

# PYTHON

## IA FRAMEWORKS

---

# CODE DEVELOPMENT FOR DATA SCIENTIST

## OBJECTIVE:

- Go further than data exploration and training model.
- Run code on real dataset with more computation power.
- Discover tool that you will need to know if you need to deploy model.

## A THREE PARTS [LAB](#).

- Write **python script**.
- Run code on **Google's Virtual Machine**.
- Run code on **Google's Virtual Machine** within **Docker container**.

# CODE DEVELOPMENT FOR DATA SCIENTIST

## OBJECTIVE:

- Go further than data exploration and training model.
- Run code on real dataset with more computation power.
- Discover tool that you will need to know if you need to deploy model.

## A THREE PARTS [LAB](#).

- —> WRITE **PYTHON SCRIPT**. <—
- Run code on **Google's Virtual Machine**.
- Run code on **Google's Virtual Machine** within **Docker container**.

# INTRODUCTION

VIRTUAL ENVIRONMENT

PYTHON SCRIPT

TP

# TOOLS

## ML Python Libraries



## Python Environment



## Viz' Python Libraries



## Framework & Tool



# VIRTUAL ENVIRONMENT

---

# MOTIVATIONS

**PROBLEM:** How to manage projects which require library at different versions ?

*I need, python 3.8 to run my AI Frameworks project, but my HDDL project is working perfectly on python 3.7.*

**SOLUTION:** Virtual Environment.

- Contains *python interpreter* and *python libraries* at a **fixed version**.
- **Various environments** coexist on the same computer with different interpreter and libraries.
- Installation and update of new libraries impact only the desired environment.

*You can have a AI Frameworks environment with python 3.8 and tensorflow-gpu and a HDDL environment with python3.7 and tensorflow-cpu.*

# HOW TO USE VIRTUAL ENVIRONMENT?

VARIOUS SOLUTIONS.

**VENV:** Native python solution ([Doc](#)).

- Works with **pip** the official package installer.
- Does not handle various python version

**VIRTUALENV:** Native python solution ([Doc](#)).

- Works with **pip** the official package installer.
- Can create virtual environment for arbitrarily installed python version.
- Community driven.

**CONDA:** A Continuum Analytics solution ([Doc](#)).

- Free solution.
- Works with both **Conda** and **pip** the official package installer.
- *Anaconda settings* for data scientist.
- Terminal or Graphic interface.



# CONDA

**Install** via anaconda [website](#) (available for Mac, unix, windows).

## Create environment

```
bguillou $> conda create -n EnvName python=x.x anaconda
```

- *-n EnvName*: Name your environment *EnvName*.
- *-python=x.x*: Which python version you want for your environment.
- *Anaconda*: Install all the the default anaconda's libraries

## Activate environment

```
bguillou $> conda activate EnvName  
(EnvName)bguillou $>
```

# CONDA

With default setting, anaconda folder is located at your user folder.

```
bguillou $> conda activate EnvName
(EnvName)bguillou $> ipython
In [1]: import sys
In [2]: sys.path
Out[2]: [ '/Users/username/anaconda3/envs/EnvName/bin' ..,
```

Every libraries can be install with conda..

```
(EnvName)bguillou $> conda install package-name
```

..or pip.

```
(EnvName)bguillou $> pip install package-name
```

# REQUIREMENTS.TXT

Install all anaconda's default libraries is practical but it's a lot of libraries.

Good practice is to list all required libraries on a *requirements.txt* file.

```
tensorflow-cpu==2.1  
pandas==1.0.5  
scikit-learn==0.23
```

All this libraries can then be installed within the environment at its creation.

```
bgui llou $> conda create -n EnvName --file requirements.txt
```

Or once the environment is created.

```
(EnvName)bgui llou $> pip install -r requirements.txt
```

# REQUIREMENTS.TXT

It's a *must have* when sharing projects *online*.

🔗 master ▾


🔗 1 branch

🏷️ 0 tags

Go to file


Add file ▾

📄 Code ▾

 **bguillouet** Readme

67cb1fa 15 minutes ago ⌚ 4 commits

📄 190107_Anomaly Detection on Time ...	Add files via upload	16 months ago
📄 README.md	Readme	15 minutes ago
📄 adits_evaluation.py	Add files via upload	16 months ago
📄 gt.csv	Add files via upload	16 months ago
📄 gt_final.csv	Add files via upload	16 months ago
📄 requirements.txt	Add files via upload	16 months ago

README.md 

## DefilnsaAirbus2020

---

### Requirements

---

```
conda create -n defiAirbus python=3.8
pip install -r requirements
```

# PYTHON SCRIPT

---

# WHY USING SCRIPT?

## *Jupyter*'s limits

- It's an exploration tool.  
*Cloud machine are accounted on an hourly base.*
- Non-Linear workflow.  
*Easy to write messy code.*
- Not designed to handle large-scale experiment.
- Not designed for production.  
*Can't be run from terminal, no test procedure.*

Exploration work:

*Jupyter*

Large-scale or production work:

Python script

# SCRIPT EXECUTION

File *script.py*

```
a = 3
b = 5
c = a + b
print("The answer is %d" % c)
```

Terminal

```
bguillou $> python script.py
bguillou $> The answer is 8
```

# PYTHON'S IDE

## **DOZENS OF SOLUTIONS !**

- Python oriented: pycharm, Spyder, idle, pydev etc..
- General: Visual studio, IntelliJ IDEA, Net Beans, eclipse etc. etc..
- At INSA : Spyder

## **GOALS**

- Write more readable code.
- Avoid errors and unused code.
- Speed up code writing (auto-completion, etc.)



TP

---

# TP – FIRST PART

Write two scripts:

- **learning.py**: learn a model, save it in the *model* directory, save *results* in the results directory.
- **prediction.py**: to generate prediction and save it in the *results* directory.

On **CatsVsDogs** data.

Good Practice:

- Ensure that **complete workflow** is **working locally** and on **small data**.

# LIBRAIRIE ARGPARSE

File *script.py*

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--a', type=int, default=5)
parser.add_argument('--b', type=int, default=3)

args = parser.parse_args()

c = args.a + args.b
print("The answer is %d" % c)
```

Terminal

```
bguillou $> python script.py
bguillou $> The answer is 8
bguillou $> python script.py --a 4
bguillou $> The answer is 7
bguillou $> python script.py --a 4 --b 2
bguillou $> The answer is 6
```

# LIBRAIRIE PICKLE

File *script.py*

```
import pickle
...
results = {"learning_time" : lt, "accuracy" : acc}
pickle.dump(results, open("/User/bguillouet/data/results.pkl", "wb"))
```

File *explore\_results.py*

```
import pickle
results = pickle.load(open("/User/bguillouet/data/results.pkl", "rb"))
print(results)
```

Terminal

```
bguillou $> ls data/
bguillou $>
bguillou $> python learning.py
bguillou $> ls data/
bguillou $> results.pkl
bguillou $> python explore_results.py
bguillou $> {"learning_time": lt, "accuracy" : acc}
```