Control System I ETHZ

Python tutorial

September 2018

1 What is Python?

Python is a high level, object oriented, interpreted language:

- **high level**: the programmer does not have to care about low-level implementation details such as registers, memory addresses but handles variables, arrays, objects, logic constructors, loops.
- **object oriented** : programming is based on the concept of objects which can contain data in form of fields known as *attributes* and functions known as *methods*.
- **interpreted** : the original program is translated in something (*bytecode*) that a virtual machine, called *interpreter* can understand and execute.

2 Why Python?

Python is:

- **Easy to learn** : few basic features and self-explanatory syntax. Would you guess what does the following code return?

```
if 3 in [1,2,3]:
    print("Hello!")
```

- **Support**: the majority of libraries are open source with a big development and support community. If you are in trouble google your problem, in the 99 % of the cases the answer is out there.
- **Object oriented**: in Python everything is an object. Thinking in a object oriented way helps to better organize the code and facilitate the transition to more complex programming languages as C/C++.

3 What Python is used for?

- **Data Science and Machine Learning**: Python is one of the most used programming language as high-level interface for machine learning and numerical algorithms libraries. During this course, we will make use of some of them: `numpy`, `scipy`, `signal` just to name a few.
- **Glue code**: it means that Python can interface and interoperate multiple pieces of code together.
- **Web browsers and applications' GUI**.

3.1 Which version?

Python 3.x is a newer version and as such supports newer features. On the other hand, python 2.7 will only receive updates until 2020 [1], and is used for facing compatibility issues. The tutorials and the course programming exercises will use python 3.x. If your PC already has another version, follow the provided installation instructions.

4 IDEs

How do we write code? Theoretically the minimum setup is given by any text editor and a Python interpreter. This solution would work but is not the best for fast development and troubleshooting. IDEs are programs designed for making our lives easier.

IDE stands for Integrated Development Environment and is a program dedicated to software development. IDEs generally supports useful features such as:

- save and reload code files
- run code within the IDE
- built-in debugger
- syntax highlighting and auto-completion
- automatic code formatting: this is an important features since python is a *indention based* programming language, meaning that

```
for i in range(2):
   print(1)
   print(2)
```

and

```
for i in range(2):
    print(1)
print(2)
```

give different results, respectively:

```
>> 1
>> 1
>> 2
>> 2
```

and

```
>> 1
>> 1
>> 2
```

A good editor will highlight code blocks and indentation errors, making debugging easier.

5 What is a Jupyter Notebook?

The Jupyter Notebook is an open-source web-based application that allow you to create and show documents that contain live code, equations, visualizations and narrative text. It is used for numerical simulations, modeling, data visualization, etc [2].

The Jupyter Notebook App can be executed on a local desktop and in addition to displaying/editing/running notebooks documents, has a "dashboard", a "control panel" showing local files and able to run and close active kernels.

In the following sections, we will guide you through the installation instructions of **PyCharm**, one of the most used python IDE and the Jupyter Notebook. The first will be used for development and main tool for coding. The second will be a useful tool for reports, data visualization and the coming python tutorials.

6 Installation instructions (Windows/Linux)

6.1 Setup Python

6.1.1 Using Anaconda

A convenient way to install python 3.x together with the Jupyter application is to use the Anaconda distribution. The advantage is that you have access to over 720 scientific packages. The system requirement is around 3 Gb of free space on disk.

Linux

1. Go to `https://www.anaconda.com/download/`, select and save Python 3.6 version

2. Open a terminal, go to the download folder and change the script mode to executable. Then execute the script:

   ```
   $ cd {installation folder path}
   installation-folder-path$ chmod +x Ananconda3-5.2.0-linux-x86_64.sh
   {installation-folder-path}$ ./Ananconda3-5.2.0-linux-x86_64.sh
   ```

3. Follow the installations instructions in the terminal and when asked to install Visual Studio Code, prompt *no*.

4. Add Anaconda path to the system PATH. Open with your favourite editor the bash file `.bashrc`. E.g, using `gedit` and append the line

   ```
   export PATH={anaconda3-install-directory}/bin:$PATH
   ```

   Save and exit. In the terminal type `source ./bashrc`. You can verify the installation typing:

   ```
   $ conda
   ```

   The `conda` command manual will show in the terminal. `conda` is a tool for managing and deploying applications, environments and packages. For installing a new package you can simply type `conda install <package_name>` in a terminal console.

Windows

1. Go to `https://www.anaconda.com/download/`, select and save Python 3.6 version.

2. Click on the downloaded executable and follow the default installation instructions. Do not install Visual Studio.

3. Add Anaconda path to the system PATH. Open a windows terminal by typing `cmd` in the search window. Click on `Command Prompt`. Type

   ```
   set PATH=%PATH%;<absolute\path\to\anaconda\installation>;
   <absolute\path\to\anacond\installation>/Scripts
   ```

   Here we provide an example:

   ```
   C:\Users\giuseppe>> set PATH=%PATH%;C:\Users\giuseppe\Anaconda3;
   C:\Users\giuseppe\Anaconda3\Scripts
   ```

6.2 Using ‘apt-get‘ and ‘pip‘ (recommended for Linux)

Ubuntu 16.04, and other versions of Debian Linux ship with both Python 3 and Python 2 pre-installed. To make sure that our versions are up-to-date, let's

update and upgrade the system with apt-get:

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
```

The `-y` flag will confirm that we are agreeing for all items to be installed, but depending on your version of Linux, you may need to confirm additional prompts as your system updates and upgrades. Once the process is complete, we can check the version of Python 3 that is installed in the system by typing:

```
$ python3 -V
```

You'll receive output in the terminal window that will let you know the version number. The version number may vary depending on whether you are on Ubuntu 16.04, or another version of Linux, but it will look similar to this:

```
Python 3.5.2
```

To manage software packages for Python, let's install `pip`:

```
$ sudo apt-get install -y python3-pip
```

A tool for use with Python, pip installs and manages programming packages we may want to use in our development projects. You can install missing Python packages typing:

```
$ pip3 install --user {package_name}
```

Here, `package_name` can refer to any Python package or library. So if you would like to install NumPy, you can do so with the command `pip3 --user install numpy`. The option user is to make sure that installation is local and not system-wide. There are a few more packages and development tools to install to ensure that we have a robust set-up for our programming environment:

```
$ sudo apt-get install build-essential libssl-dev libffi-dev python3-dev
```

6.3 Installing Jupyter

6.3.1 With Anaconda

Anaconda already comes with Jupyter Notebook app.

Linux You need to add Anaconda's bin folder to the PATH environment variable in order to find the Jupyter Notebook application (if you still did not do it in the previous steps). Modify the `.bashrc` file. You can use any text editor,

in the example we use gedit: `gedit .bashrc`. Append then the following line:

```
$ export PATH={anaconda3-install-directory}/bin:$PATH
```

Save and exit. In the terminal type `source ./bashrc`. Now you can open the Jupyter web server typing the command `jupyter notebook`

Windows In your Start menu, after Anaconda installation, you'll have a bunch of neat new tools, including an entry for Jupyter Notebook. Click to start it up and it will launch in the background and open up your browser to the notebook console.

With pip3 (Linux) First, ensure that you have the latest pip: older versions may have trouble with some dependencies:

```
$ python3 -m pip install --user --upgrade pip
```

Then install the Jupyter Notebook using:

```
$ python3 -m pip install --user jupyter
```

You can now start using Jupyter Notebook typing `jupyter notebook` in a open terminal.

7 PyCharm

PyCharm is the *standard de facto* IDE for professional development of Python applications. As a student, you are eligible for the free professional version.

1. Go to `https://www.jetbrains.com/shop/eform/students`, fill the form with your details and follow the registration instructions.
2. You can now login with your new credentials at `http://www.jetbrains.com` and access to all tools available for downlads with student license.
3. Under the tab `download` select `PyCharm 2018.2.2` and download the Linux or Windows distribution.

   Linux

   - Extract the content of the tar archive in your favourite directory

- Open a console and cd into the PyCharm folder:

```
$ cd  /{pycharm-dir}/bin/{pycharm-dir}/bin$ ./pycharm.sh
```

- Activate the program using your Jetbrains credentials.
- Follow the installation instructions and immediately start PyCharm creating the first untitled project. We require this step for creating a icon from which we can easily start PyCharm. In the IDE window, click `Tools` first and then select `Create Desktop Entry`, press `Ok`.
- You can now close the IDE and open it from the launcher icon.

Windows

- Follow the Graphical installer instructions.
- Check 32-bit launcher or 64-bit launcher according to your platform and `associate .py extension` so that Windows will automatically use Pycharm as default program for opening python scripts.
- Activate the program using your Jetbrains credentials.

7.1 Using Jupyter Notebook in PyCharm

Any python script (program) belongs to a *project*. Create a new folder called *test*, this will contain all projects files. You can create a project by opening your brand new PyCharm and selecting *New Project*. This will open the *Create Project* window. Specify as location the *test* folder by clicking on the three dots and navigating through the dialog window. Do not click on *Create* yet.

Before starting with coding , we need to select the proper interpreter. Linux comes with a preinstalled python 2.7 and Python 3. You can either choose Python 3 interpreter shipped with native Ubuntu or in case you want to use additional packages provided by Anaconda we need to change it to the Anaconda interpreter. After we add this new entry, it will not be necessary to repeat these steps for future projects.

1. In the *Create Project* window click on *Project Interpreter: Python x.x.*
2. After selecting *Existing Interpreter* specify the Anaconda interpreter if you previously installed it. For Windows users, after clicking *Existing Interpreter*, just select *System interpreter* on the left and PyCharm will authomatically select the Anaconda interpreter. In case you did not install Anaconda, just select the default python 3.x interpreter. Again, click on the three dots and navigate to the Anaconda installation folder.

3. Finally, click on *Create*.The apparently scary PyCharm IDE window will appear. Do not panic. We will not need all of its features and you will learn more by practising this great tool. The created project is empty, but not for long!

4. In the leftmost column (the *Project Explorer*) right click onto the project folder (*test*), then *New* → *Python File*. Call it *hello*.

5. Repeat the same procedure but now create a Jupyter Notebook file by selecting the corresponding entry in the scroll-down menu. Now your project will contain 2 files:

   - `test`
     - `hello.ipynb`
     - `hello.py`

8 My first Python program

Double click on `hello.py`. In the editor window, start writing your first Python program by typing:

```
print("Hello world!")
```

In order to run the code you can go to *Run* → *Run 'hello'*. The output will likely be similar to the following:

```
/home/username/anaconda3/bin/python /home/username/test/hello.py
Hello world!
Process finished with exit code 0
```

Congratulations! You have successfully run your first python program.

9 My first Jupyter Notebook

If you are a Linux user open a terminal and type `jupyter notebook`. If you are a Windows user just open the Jupyter client from the Start Menu. Jupyter dashboard will appear, showing the content of the local directory. Navigate to the folder where you want your first notebook to be saved. Than click on `New`→ `Python3`. A new notebook named `Untitled` will open. Clik on `Untitled` and give it a new name, like `FirstNotebook`. Alternatively you can navigate to the new notebook created in the `test` project. Jupyter notebooks have 2 types of line:

- *code line*: where you can write normal Python code
- *markdown line*: where you can write explanatory text in **Markdown**. You can find many Markdown cheat sheet on the web [3].

Click on the first line and type `print("Hello world!")`. Click on the plus sign, this will add a new line. Generally the newly added line will be a code line. To change its mode to Markdown, select Markdown from the scroll-down menu. Copy past the following text:

```
The command ‘print(string)‘ _prints_ to screen **string**.
```

Now select the first code line and click on the play button. This will run the code contained inside the cell. For any code cell, Jupyter associate a pair of square brackets. An asterisks appears inside if the cell is running, after the cell is run they instead contain an execution number.

You can understand in which order the code cells where executed looking at the cell numbers. Although a notebook should be designed to run all code cells in a sequential order, you are free to run any cell at any time. Note that the cell will access to the variables known to the interpreter up to the previously executed cells only. Now the following Markdown cell will be automatically selected. You can run it and visualize the highlighted text.

As you see, Markdown uses *underscores*, *asterisks*, *dashes* and *indentation* to provide test highlighting and nested lists. Title of different sizes can be obtained using multiple *hash-tags* .

That's it! Now you can open, run, edit all notebooks contained in the downloadable tutorial folder. Have fun!

10 What's next?

We will use Jupyter notebooks for give you a basic python introduction. You can walk through the tutorials, running each line sequentially. We strongly encourage you to interact with the notebook and try to change the code. In this way you can get a better understanding of how your changes affect execution and output.

PyCharm will be the main tool for coding once you became familiar with Python. It is a good practice to write a clear documentation. The reader, might it be a professor, a customer or even yourself after some time, will easily understand it and follow the logic of the implemented algorithms.

Jupyter Notebooks can be extremely useful for explaining and visualizing data. You might want to make use of it for either reporting or part of your documentation.

References

[1] `https://www.python.org/dev/peps/pep-0373/#id2`.

[2] http://jupyter.org/.

[3] https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#code.