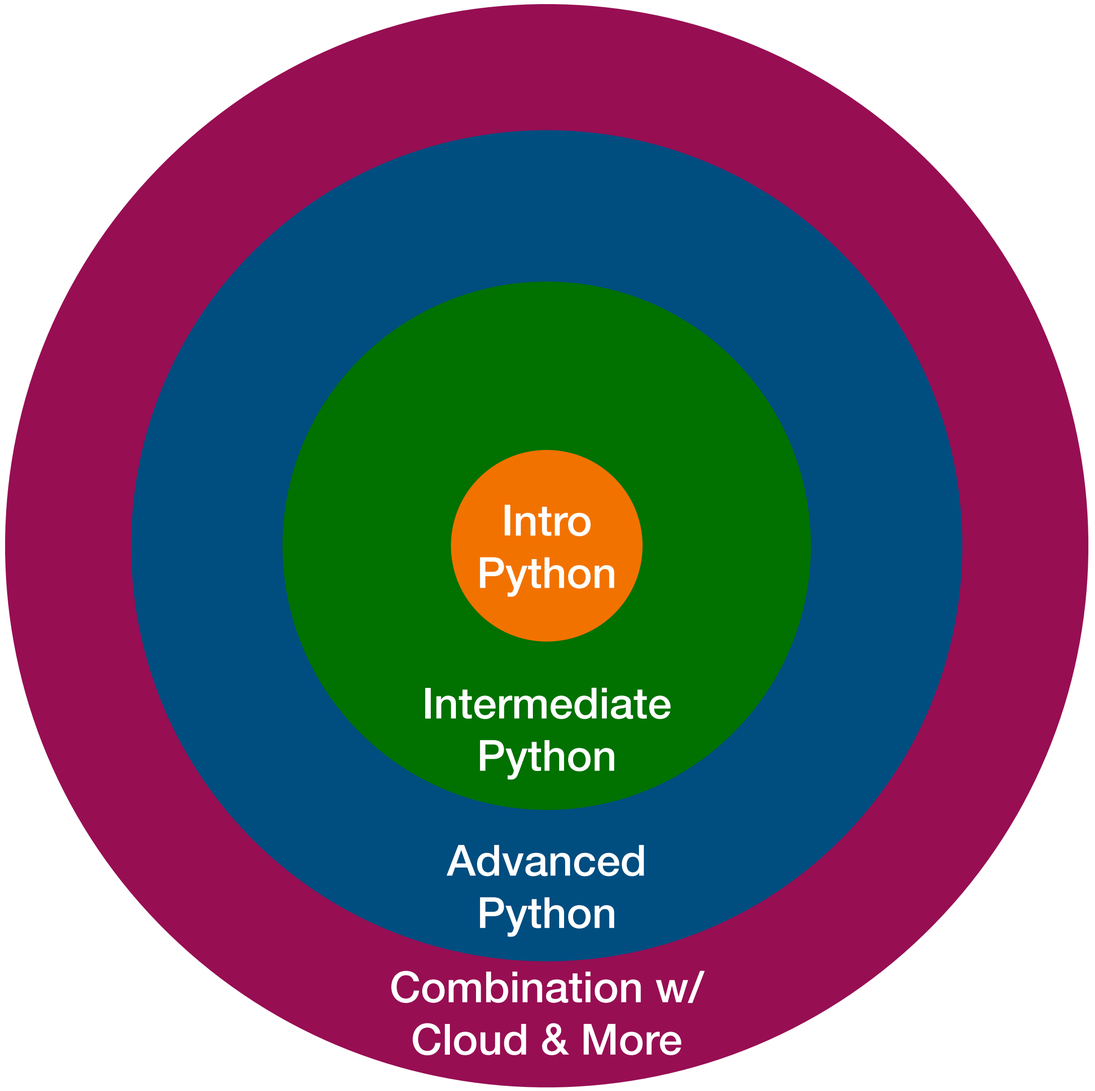
PYTHON

```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
```

esade

Topic 3: Advanced Programming/Python

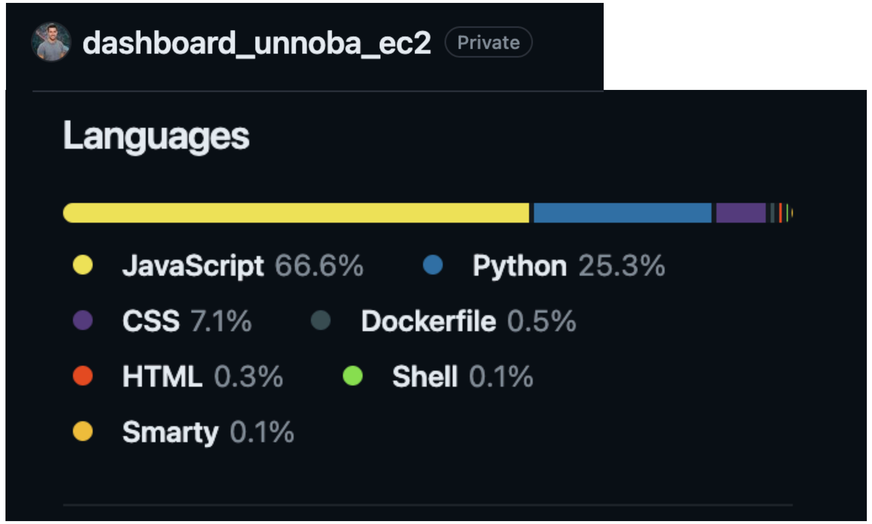
Scaling your code



- 1 Person
- 2-10 People
- 100s People
- 1000s or M People

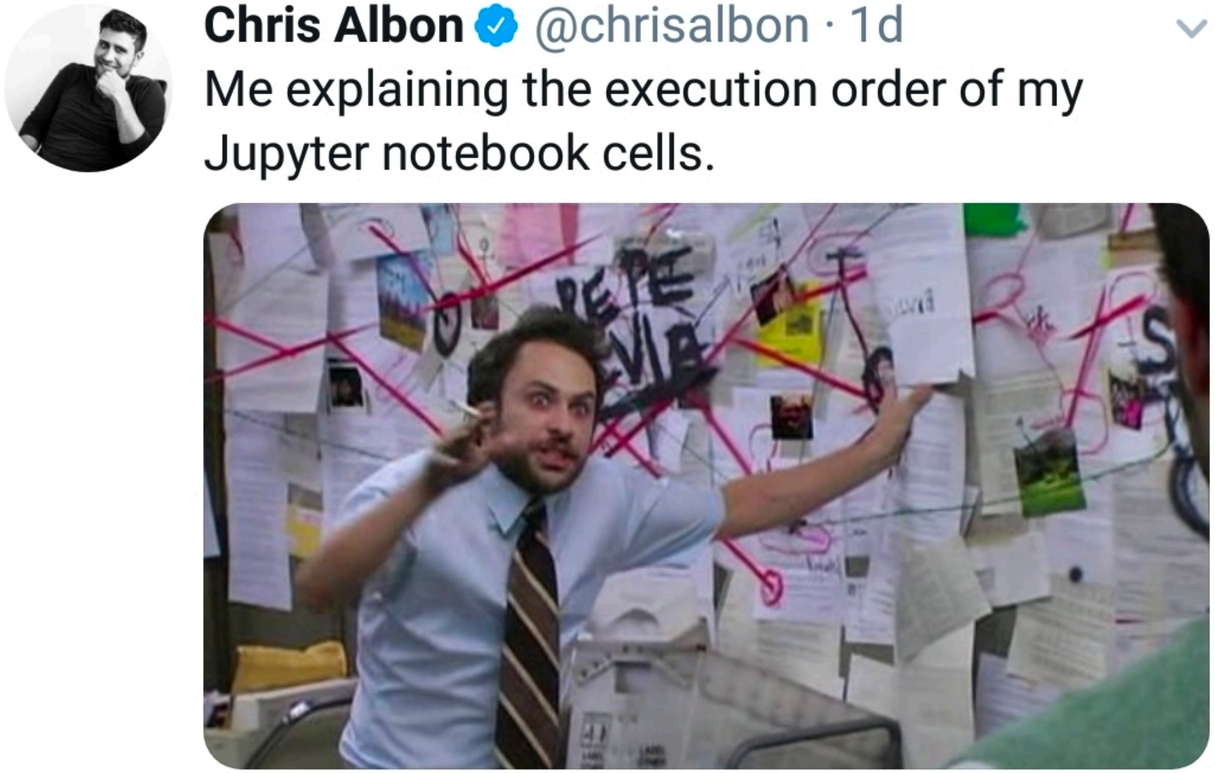
Advanced Programming is more than just Python

- When developing any type of a applications or solution for the public we may need to work on more than one programming language



We need some changes. Move away from Notebooks

- JNs are incredibly used in data projects and ML. However, keep in mind they are not easy to understand for people that did not code in the first place



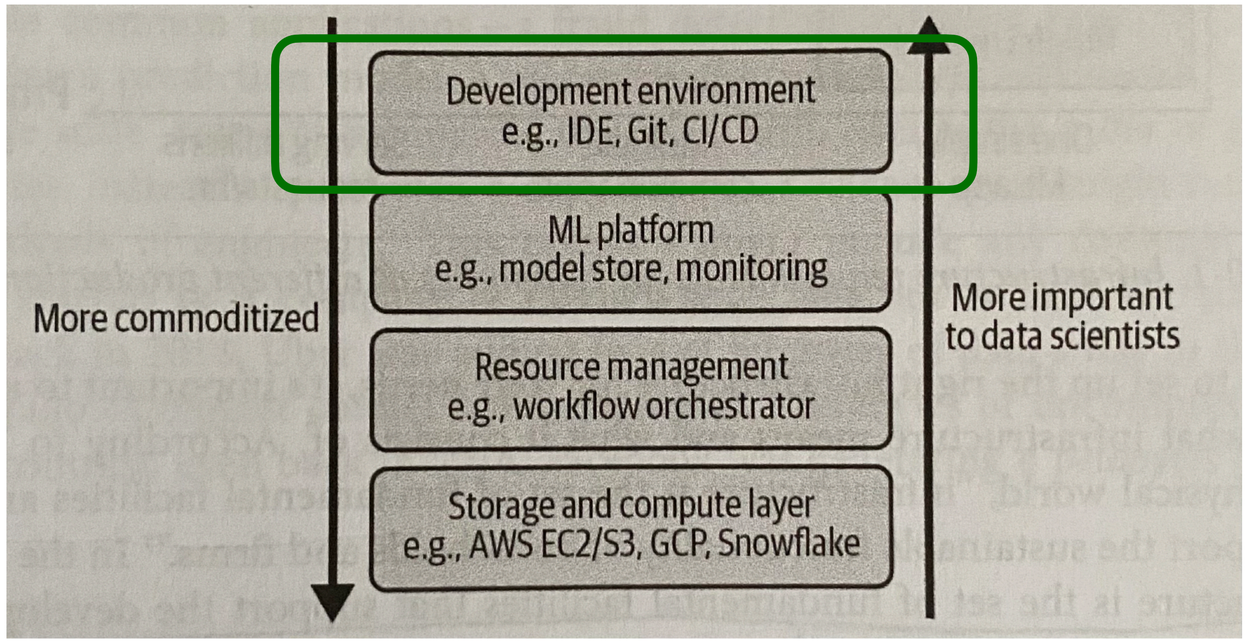
Advanced Python main topics

- Scripting (VStudio)
- Classes
- Click
- Debugging
- GitHub 1 (Version Control)
- GitHub 2 (CD & CI)
- Project Structure
- Testing
- Multiprocessing
- Exceptions + assert
- Linting + Clean Code
- Makefile

Advanced Python main topics

Okay Pepe that is a lot of information. But, where do we start?

Layers of a Scalable Infrastructure (ML in this case)



- We will focus on the development environment
- Where the code is written and run
- The code needs to be versioned and tested
- Experiments need to be tracked

The Development Environment

"If you have time to set up only one piece of infrastructure well, make it the development environment"

Ville Tuulos

IDE

- Editor where you write the code

Code Versioning

- Store & track different versions and stages of your code

CI/CD

- Continuous Improvement & Continuous Development

Let’s not only do JN - A bit of IDE hands-on

- Open Visual Studio and let’s play with it a bit

Let’s move on from JN - A bit of IDE hands-on

- Open VS
- Go to the Desktop
- Create new folder
- Create new .ipynb
- Try importing pandas (problems may arise - next slide)

Solving this:

- We need to install pandas and ipykernel and load the package for our Jupyter Notebooks in VSCode
  - **pip install pandas**
  - **pip install ipykernel**

- Option 1:
  - Go to Anaconda-Navigator. Create an environment (Environments —> Create —> give it a name + python + create).
  - Install the packages —> play button —> Open terminal —> run pip commands

- Option 2:
  - Open terminal in VScode: View —> Terminal
  - Create an environment (Maybe not needed)
  - Run the commands to install the packages

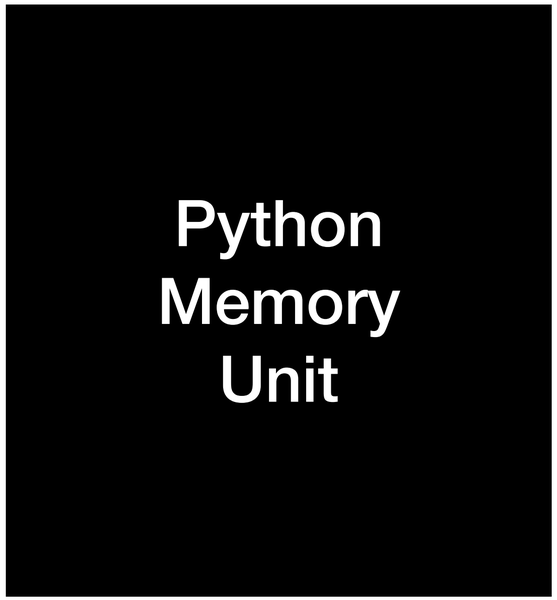
- Option 3 (For Windows):
  - Open Anaconda terminal. Build Environment and install packages

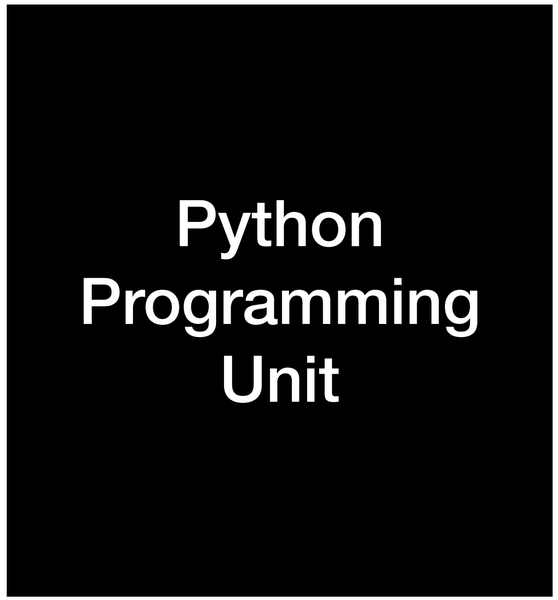
esade

Topic 4: Python Basics Recap

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





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*…

Dictionaries

- Another building block of Python for storing information
- Dictionaries are used to store data values in key-value pairs
  - my_dict = { key : value}
- To index dictionaries: my_dict[key]
- List operations: *update, pop, keys, values*

```
my_dict = {
    'Players': ['Messi', 'Ronaldo', 'Maradona'],
    'Goals' : [10, 8, 6]
}
```

```
my_dict
```

```
{'Players': ['Messi', 'Ronaldo', 'Maradona'],
'Goals': [10, 8, 6]}
```

```
my_dict['Players']
```

```
['Messi', 'Ronaldo', 'Maradona']
```

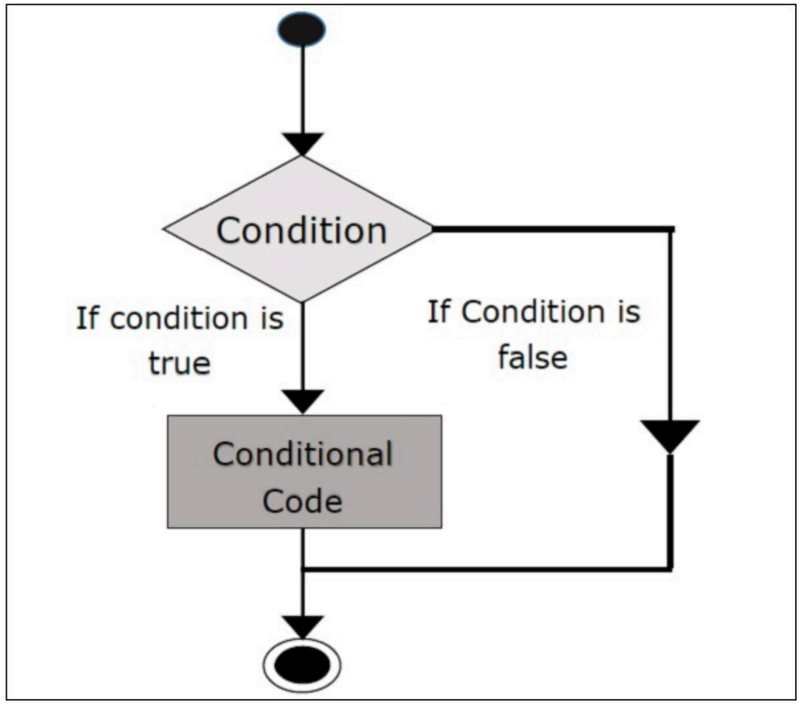
```
my_dict.update({'Assists' : [12, 3, 5]})
```

```
my_dict
```

```
{'Players': ['Messi', 'Ronaldo', 'Maradona'],
 'Goals': [10, 8, 6],
 'Assists': [12, 3, 5]}
```

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
```

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
```

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. (Do examples)

Exercise

- Create 3 functions:
    - 2 of them connected (one calls another)
    - With multiple input parameters
    - That perform different tasks

How do python libraries work

Step 0
Load the package

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

Python Memory Unit

Loads all pandas functions

Python Programming Unit

import pandas as pd

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()

```
df.columns
```

```
Index(['Rank', 'Name', 'Age', 'End_Of_Watch', 'Day_Of_Week', 'Cause',
       'Department', 'State', 'Tour', 'Badge', 'Weapon', 'Offender',
       'Summary'],
      dtype='object')
```

```
df.shape
```

```
(25623, 13)
```

```
df.index
```

```
RangeIndex(start=0, stop=25623, step=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25623 entries, 0 to 25622
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Rank           25623 non-null  object 
 1   Name           25623 non-null  object 
 2   Age            22946 non-null  float64
```

Step 3: Operations with Columns & Rows

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

You can also delete columns

```
new_list = []

for el in df['Age'].tolist():
    if el > 50:
        new_list.append(1)
    else:
        new_list.append(0)

df['Binary age'] = new_list
df.head(2)
```

```
less_50 = df[df['Age'] <= 50]
less_50.head(2)
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week	Cause
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday	Gunfire
5	Sheriff	Robert Maxwell	45.0	1797-11-12	Sunday	Gunfire

```
less_50['Age'].median(), less_50['Age'].max()
```

```
(35.0, 50.0)
```

```
new_df = df[['Rank', 'Name', 'Age']]
new_df.head()
```

	Rank	Name	Age
0	Constable	Darius Quimby	NaN
1	Sheriff	Cornelius Hogeboom	53.0
2	Deputy Sheriff	Isaac Smith	NaN
3	Marshal	Robert Forsyth	40.0
4	Deputy Sheriff	Robert Berwick	NaN

```
new_df = df.iloc[0:10]
new_df.shape
```

```
(10, 13)
```

Groupby + apply + Lambda Function

- Combining all of them gives us an easy way to go through the data
- "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

```
df.groupby('Day_Of_Week').apply(lambda x: x['Age Range'].value_counts())
```

Day_Of_Week	Age Range	Older than 24 but younger than 45	Older than 44	Younger than 25
Friday		2009	1550	181
Monday		1840	1482	193
Saturday		2188	1610	204
Sunday		2042	1507	216
Thursday		1959	1483	178
Tuesday		1882	1460	175
Wednesday		1856	1424	184

Work on your code! You are not the only one reading it!

Comment your code

```python
# Create a list of numbers
numbers = [1, 2, 3, 4, 5]

# Create a new list to store the results
results = []

# Iterate over the numbers in the list
for number in numbers:
    # Subtract 2 from each number and append the result to the results list
    results.append(number - 2)
```

1. Use clear and descriptive variable and function names. Instead of x (or y, z, l, j) put a proper name, i.e., results, number
2. Comment explaining what the code is meant to do
3. Explain complex or obscure code
4. Update comments as you make changes to your code

Which option do you prefer?

1

```python
def calculate_discounted_price(price, discount_rate):
    """
    Calculates the discounted price of an item given its original price and discount rate.
    :param price: The original price of the item
    :param discount_rate: The discount rate as a percentage
    :return: The discounted price of the item
    """
    discount_amount = price * (discount_rate / 100)
    discounted_price = price - discount_amount
    return discounted_price
```

2

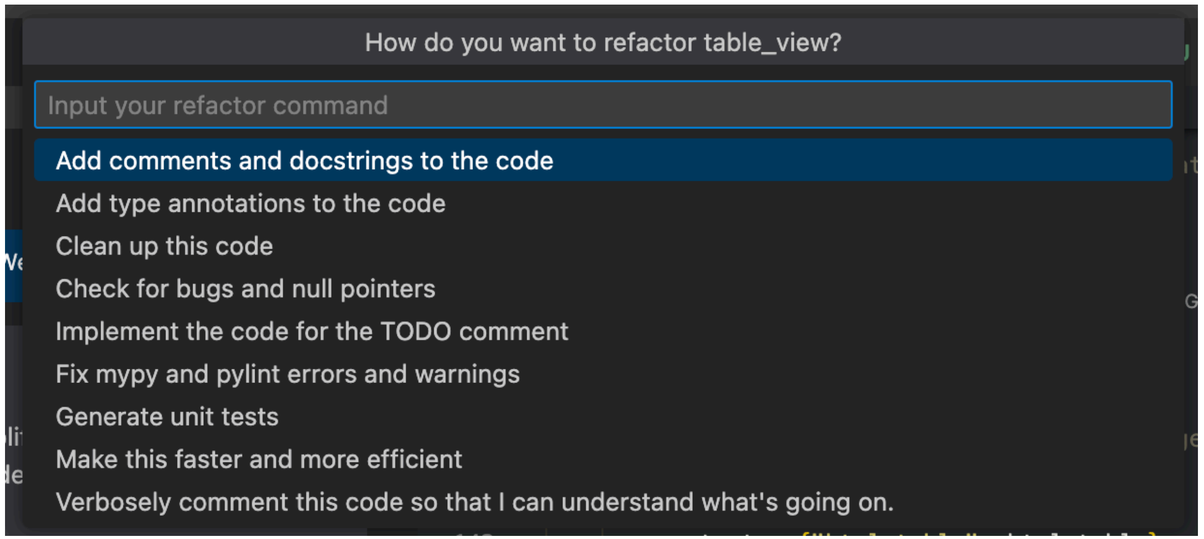
```python
def fn(a,b):
    return a - a*(b/100)
```

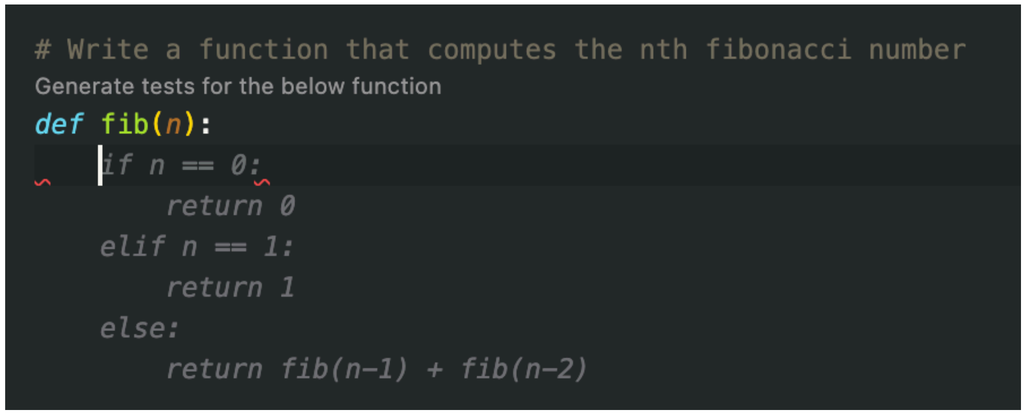
esade

Topic 5:
New Tools: VScode, Virtual Env, Command Line

VS code is much more than a text editor

Visual Studio

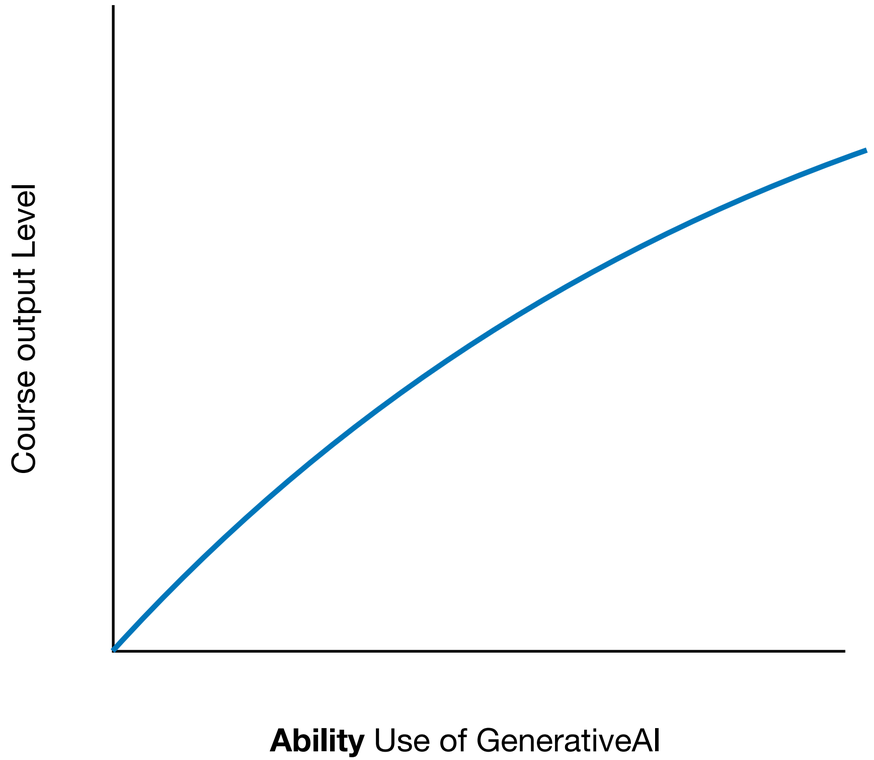




Let’s try to make use of some extensions (If time allows): CodiumAI, Codeium

Comment on GenAI (ChatGPT, Copilot…)

Positive



Negative

- If we don’t work to understand the fundamentals, at some point you will have no idea about what you are doing.
- The code will break eventually and fixing it will not be easy.
- And if you manage to get through. Users will feel it.
  - Poor code quality
  - Security and reliability concerns

Intro to Virtual Environments

- Virtual Environments allow us to load the dependencies (libraries) we need in our project. They are key to:
    - **Avoiding Version Conflicts**
    - Easier Experimentation and Testing
    - Simplified Deployment

```
Bottleneck==1.3.5
contourpy==1.0.5
cycler==0.11.0
fonttools==4.37.3
graphviz==0.20.1
joblib==1.2.0
kiwisolver==1.4.4
matplotlib==3.6.0
packaging==21.3
pandas==1.5.0
Pillow==9.3.0
psutil==5.9.2
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.2.1
scikit-learn==1.1.2
scipy==1.10.1
seaborn==0.12.0
six==1.16.0
threadpoolctl==3.1.0
```

```
Django>=4.0.1,<4.1
djangorestframework>=3.13.1,<3.14
psycopg2>=2.9.3,<2.10
drf-spectacular>=0.22.1,<0.23
Pillow>=9.1.0,<9.2
uwsgi>=2.0.20,<2.1
```

requirements.txt file examples

What do you think will happen here?

```
numpy>=1.15.3
h5py>=2.8.0
statsmodels>=0.9.0
scikit-learn>=0.20.1
tensorflow>=2.1.0
click
matplotlib
seaborn
tqdm
```

- Virtual Environments are crucial to have other people using your code

Let’s do it for our course!

- Create a requirements.txt file and store the libraries and their versions there so that we can use them to install the dependencies in our environment

Intro to Command Line

- Direct Interaction with Python Interpreter
- Running Python Scripts
- Installing and Managing Packages with pip
- Creating and Managing Virtual Environments
- Access to System Utilities and Files
- Running Unit Tests
- Version Control with Git
- Scripting and Automation

```
echo "Step 1: creating backup in VM unnoba"
ssh root@guarani-cisa.unnoba.edu.ar "pg_dump -h localhost -p 5432 -U dba -F c -b -v -f 'base.backup' guarani3"

echo "Step 2: Downloading the backup..."
scp root@guarani-cisa.unnoba.edu.ar:~/base.backup .

#Restore backups
echo "Step 3: updating the .pgpass file"
python write_pgpass.py
chmod 600 ~/.pgpass

echo "Step 4: clear database for restore..."
#Run script to update the files
python3 clear_database.py

echo "Step 5: Restoring the database..."
#Restore the databse
pg_restore --verbose --clean --no-acl --no-owner -h unnoba-anonimo-database.ctibhhezsjzw.us-east-1.rds.amazonaws.com

echo "Step 6: Running script for the ETL..."
#Run script to update the files
python3 ETL_database.py
```

Intro to Command Line

- Direct Interaction with Python Interpreter
- Running Python Scripts
- Installing and Managing Packages with pip
- Creating and Managing Virtual Environments
- Access to System Utilities and Files
- Running Unit Tests
- Version Control with Git
- Scripting and Automation

```
echo "Step 1: creating backup in VM unnoba"
ssh root@guarani-cisa.unnoba.edu.ar "pg_dump -h localhost -p 5432 -U dba -F c -b -v -f 'base.backup' guarani3"

echo "Step 2: Downloading the backup..."
scp root@guarani-cisa.unnoba.edu.ar:~/base.backup .

#Restore backups
echo "Step 3: updating the .pgpass file"
python write_pgpass.py
chmod 600 ~/.pgpass

echo "Step 4: clear database for restore..."
#Run script to update the files
python3 clear_database.py

echo "Step 5: Restoring the database..."
#Restore the databse
pg_restore --verbose --clean --no-acl --no-owner -h unnoba-anonimo-database.ctibhhezsjzw.us-east-1.rds.amazonaws.com

echo "Step 6: Running script for the ETL..."
#Run script to update the files
python3 ETL_database.py
```

Show Examples

esade

Topic 6: Python Scripting I

What is a Python Script?

- Scripts are small programs in Python that allow us to
  - **Automate** repetitive **tasks**.
  - **Integration** with web applications, databases, and even other programming languages.

```python
"""
Python script to introduce scripts
"""

import pandas as pd

Codeium: Refactor | Explain | Generate Docstring | ✕ | CodiumAI: Options | Test this function
def main():
    print('Hello')
    df = pd.read_csv('somedataset', sep=',')
    print(df.head())


if __name__ == '__main__':
    print('First I go here if you run me from the terminal')
    main()
```

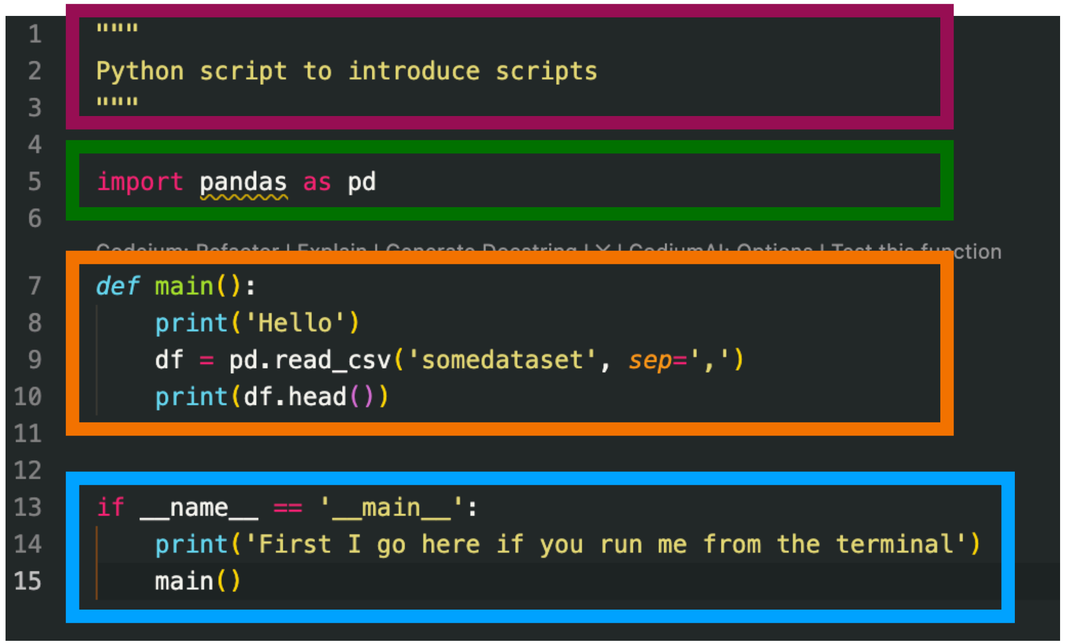
Why scripts better than JN? (Normally)

vs.

- **Reusability**: Reused and integrated into other applications or workflows.
- **Performance**: Faster and more efficient.
- **Version Control:** Scripts are more version-control friendly, making it easier to track changes and collaborate with others.
- **Deployment:** Scripts are more suitable for deployment in production environments.

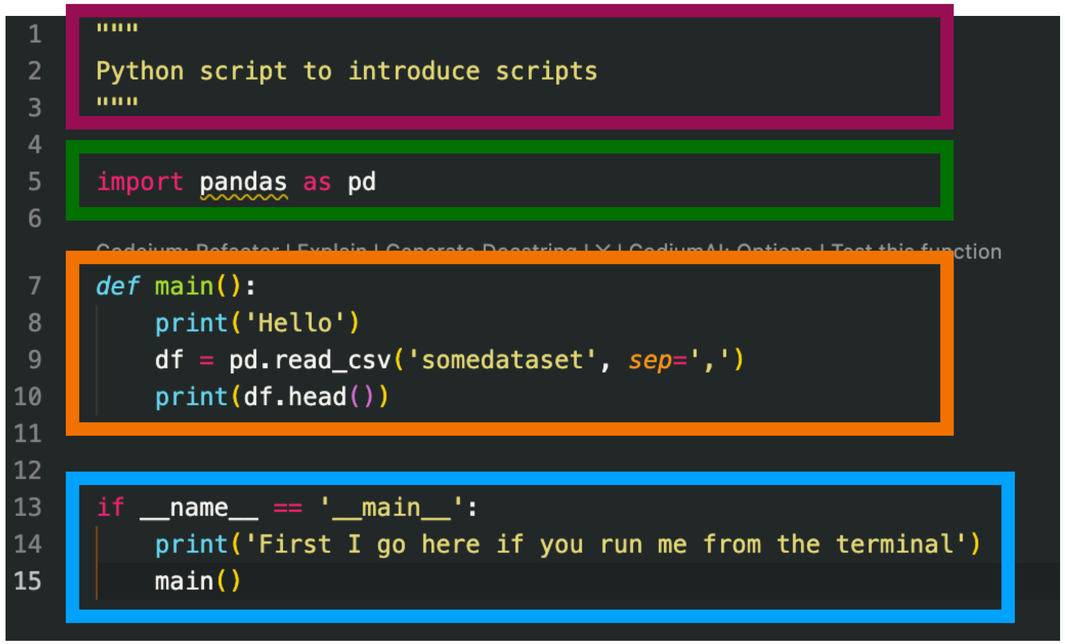
What do we need to start? The building blocks

- **Docstring:** Explains what we do in the script
- **Packages (Libraries)**: That we will use in the code
- **Main Function:** Where the code will run.
- **Python Idiom:** To check whether the script is run from the terminal



What do we need to start? The building blocks

- **Docstring:** Explains what we do in the script
- **Packages (Libraries)**: That we will use in the code
- **Main Function:** Where the code will run.
- **Python Idiom:** To check whether the script is run from the terminal



- **Let's build a Python Script**