5 APRIL 2020 / #PYTHON

How to Learn the Basics of Matplotlib from Scratch

MD.HABIBUR RAHMAN

Consider myself as a tech enthusiast, like to explore technological things. Fond of leadership and managerial activity.

What is Matplotlib?

Matplotlib is a Python library which is often used in Data Visualization. Knowledge of **Matplotlib** will definitely put you one

our **plt** alias instead.
Now, we will take two lists and plot them. The code is given below:

Snippet 1:

```
dev_x = [25,26,27,28,29,30,31,32,33,34,35]
dev_y = [38496, 42000, 46752, 49320, 53200,56000, 62316, 64928, 67
```

two lists assigned in variable `dev_x` and `dev_y` .

Snippet 2:

```
plt.plot(dev_x,dev_y)
plt.xlabel('Ages')
plt.ylabel('Median Salaray(USD)')
plt.title('Median Salary(USD) By Age')
plt.show()
```

Now we have several lines of code in **snippet 2**. In the first line, we've used the **plot()** function accessed with alias **plt** where we passed two parameters (`dev_x` is on the X-axis and `dev_y` is on the Y-axis) to be plotted on.

In the next few lines we've passed the labels of the X-axis and Y-axis that we want to see in our graph. We are depicting a survey result where it shows a `Salary vs Age graph` of a particular programming language developer.

Learning.

In this Tutorial, we are going to learn the basics of the **Matplotlib** library. If you are an absolute beginner then this tutorial is the right one for you to get started.

In order to start, you need to install **Matplotlib** on your machine. You can use any IDE or editor like Sublime Text 3 to run the python code.

First make sure you have pip installed and then install **Matplotlib**. To install both, type in the terminal:

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

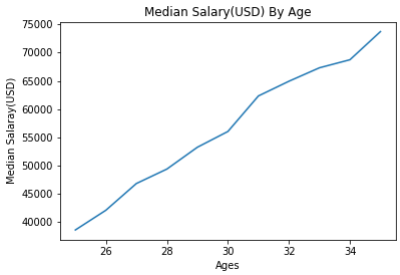
After finishing the installation, now we can start coding. **Matplotlib** helps us to visualize the data we have. We can plot any data we have with this library to visualize the nature of the data. We will see a sample plotting below.

First let's import:

```
from matplotlib import pyplot as plt
```

In this snippet, we imported the `pyplot` module of the `matplotlib` library under the alias **plt**. An alias is a kind of short form of

displayed it with the **show()** function. We can see the output below:



Now, in the following example we are going to plot two curves in a single figure where our new variables are defined as follows:

```
py_dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
py_dev_y = [45372, 48876, 53850, 57287, 63016, 65998, 70003, 70000,
```

Our code for the plotting follows.

Snippet 3:

```
from matplotlib import pyplot as plt
dev_x = [25,26,27,28,29,30,31,32,33,34,35]
dev_y = [38496, 42000, 46752, 49320, 53200,56000, 62316, 64928, 67317,
plt.plot(dev_x,dev_y)
py_dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
```

```
plt.xlabel('Ages')
plt.ylabel('Median Salaray(USD)')
plt.title('Median Salary(USD) By Age')
plt.show()
```

The output is:

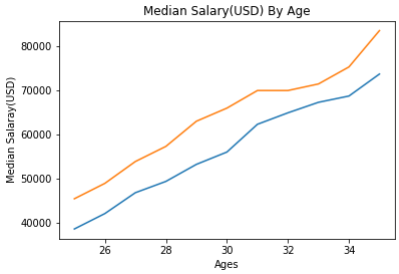


Fig: Median Salary(USD) By Age with two line curve

From the graph it is not clear what the yellow and blue curves represent. We can fix this by using the **legend()** function. We can add the following line of code just before the previous `plt.show()` line.

```
plt.legend(['All Devs','Python'])
```

The output will be like this:


```
plt.show()
```

We can also add some additional features to our graph, like line-width, color, line-style, and markers to our curves so that they look more distinguishable. To do this we need to replace the two `plt.plot()` lines in snippet 4 as follows:

Snippet 5:

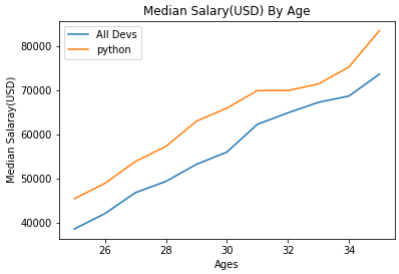
```
plt.plot(dev_x,dev_y,color='k',linestyle='-',marker='.',
linewidth=3,label='All Devs')

[same code as snippet 4 goes here]
plt.plot(py_dev_x,py_dev_y,color='b',linestyle='-',marker='o',linewidth=3
.
.
.
plt.grid(True)
plt.show()
```

Here, we've used two diff parameters in each of the new functions:

For the first and second plot() :

- color = 'k' and color = 'b' represent, respectively, the **black** and **blue** colors of the curves
- linestyle = '—' and '-' represent, respectively, the **large** and **small** dashed sign of the curves.



Now, we can see that two labels are displayed at the top left of the graph identifying the yellow curve as python and the blue curve as all developer.

Alternative Way

We can also use legend() in an alternative way to do the same thing. We can pass the info as a parameter through the plot function like below.

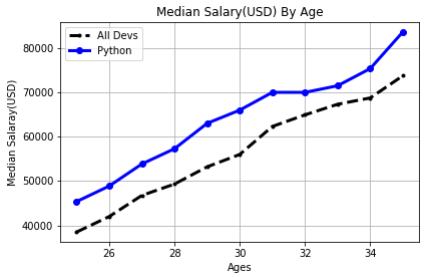
Snippet 4:

```
dev_x = [25,26,27,28,29,30,31,32,33,34,35]
dev_y = [38496, 42000, 46752, 49320, 53200,56000,62316,64928,673
plt.plot(dev_x,dev_y,label='All Devs')
py_dev_x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
py_dev_y = [45372, 48876, 53850, 57287, 63016, 65998, 70003, 700
plt.plot(py_dev_x,py_dev_y,label='Python')
.
.
[same code as previous snippet goes here]
```


shape of the data points in the graph

- **linewidth** tells us about the thickness scale of the curve.
- `plt.grid(True)` is used to add a grid at the back of your graph

The output looks like this :



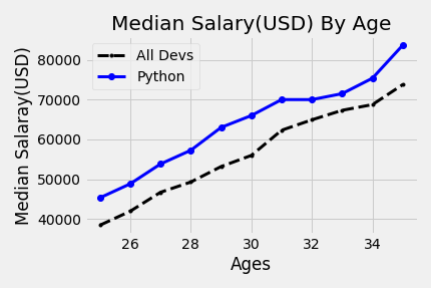
Style

We can also plot our graphs using different styles like `fivethirtyeig ht` , `ggplot` , `xkcd` and so on. Each style has its own look. All you need to is just add a single line of code before the whole snippet and run.

For example, if you add

```
plt.style.use('fivethirtyeight')
```

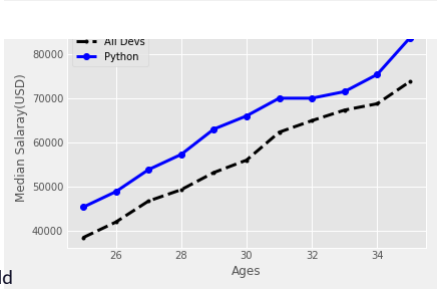
output:



If you add

```
plt.style.use('ggplot')
```

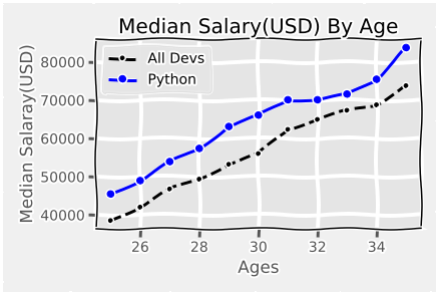
then you will have the below output:



If you add

```
plt.xkcd()
```

then you will have a cool comic look like below:



Warning! xkcd() is a method and it has braces with it.

Now, I hope you understand the differences between various styles.

Connect with me on Twitter | Linkedin

If you read this far, tweet to the author to show them you care.

Tweet a thanks

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

Get started

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.