POLITECNICO
MILANO 1863

ISPL
Image and Sound Processing Lab

Introduction to Python

Table of contents

- **Introduction**
- **Tools**
  - Conda
  - Jupyter Notebook
  - Google Colaboratory
- **Main concepts**
- **Hands-on (notebook)**
  - Standard Python
  - numpy
  - matplotlib

Introduction

Why Python?

- **Easy to learn!**
- **Easy to read**

  Almost like writing a series of instructions

- **High-level language**

  Object oriented

- **Interpreted language**

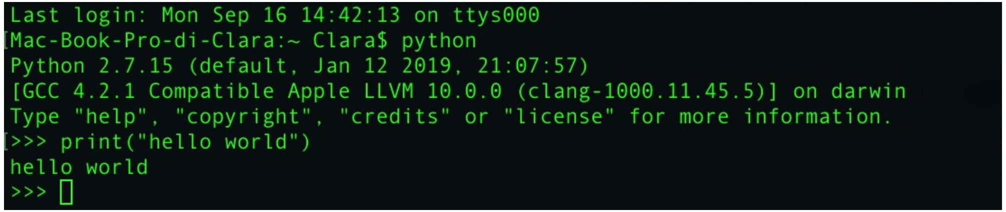
  No need to compile

- **Multi-platform**

  Linux, Windows, OSX

Python Execution

- **Interactive Mode**
  - Run the command `python` in terminal
  - Write and run sequentially each operation separately



```
Last login: Mon Sep 16 14:42:13 on ttys000
[Mac-Book-Pro-di-Clara:~ Clara$ python
Python 2.7.15 (default, Jan 12 2019, 21:07:57)
[GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print("hello world")
hello world
>>> 
```

- **Non-interactive Mode**
  - Create a file containing your code, e.g., `my_code.py`
  - Run the code using the command `python my_code.py`

Tools

Package Managers

- **Packages and Modules**
  - Python can be easily expanded through packages and modules
  - Popular libraries: scipy, numpy, scikit-learn, librosa …

- **PIP**

```
pip install numpy
```

- **Conda**

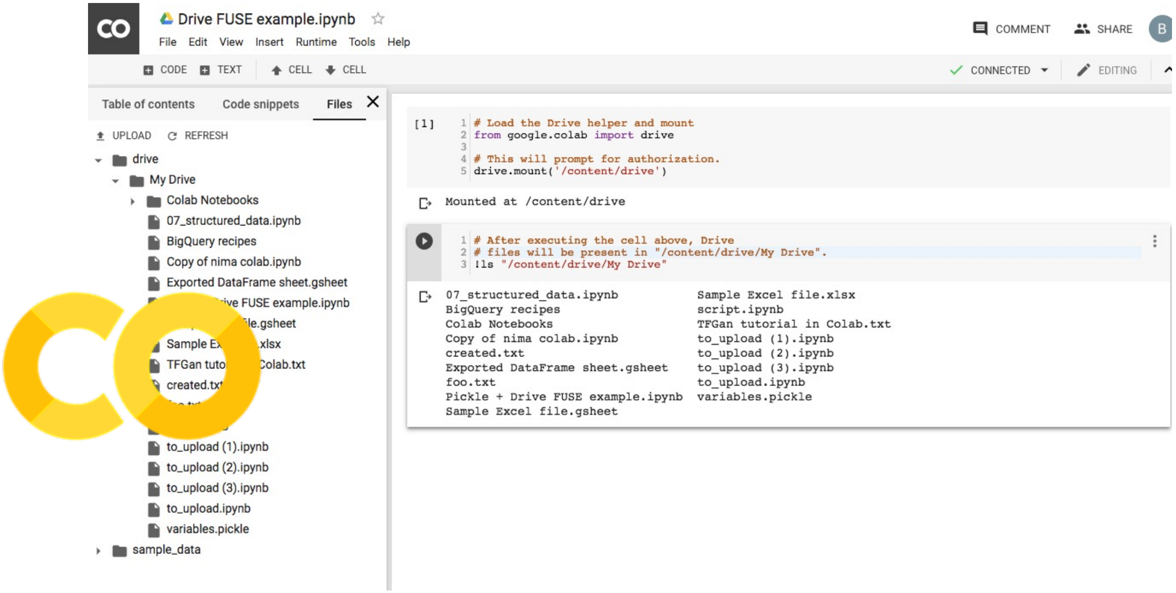
```
conda install numpy
```

- Others…

Python Tools

For the next labs you can:

- choose to install Python and its tools locally on your computer (suggested)
- use Google Colaboratory



```
# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# After executing the cell above, Drive
# files will be present in "/content/drive/My Drive".
!ls "/content/drive/My Drive"
```

```
07_structured_data.ipynb           Sample Excel file.xlsx
BigQuery recipes                   script.ipynb
Colab Notebooks                    TFGan tutorial in Colab.txt
Copy of nima colab.ipynb           to_upload (1).ipynb
created.txt                        to_upload (2).ipynb
Exported DataFrame sheet.gsheet    to_upload (3).ipynb
foo.txt                            to_upload.ipynb
Pickle + Drive FUSE example.ipynb  variables.pickle
Sample Excel file.gsheet
```

Install Python locally: Conda

Conda is an open source package management system and environment management system:

- compatible with Windows, MacOS, Linux
- install, run and updates packages and their dependencies
- creates, saves, loads and switches between environments
- created for Python but it can package and distribute software for any language

Install Python locally: Miniconda

Miniconda is a small light version of conda. It includes conda (the package manager), Python (the interpreter) and some additional packages.

https://docs.conda.io/en/latest/miniconda.html

Note: we will use Miniconda to install Python too (it is included!).

If you have previous versions of Python they will not be affected by the Miniconda installation.

Install Python locally: Miniconda Mac OS

1. Download **Miniconda3 Python 3.8 pkg version (not bash)** for MacOS
2. Run the installer
3. Open Terminal (or similar)
4. Write and run `conda update conda`
5. if you see something like `(base)` at the start of the line in your Terminal, then it means that the base environment of Conda is activated by default on startup. If you want to disable this, run:

```
conda config --set auto_activate_base false
```

Note: the installation path will be something like `/Users/name/opt/miniconda3`

Every time you will need to use Conda, you will need to open the Terminal application

Install Python locally: Miniconda Windows

1. Download **Miniconda3 Python 3.8** for Windows
2. Run the installer leaving default options
3. When it is finished, you will have a new application called **Anaconda Powershell Prompt**
4. Open Anaconda Powershell Prompt and run `conda update conda`

Note: everytime we will use Python and Conda you will need to use this prompt

Install Python locally: Create an environment

What is a conda environment and why is it useful?

Using conda, you can create an isolated python *environment* for your project. An environment is a set of packages that can be used in one or multiple projects

We can create a conda environemnt in two ways:

- manual specifications of packages
- specify environment file in YAML format

Install Python locally: Environment manual creation

```
conda create -n name_of_the_environment python=3.8
```

you can specify a version of Python, install at creation time some specific packages with specific versions of the package

Install Python locally: Creation from an yml file

- define an **yml** file (we are not going to do this today)
  https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#create-env-file-manually
- to create an environment from an yml file run simply

```
conda env create -f environment.yml
```

Install Python locally: Activation of the environment

For using the environment you will first need to activate it

```
conda activate name_of_the_environment
```

To de-activate the environment and return to your basic shell

```
conda deactivate
```

⚠ Everytime we start a lab session we will need to activate the environment.

Install Python locally: Install packages with conda

When you are **inside an environment** you can install packages using

```
conda install name_of_the_package
```

These packages are valid and will be accessible only from this environment.

Try to install some packages in the new created environment.

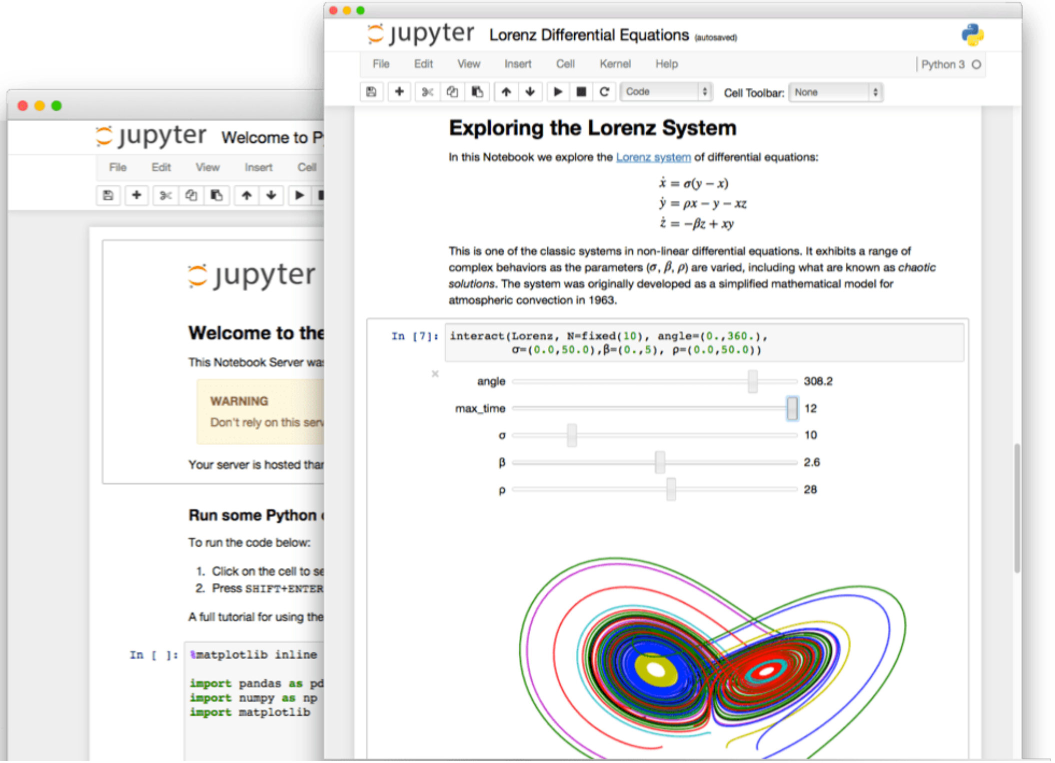
First activate the environment and then install:

- `numpy`
- `scipy`
- `jupyter notebook`

Install Python locally: IDE and Editors

Jupiter Notebook

is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.



jupyter Welcome to P
File Edit View Insert Cell
jupyter

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted tha

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2

max_time 12

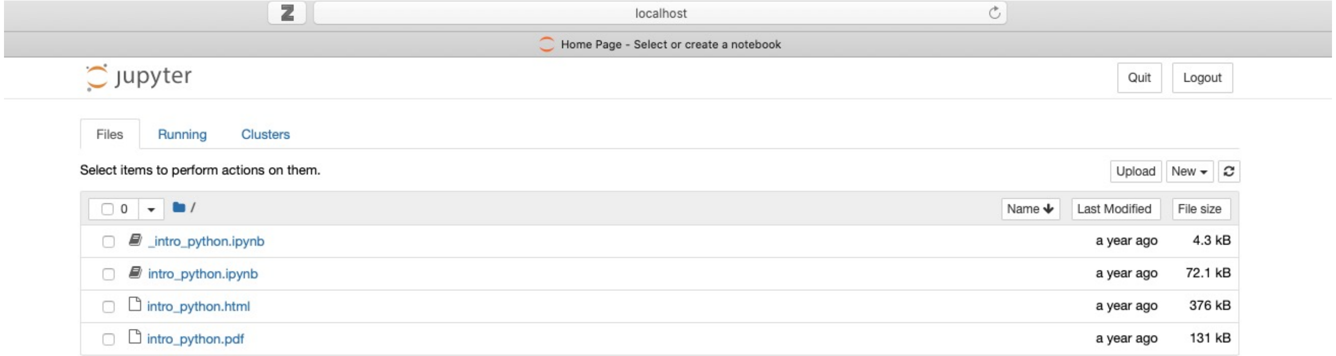
σ 10

β 2.6

ρ 28

Install Python locally: Jupyter Notebook

When you type `jupyter notebook` in your conda env →



Select items to perform actions on them.

Name	Last Modified	File size
_intro_python.ipynb	a year ago	4.3 kB
intro_python.ipynb	a year ago	72.1 kB
intro_python.html	a year ago	376 kB
intro_python.pdf	a year ago	131 kB

A new window in a browser will be opened automatically.

The windows is the user interface that access to Jupyter Notebook server on the address `localhost:8888` (usually, you can check it on the output in the terminal).

All the notebooks present in the folder **from which you launched the command** are visible and executable.

Install Python locally : Instructions for the LABS

During these labs:

- I will upload the needed notebooks and files on BeeP portal as a ZIP file. Usually it will be something like:

```
|--notebook1.ipynb
|--notebook2.ipynb
|--audio
|   |--audio1.wav
|   |--audio2.wav
|--img
|   |--img1.png
|   |--img2.png
```

Do not change this hierarchy! We will need it for loading audio files and images in our notebooks!

Python locally : Instructions for the LABS

- Download and unzip the ZIP file
- Navigate with the Terminal/Anaconda Powershell Prompt in the folder in which you have the lab folder
- Run `conda activate name-of-the-environment`
- Run `jupyter notebook`
- The notebooks I will provide you are partially empty: together, during the labs, we will write and run the code for completing them
- If you need a package in your code, you can always install it by calling `conda install name_of_the_package` inside your environment

Python on the web: Google Colab

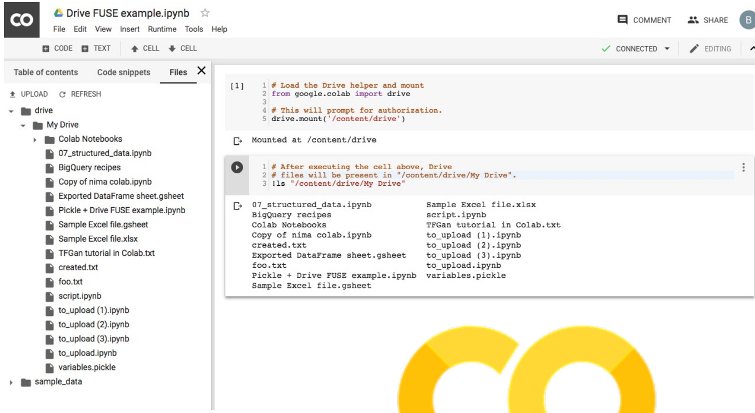
- **Google Colaboratory**

allows you to write code and test it directly on the browser. It is, in practice, a Jupyter Notebook which runs on remote servers of Google.

The notebook are automatically saved on your Google Drive.

Drawbacks:

- the service is not always guaranteed and the session expires after some time
- loading external files can be complex



[1]

```python
# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
# After executing the cell above, Drive
# files will be present in "/content/drive/My Drive".
!ls "/content/drive/My Drive"
```

```
07_structured_data.ipynb          Sample Excel file.xlsx
BigQuery recipes                  script.ipynb
Colab Notebooks                   TFGan tutorial in Colab.txt
Copy of nima colab.ipynb          to_upload (1).ipynb
created.txt                       to_upload (2).ipynb
Exported DataFrame sheet.gsheet   to_upload (3).ipynb
foo.txt                           to_upload.ipynb
Pickle + Drive FUSE example.ipynb variables.pickle
Sample Excel file.gsheet
```

Python on the web: Google Colab

- open the link https://colab.research.google.com/notebooks/intro.ipynb
- to create an empty notebook click on File / Create New Notebook (Note: you have to be logged in Google)
- to load a notebook (like the one I will provide you for the labs) use File / Load New Notebook
- write and execute you code
- if you need an external package not already included use

  ```
  !pip install name_of_the_package
  ```

  https://colab.research.google.com/notebooks/snippets/importing_libraries.ipynb
- if you need external file use the left side panel and load them directly; in alternative check this https://colab.research.google.com/notebooks/io.ipynb

Python on the web: Instructions for the LABS

- Download and unzip the ZIP file
- Load the notebooks on Google Colab
- Install necessary packages not included
- Load the audio and images files in the Google Colab session, keeping the hierarchy

Main Concepts

Comments

- **Text ignored during execution**
  - Useful to describe what is happening in the code to the reader

- **Comments start with character #**
  - … till the end of the line
  - Character # can still be used in strings

```
# this is the first comment
SPAM = 1                  # and this is the second comment
                          # ... and now a third!
STRING = “# This is not a comment”.
```

Numbers

- In interactive mode, Python prints operation outputs
- Common operators are *, +, -, /
- Characters (and) can be used to group operations
- **Warning**: Python2 vs. Python3 integer division!!

```
>>> 2*2
4
>>> (50-5*6)/4
5
>>> 7/3    #integer division returns the floor
2
>>> 2**3   #exponentiation
8
```

Variables

- It is possible to associate values to variables
- Python exploits **dynamic typing**
    - No need to declare variable type (e.g., int, float, char, etc.)
- **Warning**: do not confuse = (i.e., assignment) and == (i.e., comparison)

```
>>> a = 20
>>> b = 5*9
>>> a * b
900
```

Strings

- String is a structure containing **text data**
- Declared using ‘ or “
- Escape character \ is used to include ‘ or “ within strings
- print can be used to show string content
- **Warning**: Python2 vs. Python3 print syntax

```
>>> print(‘spam eggs’)
spam eggs
>>> print(‘doesn\’t’)
doesn’t
>>> print(“doesn’t”)
doesn’t
>>> print(‘”Yes,” he said.’)
”Yes,” he said.
>>> print(“\”Yes,\” he said.”)
“Yes,” he said.
>>> print(‘”Isn\’t,” she said.’)
“Isn’t,” she said.
```

List and tuples

- **Lists are sequences of editable variables**
- Lists can be concatenate, sorted, re-ordered, etc.
- Declared using [and]

```
>>> a = [‘spam’, ‘eggs’, 100, 1234]
>>> a
[‘spam’, ‘eggs’, 100, 1234]
>>> a[0]
‘spam’
>>> a[1:3]
[‘eggs’, 100]
>>> a[:2] + [‘bacon’, 2*2]
[’spam’, ’eggs’, ‘bacon’, 4]
>>> a.append(9.87)
>>> a
[’spam’, ‘eggs’, 100, 1234, 9.87]
```

- **Tuples are non-editable**
- Declared using (and)

Control Structures

- **If / Else**

  use operators <, >, ==, !=

```
>>> x = 5
>>> if x < 0:
...     print(’negative’)
...  elif x == 0:
...     print(‘zero’)
...  else:
...     print(‘positive’)
...
positive
```

- **For**

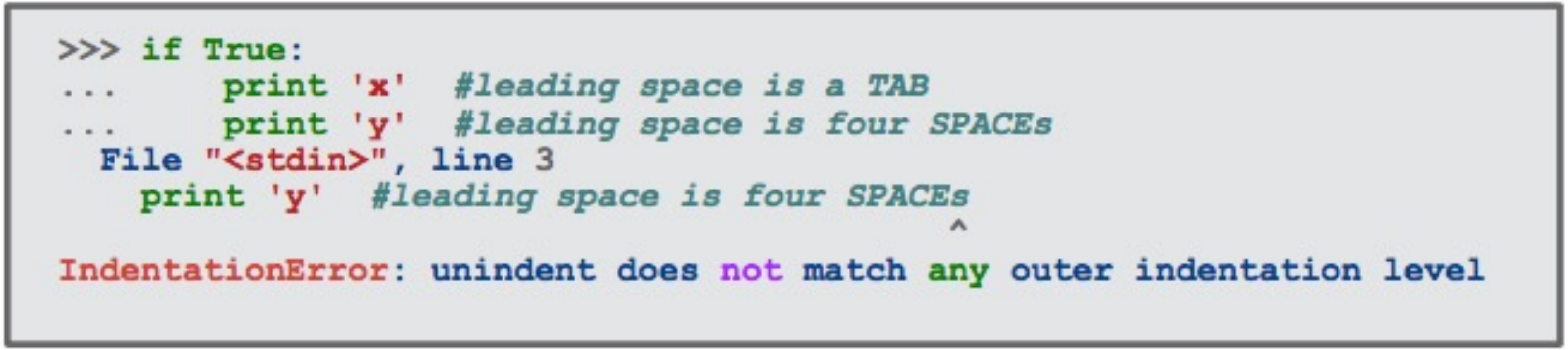
  hints: ‘zip’, ’enumerate’,

  ‘tqdm’

```
>>> # Measure some strings
...  a = [‘cat’, ‘window’, ‘defenestrate’]
>>> for x in a:
...    print(x, len(x))
...
cat 3
window 6
defenestrate 12
```

Indentation

- **Python uses indentation to group code portions**
  - Hard constraint!
- Indentation must be uniform
  - Use the same amount of tabs or spaces for each group of codes



```
>>> if True:
...     print 'x'  #leading space is a TAB
...     print 'y'  #leading space is four SPACEs
  File "<stdin>", line 3
    print 'y'  #leading space is four SPACEs
                                            ^

IndentationError: unindent does not match any outer indentation level
```

Functions

- Defined using constructor ‘def’
- Can return multiple variables as well as lists
- **Warning**: risk of changing variable values outside functions

```
>>> def power (base, exponent):
...    return base**exponent
...
>>> power(2,3)
8
>>> power(2,4)
16
>>> range(4)  #Standard Python function
[0, 1, 2, 3]
```

Modules

- Files containing multiple functions and instructions
  - also known as libraries
- Python contains some standard modules (e.g., os, math, sys, etc.)
- Modules must be imported before use (i.e., import …)
  - It is possible to import submodules (i.e., from … import …)

```
>>> import os
>>> os.getcwd()
‘/Users/Clara/’
>>> from math import factorial
>>> factorial(5)
120
```

Object-oriented Programming

- It is possible to use “special” variables that contains functions and other variables
- These are generated starting from “classes”

```
class rettangolo:

    def __init__(self, l1, l2):
        self.l1 = l1
        self.l2 = l2

    def area(self)
        return self.l1*self.l2
```

```
>>> rett1 = rettangolo(2,5)
>>> rett1.area()
10
>>> rett1 = rettangolo(4,4)
>>> rett1.area()
16
```

Hands on