Student names: . . . (please update)

Instructions: This document contains the instructions to install and get familiarized with Python programming. ***This lab is not graded****. This file does not need to be submitted and is provided for your own benefit.*

1 Course Organization

This year all the exercises for the course will be distributed through GitLab. GitLab is an online open source Git repository for maintaining and distributing code between groups of people. The access to the exercises repository is public and you can directly use your epfl credentials to login. We recommend you to use the features Git offers to collaborate with your partners and maintain a history of your own work as well. If you download the repository without cloning, then you will have to do it every week and you will end up downloading all the previous files every time you do it. This report does not contain a tutorial on how to use Git but you can find the references in section 4.2. However, we will have short demo on how to use it during the exercise hour.

The link to the git repository

2 Python Setup

This course expects you to complete the exercises using the standard Python programming language. There are many options to use Python. For those students who are starting of new with Python, we strongly suggest to follow the following installation steps of the Anaconda environment. For those familiar with Python, you are free to use an environment of your choice but we still advise you to set-up the Anaconda environment to ease the debugging process with the assistants.

2.1 Anaconda

Anaconda is an open source Python environment that sets up all the necessary packages and softwares needed for scientific computing using Python.

Follow the link to download Anaconda distribution for your specific OS.

Choose Python 2.7 for installation.

Instructions for installation :

- Windows
- MacOS
- Linux

To verify your installation :

- First open your terminal
  - Windows: Open the Anaconda Prompt (Click Start, select Anaconda Prompt)
  - MacOS: Open Launchpad, then open Terminal or iTerm.
  - Linux–Ubuntu: Open the Dash by clicking the upper left Ubuntu icon, then type “terminal”.

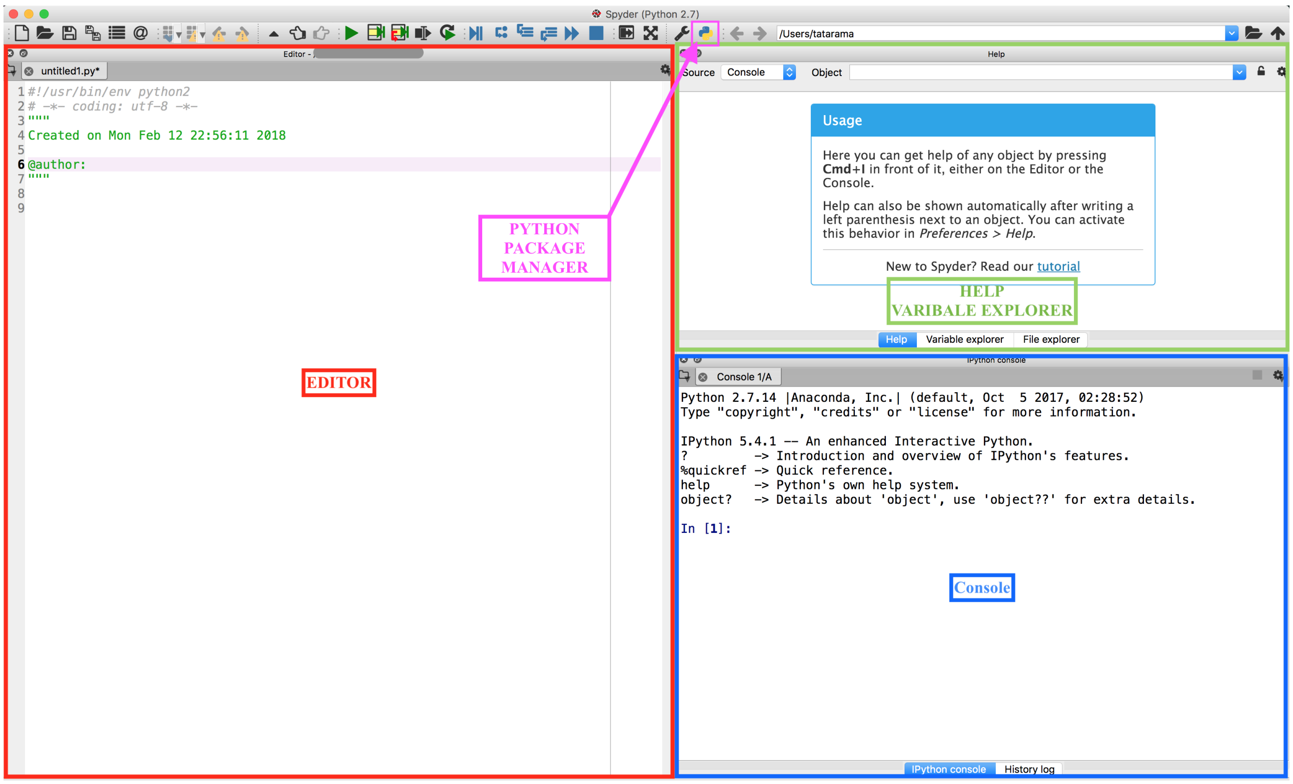
- Enter a command such as *conda list*. If Anaconda is installed and working, this will display a list of installed packages and their versions.

2.2 Spyder

Python programs can be written in several ways, it can be simply done on a terminal by running *python* or *ipython*. While this method is limited for simple programs, larger programs will be written using a text-editor or an Integrated Development Environment (IDE). Though it is not necessary to have an IDE for programming in Python, having one will bring many features that are useful while starting new. Once you have installed Anaconda successfully using the section 2.1, Spyder should be installed by default.

Spyder can be run by executing the command *spyder* in a terminal/anaconda-prompt.

Figure 1 shows the main windows when you first open Spyder. As you may have noticed, the layout is similar to Matlab. Spyder has very good documentation to get you started. It is recommended to go through the steps to get familiarized with the IDE.



```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 12 22:56:11 2018

@author:
"""
```

Usage

Here you can get help of any object by pressing **Cmd+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 2.7.14 |Anaconda, Inc.| (default, Oct  5 2017, 02:28:52)
Type "copyright", "credits" or "license" for more information.

IPython 5.4.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Figure 1: Spyder IDE overview

The three essential parts of the screen are outlined.

- **The console** (bottom right, marked in blue). You can work interactively here. Code run, either interactively or from the editor, will output any results here. Error messages will be reported here. There are two types of console: a Python console, and an IPython console. Both will run Python code, but we recommend the IPython console as it offers better visuals for debugging and has additional features.
- **The editor** (left, marked in red). You can write code to be saved to file or run here. This will suggest problems with syntax and has features to help debug and give additional information.
- **The inspector** (top right, marked in green). The Object inspector can display detailed help on specific objects (or functions, or...), and the Variable inspector can display detailed information

on the variables that are currently defined. Extremely useful when debugging.

- **Python Package Manager** (top toolbar, marked in pink) Use this menu to add new paths to the default python package paths. (NOTE : The symbol may look slightly different on your machine from the one shown in this report)

For an in depth tutorial of Spyder follow the link.

Before you begin with the programming, there is one more important step. Assuming that you have either downloaded or cloned the exercises repository from GitLab, we now have to add some python packages that will be used through out this course. To do this,

- Open the Python Package manager window (The icon marked in pink in the figure 1)
- Click on Add Path
- Navigate to the downloaded exercise repository and add the **PythonPackages** folder
- Close and re-open Spyder

3 Programming with Python

After successfully completing the installation steps in the previous section, you can now get started with programming using Python. Python is not just a computational tool but a very powerful programming language. This means having to learn a few more extra concepts to get your job done. There are a ton of references available online for those who are interested in learning Python in depth. We will try to provide the necessary references to help with the concepts that are useful during the course as and when needed.

3.1 Basic Python Concepts

In this section we will quickly go over the list of topics given below. You can open and run the individual files marked with the same topic name using Spyder. We suggest you to go through each section individually and spend time exploring each of the concepts by making changes to the code and observing the outputs.

1. HelloWorld
2. Imports
3. Data Types
4. Math
5. Conditional Statements
6. Data Containers : Lists, Tuples and Dictionaries
7. Functions
8. Loops
9. Numpy
10. Matplotlib

While you are executing each of the small exercises, try to learn how to use different features of Spyder. Especially the help and debugging feature. When you are unsure of any command, use the help service either the one built into Python or Spyder. After familiarizing yourself with the above concepts try to solve the following python exercises.

3.2 Exercise 1

Check if the following matrix M is a magic square or not?

Hint : *A magic square is a square matrix which contains distinct integers and whose sum along any of its individual rows or columns or diagonal is a constant. The constant is called as a magic constant or magic sum or magic square*

$$M = \begin{bmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{bmatrix}$$

Further Step : *Try if you can generalize your script to have a function to check any arbitrary matrix if it is a magic square or not. Import the function as a module in another script and use it to check the matrix M*

3.3 Exercise 2 - Plotting a function

Plot the following function $f(x)$ over an interval [0, 2] with proper labels and title

$$f(x) = sin(x-2)e^{-x^2}$$

4 References

4.1 Python

- NumPy for MATLAB users
- A byte of python
- A Crash Course in Python for Scientists
- The official Python documentation should be your first stop when looking for information
- numpy
- scipy
- matplotlib

4.2 Git

- Try Git!
- A Visual Git Reference