Data Analytics Course - Lesson 02

Ths. Vu Duy Khuong

Agenda

- ❑I. Matplotlib Library
- ❑II. Import Dataset
- ❑III. Export Dataset
- ❑IV. Summary Dataset

I. Matplotlib Library

1. Matplotlib Libraries

- “matplotlib” is a library specialized in plotting, open wide from numpy

- Has the goal of simplifying the work of drawing diagrams as much as possible map to “just a few lines of command”

- Supports a wide variety of chart types, especially those used in research or economics such as line, line, histograms, spectrum, correlation, error charts, scatterplots, etc.

- The structure of matplotlib consists of many parts, serving for different uses

I. Matplotlib Library

1. Matplotlib Libraries

- In addition to the charting-related APIs, matplotlib also includes several interfaces: Object-Oriented API, The Scripting Interface (pyplot), The MATLAB Interface (pylab)

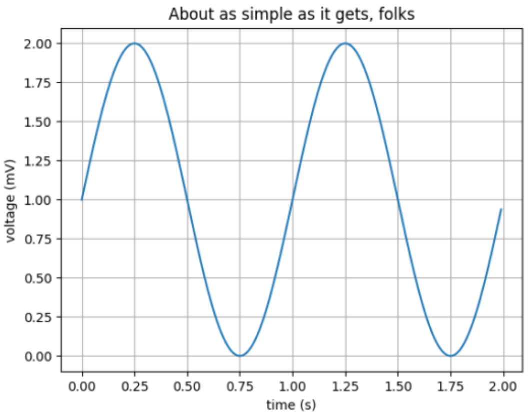
❑ These interfaces make it convenient for us to set up the index before making the chart

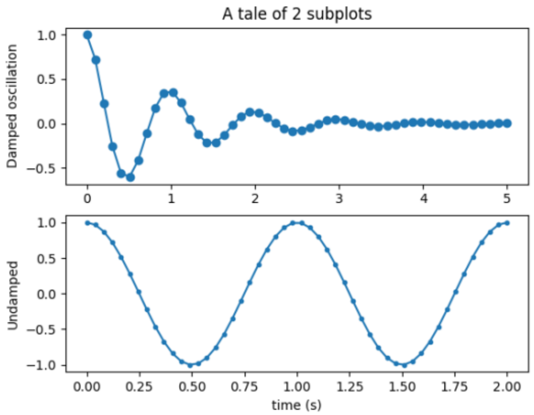
❑ The pylab interface is now out of development

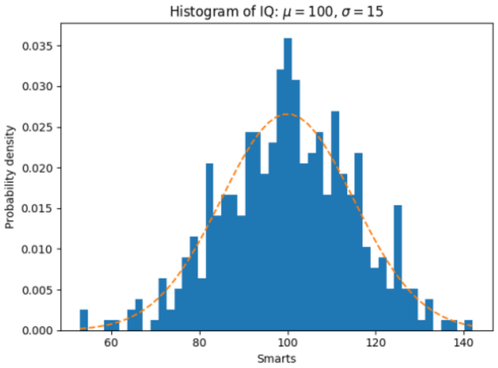
❑ Most of the examples in this slide use pyplot

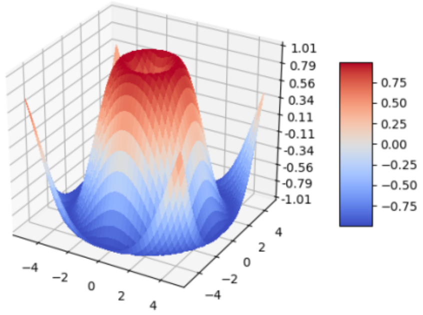
❑ Use the Object-Oriented API or directly matplotlib will allow deeper intervention in drawing chart (most projects won't have this need)

II. Some charts plotted with matplotlib

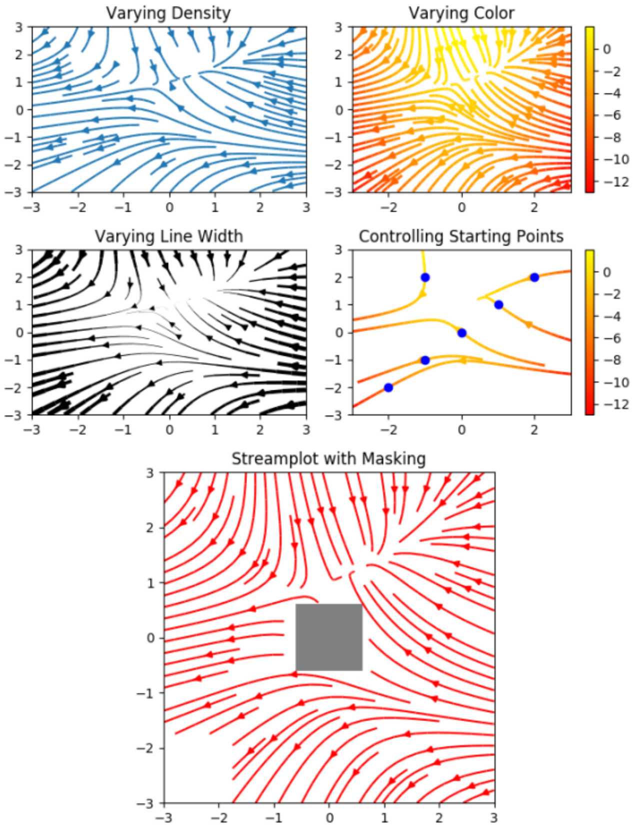

About as simple as it gets, folks

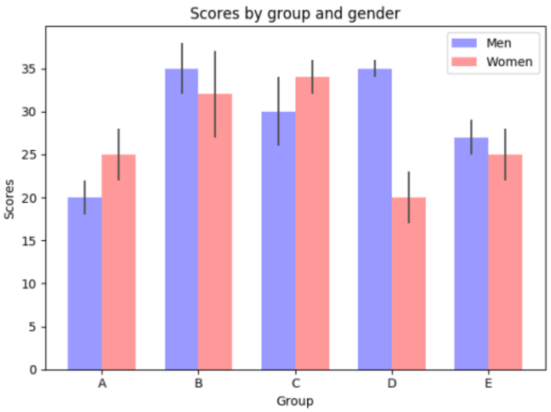

A tale of 2 subplots

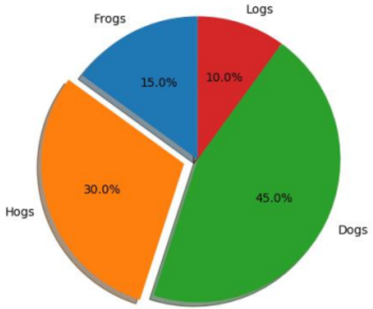

Histogram of IQ: $\mu = 100$, $\sigma = 15$



II. Some charts plotted with matplotlib







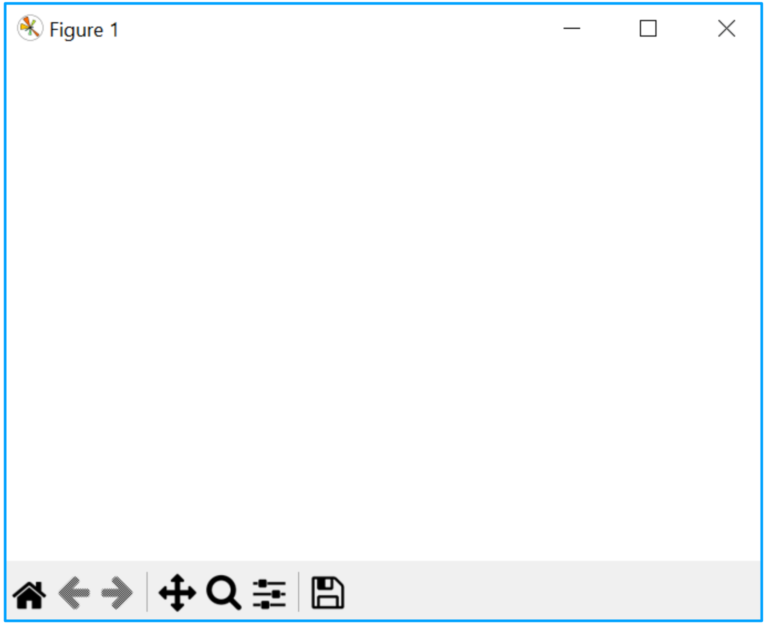
II. Figures and Subplots

Plots in matplotlib reside within a **Figure** object. We can create a new figure with

```
>>>  plt.figure()
```

Example :

```
>> plt.figure()
>> plt.show()
```

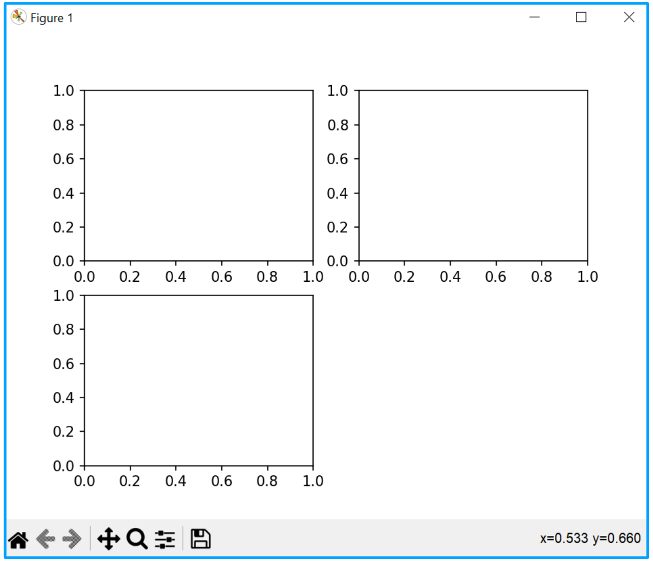


II. Figures and Subplots

We can't make a plot with a blank figure. We have to create one or more subplots using add_subplot:

Example :

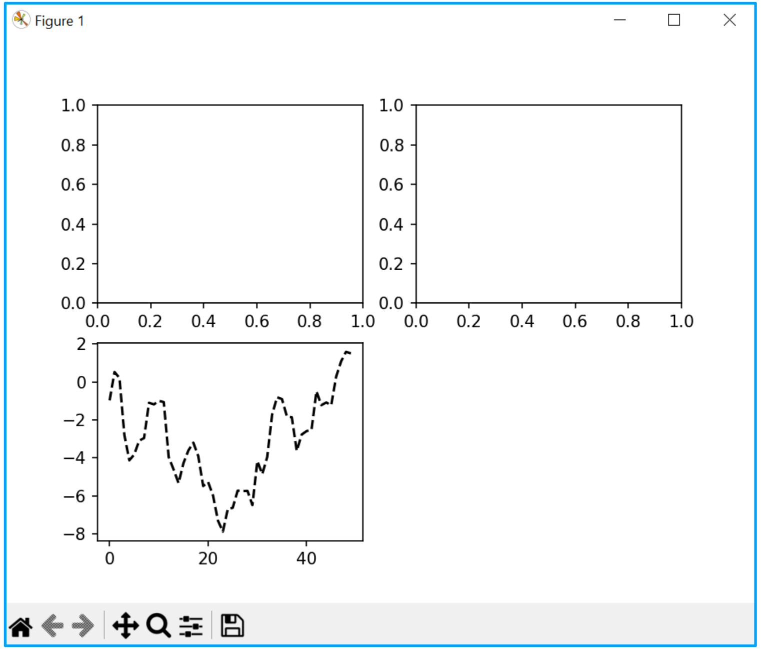
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```



This means that the figure should be 2 × 2, and we're selecting the first of 4 subplots

II. Figures and Subplots

When we issue a plotting command like `plt.plot([1.5, 3.5, -2, 1.6]),` matplotlib draws on the **last figure** and subplot used (creating one if necessary), thus hiding the figure and subplot creation



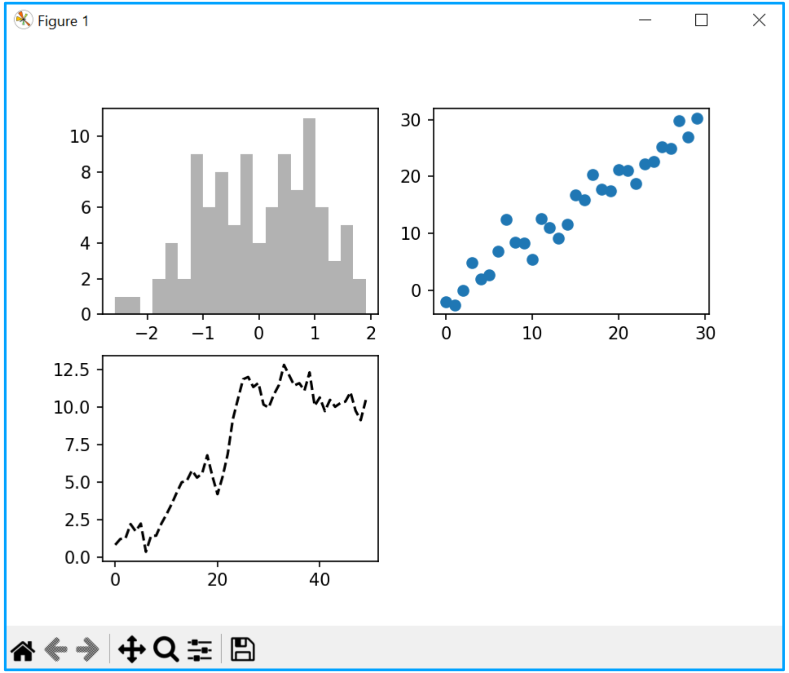
Example :

```
from numpy.random import randn
plt.plot(randn(50).cumsum(), 'k--’)
```

The **'k--'** is a style option instructing matplotlib to plot a black dashed line

II. Figures and Subplots

The objects returned by fig.add_subplot above are AxesSubplot objects, on which you can directly plot on the other empty subplots by calling each one’s instance methods



Example :

```
ax1.hist(randn(100), bins=20, color='k', alpha=0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
```

II. Figures and Subplots

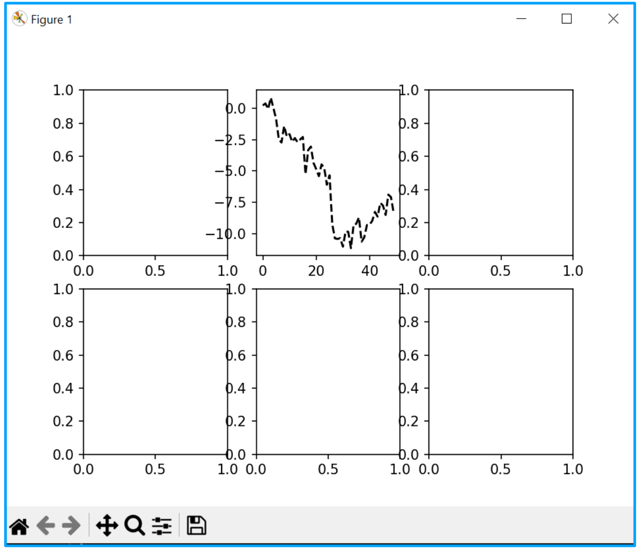
We can create many figures with multiple subplots according to a particular layout.

There is a convenience method, `plt.subplots` that creates a new figure

and returns a NumPy array containing the created subplot objects

Example :

```
fig, axes = plt.subplots(2, 3)
axes[0,1].plot(randn(50).cumsum(), 'k--')

plt.show()
```



II. Figures and Subplots

Some attribute of *plt.subplots* as belows

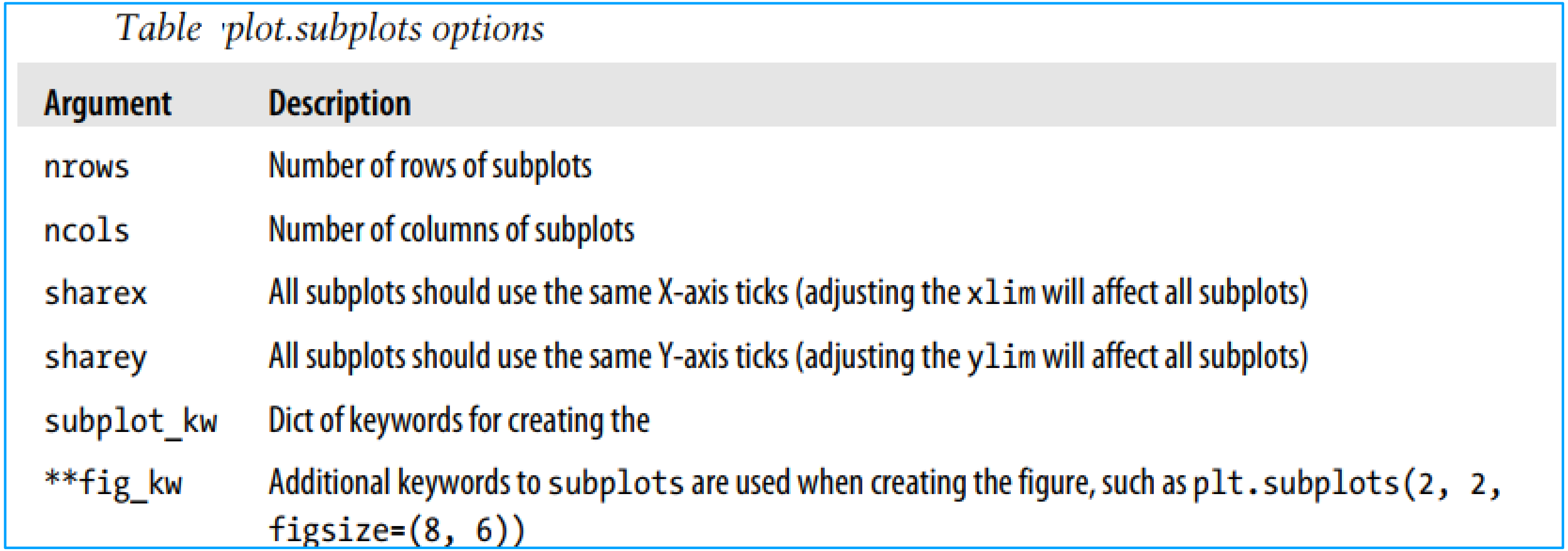


Table 'plot.subplots options

Argument	Description
nrows	Number of rows of subplots
ncols	Number of columns of subplots
sharex	All subplots should use the same X-axis ticks (adjusting the xlim will affect all subplots)
sharey	All subplots should use the same Y-axis ticks (adjusting the ylim will affect all subplots)
subplot_kw	Dict of keywords for creating the
**fig_kw	Additional keywords to subplots are used when creating the figure, such as plt.subplots(2, 2, figsize=(8, 6))

II. Figures and Subplots

Adjusting the spacing around subplots

By default matplotlib leaves a certain amount of padding around the outside of the subplots and spacing between subplots

The spacing can be most easily changed using the `subplots_adjust` Figure method, also available as a top-level function:

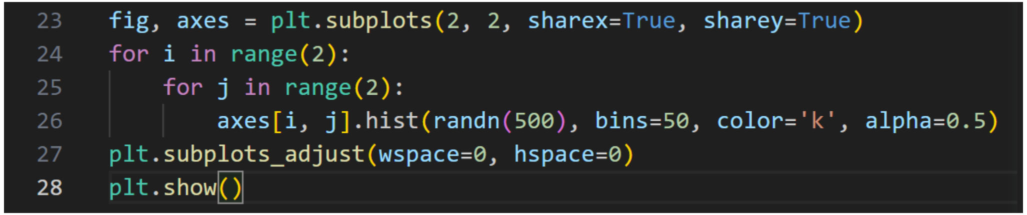
```
  subplots_adjust(left=None, bottom=None, right=None, top=None,
wspace=None, hspace=None)
```

`wspace` and `hspace` controls the percent of the figure width and figure height, respectively, to use as spacing between subplots

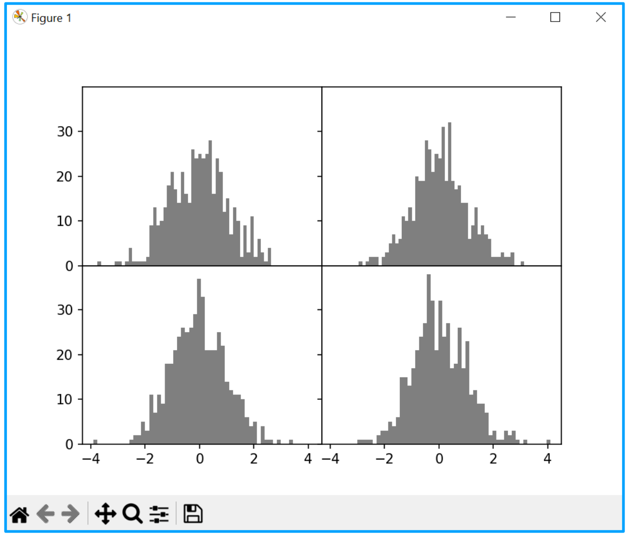
II. Figures and Subplots

Adjusting the spacing around subplots

Example : the spacing all the way to zero



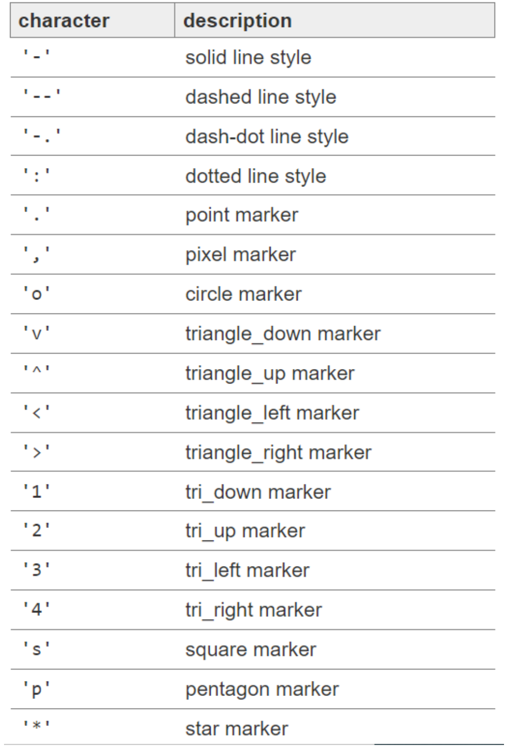
```python
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        axes[i, j].hist(randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
plt.show()
```



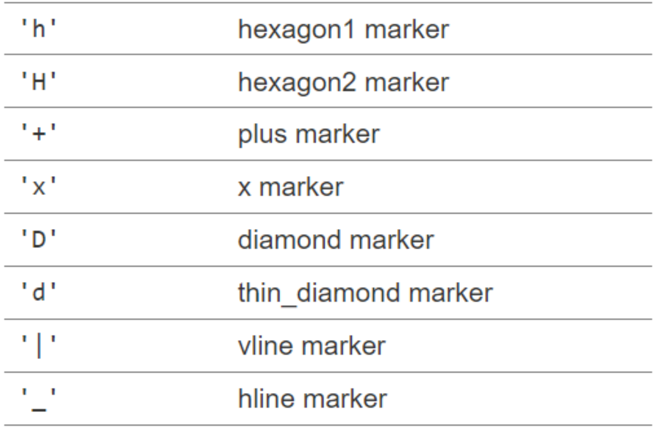
III. Colors, Markers, and Line Styles

The following format string characters are accepted to control the line style or marker:

character	color
‘b’	blue
‘g’	green
‘r’	red
‘c’	cyan
‘m’	magenta
‘y’	yellow
‘k’	black
‘w’	white



character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker



'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

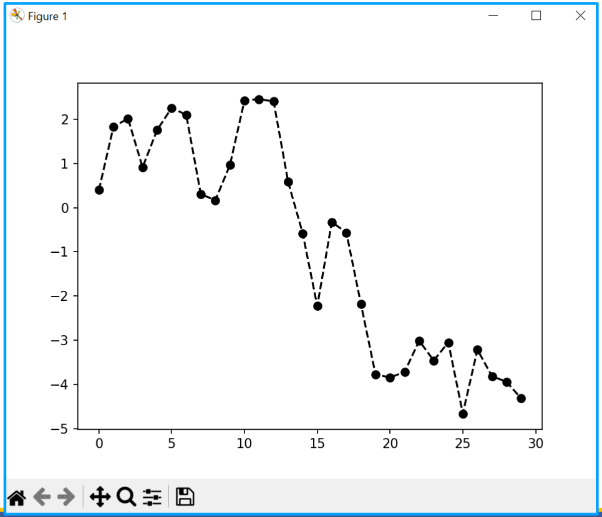
III. Colors, Markers, and Line Styles

Example 1 :

```
 ax.plot(x, y, linestyle='--', color='g’)
```

Example 2 :

```
 plt.plot(randn(30).cumsum(), color='k', linestyle='dashed',
marker='o’)
plt.show()
```



III. Ticks, Labels, and Legends

Ticks are the markers denoting data points on the axes and tick labels are the name given to ticks.

By default matplotlib itself marks the data points on the axes but it has also provided us with setting their own axes having ticks and tick labels of their choice.

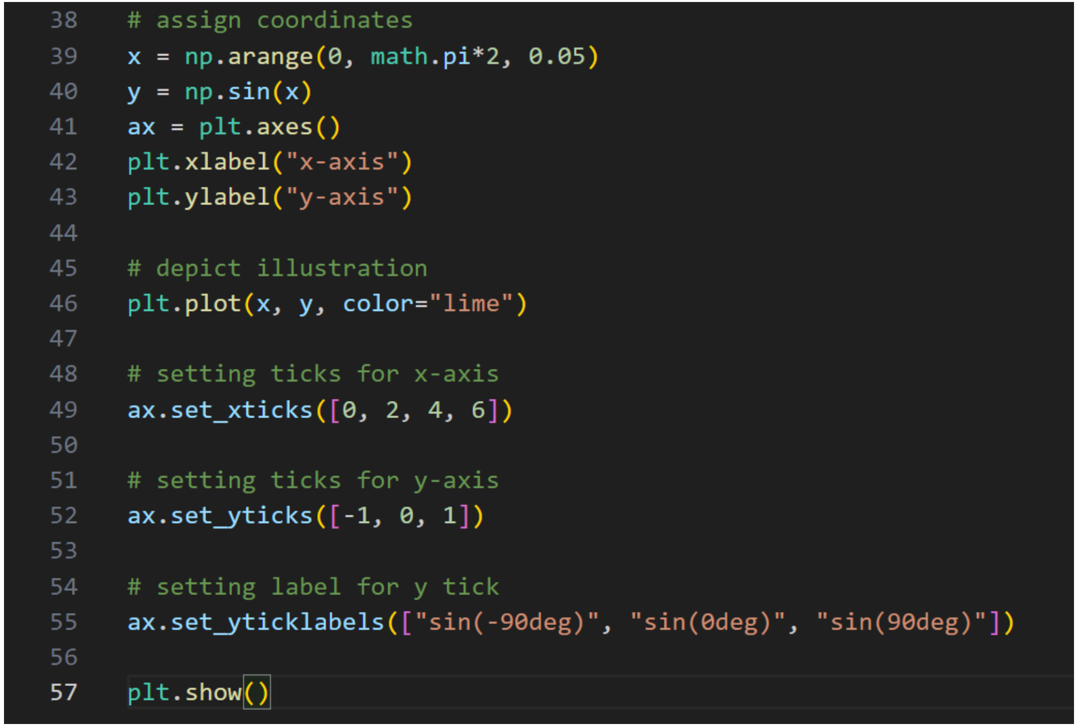
Methods used:

plt.axes().set_xticks() and **plt.axes().set_yticks()** : For setting ticks on x-axis and y-axis respectively. having data in form of a list set as parameter

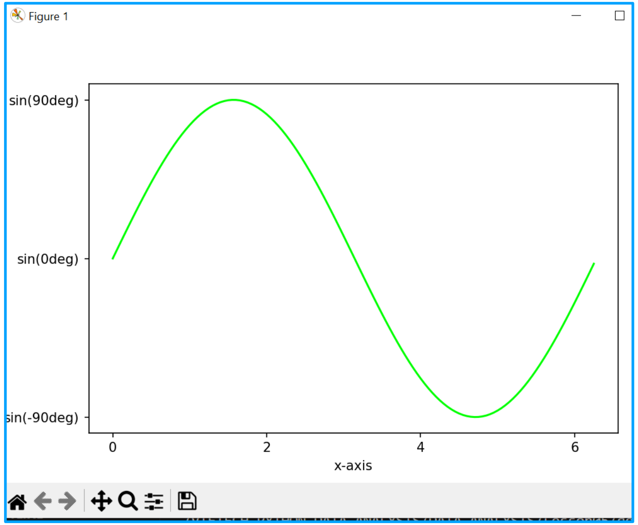
plt.axes().set_xlabels() and **plt.axes().set_ylabels()** : To set labels of our ticks with parameters in form of list.

III. Ticks, Labels, and Legends

Ticks are the markers denoting data points on the axes and tick labels are



```
# assign coordinates
x = np.arange(0, math.pi*2, 0.05)
y = np.sin(x)
ax = plt.axes()
plt.xlabel("x-axis")
plt.ylabel("y-axis")

# depict illustration
plt.plot(x, y, color="lime")

# setting ticks for x-axis
ax.set_xticks([0, 2, 4, 6])

# setting ticks for y-axis
ax.set_yticks([-1, 0, 1])

# setting label for y tick
ax.set_yticklabels(["sin(-90deg)", "sin(0deg)", "sin(90deg)"])

plt.show()
```



III. Saving Plots to File

The active figure can be saved to file using `plt.savefig`. This method is equivalent to the figure object’s `savefig` instance method

Example :

```
plt.savefig('figpath.png', dpi=400, bbox_inches='tight')
```

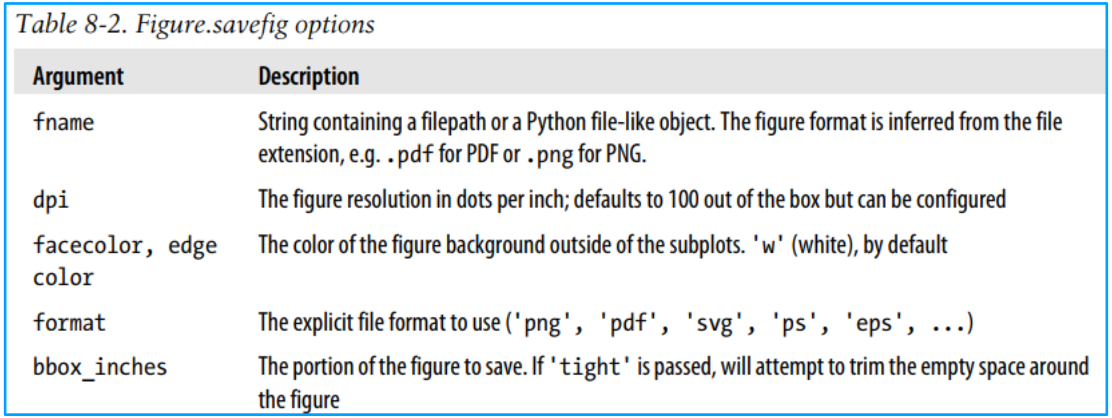
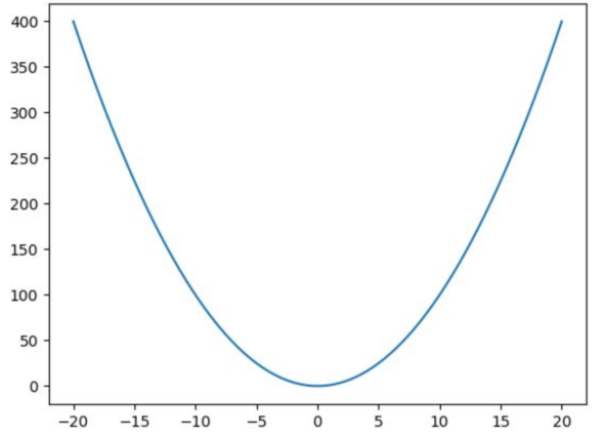


Table 8-2. Figure.savefig options

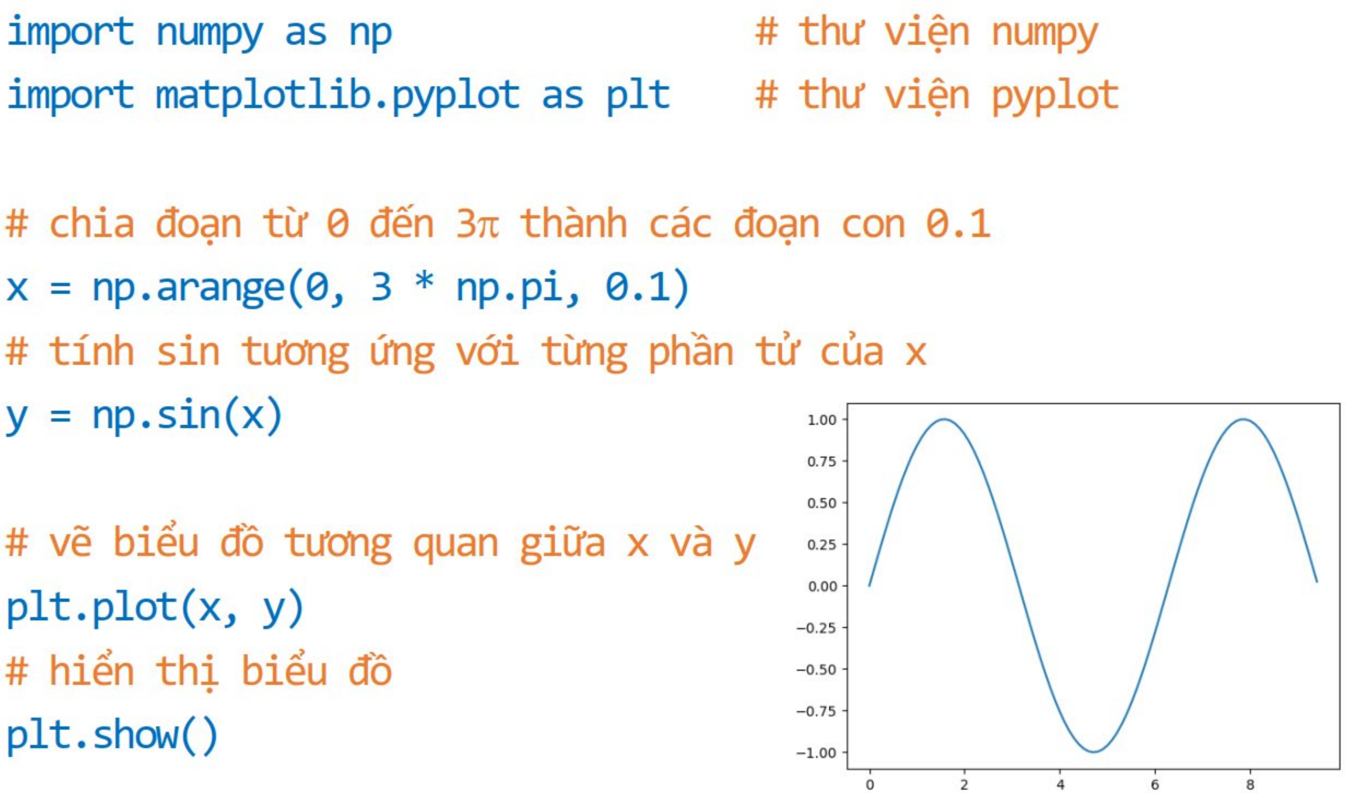
Argument	Description
`fname`	String containing a filepath or a Python file-like object. The figure format is inferred from the file extension, e.g. `.pdf` for PDF or `.png` for PNG.
`dpi`	The figure resolution in dots per inch; defaults to 100 out of the box but can be configured
`facecolor, edgecolor`	The color of the figure background outside of the subplots. `'w'` (white), by default
`format`	The explicit file format to use (`'png', 'pdf', 'svg', 'ps', 'eps', ...`)
`bbox_inches`	The portion of the figure to save. If `'tight'` is passed, will attempt to trim the empty space around the figure

IV. Example: plot y = x2

```
import numpy as np                    # thư viện numpy
import matplotlib.pyplot as plt       # thư viện pyplot

# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x * x

# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



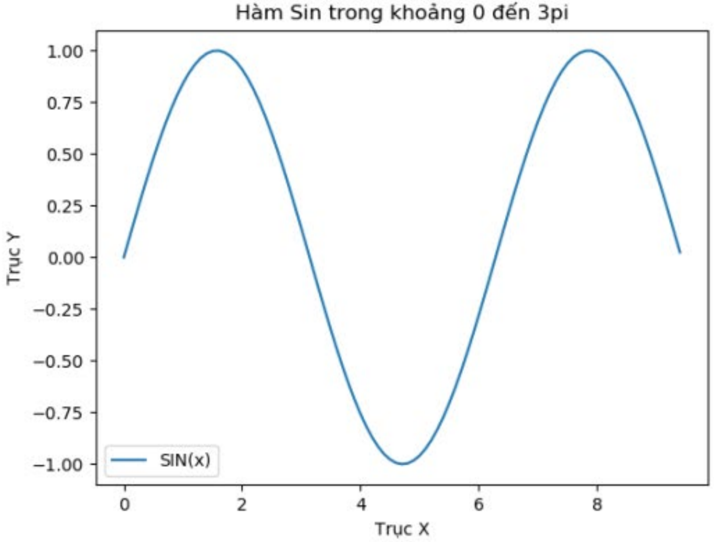
IV. Example: plot Sin



```
import numpy as np                    # thư viện numpy
import matplotlib.pyplot as plt       # thư viện pyplot

# chia đoạn từ 0 đến 3π thành các đoạn con 0.1
x = np.arange(0, 3 * np.pi, 0.1)
# tính sin tương ứng với từng phần tử x
y = np.sin(x)

# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```

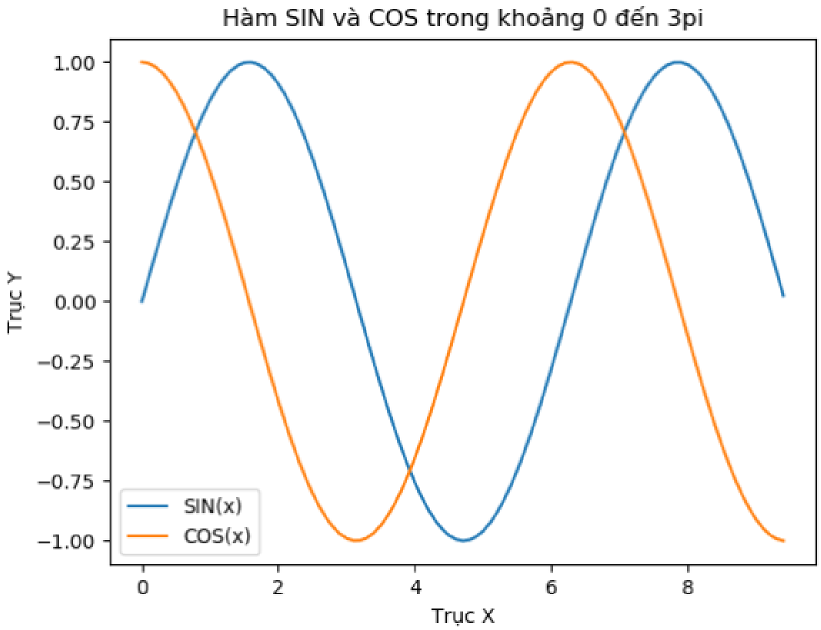
IV. Example: Full plot Sin chart

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

x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

plt.plot(x, y)
# các thông tin bổ sung cho biểu đồ
plt.xlabel('Trục X')
plt.ylabel('Trục Y')
plt.title('Hàm Sin trong khoảng 0 đến 3pi')
plt.legend(['SIN(x)'])
plt.show()
```



IV. Example: Full plot Sin & Cos chart

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

x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('Trục X')
plt.ylabel('Trục Y')
plt.title('Hàm SIN và COS trong khoảng 0 đến 3pi')
plt.legend(['SIN(x)', 'COS(x)'])
plt.show()
```



IV. Steps to plot with matplotlib

- Necessary condition: data already available
- There can be 4 basic steps:

1. Choose the right chart type
    - Depends a lot on the data type
    - Depends on the user's intended use
2. Set parameters for the chart
    - Parameters of axes, meaning, division ratio, etc.
    - Highlights on the map
    - Perspectives, swatches, colors and other details
    - Additional information
3. Draw the chart
4. Save to file

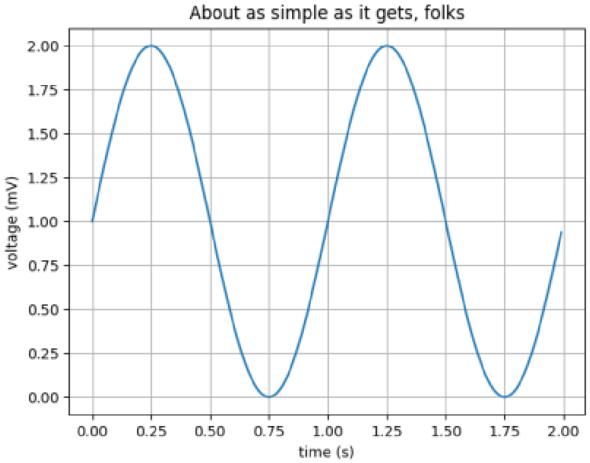
V. Some common chart types

❑ Line plot

- The graph shows the correlation between X and Y
- Syntax:

```
plot([x], y, [fmt], data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

- “fmt” is the line drawing specification
- “data” is the label of the data
- **kwargs: line drawing parameter
- Draw multiple times on one chart
- The return result is a list ofLine2D . object



V. Some common chart types

❑ Line plot: fmt

- “fmt” consists of 3 parts fmt = '[color][marker][line]’
- [color] – abbreviation of color name:
- ‘b’ – blue
- ‘g’ – green
- ‘r’ – red
- ‘c’ – cyan
- ‘m’ – magenta
- ‘y’ – yellow
- ‘b’ – black
- ‘w’ – white
- #rrggbb – specifies RGB color codes

V. Some common chart types

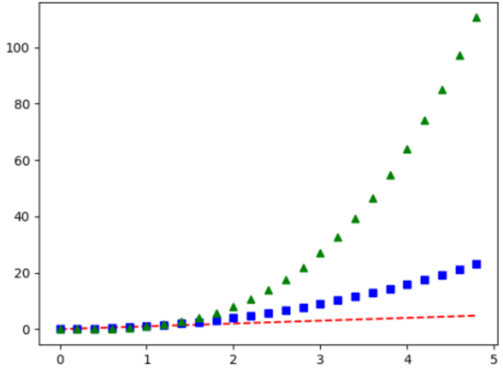
❑ **Line plot: fmt**

- [marker] – how to mark data:
  - ‘o’ – circle
  - ‘v’ – down triangle (‘^’, ‘<‘, ‘>’)
  - ‘*’ – star
  - ‘.’ – dot
  - ‘p’ – pentagon
  - …
- [line] – how to draw a line:
  - ‘-’ – solid line
  - ‘--‘ – dashed line
  - ‘-.’ – dash
  - ‘:’ – dotted line

V. Some common chart types

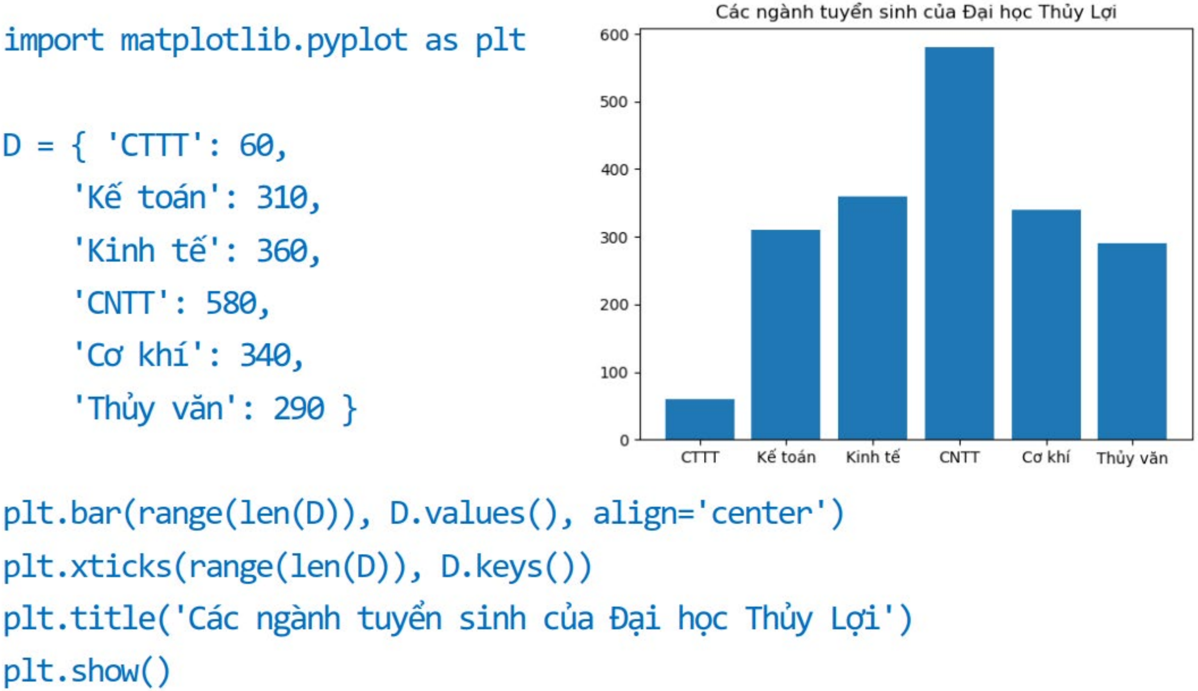
❑ Example

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

# chia đoạn 0-5 thành các bước 0.2
t = np.arange(0., 5., 0.2)

# Vẽ 3 đường:
# - màu đỏ nét đứt: y = x
# - màu xanh dương, đánh dấu ô vuông: y = x^2
# - màu xanh lá, đánh dấu tam giác: y = x^3
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



V. Some common chart types

❑ Bar plot

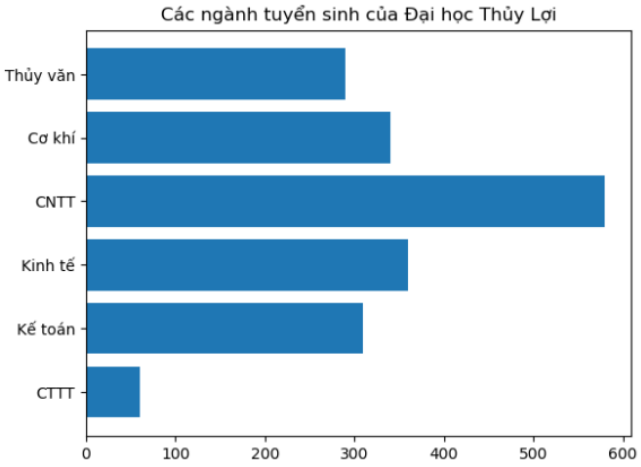


```
import matplotlib.pyplot as plt

D = { 'CTTT': 60,
      'Kế toán': 310,
      'Kinh tế': 360,
      'CNTT': 580,
      'Cơ khí': 340,
      'Thủy văn': 290 }

plt.bar(range(len(D)), D.values(), align='center')
plt.xticks(range(len(D)), D.keys())
plt.title('Các ngành tuyển sinh của Đại học Thủy Lợi')
plt.show()
```

V. Some common chart types

❑ Horizontal Bar plot

```
import matplotlib.pyplot as plt

D = { 'CTTT': 60,
      'Kế toán': 310,
      'Kinh tế': 360,
      'CNTT': 580,
      'Cơ khí': 340,
      'Thủy văn': 290 }
```

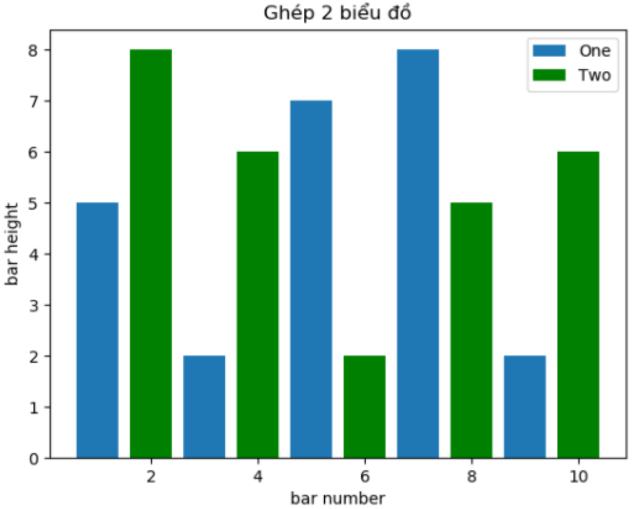


```
plt.barh(range(len(D)), list(D.values()))
plt.yticks(range(len(D)), D.keys())
plt.title('Các ngành tuyển sinh của Đại học Thủy Lợi')
plt.show()
```

V. Some common chart types

❑ Combine 2 charts

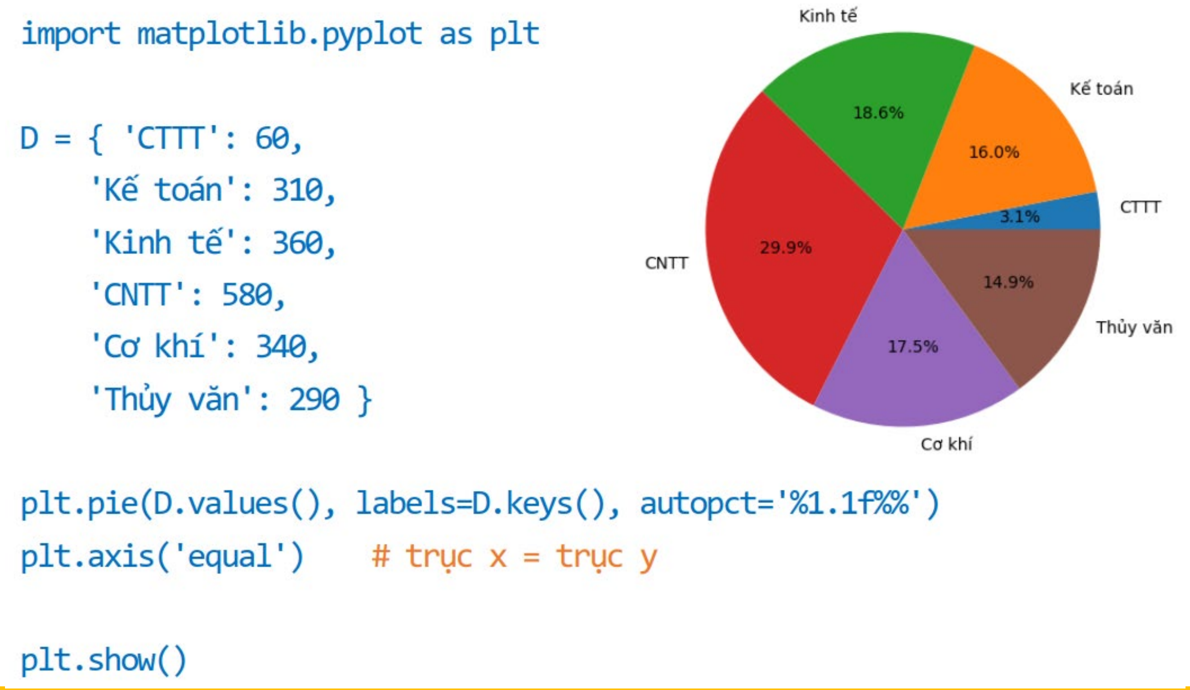
```
import matplotlib.pyplot as plt

plt.bar([1,3,5,7,9],[5,2,7,8,2], label="One")
plt.bar([2,4,6,8,10],[8,6,2,5,6], label="Two", color='g')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')

plt.title('Ghép 2 biểu đồ')

plt.show()
```



V. Some common chart types

❑ Pie charts

Clarifying detail/overall correlation

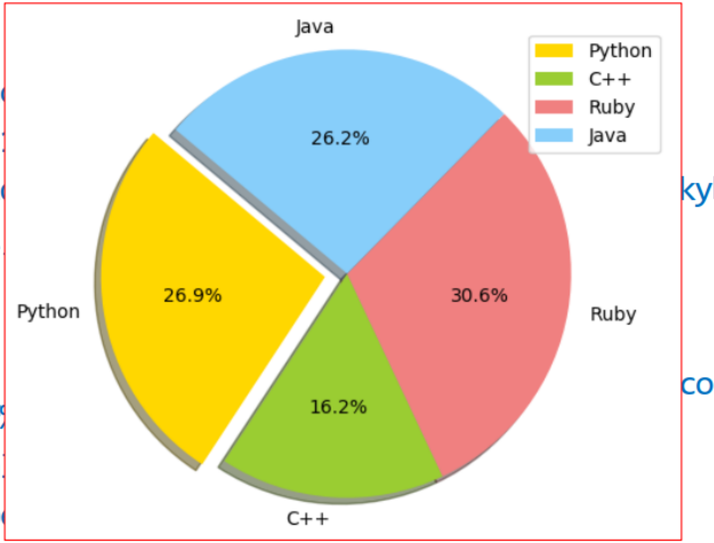


```
import matplotlib.pyplot as plt

D = { 'CTTT': 60,
     'Kế toán': 310,
     'Kinh tế': 360,
     'CNTT': 580,
     'Cơ khí': 340,
     'Thủy văn': 290 }

plt.pie(D.values(), labels=D.keys(), autopct='%1.1f%%')
plt.axis('equal')    # trục x = trục y

plt.show()
```

V. Some common chart types

❑ Full Pie charts

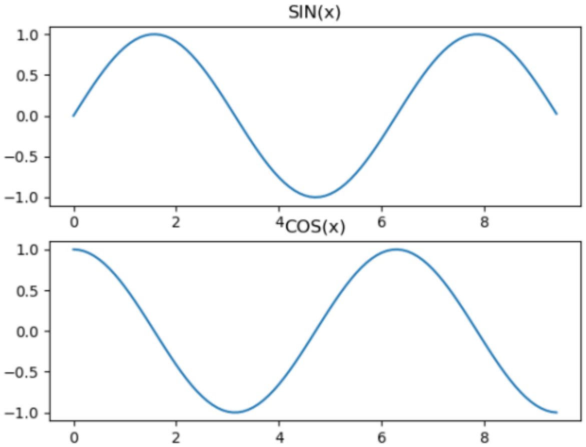
```
import matplotlib.pyplot as plt

# dữ liệu
labels = 'Pyth
sizes = [215, 
colors = ['gol                                    kyblue']
explode = (0.1

# vẽ
plt.pie(sizes,                                    colors,
autopct='%1.1f
plt.axis('equa
plt.legend(lab
plt.show()
```



VI. Some useful functions

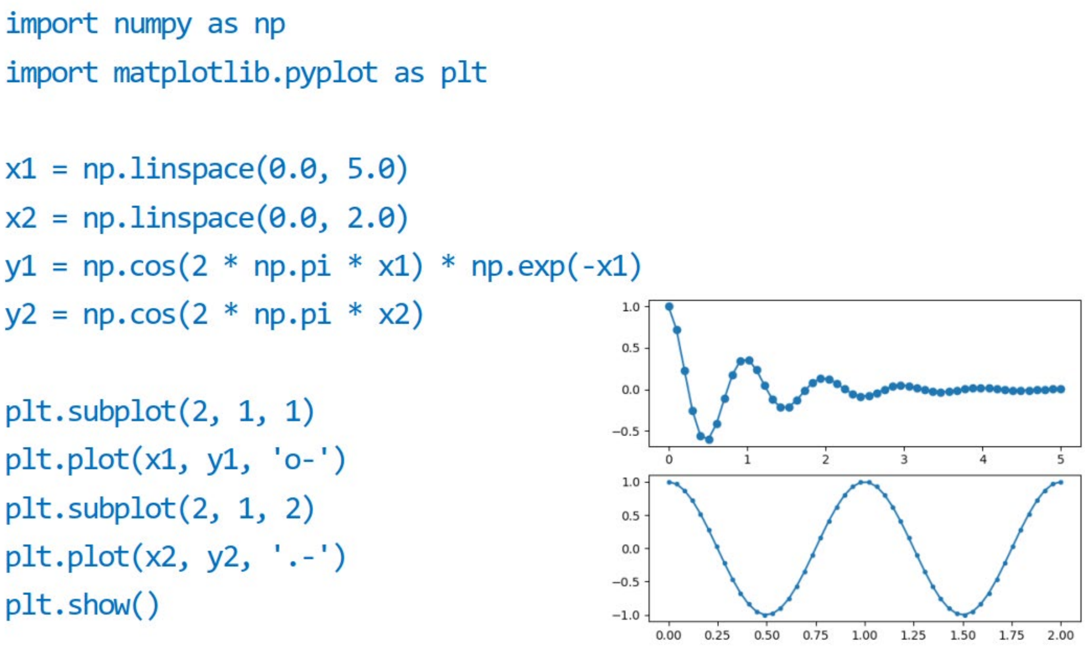
❑ Divide into subgraphs

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

x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.subplot(2, 1, 1)    # biểu đồ 1
plt.plot(x, y_sin)
plt.title('SIN(x)')
plt.subplot(2, 1, 2)    # biểu đồ 2
plt.plot(x, y_cos)
plt.title('COS(x)')
plt.show()
```



VI. Some useful functions

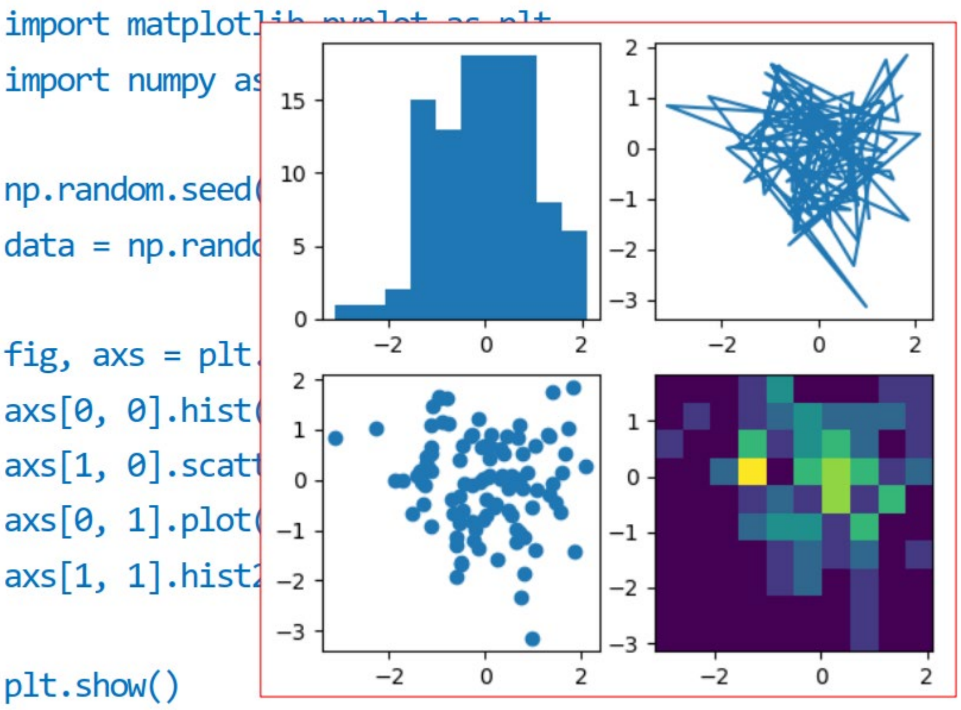
❑ Divide into subgraphs



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

x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'o-')
plt.subplot(2, 1, 2)
plt.plot(x2, y2, '.-')
plt.show()
```

VI. Some useful functions

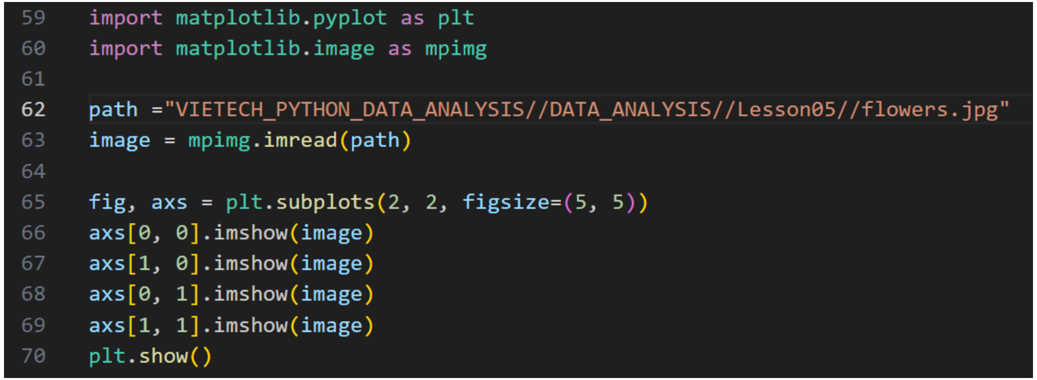
❑ Divide into subgraphs



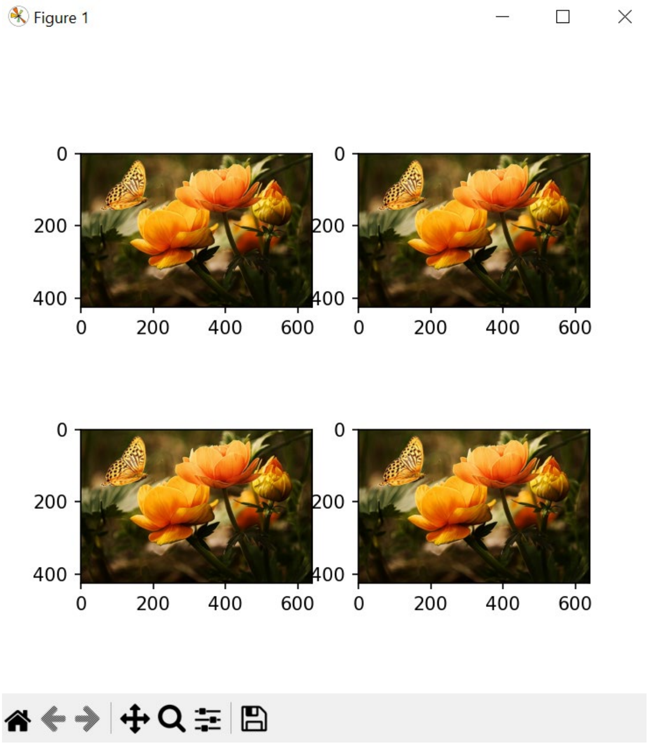
```
import matplotlib.pyplot as plt
import numpy as

np.random.seed(
data = np.rando

fig, axs = plt.
axs[0, 0].hist(
axs[1, 0].scatt
axs[0, 1].plot(
axs[1, 1].hist2

plt.show()
```

VI. Some useful functions

❑ Load image (png)

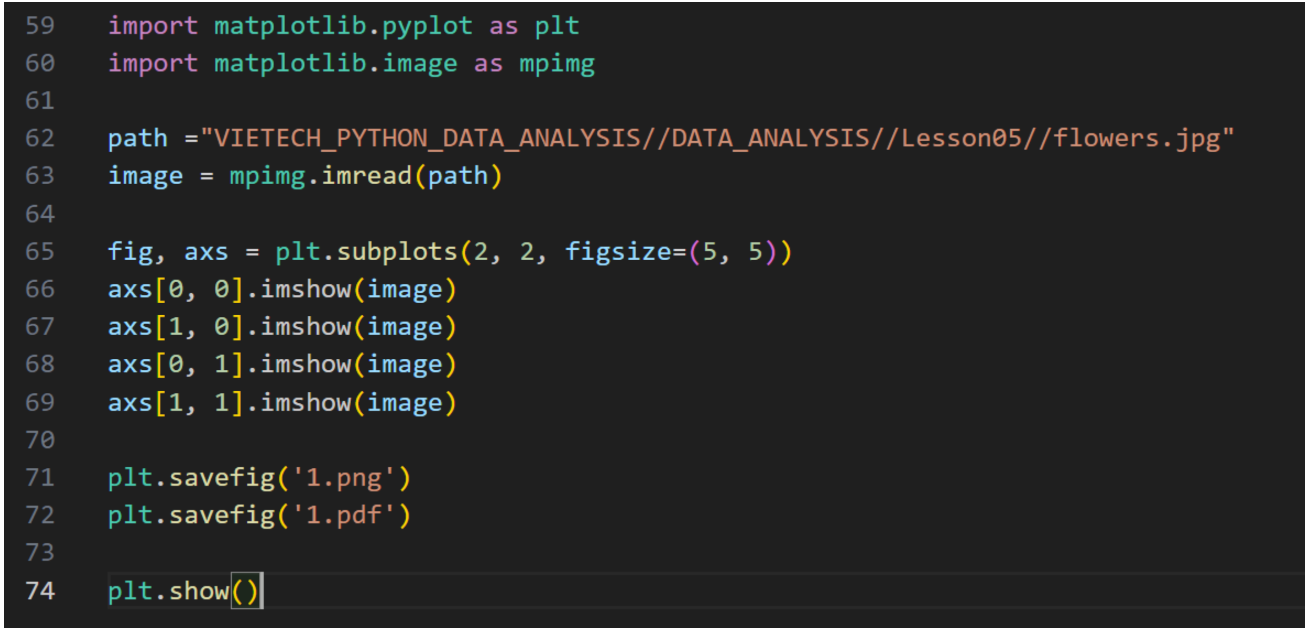


```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

path ="VIETECH_PYTHON_DATA_ANALYSIS//DATA_ANALYSIS//Lesson05//flowers.jpg"
image = mpimg.imread(path)

fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].imshow(image)
axs[1, 0].imshow(image)
axs[0, 1].imshow(image)
axs[1, 1].imshow(image)
plt.show()
```



VI. Some useful functions

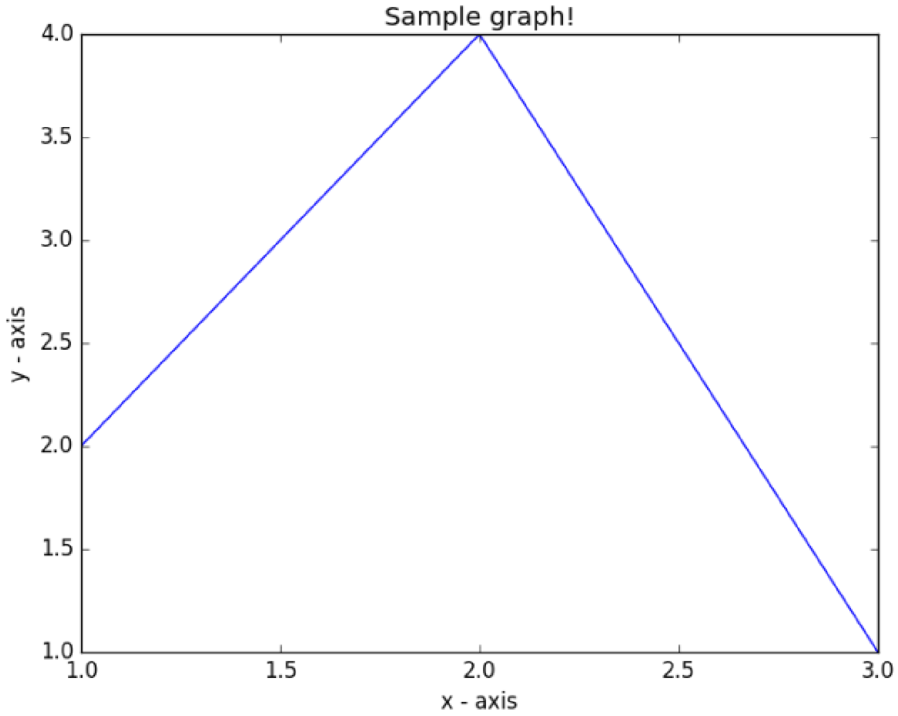
❑ Save the chart to file



```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

path ="VIETECH_PYTHON_DATA_ANALYSIS//DATA_ANALYSIS//Lesson05//flowers.jpg"
image = mpimg.imread(path)

fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].imshow(image)
axs[1, 0].imshow(image)
axs[0, 1].imshow(image)
axs[1, 1].imshow(image)

plt.savefig('1.png')
plt.savefig('1.pdf')

plt.show()
```

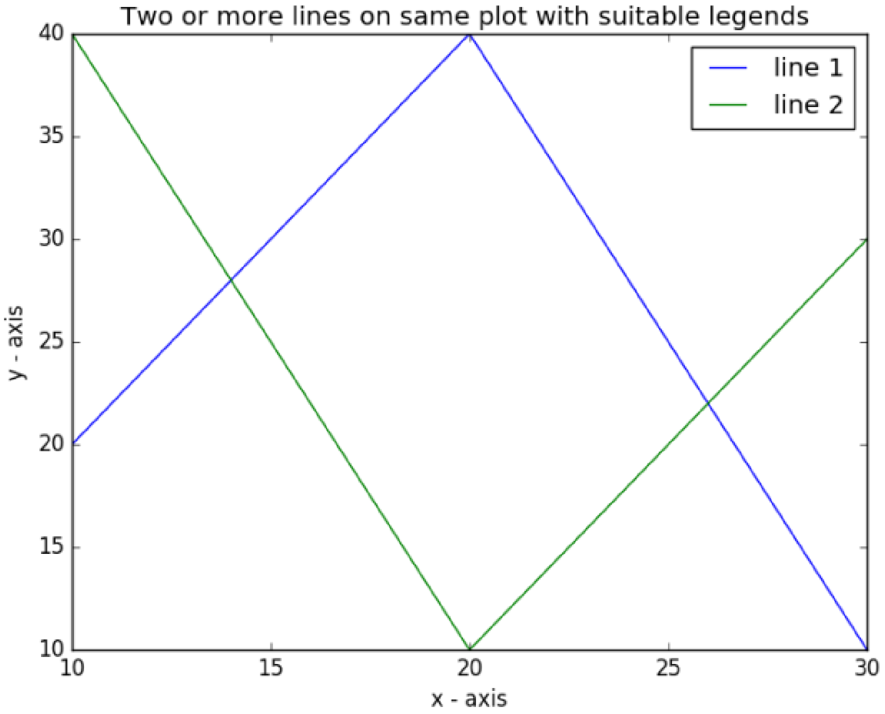
VII. Exercise

Exercise 01: Write a python program to draw the following graph



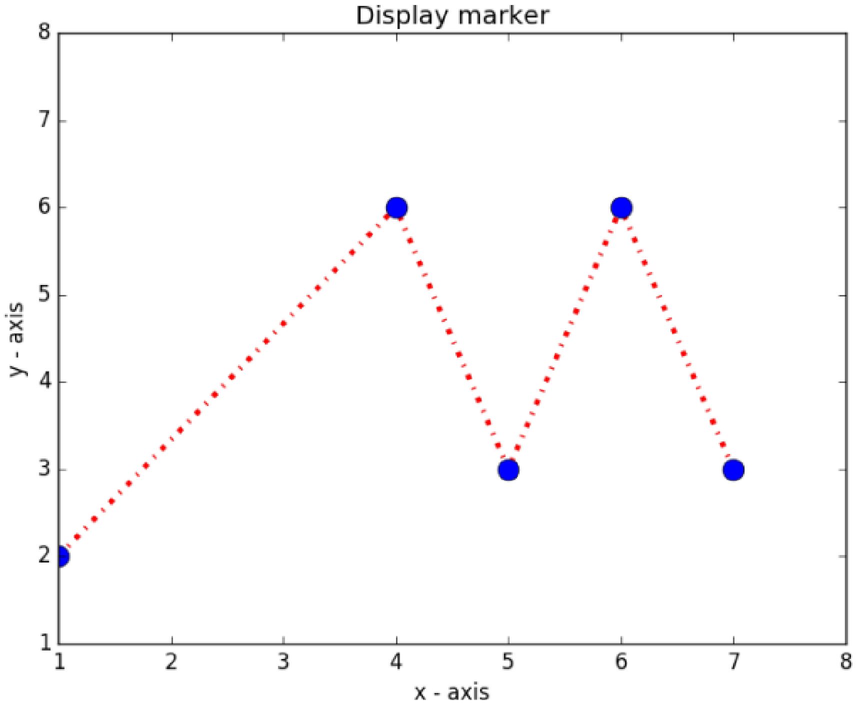
VII. Exercise

Exercise 02: Write a python program to draw the following graph



VII. Exercise

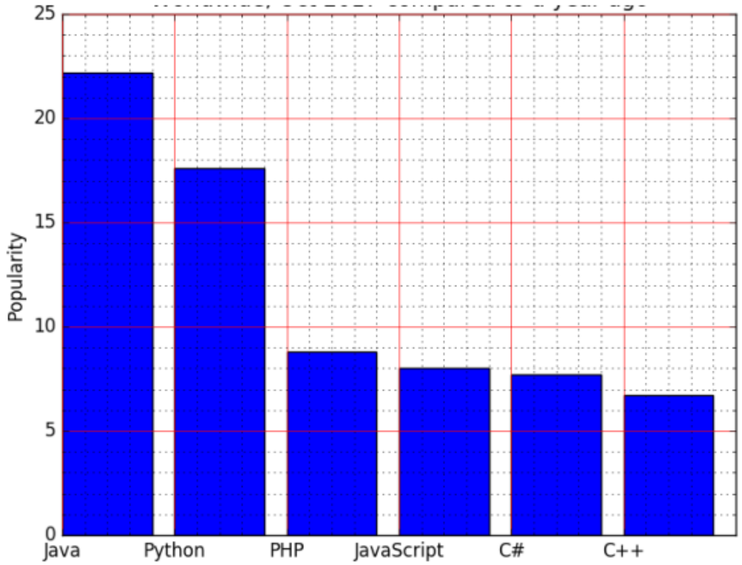
Exercise 03: Write a python program to draw the following graph



VII. Exercise

Exercise 04: Draw a graph as shown below for the following data set

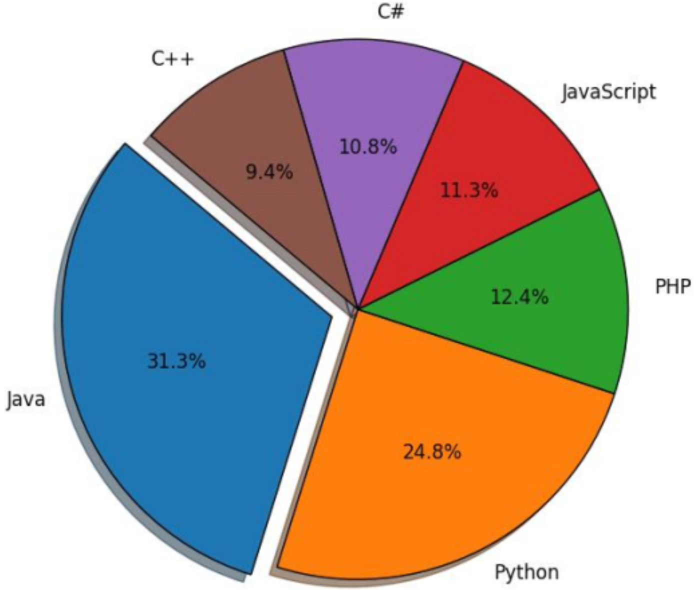
- Languages: Java, Python, PHP, JavaScript, C#, C++
- Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



VII. Exercise

Exercise 05: Draw a graph as shown below for the following data set

- Languages: Java, Python, PHP, JavaScript, C#, C++
- Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



V. Reference

Book:

Learning Pandas, chapter 2, 3, 4, 7

Q & A