Introduction to Python

Dileep Kumar G.

Contents

1. Introduction to Python
2. The Scientific Python ecosystem
3. Before starting: Installing a working environment
4. The workflow: interactive environments and text editors

Introduction to Python I

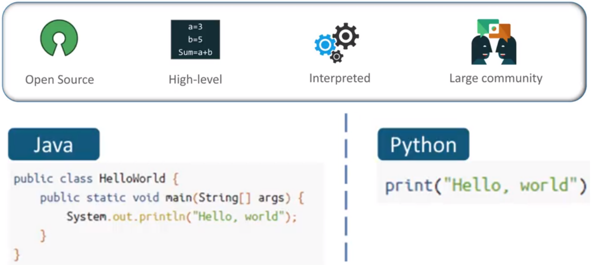
- **Python** is a high-level, interpreted, interactive and object-oriented scripting language.

- **Interpreted**: It is processed at runtime by the interpreter and you do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Interactive**: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Object-Oriented**: It supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python was developed by Guido van Rossum in the late 80s and early 90s at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Why Python? I

- **Python's strengths for Data Science and ML**
  - **Simple & Easy to learn**:



Java

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

Python

```python
print("Hello, world")
```

Why Python? II

- **Supports AI & ML libraries**:
  - Scikit-learn
  - Keras
  - Tensorflow
  - Opencv
- **Big Data**:
  - Python handles Big Data!
  - Python supports parallel computing
  - You can write MapReduce code in python
  - Libraries:

PYDOOP | DASK | PySpark

Why Python? III

- **Scripting - Automation**:
  - It is the most popular scripting language in the industry
  - Automate certain tasks in a program
  - They are interpreted rather than compiled
- **Data Science**:
  - Well suited for data manipulation and analysis
  - Deals with tabular data with heterogeneously-typed columns
  - Arbitrary matrix data
  - Observational/statistical datasets
  - Libraries:

NumPy Pandas matplotlib seaborn

Who Uses Python?

- The popular *YouTube* video sharing service is largely written in python.
- *Google* makes extensive use of Python in its web search systems.
- *Dropbox* storage service codes both its server and desktop client software primarily in python.
- The *Raspberry Pi* single-board computer promotes Python as its educational language.
- *BitTorrent* peer-to-peer file sharing system began its life as a python program.
- *NASA*, Los Alamos, Fermilab, JPL, and others use python for scientific programming tasks.
- *NSA*[1] uses python for cryptography and intelligence analysis.

[1] National Security Agency, USA

The Scientific Python ecosystem I

- Unlike Matlab, or R, Python does not come with a pre-bundled set of modules for scientific computing.
- Below are the basic building blocks that can be combined to obtain a scientific computing environment:

1. **Python, a generic and modern computing language**
    - The language: flow control, data types (string , int), data collections (lists, dictionaries), etc.
    - Modules of the standard library: string processing, file management, simple network protocols.
    - A large number of specialized modules or applications written in Python: web framework, etc... and scientific computing.
    - Development tools (automatic testing, documentation generation)
2. **Numpy**: Numerical computing with powerful numerical arrays objects, and routines to manipulate them. `http://www.numpy.org/`
3. **Scipy** : High-level numerical routines. Optimization, regression, interpolation, etc `http://www.scipy.org/`
4. **Matplotlib** : 2-D visualization, publication-ready plots. `http://matplotlib.org/`

The Scientific Python ecosystem II

5. **Advanced interactive environments**
   - IPython, an advanced interactive Python console. `http://ipython.org/`
   - Jupyter, an advanced interactive notebooks in the browser. `http://jupyter.org/`
6. **Domain-specific packages**
   - **Mayavi** for *3-D visualization*
   - **pandas, statsmodels, seaborn** for *statistics*
   - **sympy** for *symbolic computing*
   - **scikit-image** for *image processing*
   - **scikit-learn** for *machine learning*

Before starting: Installing a working environment I

- You need to set up a **Python environment**.
- The easiest way to do this is by installing **Anaconda**[2].
  - `https://www.anaconda.com/download/`
- We'll be using **Jupyter notebooks**.
  - They interleave documentation (in markdown) with executable Python code, and they run in your browser. That means that you can easily edit and re-run all the code in this course.
- If you use Anaconda, Jupyter is already installed.

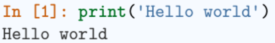
Python 3 or Python 2?

- In 2008, Python 3 was released. It is a major evolution of the language that made a few changes.
- Some old scientific code does not yet run under Python 3.
- However, this is infrequent and Python 3 comes with many benefits. I advise that you install Python 3.

[2] Popular Python Data Science Platform

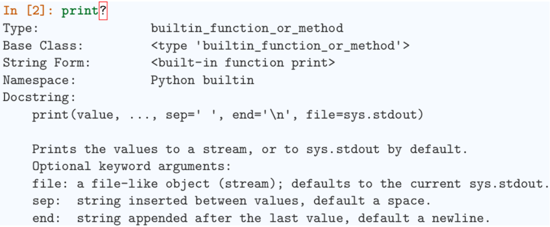
The workflow: interactive environments and text editors I

- There is not one blessed environment to work in, and not only one way of using it.
- Interactive work
  - I recommend an interactive work with the IPython console, or its offspring, the Jupyter notebook. They are handy to explore and understand algorithms.
    - To start IPyhon: Type "ipython" in a terminal
    - To start Jupyter: Type "jupyter notebook" in terminal
  - Under the notebook: To execute code, press "shift enter".



```
In [1]: print('Hello world')
Hello world
```

  - Getting help by using the ? operator after an object:



```
In [2]: print?
Type:               builtin_function_or_method
Base Class:         <type 'builtin_function_or_method'>
String Form:        <built-in function print>
Namespace:          Python builtin
Docstring:
    print(value, ..., sep=' ', end='\n', file=sys.stdout)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
```

The workflow: interactive environments and text editors II

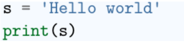
- See also:
  - IPython user manual: `http://ipython.org/ipython-doc/dev/index.html`
  - Jupyter Notebook QuickStart: `http://jupyter.readthedocs.io/en/latest/content-quickstart.html`

The workflow: interactive environments and text editors III

- Elaboration of the work in an editor
  - As you move forward, it will be important to not only work interactively, but also to create and reuse Python files.
  - Here are several good easy-to-use editors:
    - Spyder: integrates an IPython console, a debugger, a profiler...
    - PyCharm: integrates an IPython console, notebooks, a debugger...
    - Atom

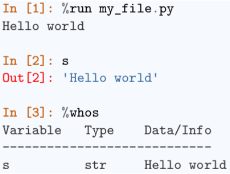
The workflow: interactive environments and text editors IV

- As an exercise, create a file my_file.py in a code editor, and add the following lines:



```python
s = 'Hello world'
print(s)
```

  - Now, you can run it in IPython console or a notebook and explore the resulting variables:



```
In [1]: %run my_file.py
Hello world

In [2]: s
Out[2]: 'Hello world'

In [3]: %whos
Variable   Type    Data/Info
----------------------------
s          str     Hello world
```

Q & A Session