Python: A "brief" introduction

Day 1

Álvrarez Morales, Gonzalo

UPM
Ingemat

21 11 2022

Outline

1. Why Python?
2. Variable types
3. Scripts
4. Classes
5. External files

Outline

1 Why Python?

2 Variable types

3 Scripts

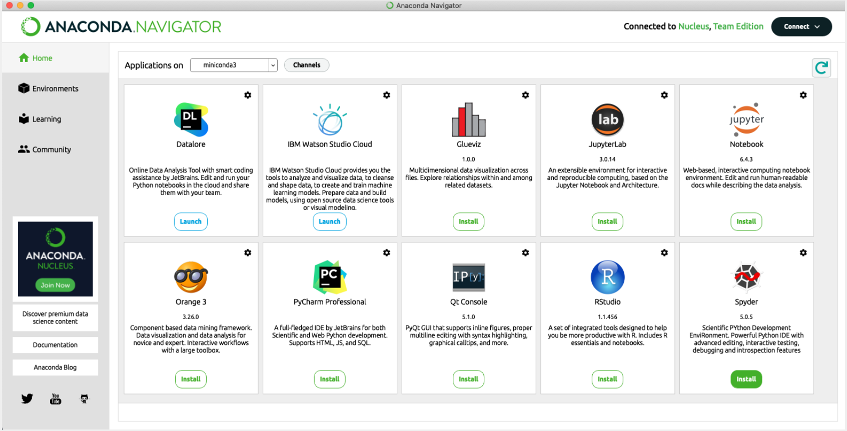
4 Classes

5 External files

Why Python?

- Python es an interpreted language.
- Is Open-Source.
- Easy and beginner-friendly
- Widely usage
- Versatility
  - Plenty of documentation
  - Multiple selection of developed libraries
- Compatible with several other languages.

Environments



Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

Launch

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

Launch

Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install

JupyterLab

3.0.14

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Install

Notebook

6.4.3

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Install

Orange 3

3.26.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install

PyCharm Professional

A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

Install

Qt Console

5.1.0

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Install

RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install

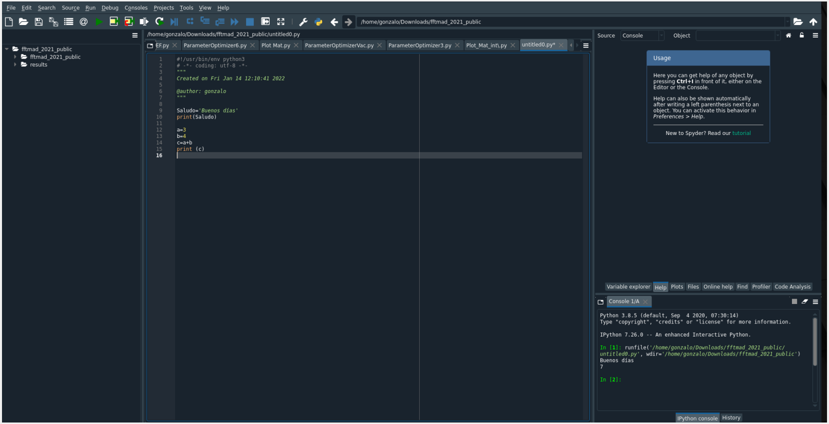
Spyder

5.0.5

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Install

Spyder



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 14 12:10:41 2022

@author: gonzalo
"""

Saludo='Buenos días'
print(Saludo)

a=3
b=4
c=a+b
print (c)

```

Usage

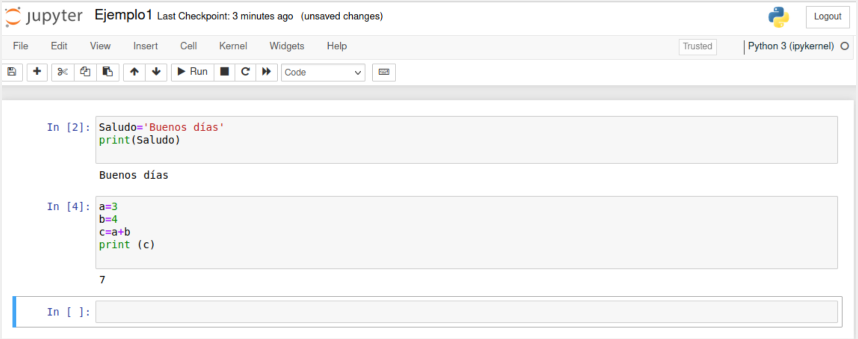
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

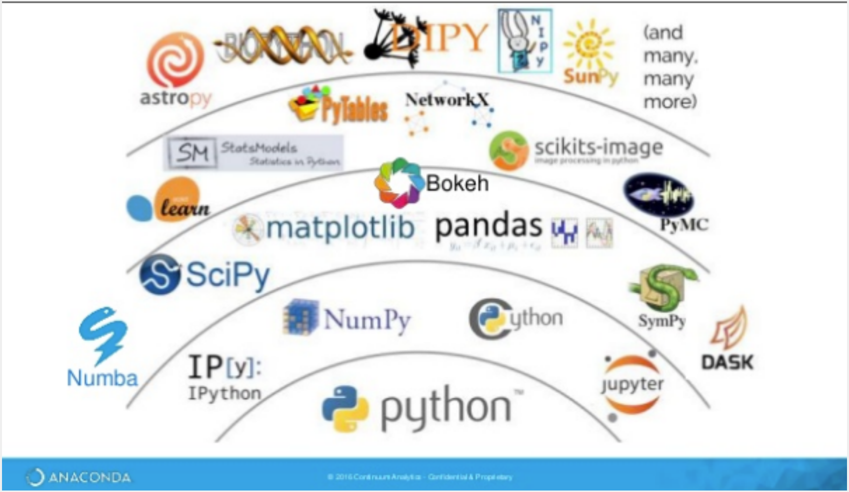
```
Python 3.8.5 (default, Sep  4 2020, 07:30:14)
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('/home/gonzalo/Downloads/fftmad_2021_public/
untitled0.py', wdir='/home/gonzalo/Downloads/fftmad_2021_public')
Buenos días
7

In [2]:
```

Jupyter Notebook



```
In [2]: Saludo='Buenos días'
        print(Saludo)

        Buenos días

In [4]: a=3
        b=4
        c=a+b
        print (c)

        7

In [ ]:
```

LIBREARIES


astropy
BIOPYTHON
DIPY
NIPY
SunPy
(and many, many more)
PyTables
NetworkX
StatsModels Statistics in Python
scikits-image image processing in python
Bokeh
learn
matplotlib
pandas
PyMC
SciPy
NumPy
Cython
SymPy
Numba
IP[y]: IPython
python
jupyter
DASK
ANACONDA
© 2016 Continuum Analytics - Confidential & Proprietary

Outline

1 Why Python?

2 Variable types

3 Scripts

4 Classes

5 External files

Numeric

- Integers
  - Signed or Unsigned: int_ / uint_
  - Bit-size (8,16,32,64)
    e.g. int8, int32, uint64
- Floating point, float
  - Reals or complex: float_ / cfloat_
  - Bit-size (8,16,32,64)

Other individual variables

- Booleans (bool)
- Flexibles
  - Void
  - Characteres
    - String
    - Unicode

"\N{Character_name}"	"\N{GREEK CAPITAL LETTER DELTA}"
"\u+16-bit hex value"	"\u0394"
"\u+32-bit hex value"	"\u00000394"

Collections

In python, indexing begins at 0 (zero index language)

Types of collections	Ordered	Non ordered
Mutable	List	Dictionary
Immutable	Tuple	Set

+

numpy.arrays

Ordered and mutable list or homogeneous object both in kind and size. They can have any dimensionality and are the core of numpy library.

Colección	Declaración
List	L=[item_0, item_1, item_2, ... item_n]
Tuple	T=(item_0, item_1, item_2, ... item_n)
Dictionary	D={ Key : item, ...}/ dict({k:i,...})/ dict([(k, i), $\ldots$])
Set	S=set ([item_0, item_1, item_2, ... item_n])

Outline

1. Why Python?
2. Variable types
3. Scripts
4. Classes
5. External files

Scripts

Scripts are sequential list of instructions (Exception function declaration)
Most common structures:

- Conditionals
- Loops
- Functions

In Python, structures are not ended by an specific command such as end. They are controlled by indentation:
All the inside of a block is one level deeper than its declaration, and the block ends after the ending of the indentation.
Blocks can be nested by indenting to deeper levels.

Conditionals

Answer to the following structure:

Condition_i must be a boolean

```
if (Condition_0):
    Set of instructions to follow if Condition_0 is True
elif (Condition_1):
    Set of instructions to follow if Condition_1 is True but all condition_< 1 were False
elif (Condition_2):
    Set of instructions to follow if Condition_2 is True but all condition_< 2 were False
else :
    Set of instructions to follow if none of the previous conditions were True
```

Loops

For

For loops are used when some instructions should be repeated a known fixed number of times. The loop is repeated by replacing a dummy variable by the successive elements of a list, resulting in a total amount of iterations equal to the length of said list.

While

While loops are used when some instructions should be repeated an unknown number of times, until some criterion is fulfilled.
The while loop will be repeated while an initial condition is True
The initial condition should be an operation with a boolean as its result.
Careful, While statements may loop forever

Loops

For

```
for i in L:
    $0^{th}$ instruction to repeat
    $1^{st}$ instruction to repeat
    $2^{nd}$ instruction to repeat
    $3^{rd}$ instruction to repeat
Instructions outside of the loop
```

L=$[v_0, v_1, v_2, v_3, v_4, ..., v_n]$

While

```
while Condition:
    $0^{th}$ instruction to repeat
    $1^{st}$ instruction to repeat
    $2^{nd}$ instruction to repeat
    $3^{rd}$ instruction to repeat
Instructions outside of the loop
```

Condition should be result of a logical operation (boolean)
i.e.: while error$<$1e-8
error variable is updated every iteration inside the loop

Functions

A function is an encapsulated block of code to be executed when called inside another code by its name.

Internal variables inside a function are removed after ending the function.

Declaration of a function is separated from its calling

Declaration:

```
def function($input_0$, $input_1$, $input_2$, ..., $input_n$)
    Function instructions
    Function instructions
    Function instructions
    Function instructions
return ($output_0$, $output_1$, $output_2$, ..., $output_m$)
```

Calling:

O_0, O_1, O_2, ..., O_m= function(I_0, I_1, I_2, ..., I_n)

Scope

Variables with the same name cannot coexist in the same place at the same time, as the new definition would overwrite the previous statement.
However variables with different scopes can coexist: i.e.

Example code

```
x = 5
def foo():
    x = 10
    print("local x:", x)
foo()
print("global x:", x)
```

Output:

```
local x: 10
global x: 5
```

Outline

1 Why Python?

2 Variable types

3 Scripts

4 Classes

5 External files

Classes/Objects

Python is an object oriented language.
Classes are Python’s objects.

Code:

```
class Person:
    def__init__(self, name, age):
        self.name = name
        self.age = age
    def myfunc(self):
        print("Hello my name is " + self.name)
p1 = Person("John", 36)
p1.myfunc()
```

Output:

```
"Hello my name is John"
```

Outline

1 Why Python?

2 Variable types

3 Scripts

4 Classes

5 External files

External files

To access an external file in Python use the command open():
open('file_name.extension','options')
'file_name' should include the path to the file if said file is not in the same folder as the running code.
Available options are:

- Modes
  - Reading: 'r'
  - Writing: 'w'
  - Appending: 'a'
  - Creating: 'c'
- Reading/writing format
  - Plain text 't'
  - Binary 'b'

By default, options 'r' & 't' are considered
i.e. f = open("demo.txt", "rt") is the same as f = open("demo.txt")

External files

To close the file use the command close()

i.e. f.close()

To read the lines of a file use the command readlines(n).

The command readlines(n) reads the first n lines of a file and returns a list of strings with the content of each line as each entry.

i.e. fl=f.readlines()   by default n=-1 (reading up to the last line)

To process data stored in special formats some libraries have been developed

i.e. pandas is a great library to handle data stored in .csv files.

Thank you for you attention!

All related information can be easily found with a brief search over the internet.

Only the very basics have been covered, you are expected to further develop the contents to suit your individual goals.

If you have any question please don’t hesitate to ask

On the next session:

- Hoe to plot data and make figures **Matplotlib.pyplot**
- Deal with data files (.csv, excel, ...) **Pandas**
- Data treatment for an engineering problem.