Quantitative methods in finance - beginner python exercise with numpy

Eric Vansteenberghe

September 11, 2017

Link to the source codes and the data sets used in this lecture

1 Introduction

These are some exercises to check that some basic python concepts are mastered.

2 Work with numpy

Start in spyder with a new and empty file.

2.1 Import the package

First we want to import the package numpy.

```
import numpy
```

2.2 Work with exponential function

With the package numpy you already have a lot of functions implemented and ready to use.
For example, the function **exponential**:

$$\exp : x \rightarrow \exp(x)$$

read "funtion exponential that to a variable x associates the value "exponential of x".
If we want to know the value of $exp(1.5)$:

2.3 y=f(x)

Now we want to use x as a vector containing different values and then for each value of x compute the associated y=f(x), with f being a function (in our case we work with exponential function).
We create a vector called x, ranging from 0 to 9 with steps of 1, either we use:

```
x = numpy.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

or equivalently:

```
x = numpy.arange(10)
```

This creates:

$$X = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix} \tag{1}$$

Then we do y=f(x):

```
y = numpy.exp(x)
```

This returns:

$$Y = \begin{pmatrix} e^0 & e^1 & e^2 & e^3 & e^4 & e^5 & e^6 & e^7 & e^8 & e^9 \end{pmatrix} \tag{2}$$

2.4 Plot y=f(x)

Now we want to use another package to be able to plot y=f(x):

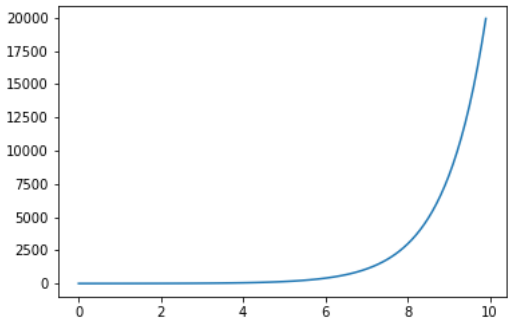
```
import matplotlib.pyplot as plt
```

We want to plot y as a function of x:

```
plt.plot(x,y)
```

We might want a smoother plot, in this case, we will use more computing power and have smaller steps between 0 and 10, let's say we want to plot by steps of 0.1:

```
x3 = numpy.arange(0,10,0.1)
```



3 Create you own function

Now you should create your own function, that you could call my_f:

$$\text{my\_f} : x \to x^2 + x - 2$$

```
def my_f(x):
    y=x**2+x-2
    return y;
```

Now apply this function to create a y_f:

```
y_f = my_f(x)
```

3.1 Plot y=f(x)

You could apply your function into y2:

```
y2=my_f(x)
```

Then plot y2 as a function of x.

3.2 Find x where f(x)=0

Now we want to solve for $f(x) = 0$. We can try if a=b using the operator "==":

```
my_f(0)==0
```

we find that this is not the case: FALSE

```
my_f(1)==0
```

we find that this is the case: TRUE
Indeed, $x^2 + x - 2 = 0$ when $x = 1$.
Now we can loop through all the element in the vector x $(0, 1, \ldots, 9)$ and print which element solve: f(element)=0:

```
for element in x:
    if my_f(element)==0:
        print('0=f(x) for x=',element)
```

3.3 fsolve

The good news is that there is already a function to do the above: fsolve.
You need to first import the function from the right package:

```
from scipy.optimize import fsolve
```

Now to find the solution to $f(x) = 0$, it is in one line:

```
fsolve(my_f,0)
```

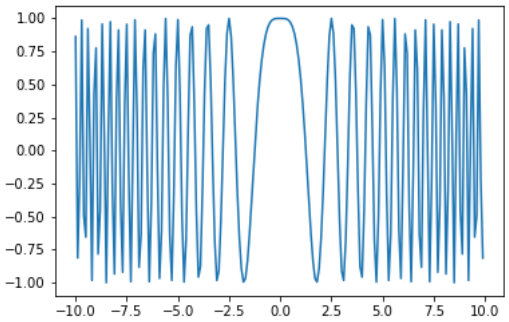
Not that ",0)" here tells fsolve **where** to start searching for the root in the abscisse line.
You can get help on this function using the console:

```
?fsolve
```

x0 = 0 is *The starting estimate for the roots of 'func(x) = 0'*

3.4 Define another function and find the root closest to x=10

Define a function: $f(x) = \cos(x^2)$, plot it and find the solution of $f(x) = 0$ close to $x = 10$.


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0