Macroeconomics 3
TA session 1, Exercise set

Prof.: Marco Maffezzoli *TA: Andrea Pasqualini*[*]

PhD in Economics and Finance, A.y. 2017-2018
Bocconi University, Milan

This is a set of exercises to get you started with Python and the modules `numpy` and `matplotlib`. **No hand-in required**. For every exercise, there is more than one way to obtain the intended result, so experimenting is encouraged. The first seven exercises are simply intended to warm you up with Python (they're silly, I know), while the others are a bit more serious and focused on what we will do. If you have questions, feel free to ask me.

0. Create a dictionary with the name of 4/5 students and years of birth and derive a list with the ages by using a list comprehension. Obtain also a new dictionary with the same keys as the original containing the ages computed above.

1. Define a function `myMax()` that takes two numbers as arguments and returns the largest of them, and complains if the type of inputs is incorrect.

2. Write a function that takes a character (i.e., a string of length 1) and returns True if it is a vowel, False otherwise. *[Extra: validate your inputs, i.e., check for unsuitable inputs]*

3. Write a function that takes a list and returns a new list that contains all the elements of the original list without duplicates (if any). *[Extra: validate your inputs]*

4. Write a function that computes the dot product of two given lists leveraging the zip() function. *[Extra: validate your inputs]*

5. Write a recursive function `fibonacci(n)` that returns the n-th element of the Fibonacci series.

6. Enclose the answer to the previous exercise into a `Class` that you will call `Fibonacci`. This class should have two methods: `element` and `series`. `Fibonacci.element(n)` will return only the n-th element of the series, while `Fibonacci.series(n)` will return a tuple (not a list!) that consists of the Fibonacci sequence up to the n-th entry.
   [Hint: remember that Python starts counting from zero]
   [Extra: validate your inputs]

7. Suppose a certain Markov Chain has the following right-stochastic transition matrix:

$$\Pi = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix}$$

[*] Author of this document. Comments are welcome: andrea.pasqualini@phd.unibocconi.it.

Create the matrix using numpy and compute the N-step ahead transition matrix for some arbitrary N. *Extras:*

(a) *write code that simulates the markov chain;*

(b) *write code that finds the stationary distribution;*

(c) *write everything using a class, with input validation, i.e.,* Π *must be a square matrix and right-stochastic.*

[Hint: Google is your best friend]

8. Suppose that you want to evaluate a polynomial $p(x)$ defined as

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \\ &= \sum_{i=0}^{n-1} a_i x^i. \end{aligned}$$

You do not want to choose a value x necessarily. Therefore, write a `Class` that, for a given `tuple` or `list` a:

- has a method that evaluates $p(x)$ at a given x_0;
- has a method that computes the first derivative of $p(x)$ at a given x_0;
- plots the function in a given domain.

You can only use the following external help: `from numpy import array, linspace` and `from matplotlib.pyplot import subplots`

9. Let $X = [0, 1] \subset \mathbb{R}$ and consider the function $f : X \to X$ such that

$$f(x) = \sqrt{1+x} - \frac{2}{3}, \qquad x \in X.$$

As you know, we can construct a sequence $(x_n)_{n=0}^{\infty}$ through the iteration $x_{n+1} = f(x_n)$, given an initial condition x_0. The sequence so obtained is convergent because f is a contraction. The limit is the fixed point of the contraction.

Write some code that implements the iteration and finds the fixed point: start from an arbitrary initial condition x_0 and stop when x_{n+1} and x_n are "close enough." Keep track of the (finite) sequence $(x_n)_{n=0}^{N-1}$ visited by the algorithm. You can only use the following import statement: `from numpy import sqrt, array`. *[Extra: define a function that performs the iteration for a given "function handle" and initial condition]*

10. Using the answer to the previous exercise and the path converging to the fixed point you obtained with the iterative algorithm, replicate *exactly* Figure 1.[1] The original figure is a PDF file 5 inches wide and 4 inches tall. You can use anything that you find useful in `numpy` and `matplotlib`. *[Hint: Google is your best friend]*

[1] *Exactly* means that you must replicate the figure in every single detail, including the absence of the box around the graph, the order in which the lines have been drawn, the text written with LaTeX, the width of the lines, the size of the overall figure, etc.

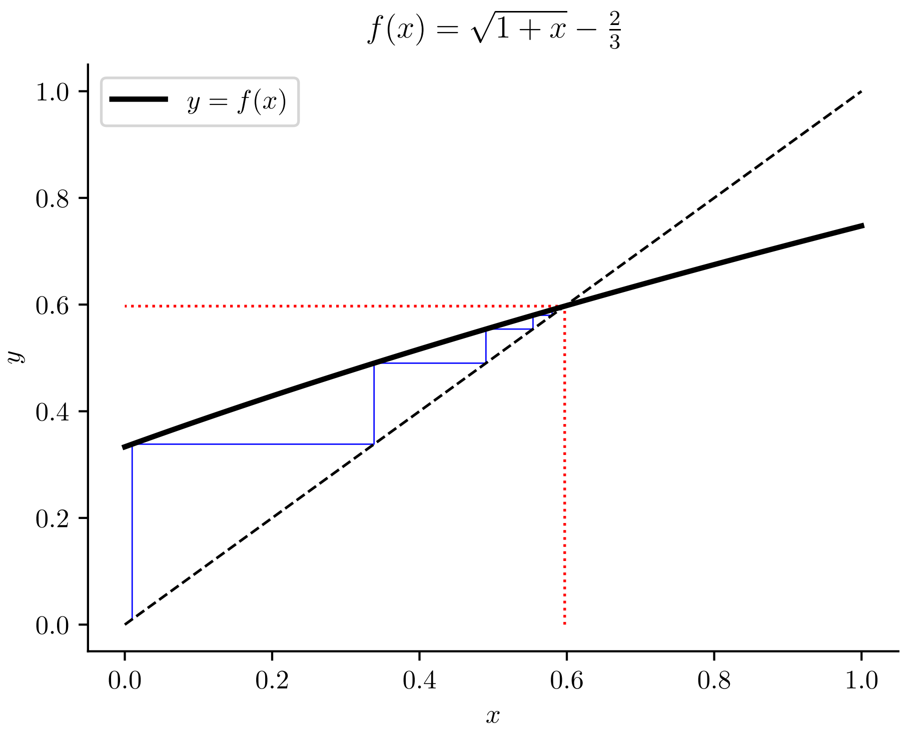

Figure 1: Path visited by the algorithm $x_{n+1} = f(x_n)$ and location of the fixed point. The initial condition is $x_0 = 0.01$.