Problem Set 0: A first start on Programming and Python

Albert Rodriguez-Sala and Jacob Adenbaum
Programming for Economics—ECNM10106
The University of Edinburgh

This problem set is designed to get you started in Python, ensure that you correctly download Anaconda, and that you familiarize yourself with Spyder, Jupyter Notebooks, and some of the main Python libraries.
This problem set will not be graded and it will not count for the final grade—don't worry if you cannot solve it, this is to practice! We recommend you work on this homework individually but collaboration with classmates is allowed. We recommend working on this homework in a single script (*.py* file) in Spyder. Once you are finished, open Jupyter Notebooks and reproduce the exercises in different cells (with adding markdown comments) so that the outcome is a nice file containing comments, the code, and the solutions in *.ipynb*,or *.pdf* format.

Exercise 1. Preliminary exercise.

Set your working directory (equivalent to the cd in Stata):

```
import os
os.chdir("your folder path")
```

Read the Zen of Python by running the following line

```
import this
```

What does it say about programming? How does it relate to some of the good coding practices we saw in class?

Exercise 2. Copy-paste the following paragraph into a string variable called *str_x*. Count the number of times each of the following words appears in the string: *wealth*, *distribution*, *households*, and *assets*. **Hint:** Google how to create a string in Python. Use the count method of a str class.

"Facts about the distribution of wealth have been highlighted in a large number of studies, including Wolff (1992, 1998), Cagetti and De Nardi (2008), and Moritz and Rios-Rull (2015). A striking aspect of the wealth distribution in the US is its degree of concentration. Over the past 30 years or so, for instance, households in the top % of the wealth distribution have held about one-third of the total wealth in the economy, and those in the top 5% have held more than half. At the other extreme, more than 10% of households have little or no assets. While there is agreement that the share held by the richest few is very high, the extent to which the shares of the richest have changed over time (and why) is still the subject of some debate (Piketty 2014, Kopczuk 2014, Saez and Zucman 2014, and Bricker et al. 2015)."

This paragraph is from the column "Quantitative macro models of wealth inequality: A survey" by Mariacristina de Nardi at VOXEU-CEPR, 2015.

Exercise 3. Compute and plot 1-D functions.

a. Create a grid—or linear space—x from 0 to 10, with 100 points. Create a variable y that is equal to the log of x, and a variable z that is equal to the sinus of x. Explore your outcomes on the variable explorer. What type is y? what size is y? and z? **Hint:** Check the NumPy documentation and familiarize yourself with the NumPy *linspace*, *log*, and *sin* routines. To import the NumPy library in your file run

```
import numpy as np
```

and then call the NumPy functions via np.linspace, np.log, and np.sin.

b. Go to Matplotlib documentation. Use some of the routines in the library to plot (y, x) and (z, x).

Exercise 4. Compute the values of z, $z = f(x, y) = log(x) + sin(y)$, where both x and y are grids from 0 to 10 with 100 points. Make a 3-D plot of $f(x, y)$ using Matplotlib 3D surface.

Exercise 5. Let's solve the following consumer problem. The consumer maximizes utility over goods x_1, x_2 subject to the budget constraint.

$$Max_{x_1,x_2} x_1^{\alpha} x_2^{\alpha-1}$$
$$\text{St: } p_1 x_1 + p_2 x_2 \leq y$$

Where y is the consumer's income and x_i, p_i, and y are strictly positive numbers. First, set up the problem in Python using the following code

```
import numpy as np
# Parameters values
alpha = 0.5
y = 10
p1 = 1
p2 = 2

def utility_function(x1,alpha,y,p1,p2):
    x2 = (y-p1*x1)/p2  # Income not spent first good is spent on the second
    utility = x1**alpha * x2**(1-alpha)
    return utility
```

Go to SciPy.optimize documentation and use *minimize_scalar* to find the x_1 and x_2 that solve the consumer problem. **Hint:** Note that maximizing a function f is the same a minimizing a function $g = -f$. Note that to apply minimize scalar you need to write a new function based on the *utility_function* with x1 as the only argument. One way to this is the following

```
obj_func = lambda x1: -utility_function(x1,alpha,y,p1,p2)
```