Data Visualization with Python

Session-1

CLARUSWAY©
WAY TO REINVENT YOURSELF

Data Visualization with Python
Course Info

Course Duration 14 May - 24 May

6 Session, 2 Lab, **20 Hours in Total**

Structure of Course

- **Matplotlib (2 Session and 1 lab)**
- **Seaborn (3 Session and 1 lab)**
- **Project Solution (1 Session)**

Course Assignment and Projects

- **2 Assignment**
- **1 Mini Project**

Part 1 - Matplotlib

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

I have completed the **pre-class content.**

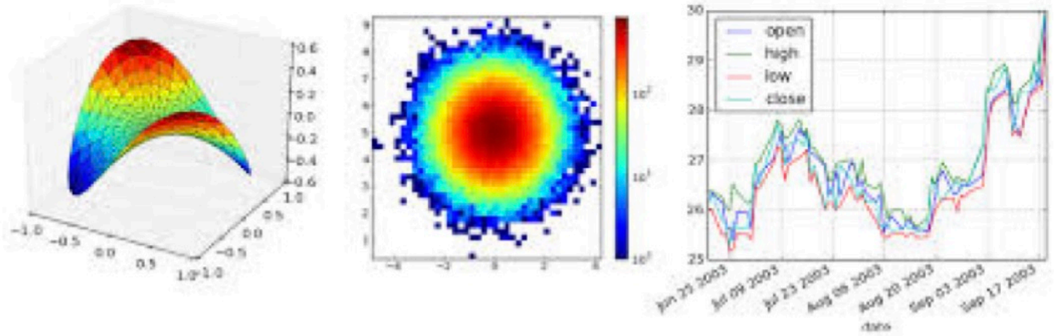
Students choose an option

Pear Deck Interactive Slide
Do not remove this bar

No Multiple Choice Response
You didn't answer this question

Do you have experience about Matplotlib ?

matplotlib



Students choose an option

Pear Deck Interactive Slide
Do not remove this bar

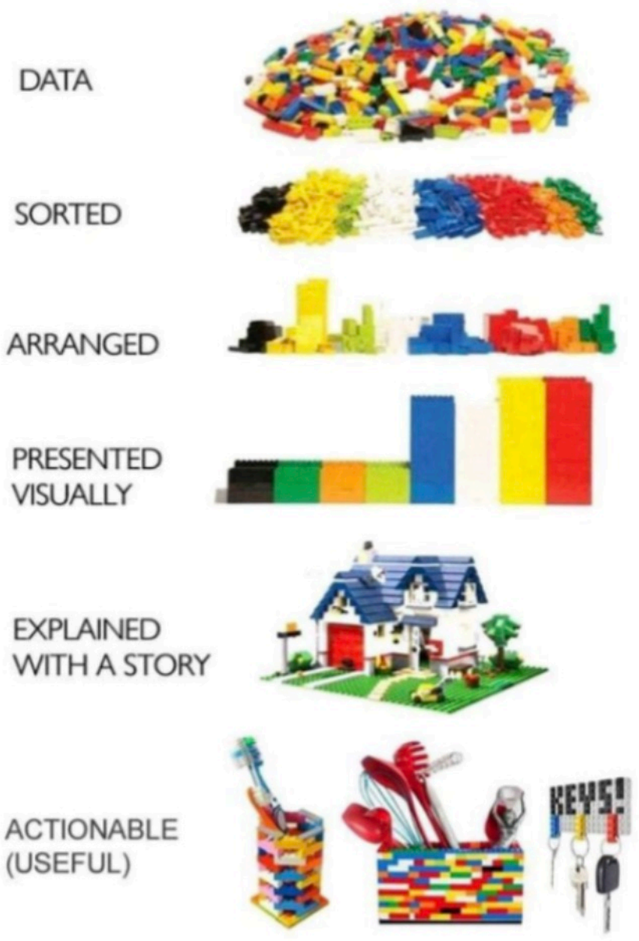
No Multiple Choice Response
You didn't answer this question

Why Data Visualization?

- Displaying statistical summaries
- Exploring data structure
- A picture is worth a thousand words
- Identifying patterns and clusters
- Evaluating model performance
- Interpreting the data to gain information
- Presenting results

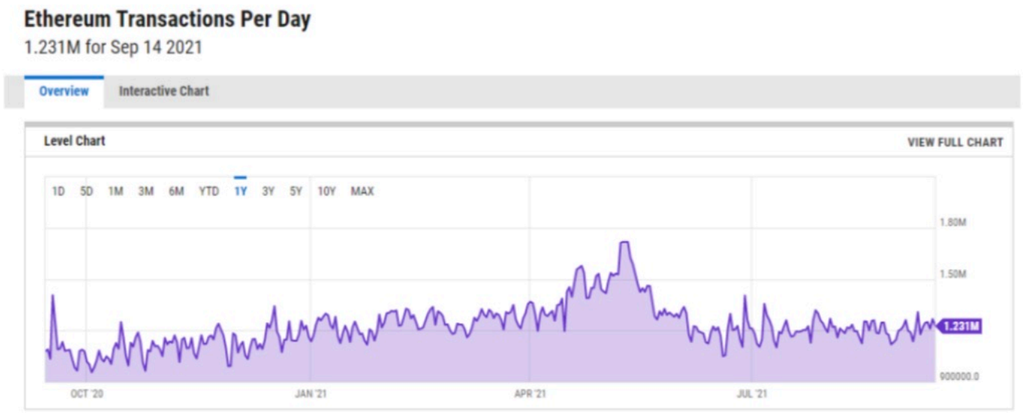
Why Data Visualization?

Explains data with Lego!



What Do you Understand?

- Ethereum Transactions Per Day reflects the daily number of transactions completed on the Ethereum network.
- Ethereum Transactions Per Day is at a current level of 1.231M, down from 1.267M yesterday and up from 1.080M one year ago.
- This is a change of -2.86% from yesterday and 14.04% from one year ago.



Students, write your response!

No Text Response

You didn't answer this question

Scope of the Course


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

```
import seaborn as sns
```

```
import pandas as pd
```

Create figure, axes, subplots

Built on matplotlib and can be used together with it

No need to import matplotlib or seaborn

Seaborn vs. Matplotlib

FEATURES	MATPLOTLIB	SEABORN
Functionality	It is utilized for making basic graphs. Datasets are visualised with graphs styles. • Bar graphs, • Histograms, • Pie charts, • Scatter plots, • Lines and so on.	Seaborn contains a number of patterns and plots for data visualization. It uses fascinating themes. It helps in compiling whole data into a single plot.
Syntax	It uses comparatively complex and lengthy syntax.	It uses comparatively simple syntax which is easier to learn and understand.

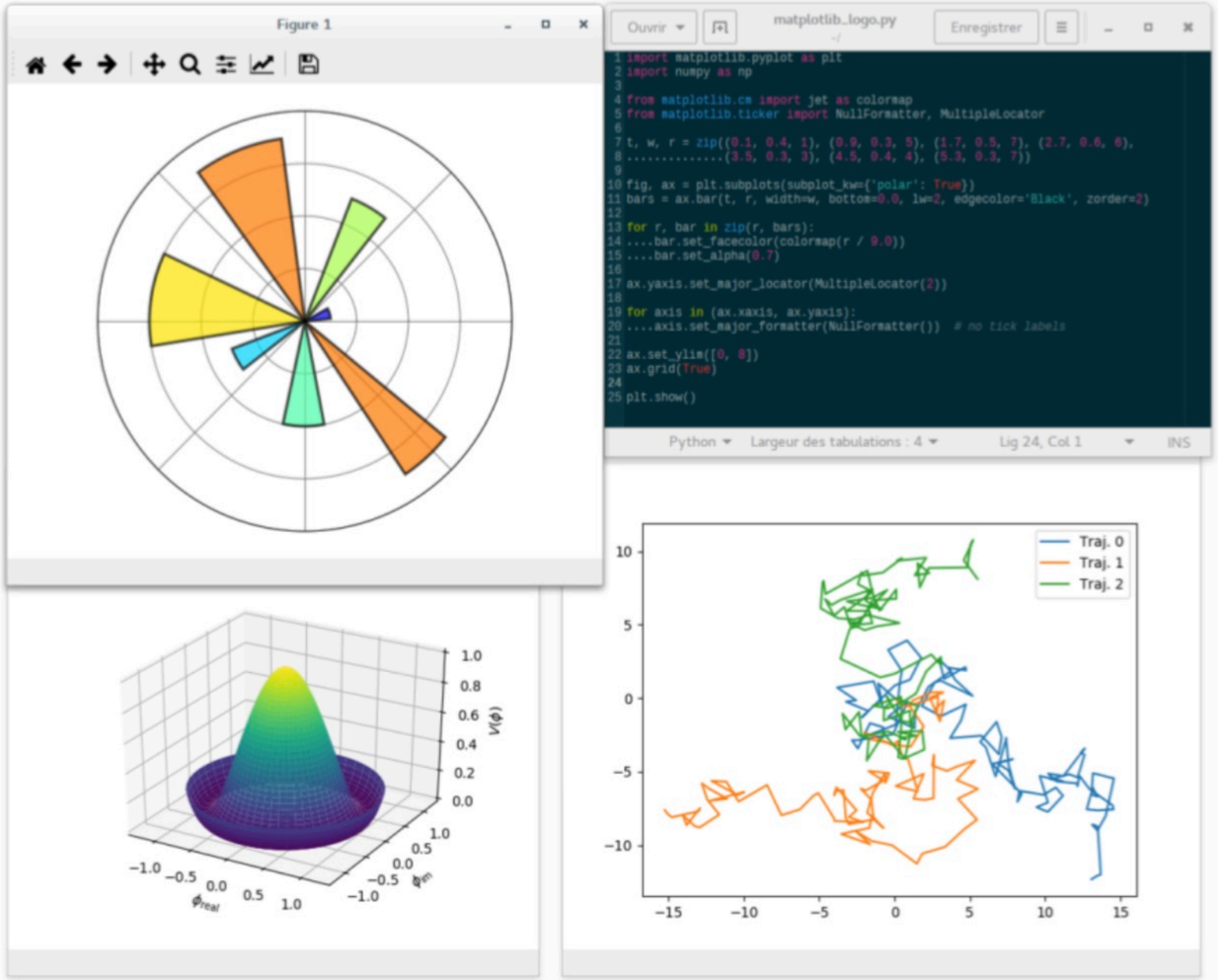
Seaborn vs. Matplotlib

FEATURES	MATPLOTLIB	SEABORN
Visualization	Matplotlib is well connected with Numpy and Pandas and acts as a graphics package for data visualization in python. Pyplot provides similar features and syntax as in MATLAB. Therefore, MATLAB users can easily study it.	Seaborn is more comfortable in handling Pandas data frames. It specializes in statistics visualization.
Pliability	Matplotlib is a highly customized.	Seaborn avoids overlapping of plots with the help of its default themes.

Introduction to Matplotlib

- In 2002 John Hunter created Matplotlib to try to replicate MatLab's plotting capabilities in Python.

- https://matplotlib.org/stable/gallery/index.html



Install Matplotlib

- Install command line in jupiter : !pip install matplotlib
- Install in conda prompter : conda install matplotlib

The version string is stored under __version__ attribute.

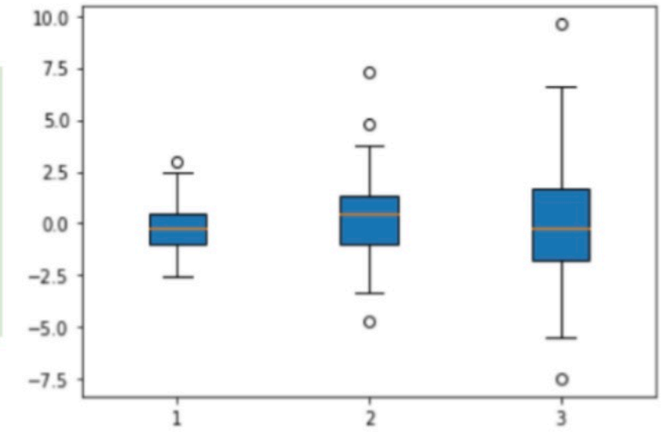
```
import matplotlib
print(matplotlib.__version__)
```

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

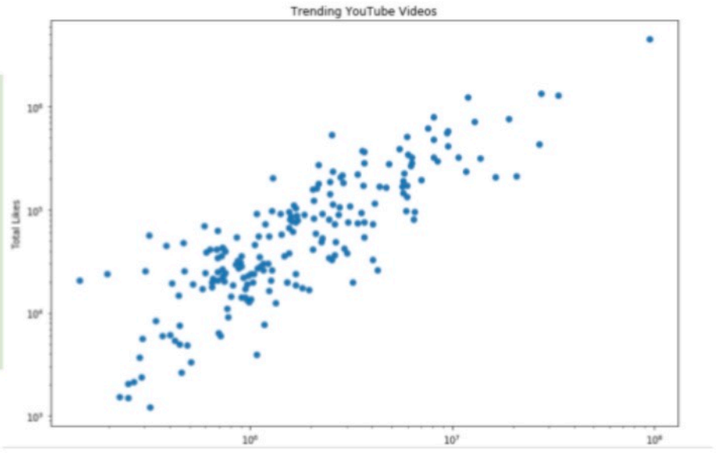
```
import matplotlib.pyplot as plt
```

Special Plot Types

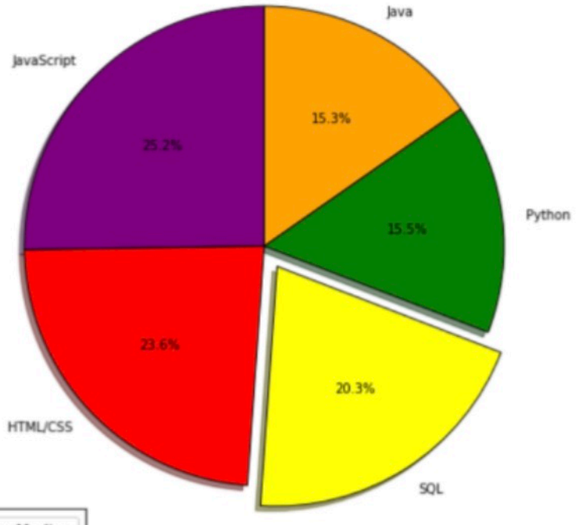
Box Plot



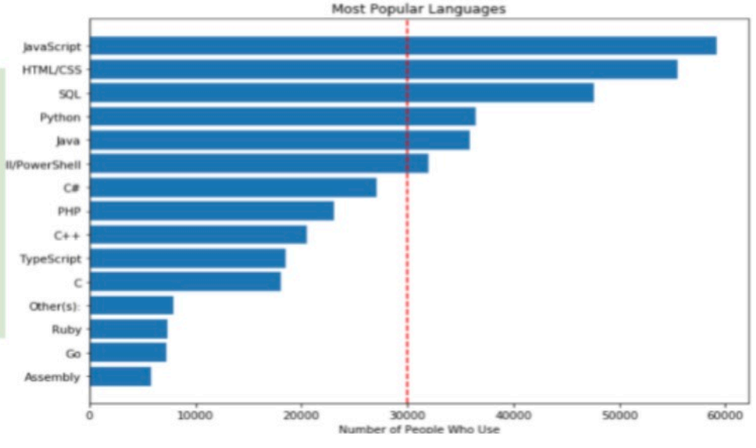
Scatter Plot



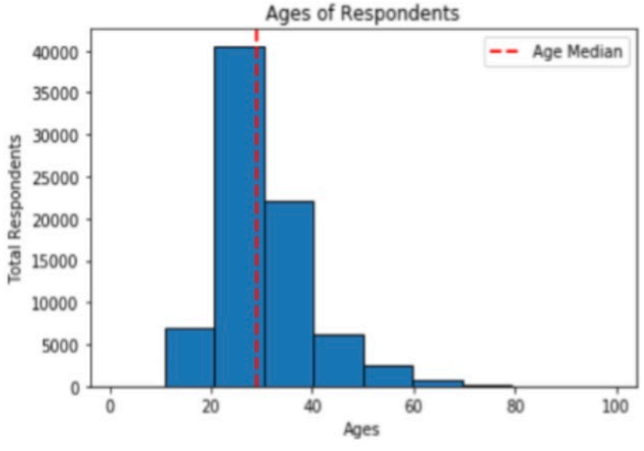
Pie Chart



Bar Plot

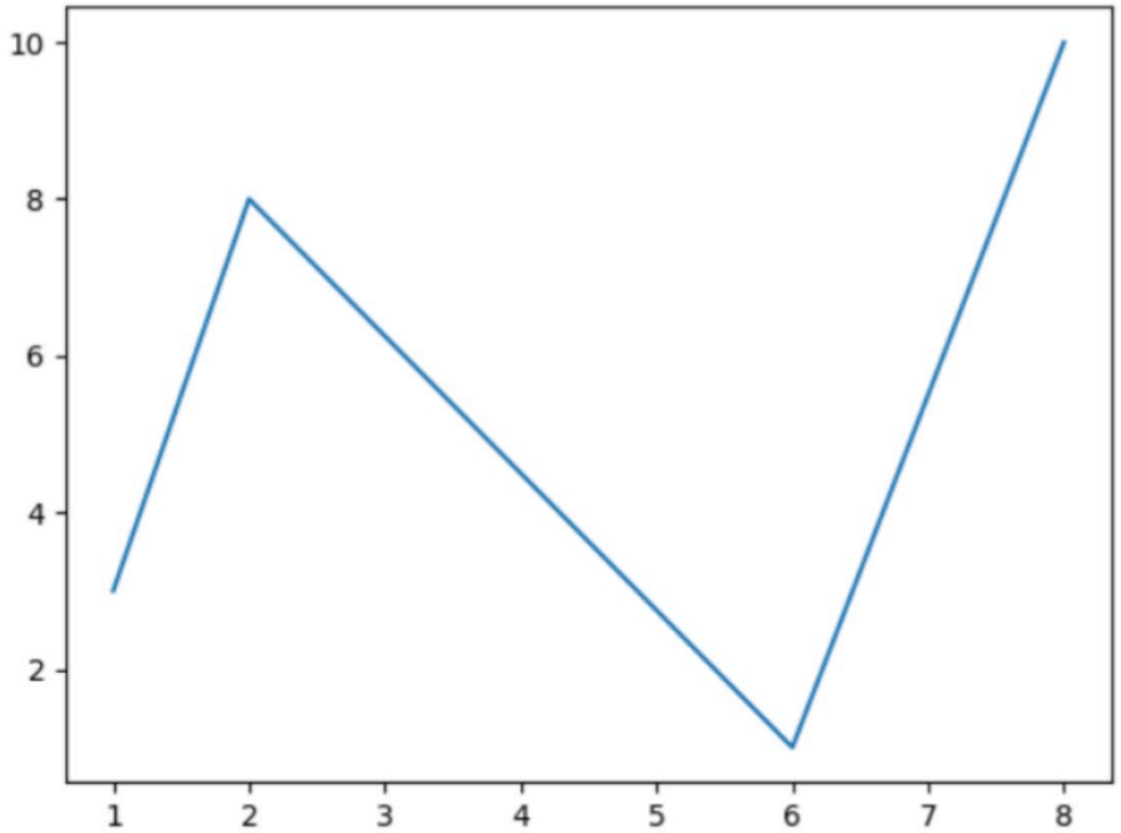


Histogram Plot



Matplotlib Plotting

```
import matplotlib.pyplot as plt
import numpy as np

x_points = np.array([1, 2, 6, 8])
y_points = np.array([3, 8, 1, 10])

plt.plot(x_points, y_points)
plt.show()
```



Two Methods

Functional

```
plt.plot(age, salary)
plt.xlabel("age")
plt.ylabel("salary")
plt.title("Salary by Age")

plt.show()
```

Object Oriented

```
fig, ax = plt.subplots()

ax.plot(age, salary, "r")
ax.set_xlabel("Age")
ax.set_ylabel("Salary")
ax.set_title("Salary by Age")
```

Two Methods

A note on the Object-Oriented API vs. Pyplot

Matplotlib has two interfaces. The first is an object-oriented (OO) interface. In this case, we utilize an instance of `axes.Axes` in order to render visualizations on an instance of `figure.Figure`.

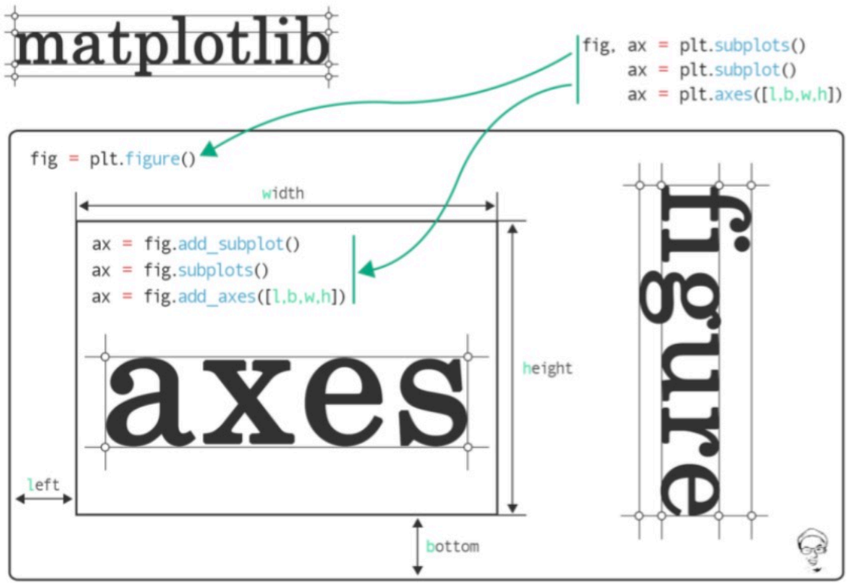
The second is based on MATLAB and uses a state-based interface. This is encapsulated in the `pyplot` module. See the pyplot tutorials for a more in-depth look at the pyplot interface.

Most of the terms are straightforward but the main thing to remember is that:

- The Figure is the final image that may contain 1 or more Axes.
- The Axes represent an individual plot (don't confuse this with the word "axis", which refers to the x/y axis of a plot).

source: https://matplotlib.org/stable/tutorials/introductory/lifecycle.html

What do you think?



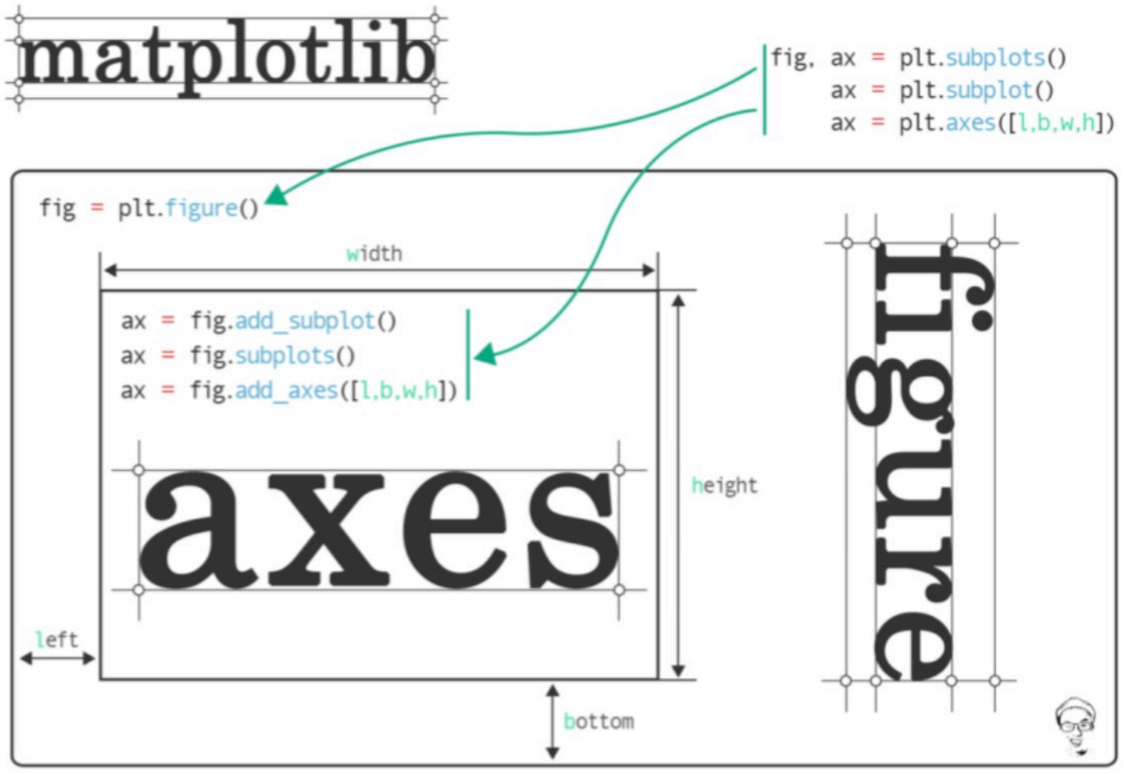
How do you describe figure & axes?

No Text Response

You didn't answer this question

Students, write your response!

Figure and Axes Notions

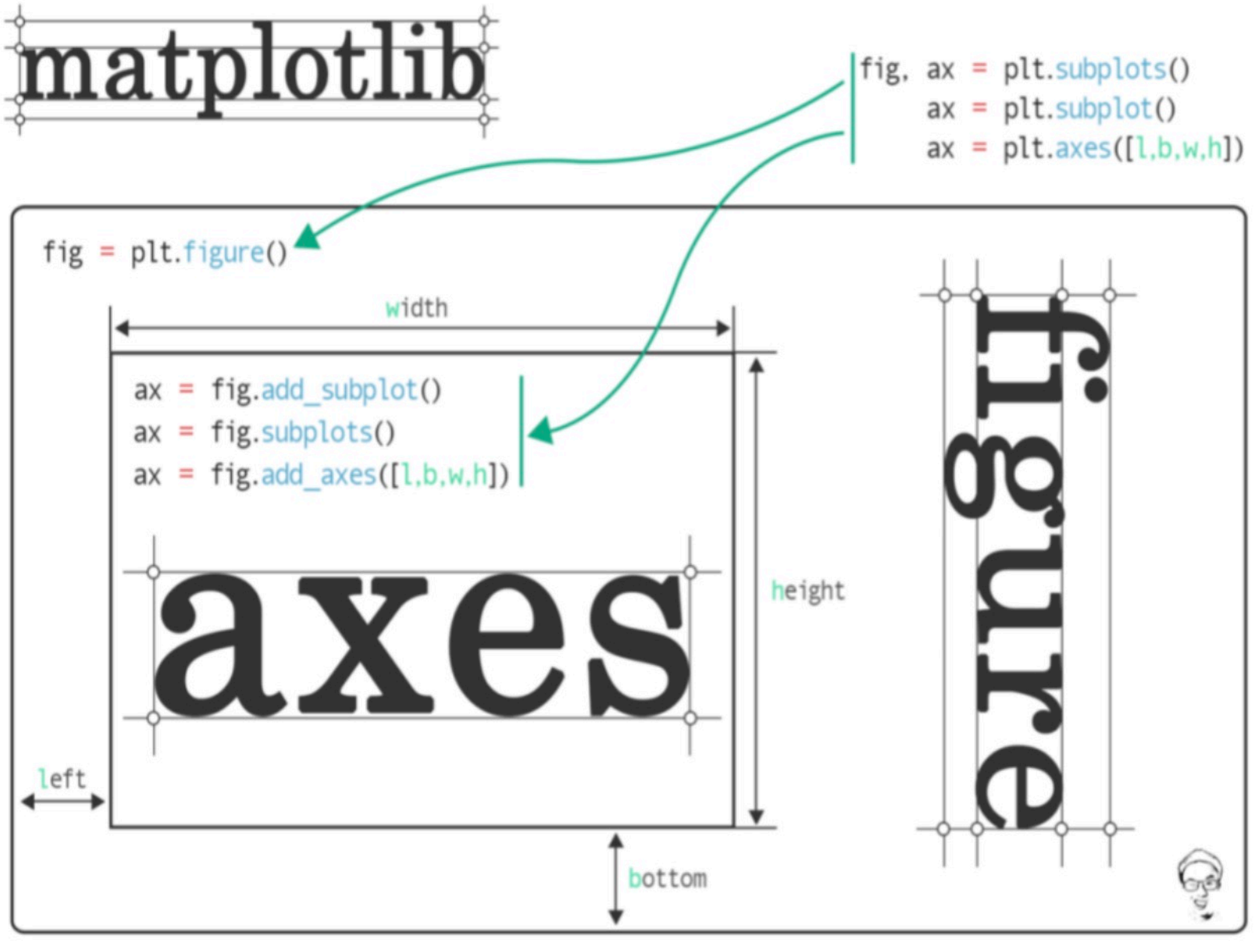


`Figure`

The top level container for all the plot elements.

- The Figure is the final image that may contain 1 or more Axes.
- The Axes represent an individual plot (don't confuse this with the word "axis", which refers to the x/y axis of a plot).

Create “figure”, Add “axes”



Manual

```
# Create Figure (empty canvas)
fig = plt.figure()
# Add set of axes to figure manually
axes = fig.add_axes([0.1, 0.1, 0.8, 0.8])
```

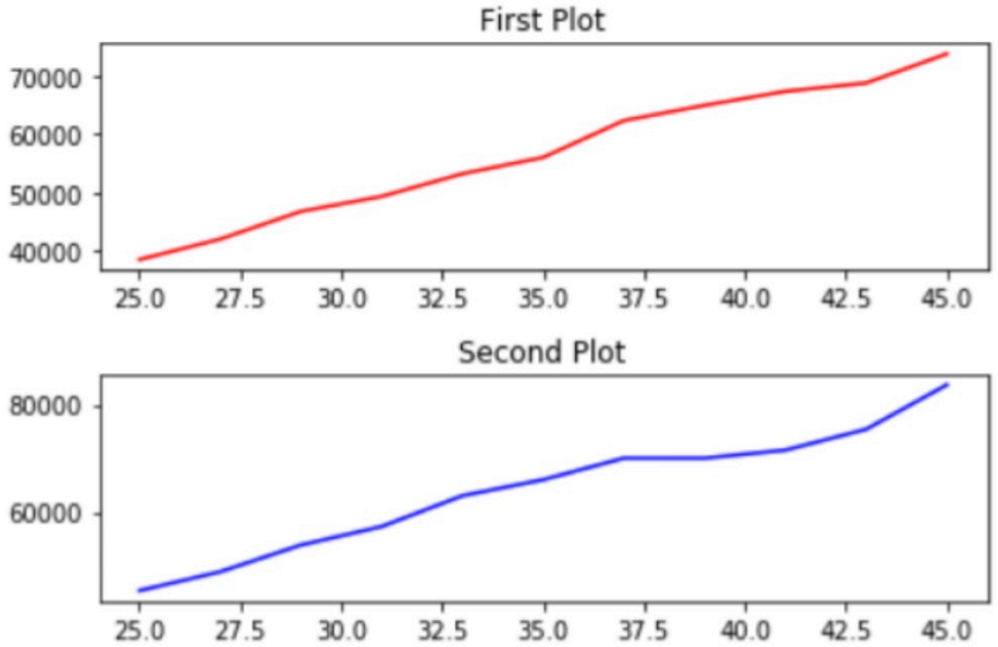
Automatic

```
fig, ax = plt.subplots()
```

Figure and Axes Notions

1 Figure, 2 Axes

```
fig, ax = plt.subplots(nrows=2, ncols=1)
ax[0].plot(age, salary, 'r')
ax[0].set_title('First Plot')

ax[1].plot(age, salary_2, 'b')
ax[1].set_title('Second Plot')
plt.tight_layout()
```

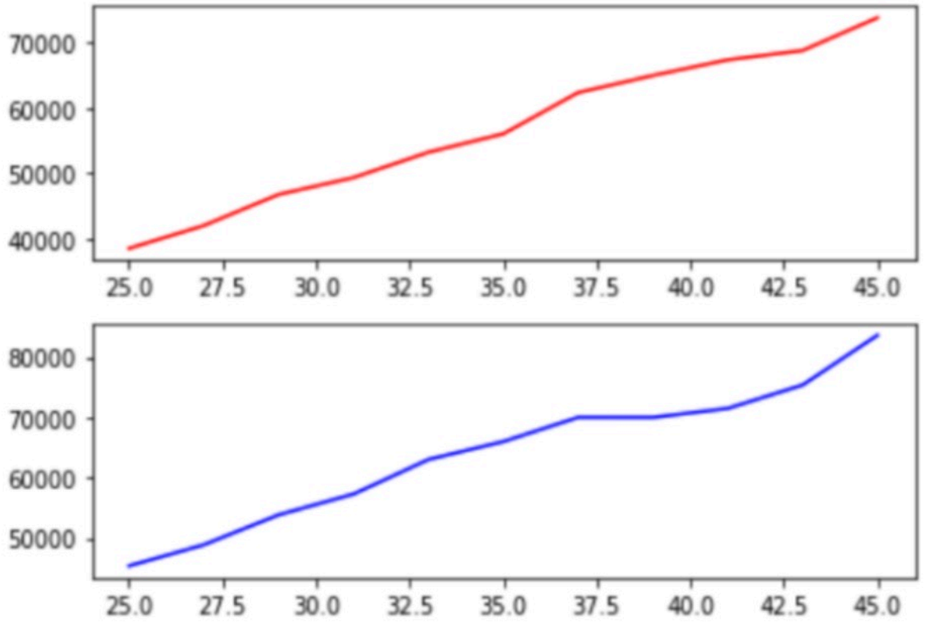


- The Figure is the final image that may contain 1 or more Axes.
- The Axes represent an individual plot (don't confuse this with the word "axis", which refers to the x/y axis of a plot).

Subplots

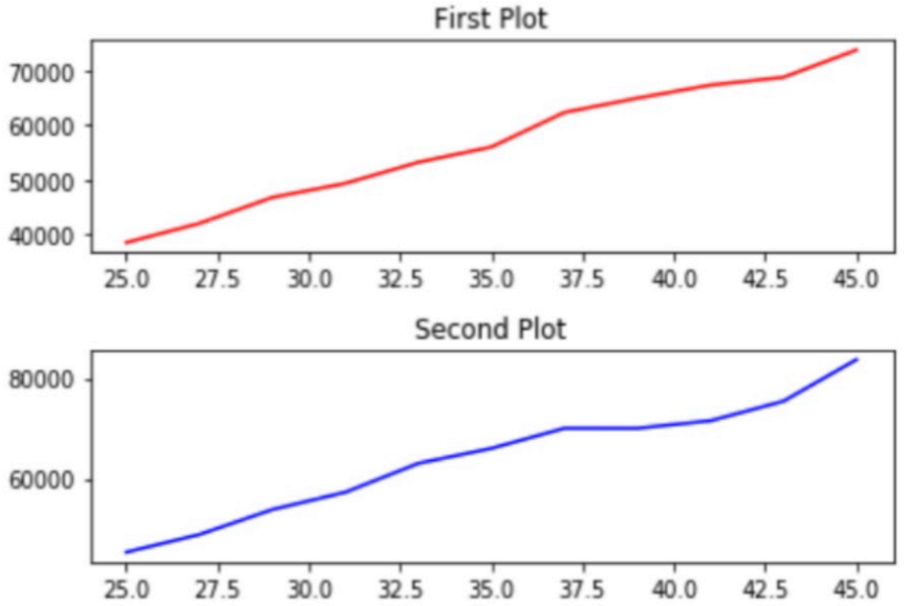
Functional

```
# plt.subplot(nrows, ncols, plot_number)
plt.subplot(2,1,1)
plt.plot(age, salary, 'r') # More on color options later
plt.subplot(2,1,2)
plt.plot(age, salary_2, 'b')
plt.tight_layout()
```

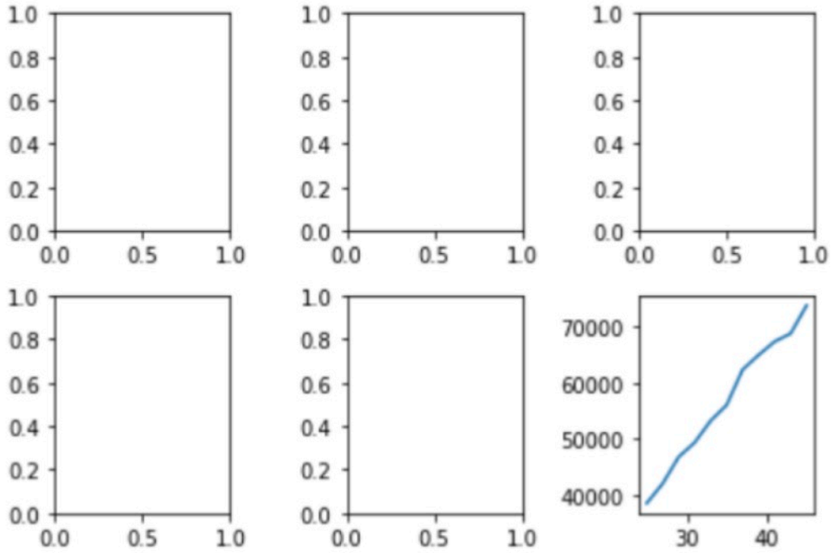


Object Oriented

```
fig, ax = plt.subplots(nrows=2, ncols=1)
ax[0].plot(age, salary, 'r')
ax[0].set_title('First Plot')

ax[1].plot(age, salary_2, 'b')
ax[1].set_title('Second Plot')
plt.tight_layout()
```



What do you think?



How to complete the code?

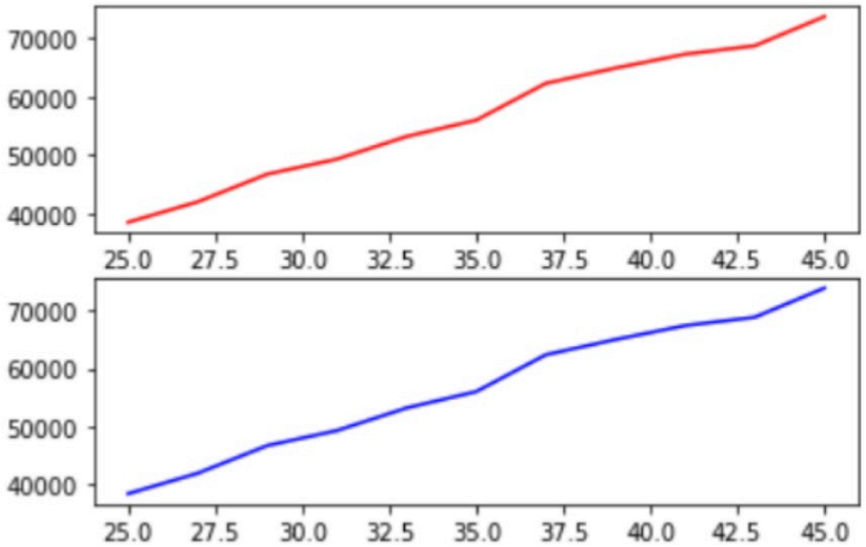
```
fig, ax = plt.subplots(2, 3)
…….. .plot(age, salary)
plt.tight_layout()
```

Students choose an option

No Multiple Choice Response

You didn't answer this question

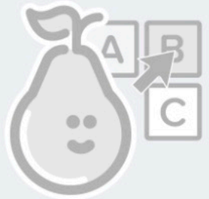
What do you think?



How to complete the code?

```
plt.subplot(.....)
plt.plot(age, salary, 'r')
plt.subplot(2,1,2)
plt.plot(age, salary, 'b')
```

Students choose an option



No Multiple Choice Response

You didn't answer this question

Figure Size

```
fig, ax = plt.subplots(figsize=(6,3)) #change the figsize

ax.plot(age, salary, 'r')
ax.set_xlabel('age')
ax.set_ylabel('salary')
ax.set_title('title')
```

```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(6,3))

ax[0].plot(age, salary)
ax[0].set_xlabel('age')
ax[1].plot(age, salary_2)
ax[0].set_title('First')
ax[1].set_title('Second')
ax[1].set_xlabel('age')
plt.tight_layout()
```

Linestyles, Markers

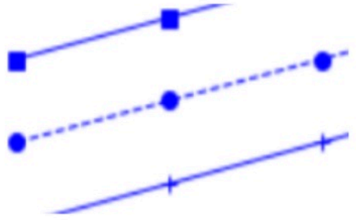
ls (linestyle)

```
linestyle='-')
ls='-.')
ls=':')
```

Character	Description
'-'	Solid line
'--'	Dashed line
'-.'	Dash-dot line
':'	Dotted line
'H'	Hexagon marker

marker

```
marker='+')
 marker='o')
marker='s')
```



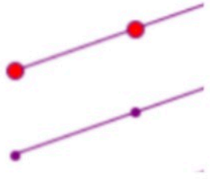
Character	Description
'.'	Point marker
'o'	Circle marker
'x'	X marker
'D'	Diamond marker
'H'	Hexagon marker

Linewidths, Markersize, Color

lw (linewidth)

```
lw=0.25)
lw=0.50)
linewidth=1.00)
```

markersize

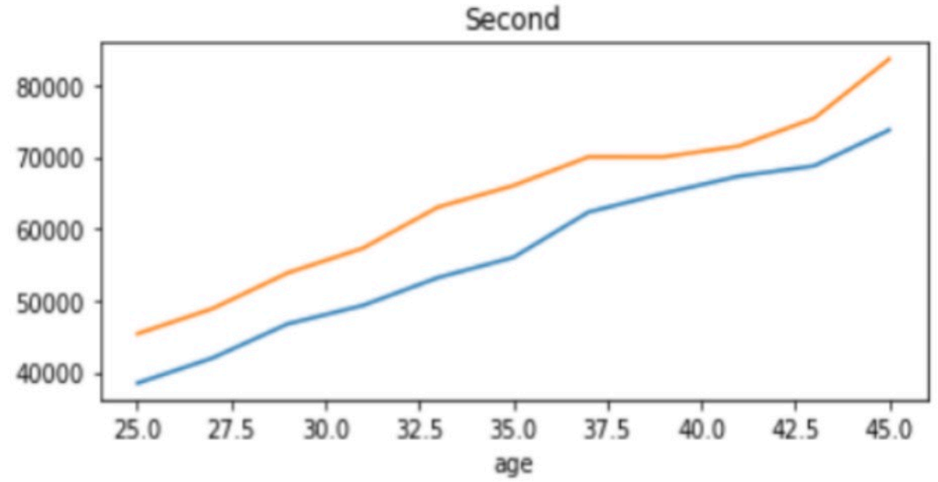


```
markersize=2)
markersize=4)
```

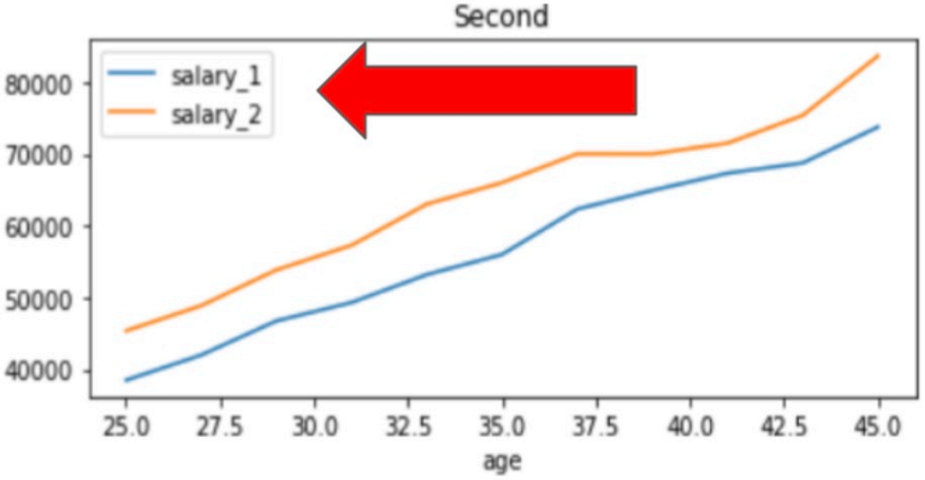
color

Character	Color
'b'	Blue
'g'	Green
'r'	Red
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'b'	Blue
'w'	White

Legends

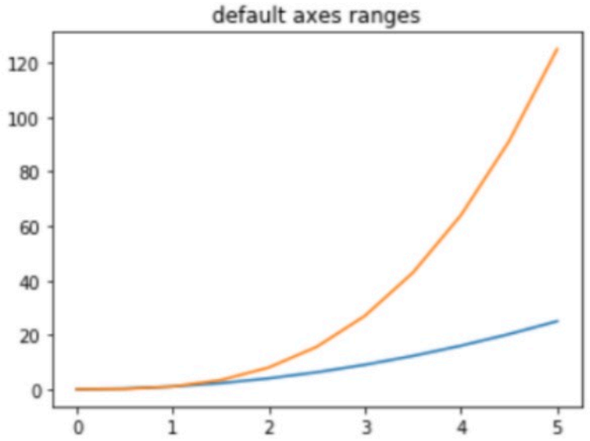


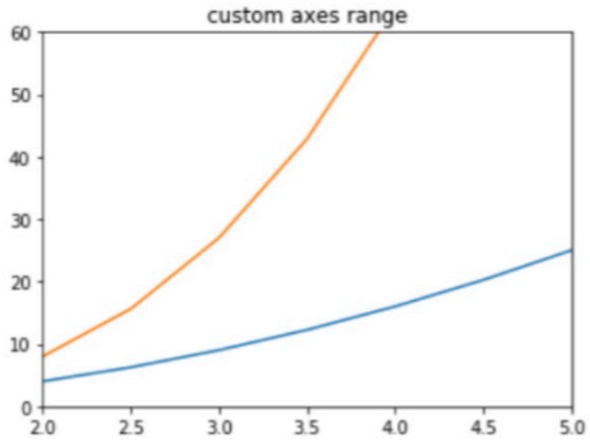

```
fig, ax = plt.subplots(figsize=(6,3))

ax.plot(age, salary)
ax.set_xlabel('age')
ax.plot(age, salary_2)
ax.set_title('First')
ax.set_title('Second')
ax.set_xlabel('age')
plt.tight_layout()
```




```
fig, ax = plt.subplots(figsize=(6,3))

ax.plot(age, salary, label="salary_1")
ax.set_xlabel('age')
ax.plot(age, salary_2, label="salary_2")
ax.set_title('First')
ax.set_title('Second')
ax.set_xlabel('age')
ax.legend()
plt.tight_layout()
```

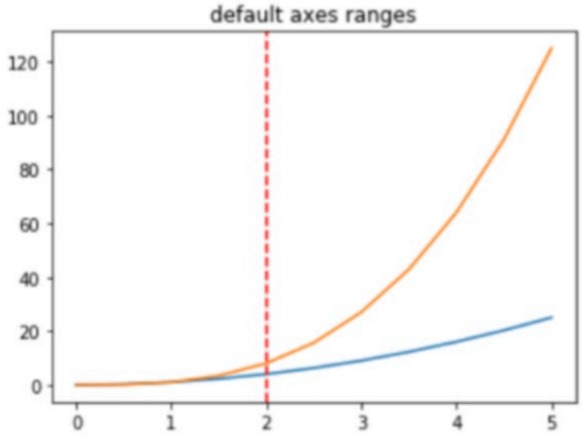
Plot Range and Adding Extra Line

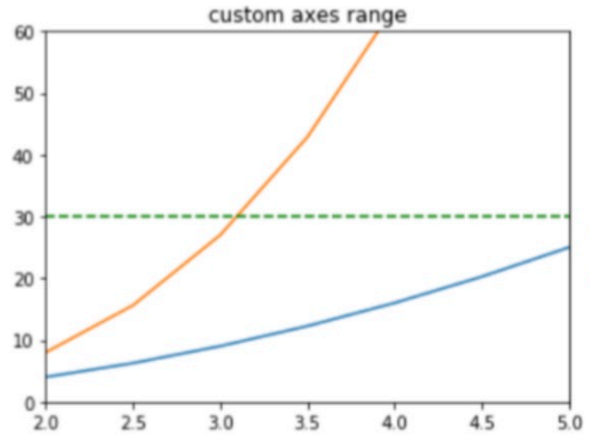




Plot Range

```
ax[1].plot(x, x**2, x, x**3)
ax[1].set_ylim([0, 60])
ax[1].set_xlim([2, 5])
ax[1].set_title("custom axes range")
```

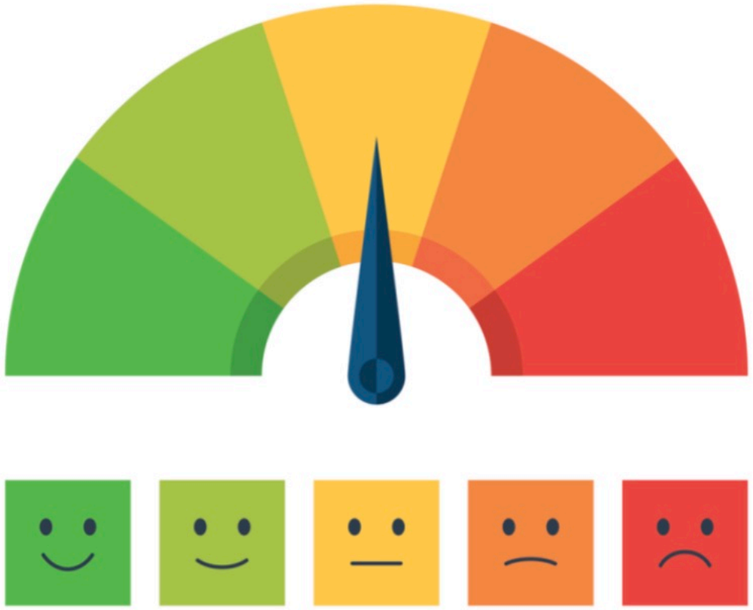




Adding Horizantal-Vertical Lines

```
ax[0].axvline(x=2, color='red', ls='--')
ax[1].axhline(y=30, color='green', ls='--')
```

Is everything clear so far?



Students choose an option

Pear Deck Interactive Slide
Do not remove this bar

No Multiple Choice Response
You didn't answer this question