Lab0 - Introduction_Python

February 5, 2019

1 Urban Computing - Introduction to Python

The current document contains initial information about Python programming language. Students who are familiar with it can skip some steps.

1.0.1 1. Install Python

You can download and install the newest version of python from Python.org or the latest python anaconda distribution from anaconda.com. Keep in mind that anaconda distribution includes various libraries for scientific computing. Check if python works correctly by typing `python` in a terminal. You have to see something like the following:

```
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 13 2017, 12:02:49)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

1.0.2 2. Install Jupyter Notebook

Students who have installed the full anaconda distribution package (not mini-conda), they already have the jupyter-notebook installed in their machines. The rest can follow the instructions and install the notebook using this link.

1.0.3 3. Get familiar

Open a jupyter notebook file and try to write simple blocks of python code. Try to use the text functionality of jupyter notebook using the keys `Esc` , `b` and `m`. This will create a new block for text and you can use it by hitting `Enter`, check the documentation. If you want to open a new python cell below the current one, just hit `Esc` and then `b`

1.0.4 4. Basic Python

- Basic data types and containers

```
In [1]: x = 3
        print(x, type(x))
```

```
3 <class 'int'>
```

```
In [2]: y=1.0
        print(y, type(y))

1.0 <class 'float'>


In [3]: T = True
        print(x, type(x))

3 <class 'int'>


In [4]: s = 'urban_computing'
        print(s, type(s))

urban_computing <class 'str'>


In [5]: l = [1,2,3,4]
        print(l, type(l))

[1, 2, 3, 4] <class 'list'>


In [6]: t = (1,2,3)
        print(t, type(t))

(1, 2, 3) <class 'tuple'>


In [7]: d = {'age': [24,18], 'sex': [0,1], 'job':['statistician', 'computer_science']}
        print(d, type(d))

{'age': [24, 18], 'sex': [0, 1], 'job': ['statistician', 'computer_science']} <class 'dict'>
```

- Basic operations

```
In [8]: x_ = x+3           # Addition
        y_ = y+x-3.4
        T_ = T and False   # Bool operation
        s_ = s + '_course' # String concatenation
        X = x_ / 2         # Division
        Y = y_*y           # Multiplication
        XX = x**3          # Power
        l_ = l[0] + 2      # Index the first element of the list and add the number 2 to it.
        print('x =' + str(x) + '\nx_=' + str(x_) + '\ny =' + str(y) + '\ny_=' + str(y_)+\
                  '\nT ='+str(T)+'\nT_='+str(T_)+'\ns ='+str(s)+'\ns_='+str(s_)+ '\nX ='+str(X)
                 '\nY =' + str(Y) + '\nXX='+str(XX)+ '\nl ='+str(l)+'\nl_='+str(l_))
```

```
x =3
x_=6
y =1.0
y_=0.6000000000000001
T =True
T_=False
s =urban_computing
s_=urban_computing_course
X =3.0
Y =0.6000000000000001
XX=27
l =[1, 2, 3, 4]
l_=3
```

- Loops

```
In [9]: for i in l:
            print(i)

1
2
3
4
```

```
In [10]: i = 0
         while i<4:
             print(l[i] + 10)
             i+=1

11
12
13
14
```

- Functions and Plots Build a function with the name `function` which asks from the user to provide integers for variables `x` & `y`. Then loop `i` through the range `[0,x]` and calculate the equation: $y \leftarrow y^{\frac{i+1}{4}}$ inside the routine. Append the outcome of `y` into a `list` and return it at the end.

```
In [11]: def function():
             x = input('Provide a number between 5 and 10 for x: >_')
             y = input('Provide a number for y: >_')
             x,y = int(x), int(y)
             l = [y]
             for i in range(x):
                 y = y**(i+1)/4
```

```
            l.append(y)
        return l
    function()

Provide a number between 5 and 10 for x: >_6
Provide a number for y: >_8
```

```
Out[11]: [8, 2.0, 1.0, 0.25, 0.0009765625, 2.220446049250313e-16, 2.996272867003007e-95]
```

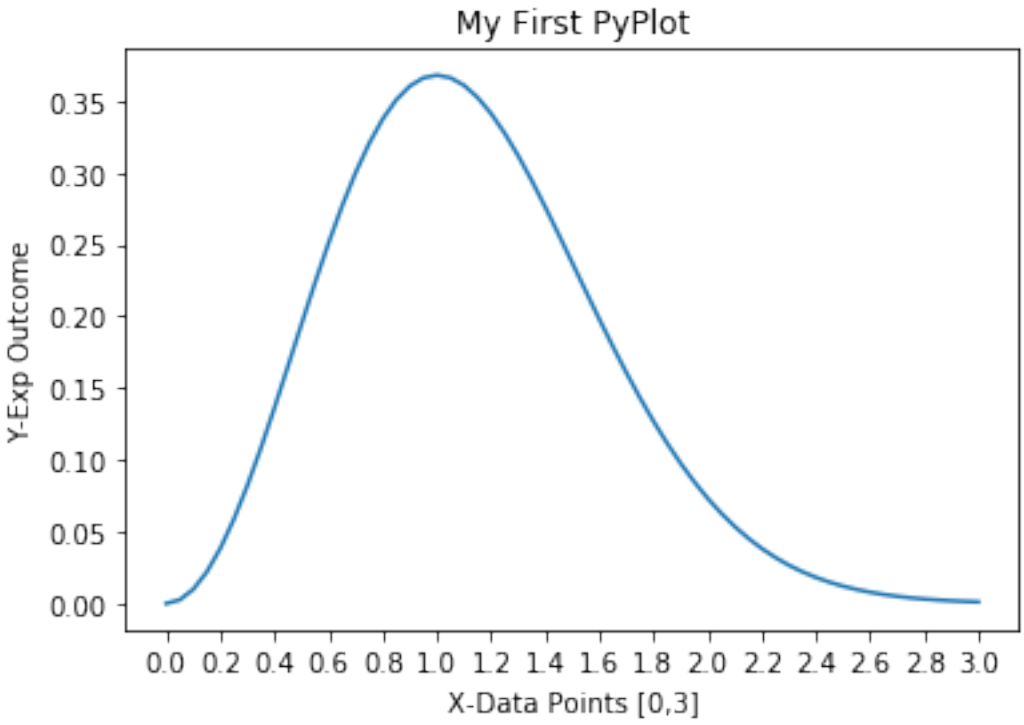
- Use functions Construct a function which calculates the equation: $y = x^2 \cdot e^{-2x}$ where $x \in [0, 3]$. First generate continuous noise data for the variable x (e.g 60 intervals) using the `numpy` library. Then compute the equation, producing the exponential values for y.

```
In [12]: import numpy as np

    def f(x):
        return x**2*np.exp(-x**2)

    x = np.linspace(0, 3, 61)    # 61 points between 0 and 3
    y = f(x)
```

- Visualize function's outcome

- Plot the vectors x,y using the `matplotlib` library.
- Provide titles and labels
- Save the graph locally
- Try to change xticks and yticks

```
In [13]: import matplotlib.pyplot as plt

    # First open an empty plot figure
    plt.figure()
    # Generate the plot
    plt.plot(x,y)
    plt.xticks(np.arange(0, 3.1, step=0.2))
    # Include main title of the graph
    plt.title('My First PyPlot')
    # State titles for X.axis and Y.axis
    plt.xlabel('X-Data Points [0,3]')
    plt.ylabel('Y-Exp Outcome')
    # Save the figure locally (pdf or png)
    plt.savefig('first_plot.png')
    plt.savefig('first_plot.pdf')
    # Visualize the plot in the console
    plt.show()
```



1.0.5 Open Lab1 using your jupyter notebook and work on the assignments >_ .