Introduction to Python

Andrea Esuli

What is Python?

Python is a programming language created by Guido van Rossum in 1991.

It is a high level, interpreted, dynamic typed, strong typed, garbage collected language, supporting many programming styles (imperative, functional, procedural, object oriented programming).

The name comes after the Monty Python Flying Circus.

Why Python?

Python has a simple, clean syntax. It's easy to learn.

The type system doesn't get in the way of coding, if not required.

Python has rich standard libraries and a huge amount of packages:

- Builtin data types for numbers, strings, lists, tuples, sets, dictionaries
- Strong numeric processing capabilities, with support to fast CPU/GPU parallel processing.
- State-of-the-art packages for NLP, data processing, statistical machine learning, deep learning.

It recently had a fast growth and it is currently among the most used and most appreciated languages.

Which Python version should I use?

3

Which Python version should I use?

"But… I already have code written in Python 2"

Although Python 3 is not fully backward compatible with Python 2, there are few differences and a key aspect in favor of Python 3:

- **Strings, are not just bytes sequences, they are Unicode sequences.**
  - **This is huge improvement for us working with text!**

*"Python 2.x is legacy, **Python 3.x is the present and future of the language**"*

  - Python 3 has been first released in 2008 (3.4 in 2014), it is not a recent novelty.
  - Python 2 had its last release, 2.7, in 2010, since then it is on end-of-life support.

Instagram has moved its 400M users platform from Py2 to Py3

Installation

Installation

The open source reference implementation of python is available from the python foundation.

However, I warmly suggest you to install the Anaconda distribution.

Anaconda can be installed without super user privileges, and it does not conflicts with existing python installations.

If you use Windows, anaconda solves many issues with native compilation of C,C++ portions of code that may be part of packages, specially ML ones.

The **conda** management tool for environments and packages is simple to use, and it provides **precompiled** packages for many platforms.

Installation - environments

Environments allow to have multiple, distinct, independent installations of Python, each one with its selection of installed packages:

```
>conda create -n ta python
```

In this way you can manage a dedicated setup for each of your projects. Messing up one environment does not affects the others.

When you want to use an environment you **activate** it:

```
mac/linux>conda activate ta
windows>activate ta
```

Installation

The conda command can be used to install/remove packages:

```
>conda install nltk scikit-learn matplotlib gensim keras
 beautifulsoup4 pandas
```

When a package is not available from the main anaconda repository, it may be installable from dedicated channels:

```
>conda install pytorch -c pytorch
```

Otherwise it can be installed using the pip tool, the <u>standard package manager for python</u>:
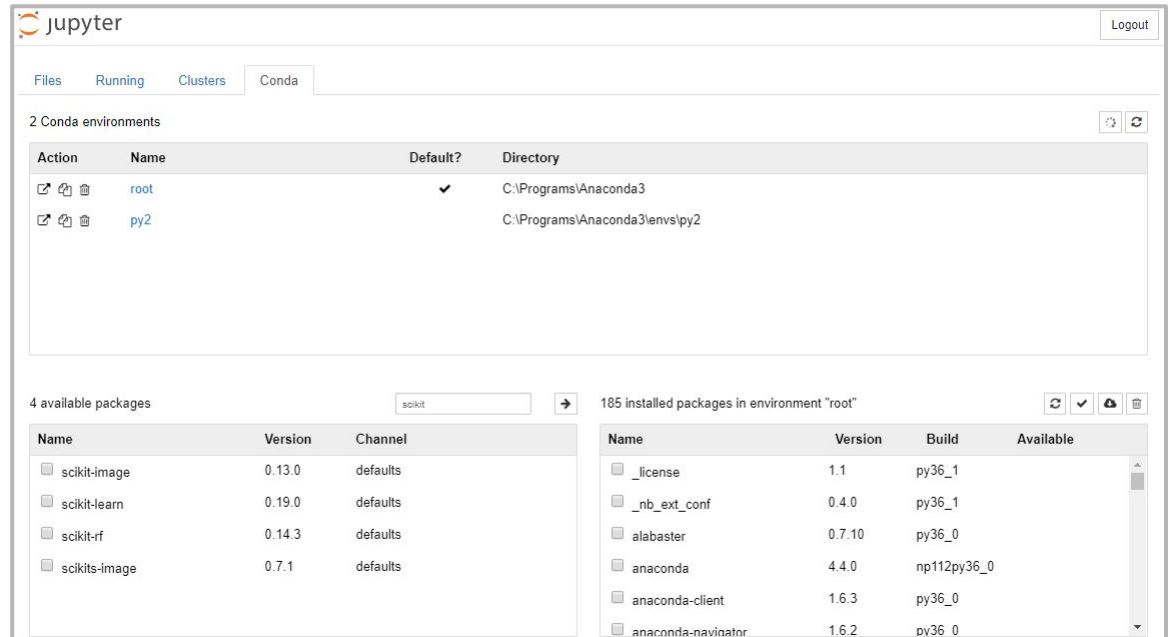
```
>pip install tweepy
```

Installation

Packages and environments can be managed also from jupyter:

From a dedicated panel:

>conda install nb_conda

Installation
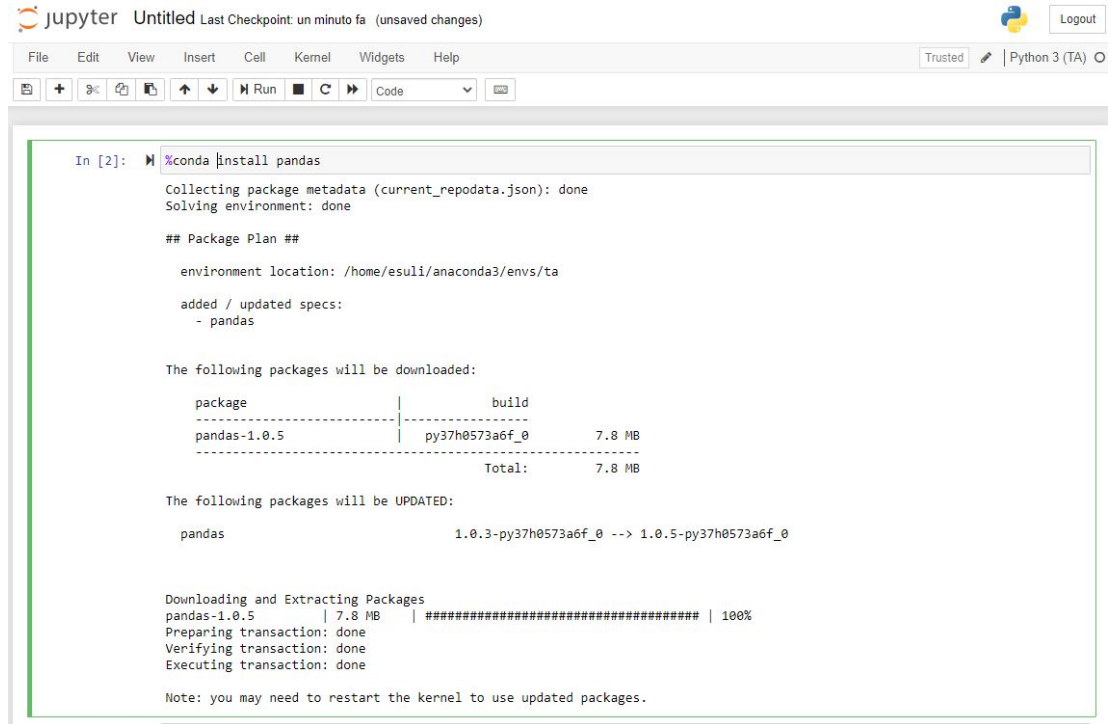
Packages and environments can be managed also from jupyter:

Running shell commands directly in the notebook:

Running Python

Console

Python can be run as an interactive command interpreter:

```
>ipython
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]: a = 5
In [2]: a**2
Out[2]: 25
In [3]: for i in range(5):
   ...:     print(i, i**2)
   ...:
0 0
1 1
2 4
3 9
4 16
In [4]: exit()
```

Notebooks

A notebook is an interactive computational environment, in which pieces of code are organized in "code cells" whose output is shown "output cells" the notebook itself.

Notebooks can contain many types of cells, such as rich text, plots, animations.

Notebook are useful for exploration, experimentation, and reporting results.

You can install Jupyter notebooks on your computer or use a hosted service, e.g., Colab.

Strings

Strings have already been discussed in Chapter 02, but can also be treated as collections similar to lists and tuples. For example

```
In [1]:
```

```
In [4]: S = 'Taj Mahal is beautiful'
        print([x for x in S if x.islower()]) # list of lower case charactes
        words=S.split() # list of words
        print("Words are:",words)
        print("--".join(words)) # hyphenated
        " ".join(w.capitalize() for w in words) # capitalise words
        ['a', 'j', 'a', 'h', 'a', 'l', 'i', 's', 'b', 'e', 'a', 'u', 't', 'i', 'f', 'u', 'l']
        Words are: ['Taj', 'Mahal', 'is', 'beautiful']
        Taj--Mahal--is--beautiful
Out[4]: 'Taj Mahal Is Beautiful'
```

String Indexing and Slicing are similar to Lists which was explained in detail earlier.

```
In [3]: print(S[4])
        print(S[4:])

        M
        Mahal is beautiful
```

Dictionaries

Dictionaries are mappings between keys and items stored in the dictionaries. Alternatively one can think of dictionaries as sets in which something stored against every element of the set. They can be defined as follows:

To define a dictionary, equate a variable to { } or dict()

```
In [5]: d = dict() # or equivalently d={}
        print(type(d))
        d['abc'] = 3
        d[4] = "A string"
        print(d)

        <class 'dict'>
        {4: 'A string', 'abc': 3}
```

As can be guessed from the output above. Dictionaries can be defined by using the { key : value } syntax. The following dictionary has three elements

```
In [6]: d = { 1: 'One', 2 : 'Two', 100 : 'Hundred'}
        len(d)
Out[6]: 3
```

Scripts

A script is a Python source file, i.e., text file, with .py extension, that defines a directly executable program and/or a module declaring functions and classes.
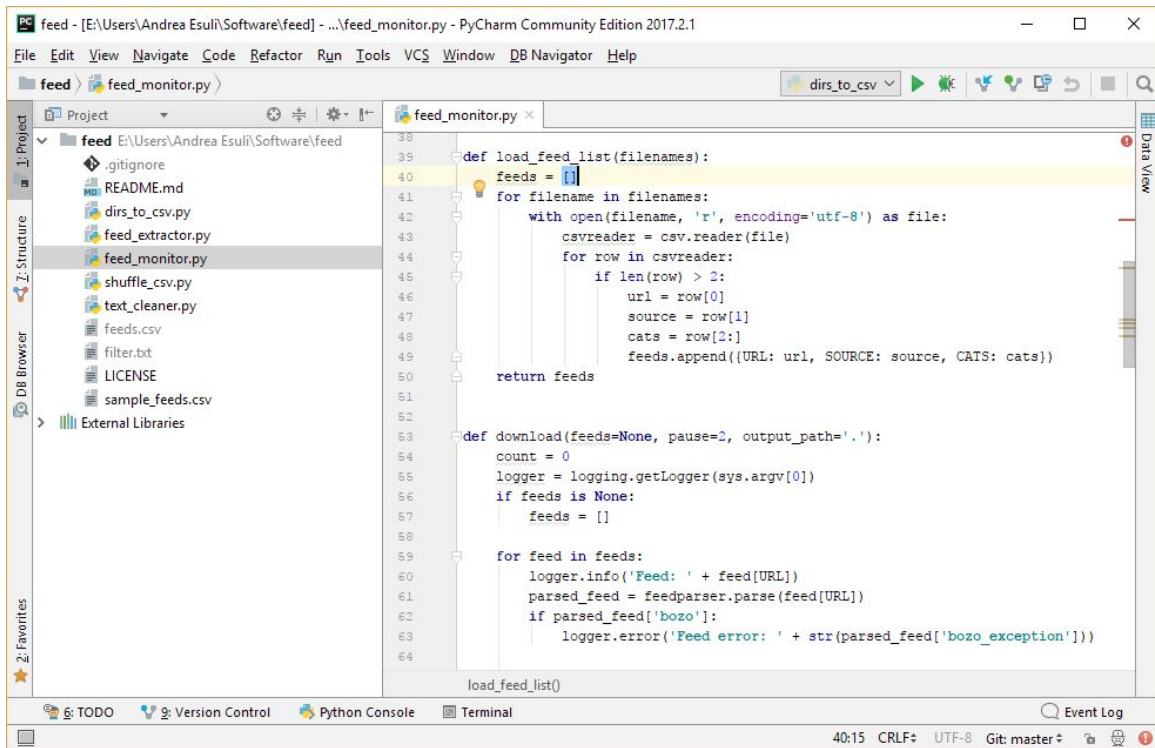
Content of a hello.py file:

```
def hello():
    print('Hello world!')

hello()
```

Execution:

```
>python hello.py
Hello world!
>
```

Which editor?

I warmly suggest [PyCharm](#).

Detailed tutorials

Here you can find a list detailed tutorials that introduce all the basic concepts of Python.

Introduction to Python (Learning Path) – Real Python
Learn fundamental concepts for Python beginners that will help you get started on your journey to learn Python. These tutorials focus on the absolutely essential things you need to know about Python.

realpython.com