Brief introduction to Python

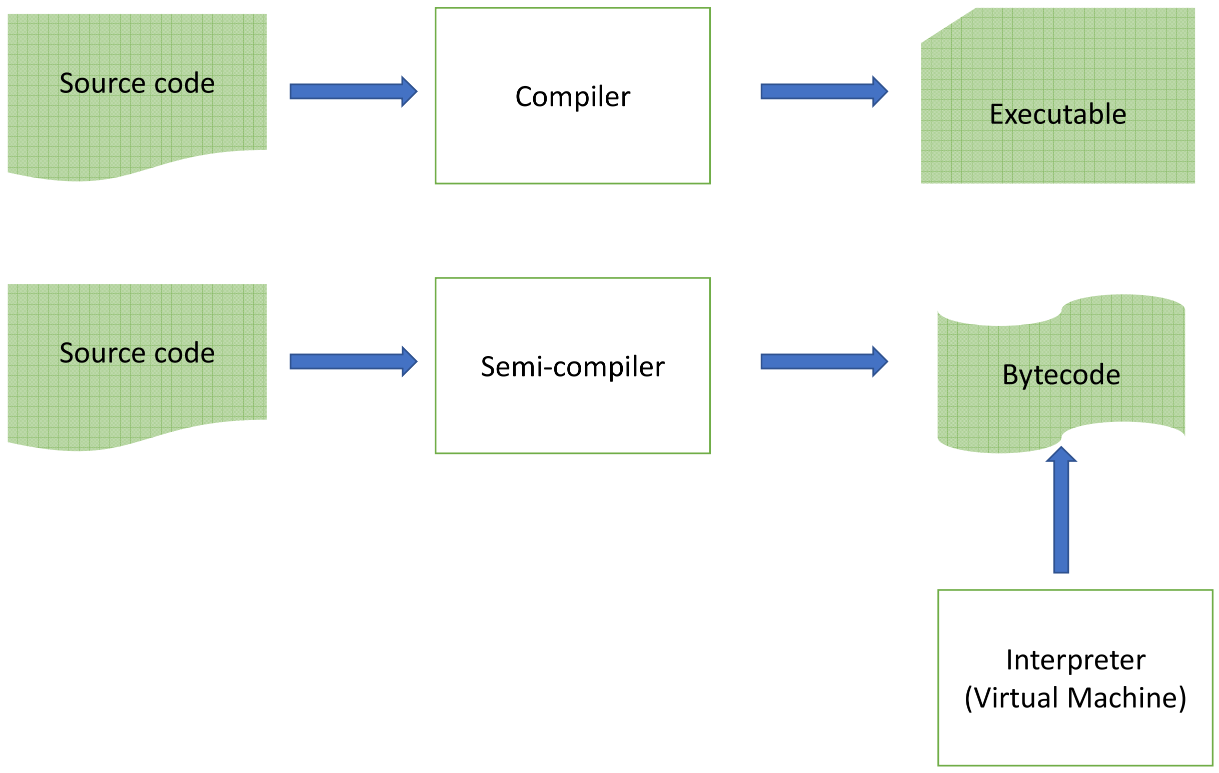
Author: Michele Moro UPD Version 1.0 November 2021

1. What's Python

1.1. The language and its executor

Python is a general-purpose programming language initially developed by Guido van Rossum during the 90's. It aims at being dynamic, simple, easy to read, flexible, modern in the sense that it allows forms of object oriented programming, structured programming, functional programming and implements other interesting paradigms. One aspect that distinguishes Python from other widespread languages is that it does not associate a fixed data type to a variable: as a consequence, a form of dynamic strong typing is applied (i.e. an object type is checked only at runtime). The runtime includes also a garbage collector to recover memory which is no longer referred by any variable of the running program.

Python can be considered an *interpreted* language: this means that a given instruction can be immediately recognized and executed by a sort of virtual machine (the executor). For the sake of efficiency, the high level source code is actually first translated into a more synthetic *bytecode* and then sent to the executor. This translation is essentially transparent to the user, so the impression is that an instruction is immediately executed and no form of compilation appears to be required before running the program.



1.2. Python's strength points

- General purpose: not just for scientific applications
- Portable: it runs on all major PC OS and also on small hardware like Raspberry PI and even smaller microcontrollers (there are special reduced versions like micropython)
- Accessible to the beginners but powerful for advanced users

- Simple syntax: not too many keywords and indentation used for block delimitation (instead of the more common use of curly brackets or pairs of keywords)
- Integrating modern approaches such as Object oriented programming, exception handling, interpretation through a virtual machine using runtime checks and a garbage collector, etc.
- Promoting the development of a wide range of libraries and applications: effective use of Python for Mathematics, Statistics, Astronomy, Climate, GIS, Machine learning and also for controlling small and big devices for special uses
- Free and open source: this leads to a huge community of practitioners and developers

2. How to start

2.1. Installation

The Python basic package can be downloaded from www.python.org python™ and easily installed on your machine (detailed instructions on another document). It is suggested to use the 'pip' tool to subsequently install other library modules requested by your applications. The most common ones are:

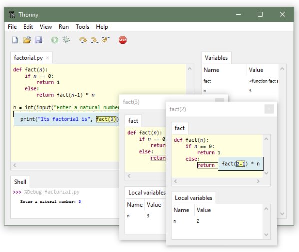
- *matplotlib*, for plotting features (which includes also numpy, core library for scientific computing)
- *pandas*, it provides ready to use high-performance data structures and data analysis tools
- *statistics*, for simple statistics
- *scipy*, it provides algorithms for optimization, integration, interpolation, linear algebra elaborations, algebraic equations, differential equations, statistics and many other classes of problems

2.2. Tools

When you install a version of the Python package from python.org, you install the interpreter and IDLE, its Integrated Development and Learning Environment. This latter includes a command-line based interface through which you can immediately try to execute Python instructions and see their effects, and a text editor to prepare, save programs in source code text files, and run them. The standard extension for these files is .py.

In order to have more advanced editing features it is possible to get other IDE (Integrated Development Environment), some of them freely usable for not commercial purposes. The more sophisticated of them are particularly useful for developing large projects and help to follow and support the entire process of development. A partial list of such IDE's follows:

- Thonny (www.thonny.org)



```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number
    print("Its factorial is", fact(3))
```

```
>>> %Debug factorial.py
  Enter a natural number: 3
```

Name	Value
fact	<function fact a
n	3

- Wing (www.wingware.com)

Wing Python IDE was designed from the ground up for Python, to bring you a more productive development experience.

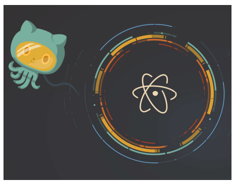
Current version: 8.1.0

- PyCharm Edu (www.jetbrains.com/pycharm-edu/)

- Visual Studio Code (https://code.visualstudio.com/)

- Vim (www.vim.org)

- Atom (https://atom.io/)



- Eclipse + PyDev + LiClipse (www.eclipse.org www.pydev.org www.liclipse.com)

- GNU Emacs (www.gnu.org/software/emacs/)

- Spyder (component of Anaconda www.spyder-ide.org)

2.3. Documentation

- The official Python language documentation and tutorial are available on docs.python.org/3/
- A language reference by w3schools https://www.w3schools.com/python/python_reference.asp
- Another language reference https://python-reference.readthedocs.io/en/latest/
- A language reference card https://perso.limsi.fr/pointal/_media/python:cours:abregepython-english.pdf
- Various useful resources https://www.reddit.com/r/learnpython/wiki/index

3. Python and Jupyter Notebook

3.1. What's Jupyter Notebook

The Jupyter Notebook is an incredibly powerful tool: differently from other teaching approaches, it allows to present tutorials, examples, exercises, even complex projects in an intuitive and interactive way. You can combine in just one place description/narrative text, mathematical equations, runnable code, visualizations, and other media. By integrating code and its output into a single document, the learner is helped to fully understand the presented content without continuously switching from one tool to another, making the explanations more understandable, repeatable, and shareable.

3.2. How to run a notebook in the Python environment

A notebook is namely a file with extension *.ipynb* and must be interpreted by a server. The server can run on your own machine. For example, if you have installed the Anaconda Python platform (www.anaconda.com/products/individual), you got also the Jupyter notebook software; on Windows Anaconda adds a shortcut to run the server.

If you have Python and other modules already installed, as suggested above, you can use again the pip tool to add the *jupyterlab* module to your system. Then, on a terminal window (command prompt), you give the command:

jupyter notebook

to run the server. When started, it activates your predefined browser to connect to the address http://localhost:8888 which is the port connected to the server. In the Files tab of the showed page you see your working directory, so you can navigate through your folders to select a notebook (a .ipynb file) to run and/or edit. The Running tab lists all the active (running) notebooks, while the Clusters tab is not of our current interest.