

# PYTHON FOR MACHINE LEARNING, DATA SCIENCE & DATA VISUALIZATION

## Bài 6:Trực quan hóa dữ liệu - Matplotlib

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-csdl/Python-for-Machine-Learning-Data-Science--Data-Visualization-Python-cho-may-hoc-Khoa-hoc-du-lieu-va-Truc-quan-hoa-du-lieu\\_191](https://csc.edu.vn/lap-trinh-va-csdl/Python-for-Machine-Learning-Data-Science--Data-Visualization-Python-cho-may-hoc-Khoa-hoc-du-lieu-va-Truc-quan-hoa-du-lieu_191)

2019



# Nội dung

---

1. Vai trò của trực quan hóa dữ liệu
2. Matplotlib
3. Quy trình tạo biểu đồ

# Vai trò của trực quan hóa dữ liệu

---

## □ Định nghĩa:

- “The use of computer-supported, visual representations of abstract data to amplify cognition.” [Card et al., 1999]
- Trực quan hóa dữ liệu: Là việc sử dụng các biểu diễn trực quan của dữ liệu trừu tượng thông qua sự hỗ trợ của máy tính để mở rộng nhận thức. (tạm dịch)

# Vai trò của trực quan hóa dữ liệu

---

## □ Định nghĩa:

- “The **representation and presentation** of data to **facilitate understanding**.” [Kirk, 2016]
- Là sự diễn tả và trình bày dữ liệu để tạo điều kiện cho việc hiểu biết về dữ liệu. (tạm dịch)

# Vai trò của trực quan hóa dữ liệu

## ❑ Tại sao cần trực quan hóa dữ liệu?



Dữ liệu khó hiểu...

# Vai trò của trực quan hóa dữ liệu

## ❑ Tại sao cần trực quan hóa dữ liệu?



Quá nhiều số...

# Vai trò của trực quan hóa dữ liệu

## ❑ Tại sao cần trực quan hóa dữ liệu?

	A	B	C	D	G
1	Sales Team Review				
2	Salesperson	Region Covered	February 2017 Sales	Cost of Sales	
3	Jeffrey Burke	Oklahoma	\$ 28,000	\$ 2,460	
4	Amy Fernandez	North Carolina	\$ 23,138	\$ 1,521	
5	Mark Hayes	Massachusetts	\$ 25,092	\$ 1,530	
6	Judith Ray	California	\$ 21,839	\$ 1,923	
7	Randy Graham	South Carolina	\$ 23,342	\$ 2,397	
8	Christina Foster	Delaware	\$ 23,368	\$ 1,500	
9	Judy Green	Texas	\$ 21,510	\$ 1,657	
10	Paula Hall	Virginia	\$ 21,314	\$ 2,418	
11	February Sales Total		\$ 187,603	\$ 15,406	
12					

Các cuộc họp nhàm chán với các spreadsheet...

# Vai trò của trực quan hóa dữ liệu

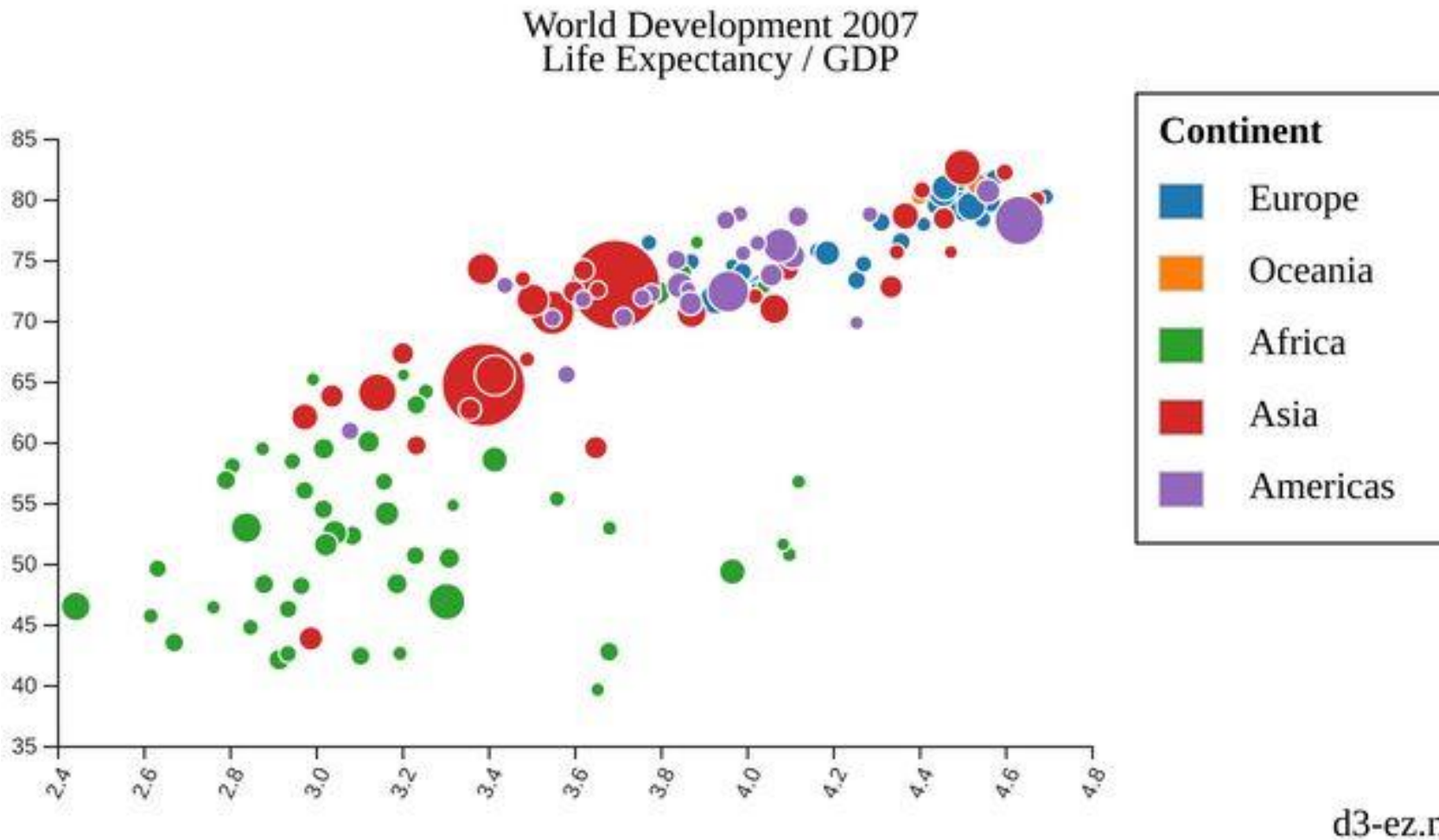
---

## □ Ưu điểm

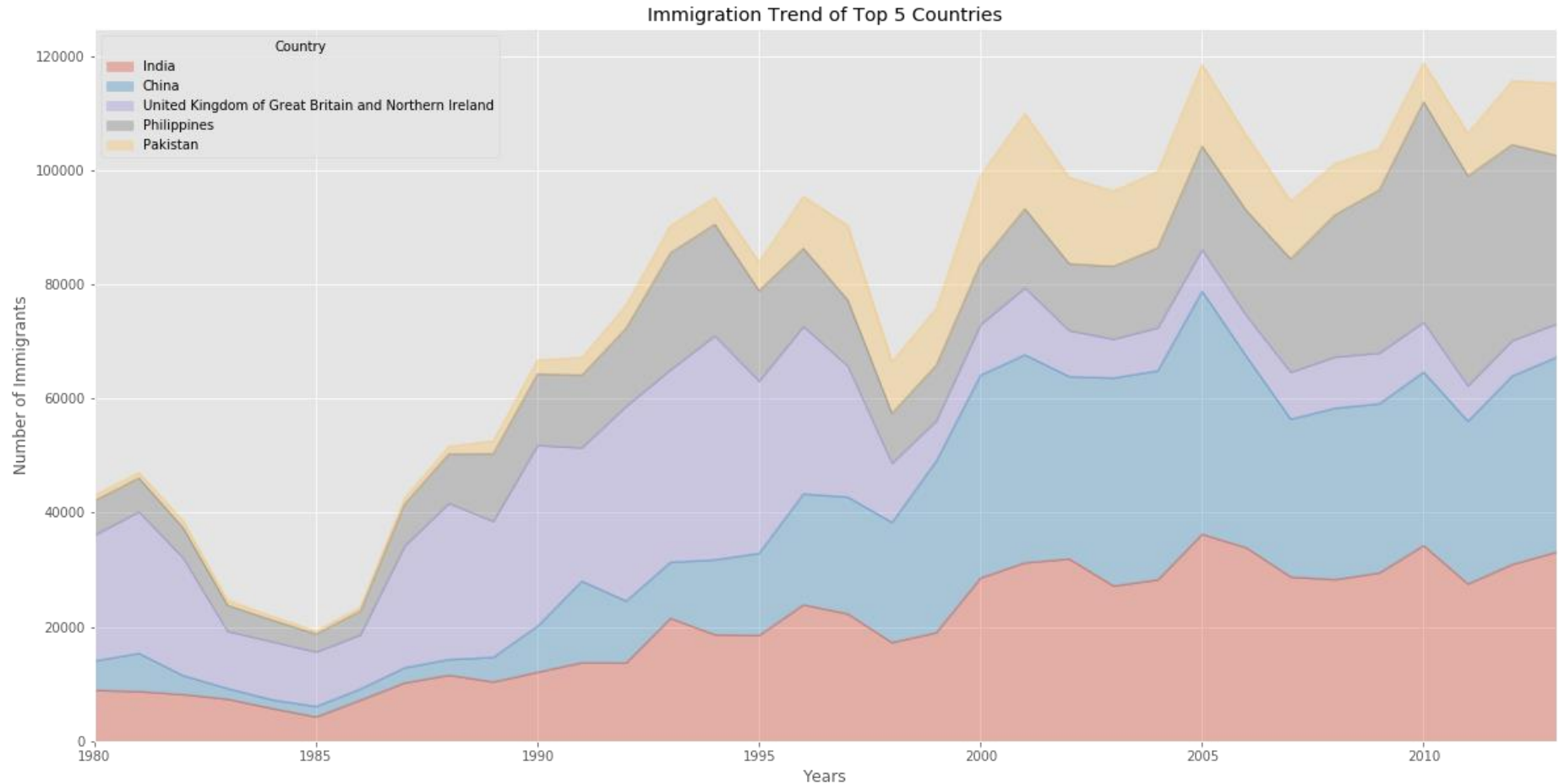
- Giúp người xem thấy trực quan và dễ hiểu
- Giúp giải quyết vấn đề
- Kể câu chuyện về dữ liệu trong thời gian rất ngắn



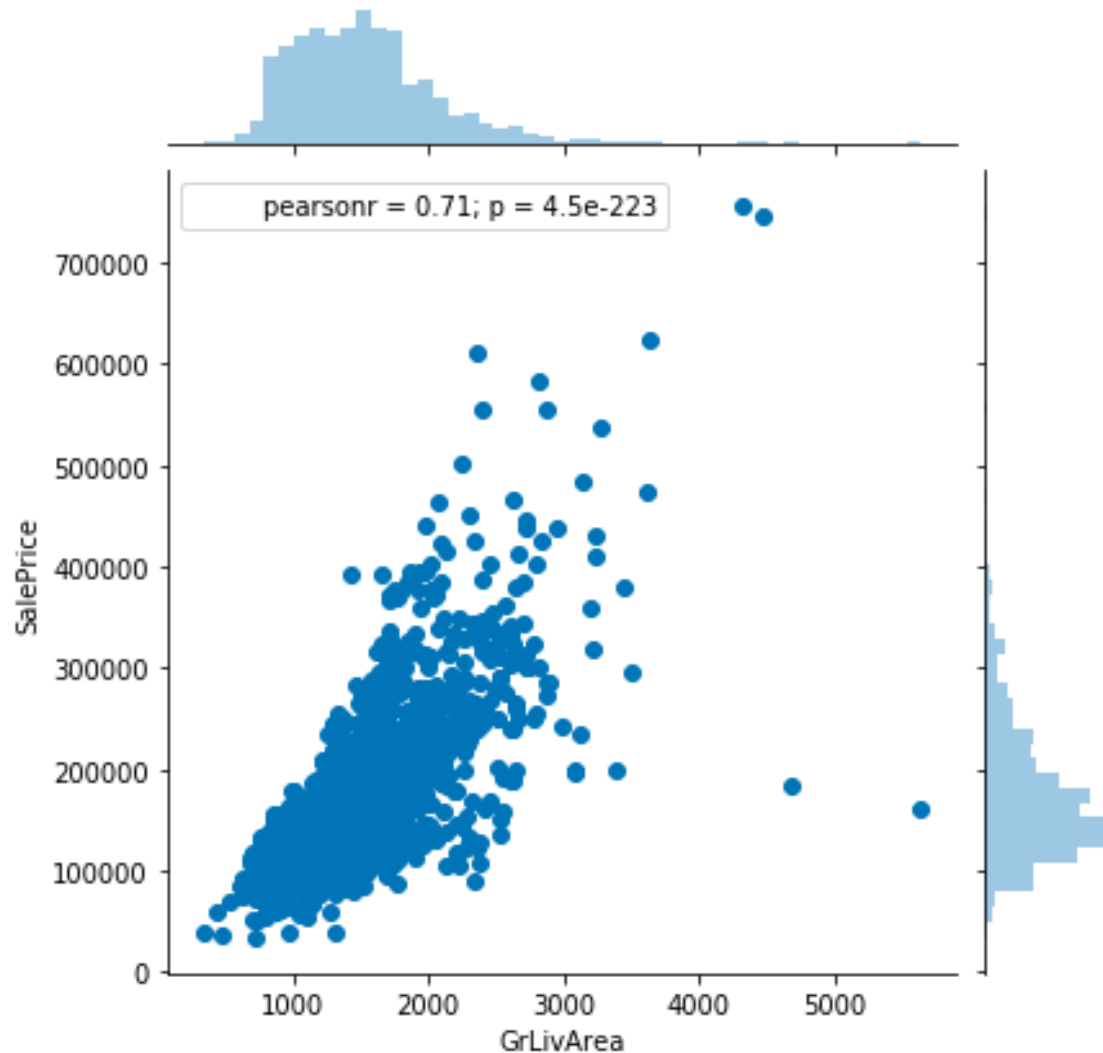
# Vai trò của trực quan hóa dữ liệu



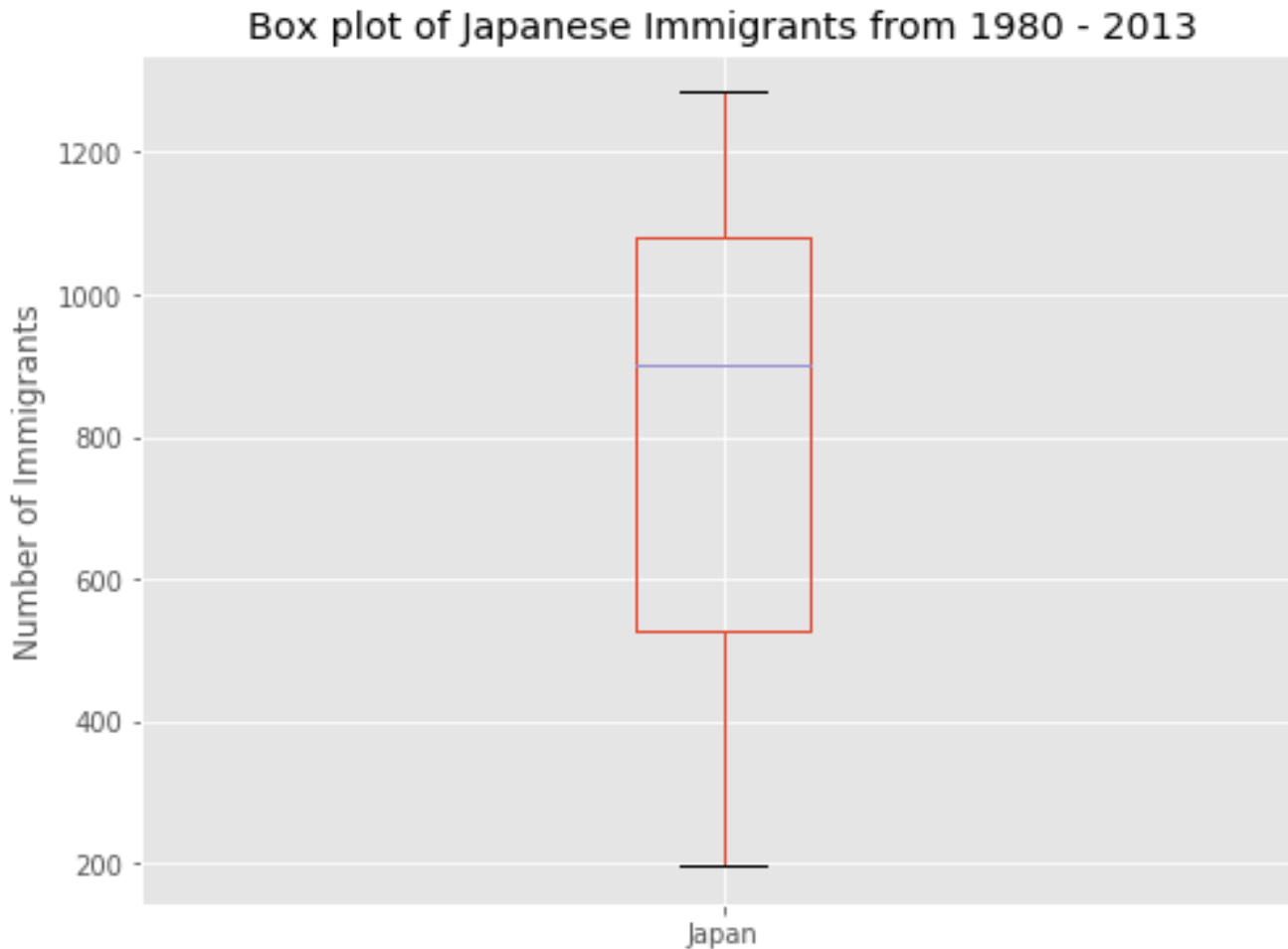
# Vai trò của trực quan hóa dữ liệu



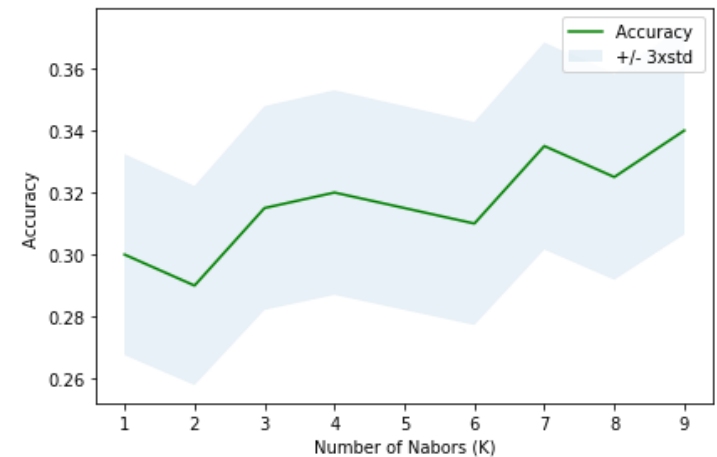
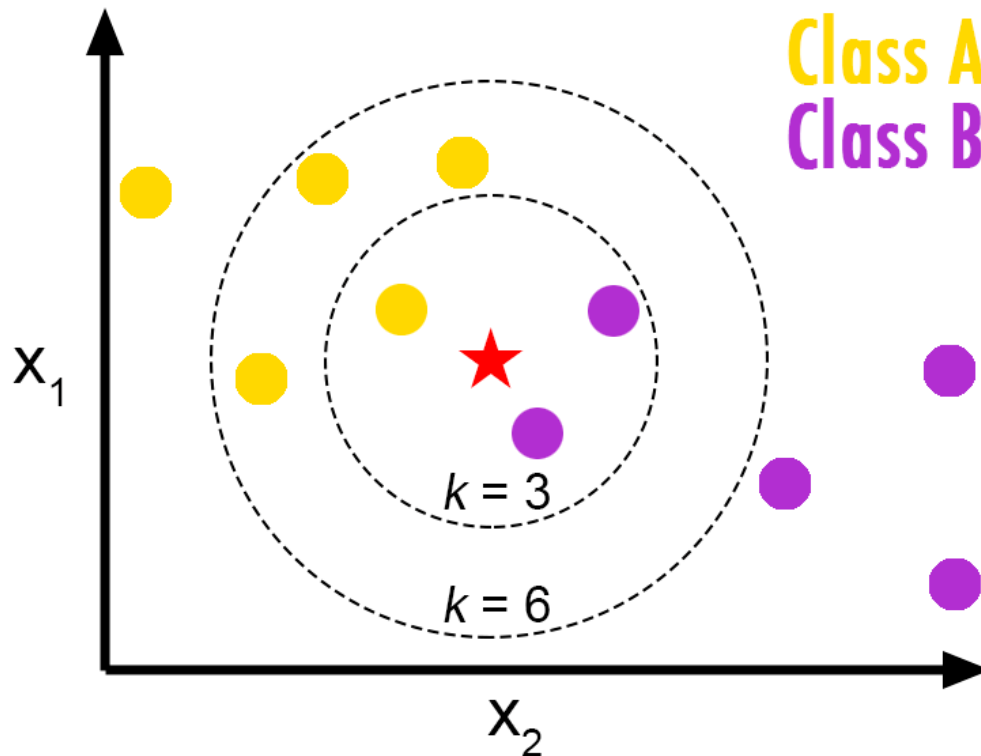
# Vai trò của trực quan hóa dữ liệu



# Vai trò của trực quan hóa dữ liệu



# Vai trò của trực quan hóa dữ liệu



# Vai trò của trực quan hóa dữ liệu

---

## □ Nguyên tắc thiết kế:

- "Good data visualization is:
  - 1. Trustworthy
  - 2. Accessible
  - 3. Elegant" - Andy Kirk\*

***Theo: Kirk, A. Data Visualisation: A handbook for Data Driven Design. SAGE publications, 2016.***

# Vai trò của trực quan hóa dữ liệu

---

## □ Nguyên tắc thiết kế:

### ● Trustworthy

- Sự tin tưởng khó kiếm, dễ mất
- Tính trung thực và toàn vẹn cần phải có ở mọi nơi trong quá trình thực hiện data science

# Vai trò của trực quan hóa dữ liệu

---

## □ Nguyên tắc thiết kế:

- Accessible

- Cần phải biết ai là người sẽ xem các biểu đồ
- Hiểu mục đích của việc trực quan hóa



# Vai trò của trực quan hóa dữ liệu

---

## □ Nguyên tắc thiết kế:

- Elegant

- Tập trung vào các yếu tố liên quan
- Có phong cách riêng nếu có thể
- Suy nghĩ cách thiết kế trước khi thể hiện

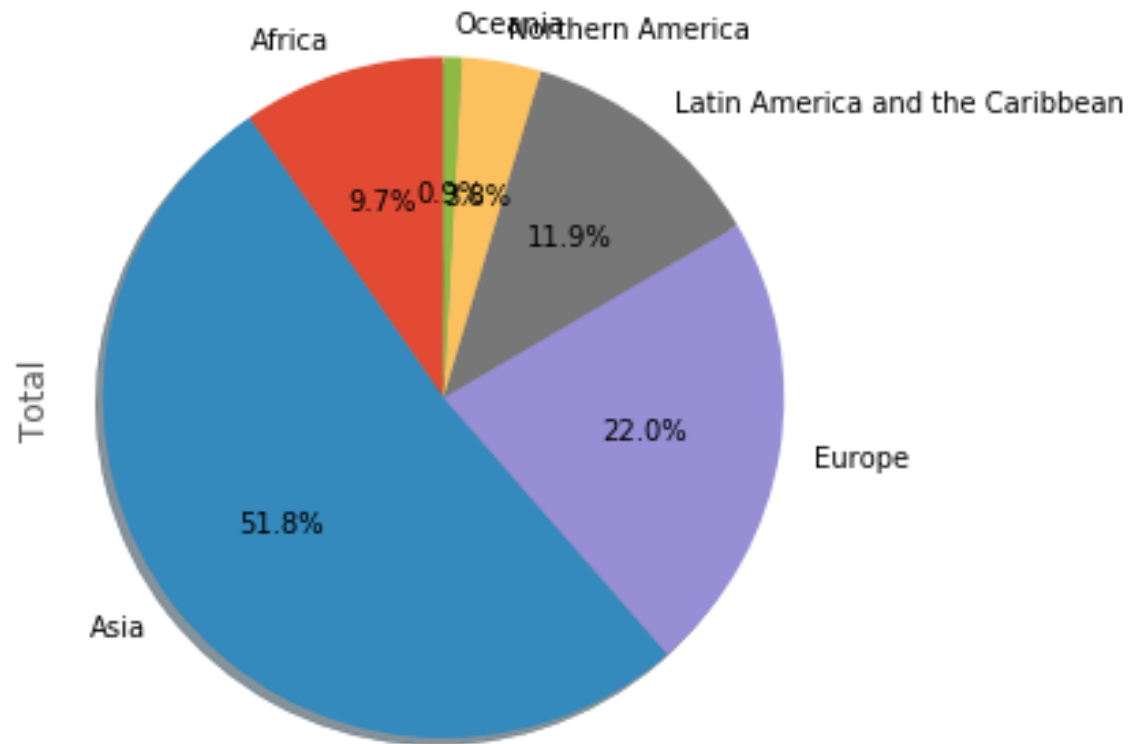
# Vai trò của trực quan hóa dữ liệu

---



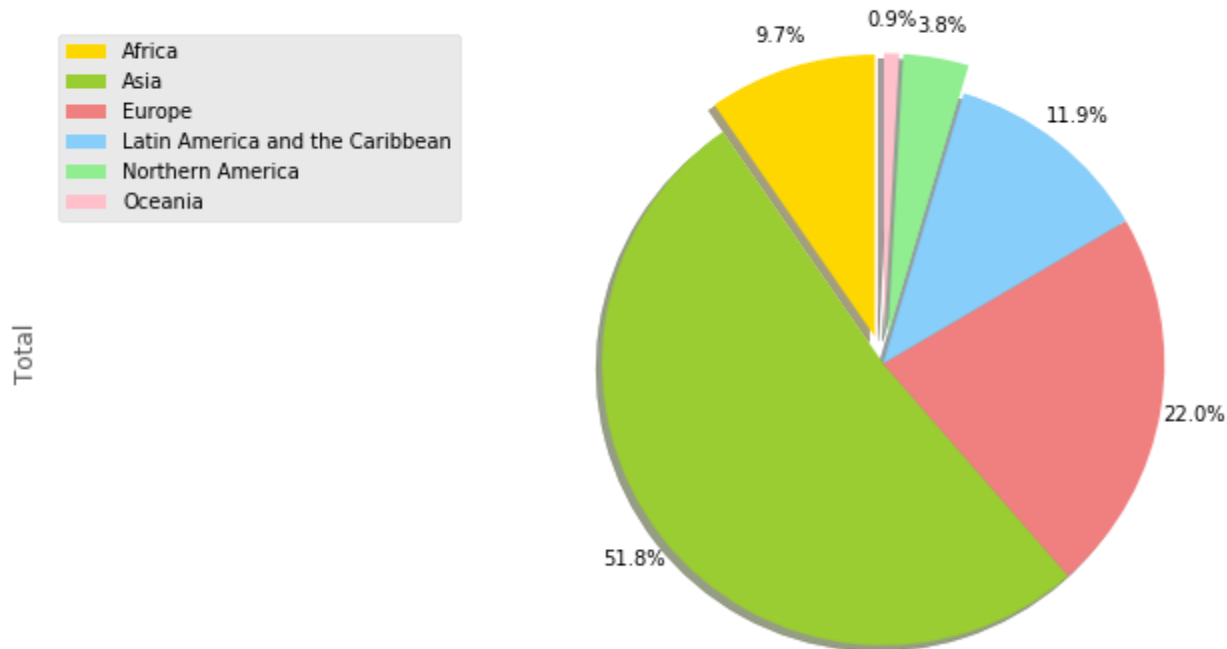
# Trực quan hóa dữ liệu

Immigration to Canada by Continent [1980 - 2013]



# Trực quan hóa dữ liệu

Immigration to Canada by Continent [1980 - 2013]



# Thư viện trực quan hóa dữ liệu

## Visualization Libraries in Python

### 2. Visualization Libraries



#### **Matplotlib**

(plots & graphs, most popular)



#### **Seaborn**

(plots : heat maps, time series, violin plots)

# Nội dung

---

1. Vai trò của trực quan hóa dữ liệu
2. Matplotlib
3. Quy trình tạo biểu đồ

# Matplotlib

---

## □ Giới thiệu

- Matplotlib là một thư viện vẽ biểu đồ 2D của Python, tạo ra các dạng hình vẽ khác nhau có chất lượng tốt.
- Matplotlib có thể được sử dụng trong script Python (Python/ Ipython ), Jupyter Notebook, ứng dụng web, và 4 bộ tool GUI.
- Người dùng có thể kiểm soát line style, font, các thuộc tính trên axes... thông qua giao diện hướng đối tượng hoặc thông qua các function

<https://matplotlib.org/>

# Matplotlib

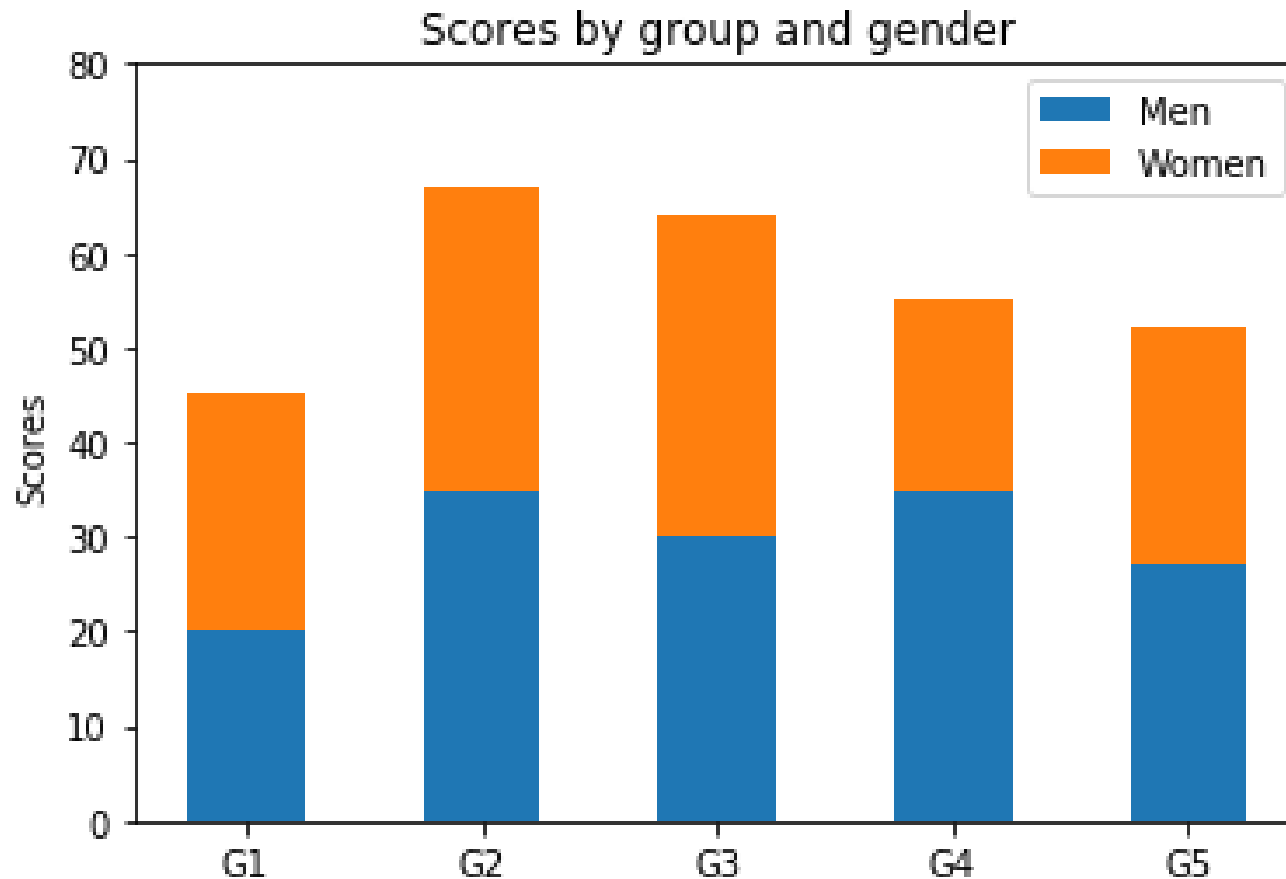
---

## □ Lý do chọn:

- “Matplotlib tries to **make easy things easy** and **hard things possible**. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.” – theo <https://matplotlib.org/>
- Cố gắng làm cho mọi thứ dễ dàng trở nên dễ dàng hơn và những thứ khó khăn trở nên khả thi. Bạn có thể tạo các biểu đồ dạng plot, histogram, power spectra, bar chart, errorchart, scatterplot... với chỉ vài dòng code.



# Quan sát biểu đồ



# Quan sát biểu đồ

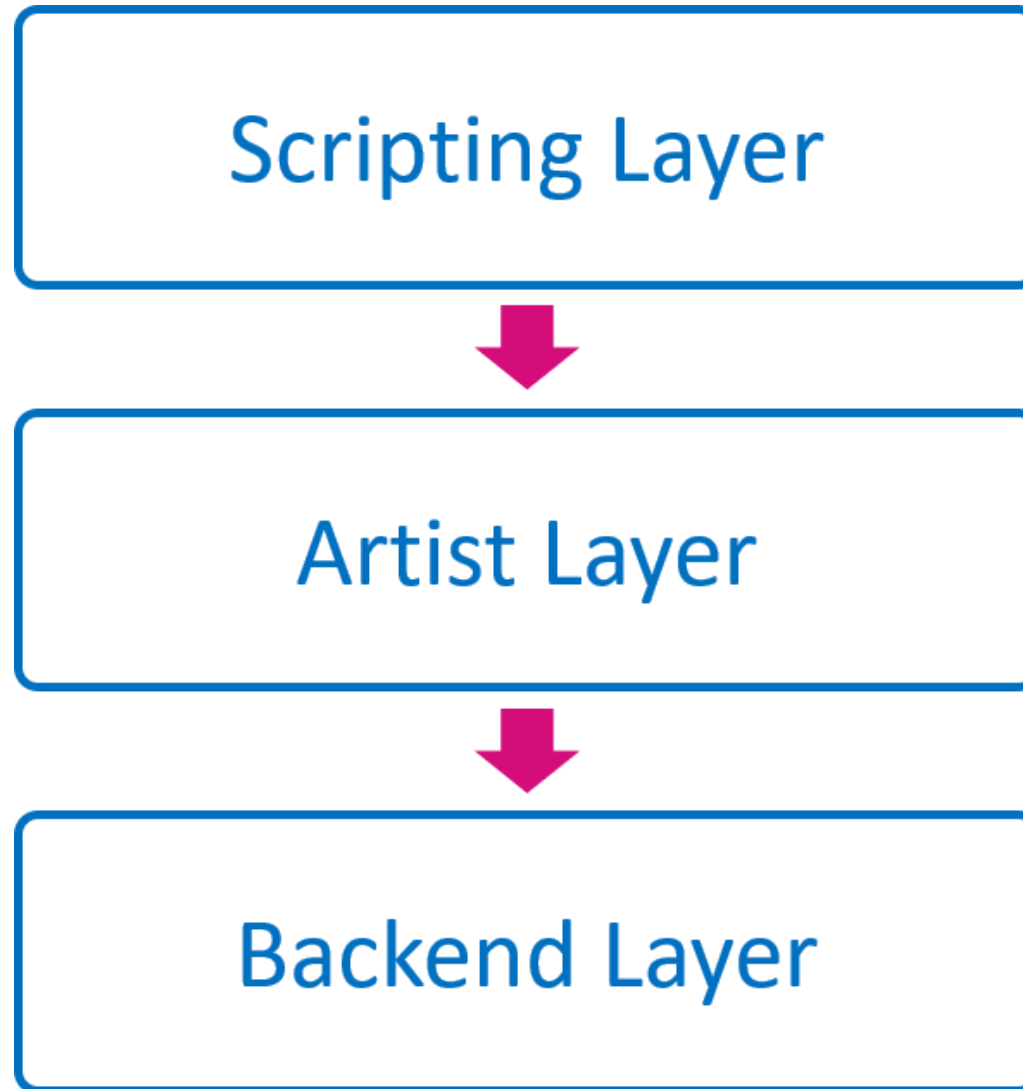
---

## ❑ Các thành phần chung trong biểu đồ:

- Loại biểu đồ (bar, line, scatter, histogram, pie...)
- Dữ liệu trên các trục (Axes data ranges)
- Nhãn trên trục (Axes labels)
- Ghi chú (Legend)
- Các chú thích (Annotations)

# Kiến trúc Matplotlib

---




# Scripting Layer

---

Gọi trực tiếp các lệnh qua pyplot module



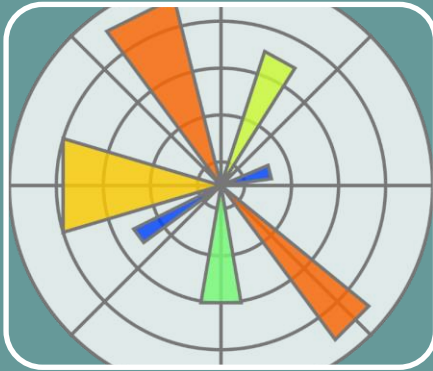
Matplotlib tự lưu vết của trạng thái hiện hành



Thực hiện các thay đổi trên Figure và Axes hiện hành

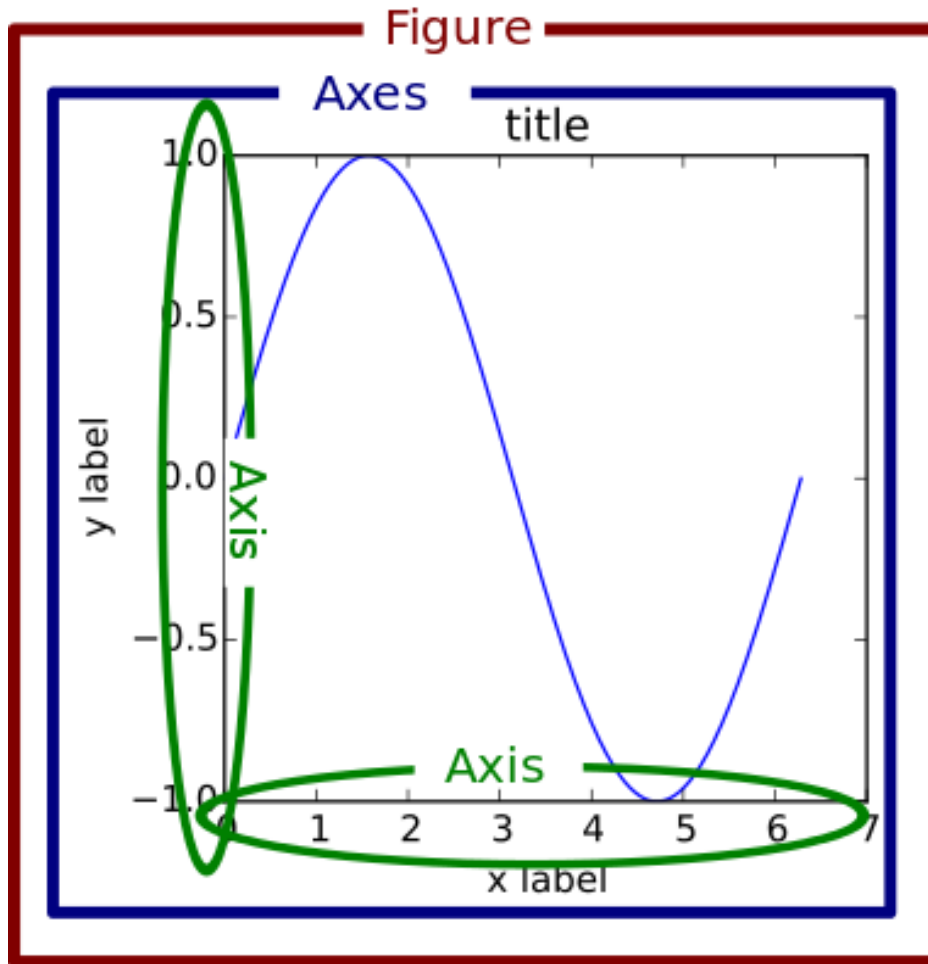
# Artist Layer

## Stateless Interface



- Còn gọi là object-oriented interface
- Sử dụng các biến đối tượng một cách tường minh
- Mỗi biến sẽ giúp thay đổi một số thuộc tính của đồ thị
- Rõ ràng
- Dễ dàng tùy biến các thuộc tính của biểu đồ

# Cấu trúc của biểu đồ



- Cây phân cấp đối tượng
- Figure và Axes là 2 đối tượng chính
- Figure là không gian thể hiện, cho phép trên đó có nhiều biểu đồ.
- Axes là biểu đồ thật sự với các thông tin: trục x-y, ghi chú, marker, ...

# matplotlib.pyplot

---

- ❑ `import matplotlib.pyplot as plt`
- ❑ **Hiển thị biểu đồ** : `plt.show()`
- ❑ **Tạo figure, axis**
  - `fig = plt.figure()`
  - `ax = plt.axes()`
- ❑ **Lưu figure** : `plt.savefig(<tên tập tin>)`

# Matplotlib

## ❑ Line plot: `matplotlib.pyplot.plot()`

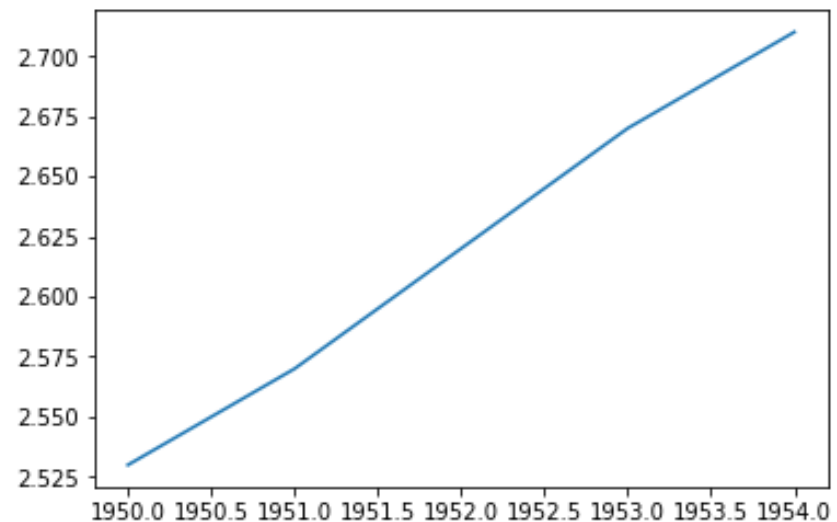
### ● Vẽ y và x theo dạng line hoặc/và marker

#### ■ Dạng mặc định

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

```
year = [1950, 1951, 1952, 1953, 1954]  
pop = [2.53, 2.57, 2.62, 2.67, 2.71]
```

```
# plot year and pop using default line style and color  
plt.plot(year, pop)  
plt.show()
```

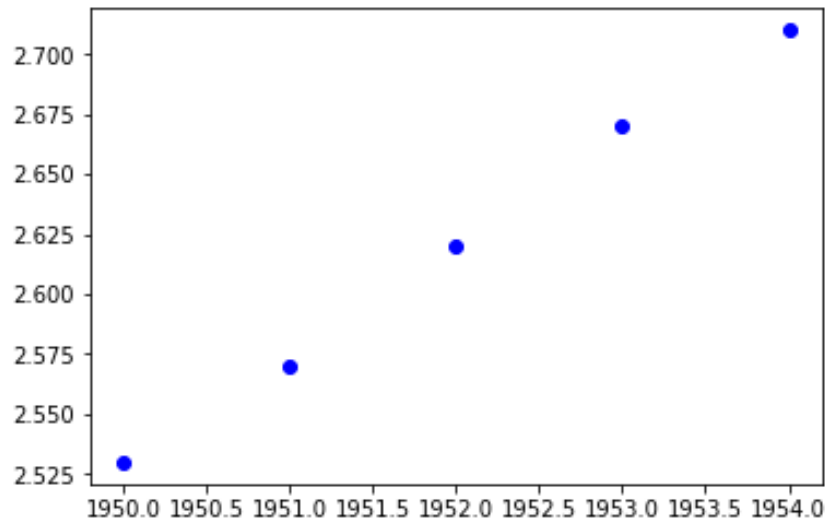




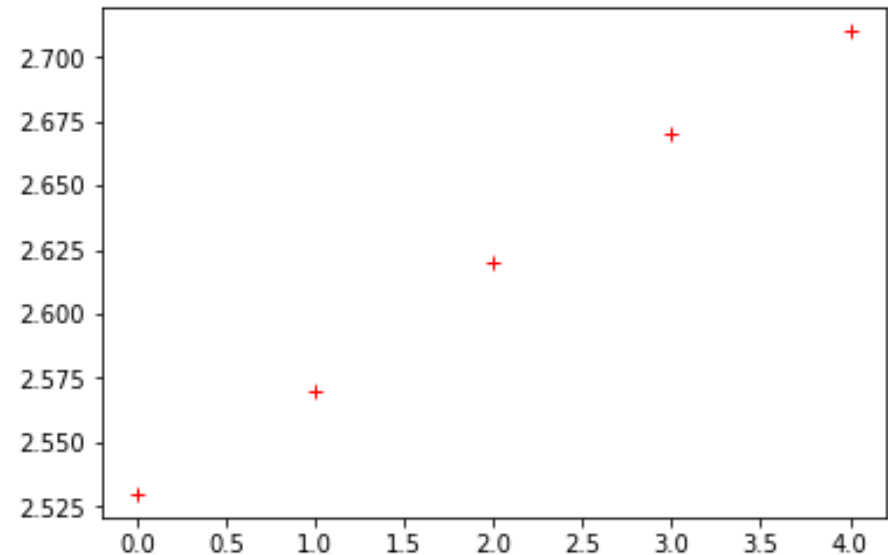
# Matplotlib

## ■ Định dạng

```
plt.plot(year, pop, 'bo')  
plt.show()
```



```
# ditto, but with red plusses  
plt.plot(pop, 'r+')  
plt.show()
```



```
plt.plot([x], y, [fmt])
```

```
"[color][marker][line]"
```

# Matplotlib

Format	Colors
'b'	blue
'r'	red
'g'	green
'm'	magenta
'c'	cyan
'b'	black
'w'	white
'y'	yellow

• point marker	='.'
○ circle marker	='o'
· pixel marker	=','
▼ triangle_down marker	='v'
► triangle_right marker	='>'
◄ triangle_left marker	='<'
⋏ tri_down marker	='1'
⋐ tri_up marker	='2'
⋑ tri_left marker	='3'
⋒ tri_right marker	='4'
⬡ hexagon1 marker	='h'
⬢ hexagon2 marker	='H'
⬠ pentagon marker	='p'
■ square marker	='s'
⊕ plus marker	='+'
★ star marker	='*'
✕ x marker	='x'
◆ diamond marker	='D'
◇ thin_diamond marker	='d'
vline marker	=' '
— hline marker	=' _'

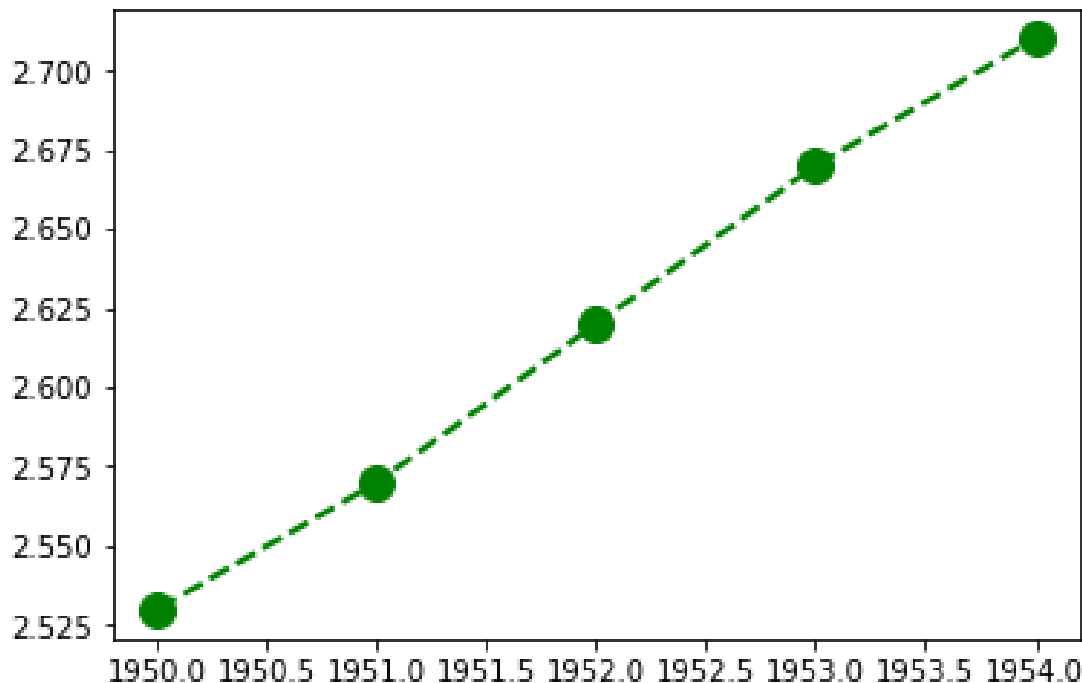
— solid line style	=' _'
- · - dash-dot line style	=' - .'
- - - dashed line style	=' - -'
· · · · · dotted line style	=' : .'

**"[color] [marker] [line]"**

# Matplotlib

- Dạng có dùng color, marker, markersize, linestyle, linewidth

```
plt.plot(year, pop, color='green', marker='o', linestyle='dashed',  
         linewidth=2, markersize=12)  
plt.show()
```

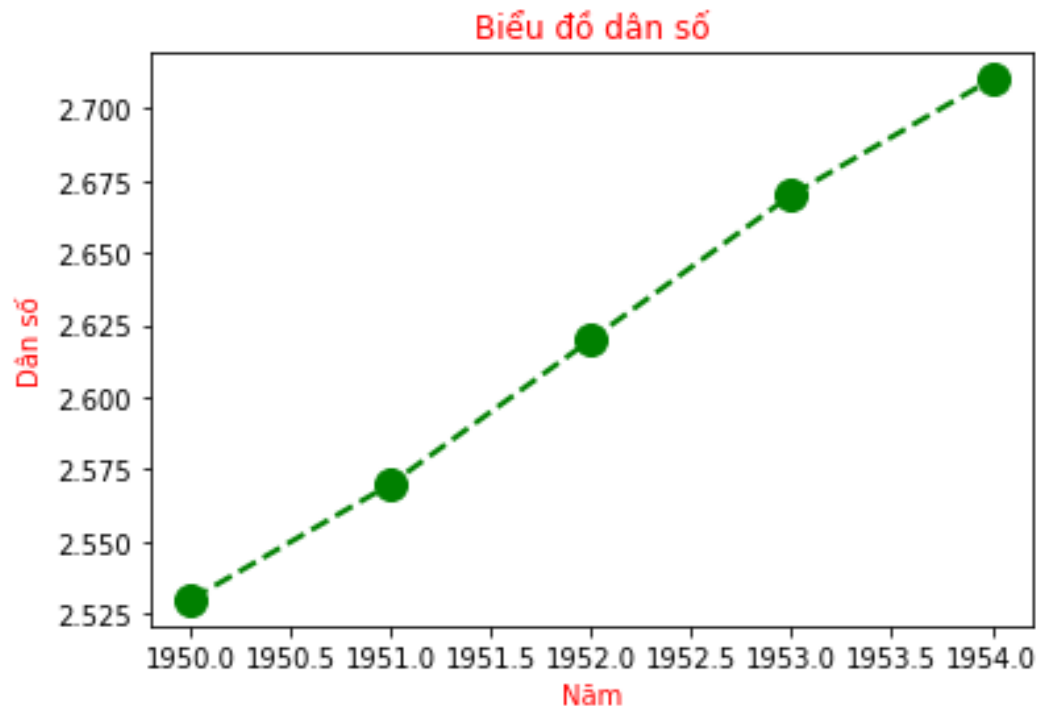


character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

# Matplotlib

## ■ Dạng có title, ylabel, xlabel

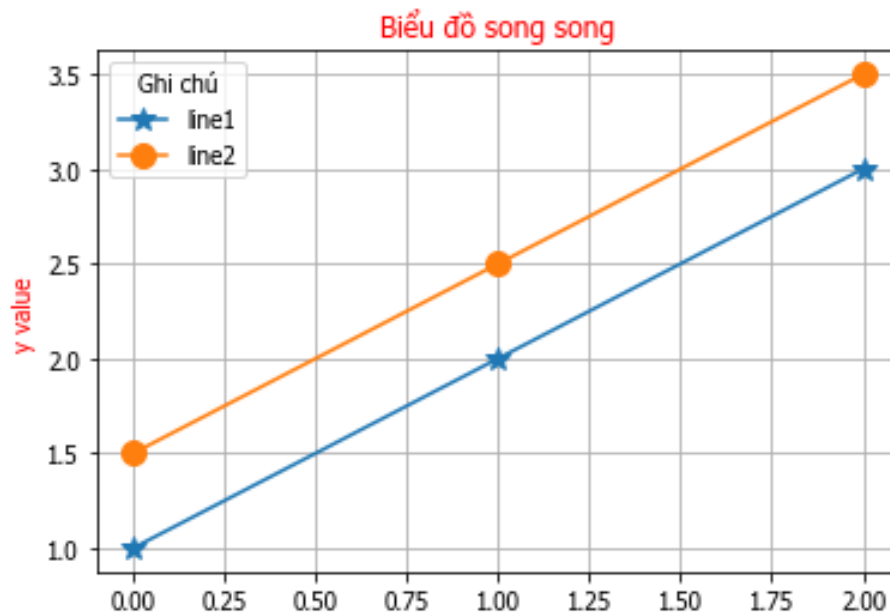
```
plt.plot(year, pop, color='green', marker='o', linestyle='dashed',  
         linewidth=2, markersize=12)  
plt.title('Biểu đồ dân số', color="red")  
plt.ylabel('Dân số', color="red")  
plt.xlabel('Năm', color="red")  
plt.show()
```



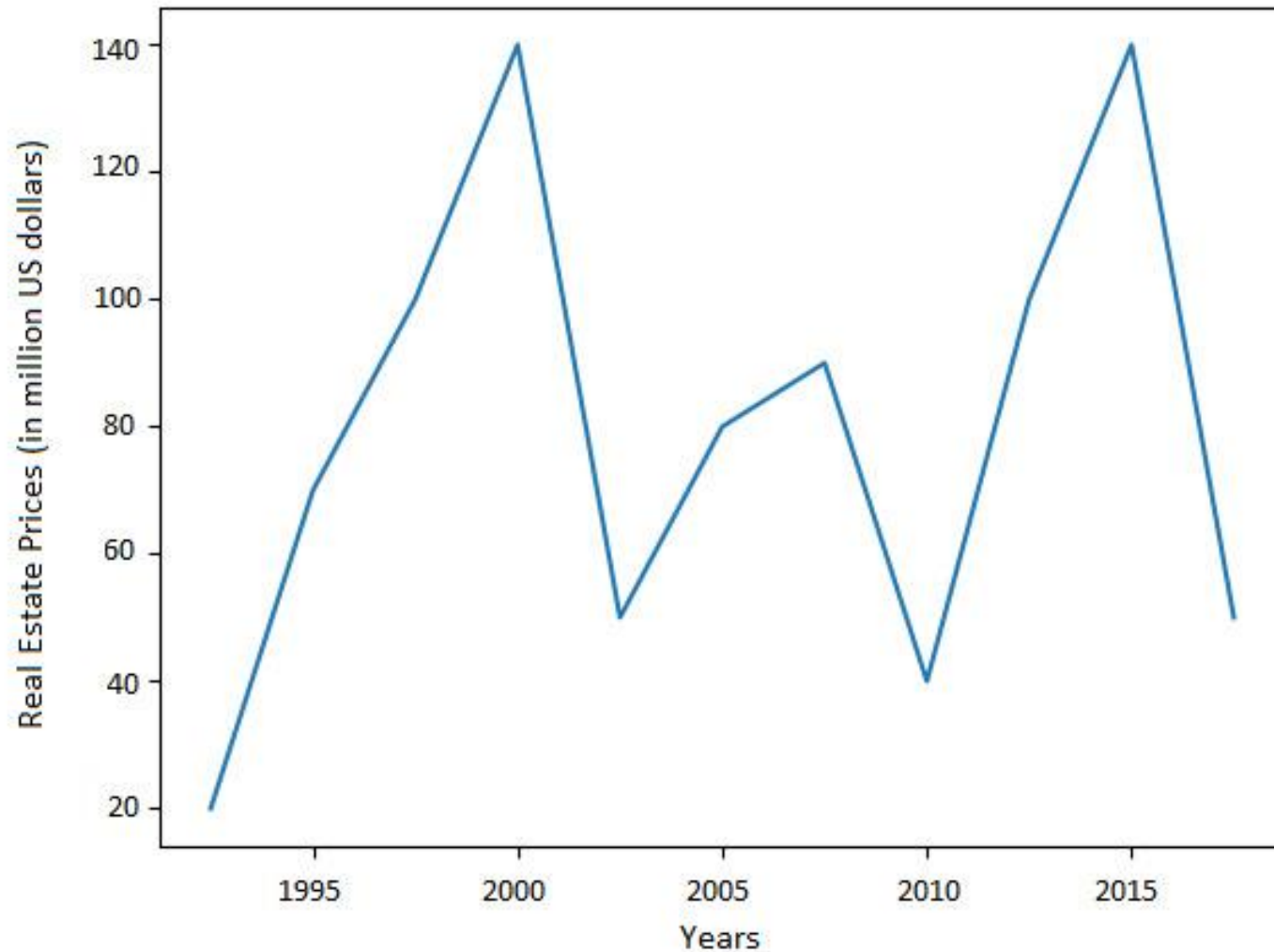
# Matplotlib

- **Dạng có nhiều line, kèm legend (ghi chú), grid**

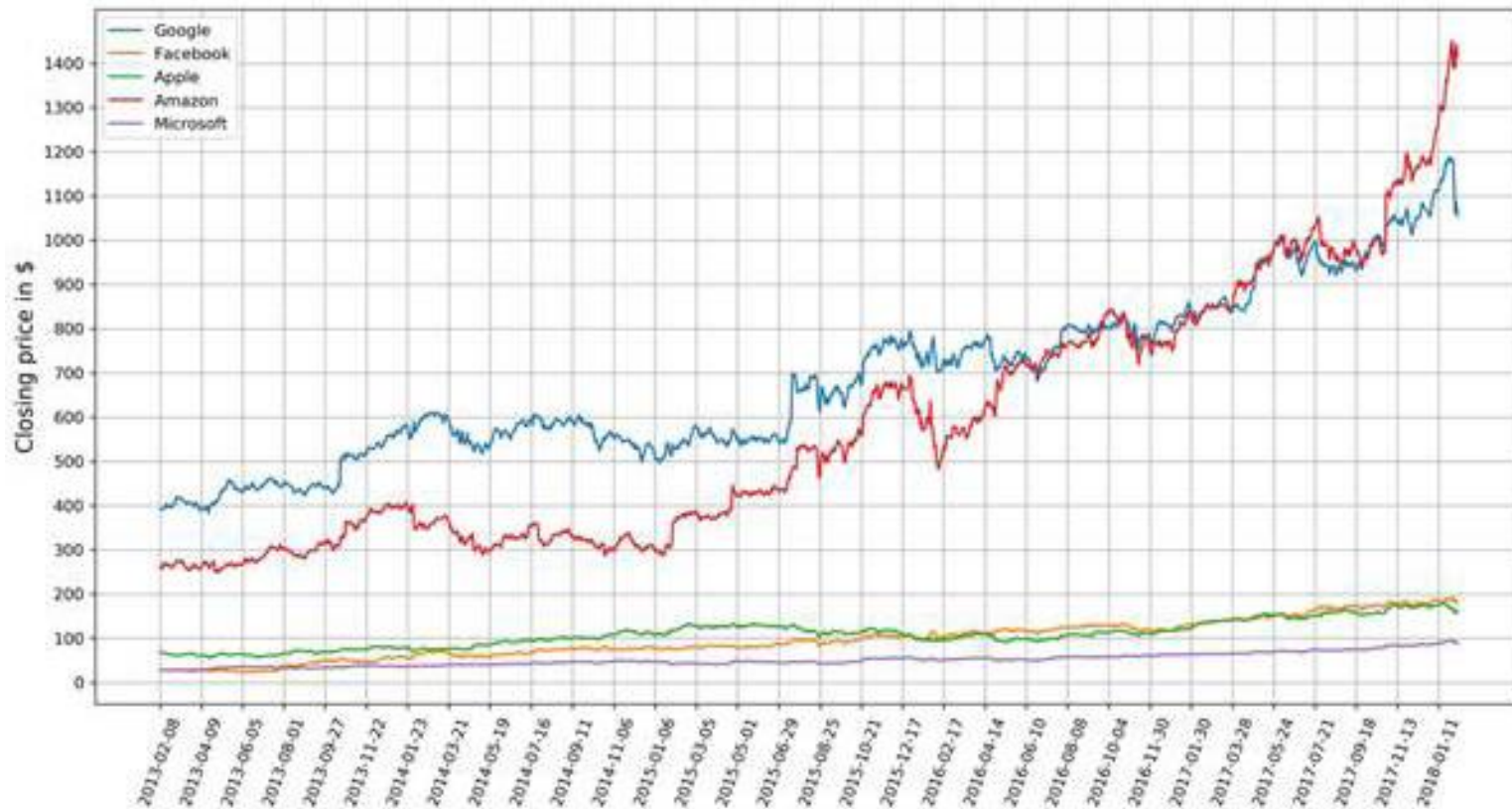
```
# Plotting multiple sets of data
plt.plot([1, 2, 3], label='line1', marker='*', markersize=10)
plt.plot([1.5, 2.5, 3.5], label='line2', marker='o', markersize=10)
plt.title('Biểu đồ song song', color="red")
plt.ylabel('y value', color="red")
plt.legend(title="Ghi chú")
plt.grid(True)
plt.show()
```



# Line Chart



# Line Chart



# Matplotlib

---

## □ Area plot:

### `matplotlib.pyplot.fill_between()`

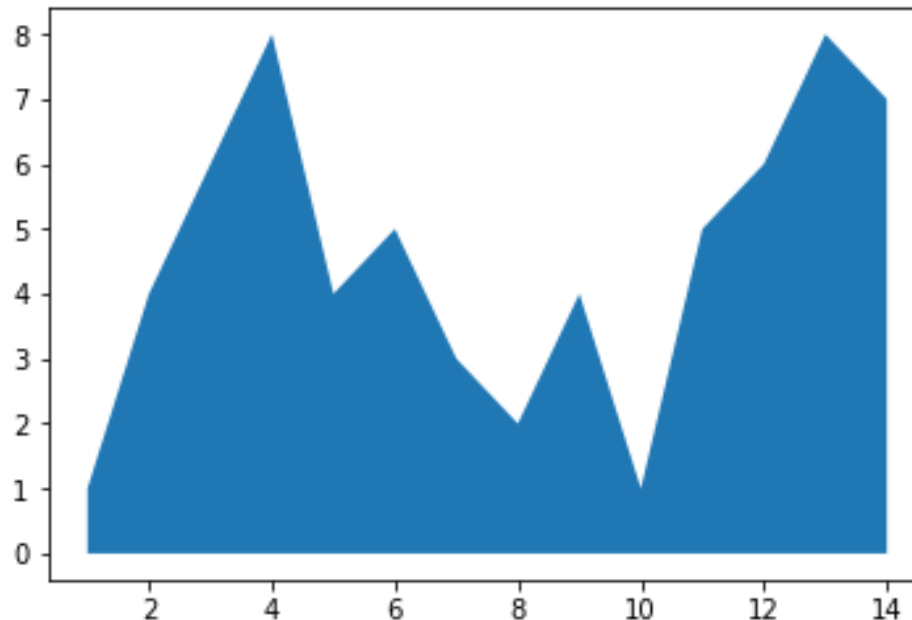
- Biểu đồ area tương tự như biểu đồ line, chỉ khác là vùng giữa trục X và line được tô màu. Nó đại diện cho sự tiến hóa của một biến số. Về cơ bản, trục X biểu thị thời gian hoặc một biến có thứ tự và trục Y biểu thị giá trị của biến khác.



# Matplotlib

- Dạng mặc định

```
# Create data  
x=range(1,15)  
y=[1,4,6,8,4,5,3,2,4,1,5,6,8,7]  
  
# Area plot  
plt.fill_between(x, y)  
plt.show()
```



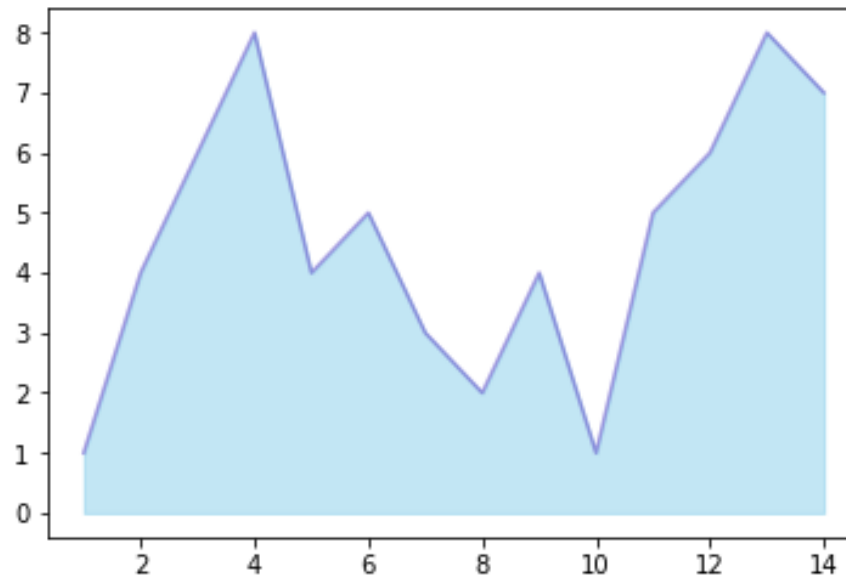
# Matplotlib

- Thay đổi màu, độ trong suốt, thêm line

```
# Create data
x=range(1,15)
y=[1,4,6,8,4,5,3,2,4,1,5,6,8,7]

# Change color, and its transparency
plt.fill_between( x, y, color="skyblue", alpha=0.5)
# Add a stronger line on top
plt.plot(x, y, color="Slateblue", alpha=0.6)
```

```
[<matplotlib.lines.Line2D at 0x13e05834710>]
```

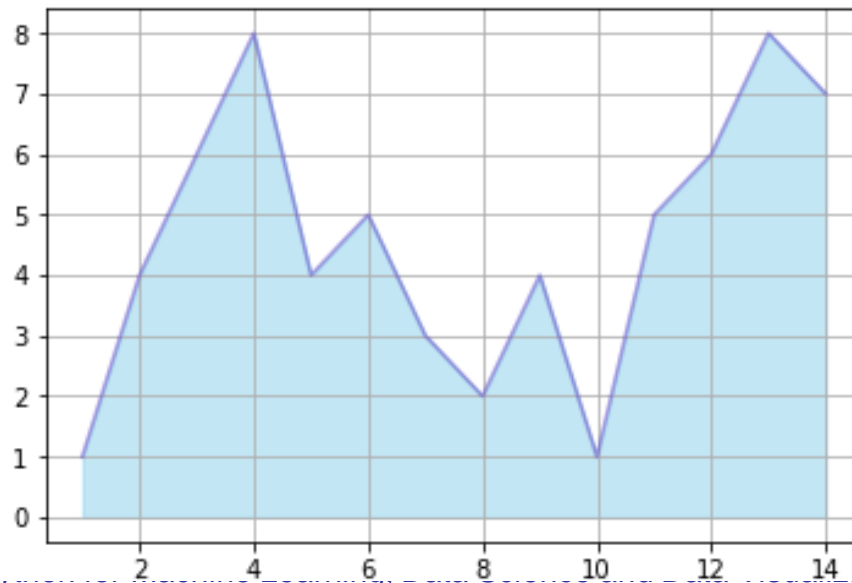


# Matplotlib

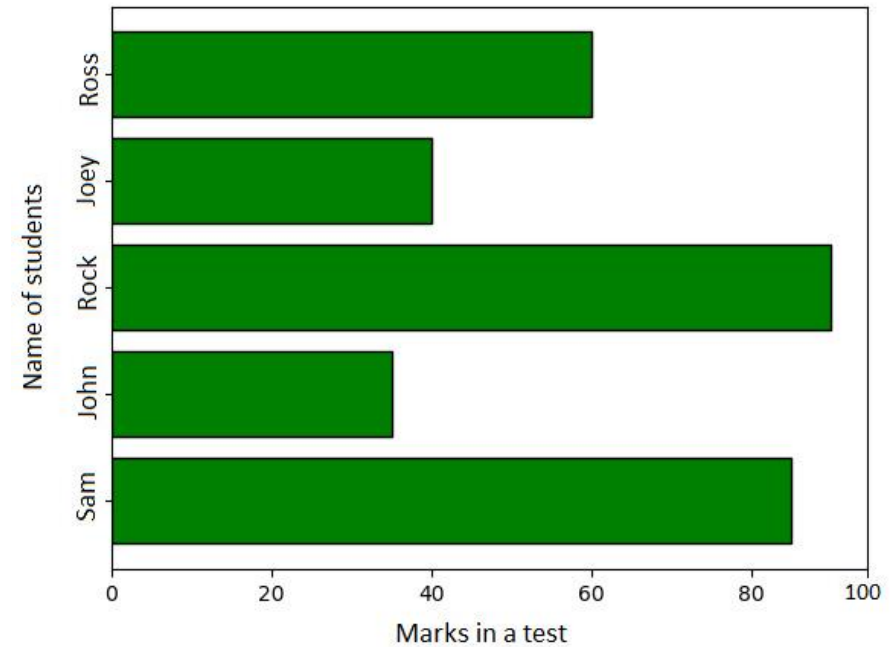
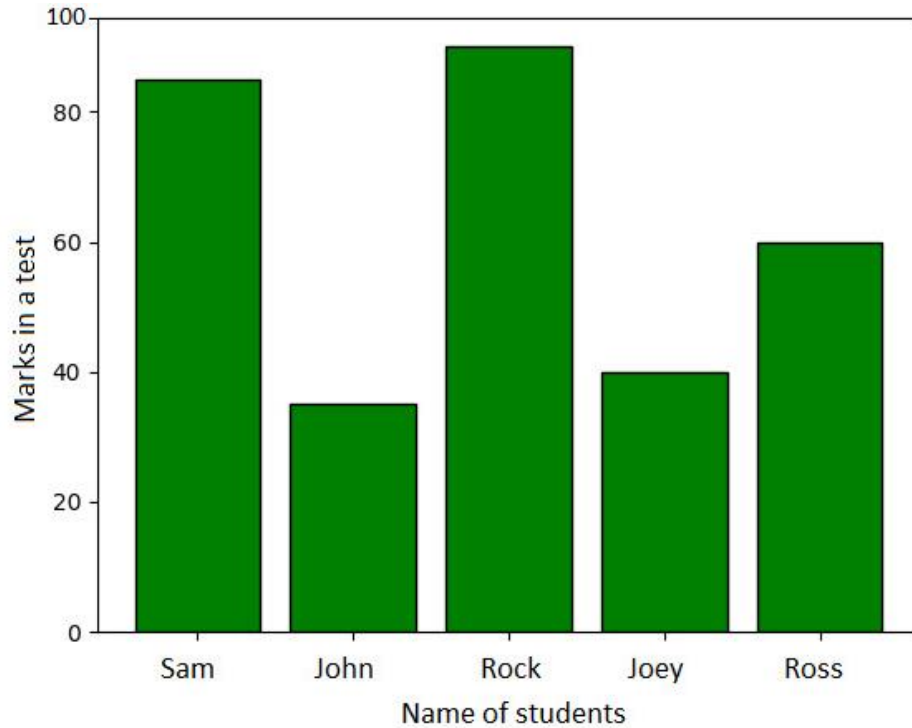
- Thay đổi color, transparency; thêm line, grid

```
# Create data
x=range(1,15)
y=[1,4,6,8,4,5,3,2,4,1,5,6,8,7]

# Change color, and its transparency
plt.fill_between( x, y, color="skyblue", alpha=0.5)
# Add a stronger line on top
plt.plot(x, y, color="Slateblue", alpha=0.6)
plt.grid(True)
plt.show()
```



# BAR CHART



# Matplotlib

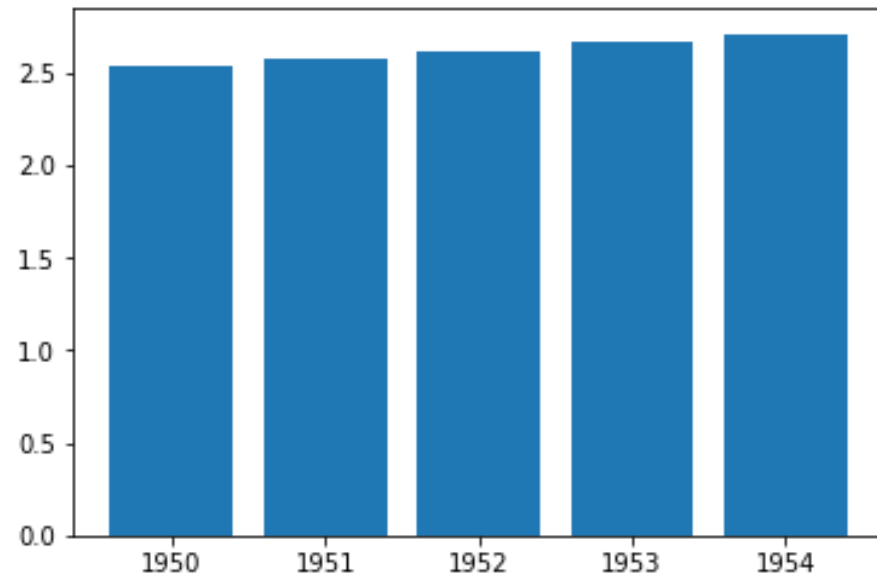
## □ Bar chart: `plt.bar(x, height, [width])`

- Vẽ biểu đồ khối
  - Dạng mặc định

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

```
year = [1950, 1951, 1952, 1953, 1954]  
pop = [2.53, 2.57, 2.62, 2.67, 2.71]
```

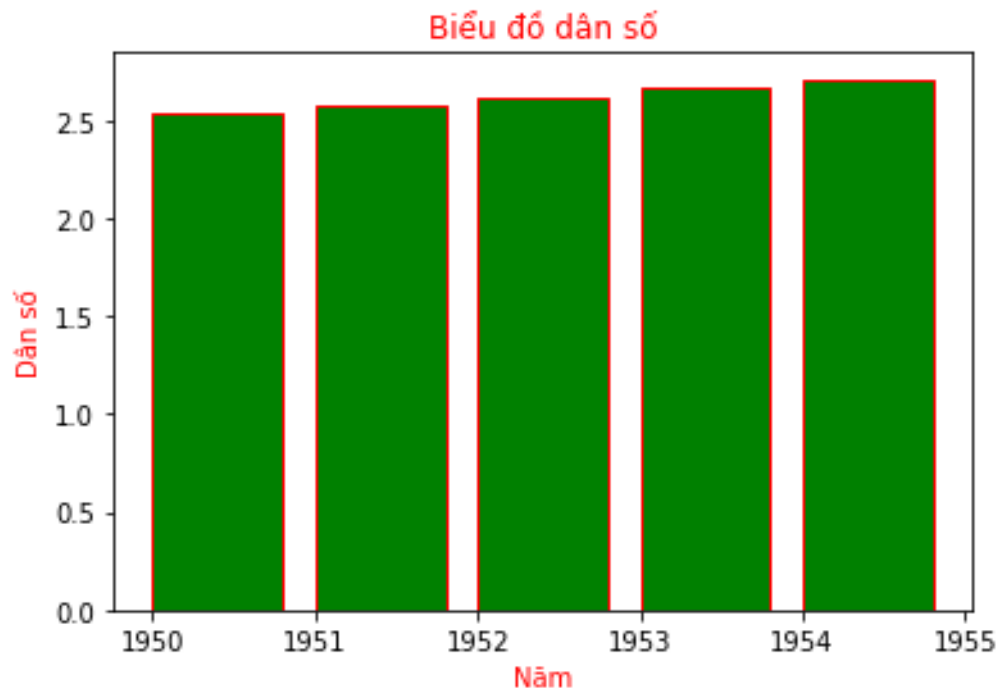
```
plt.bar(year, pop)  
plt.show()
```



# Matplotlib

## ■ Dạng có align, edgecolor

```
# align edge
plt.bar(year, pop, color='g', align='edge', edgecolor="red")
plt.title('Biểu đồ dân số', color="red")
plt.ylabel('Dân số', color="red")
plt.xlabel('Năm', color="red")
plt.show()
```



# Matplotlib

## ■ Dạng cột chồng nhau - Stacked bar chart

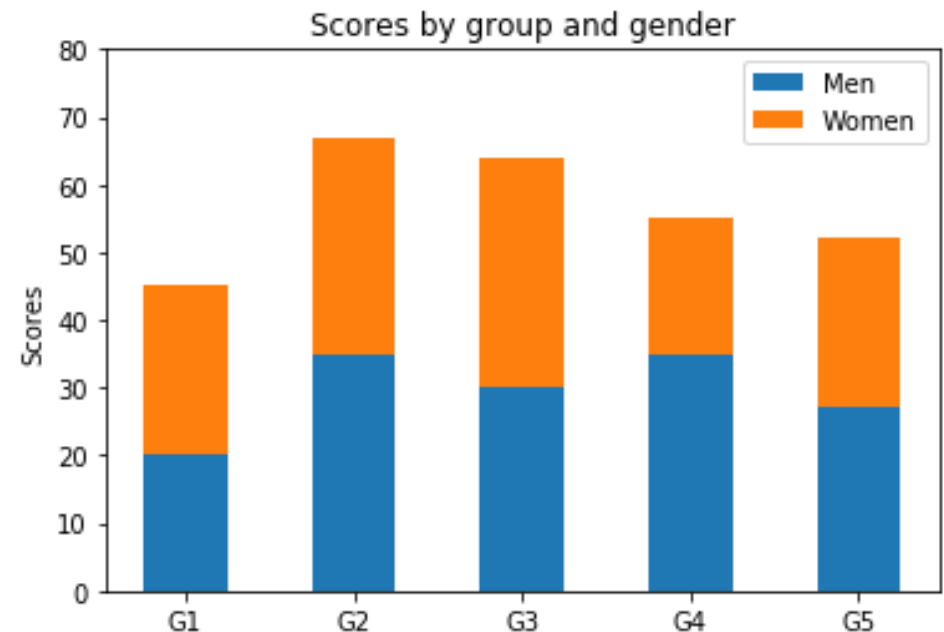
```
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)

ind = np.arange(N)    # the x locations for the groups
width = 0.5           # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, menMeans, width, label="Men")
p2 = plt.bar(ind, womenMeans, width, bottom=menMeans, label="Women")

plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend()

plt.show()
```



# Matplotlib

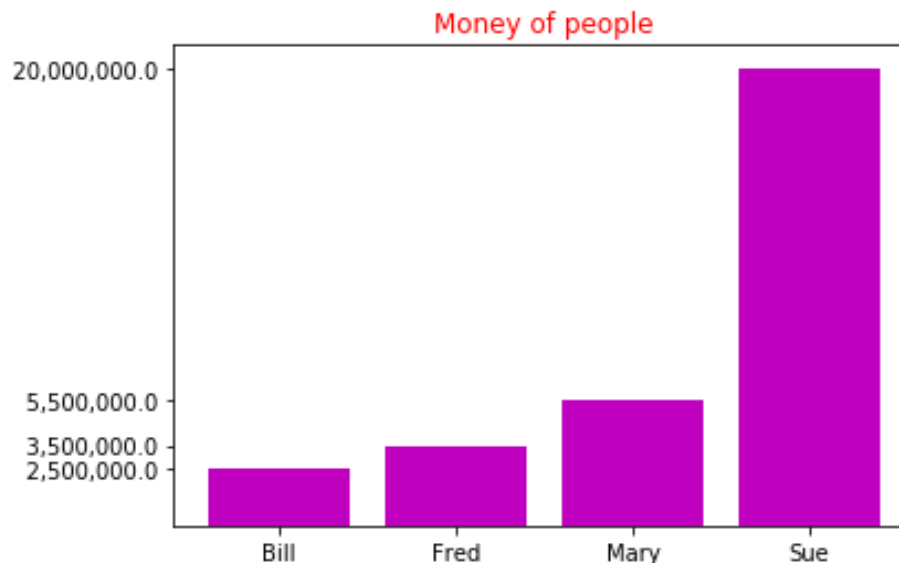
## ■ Dạng có định dạng tiền tệ

```
import matplotlib.pyplot as plt

import pandas as pd
name = ['Bill', 'Fred', 'Mary', 'Sue']
money = pd.Series([2.5e6, 3.5e6, 5.5e6, 2.0e7])

plt.bar(name, money, color="m")
plt.xticks(name)
plt.yticks(money, money.map(lambda x: '{:15,.1f}'.format(x)))
plt.title("Money of people", color="red")

plt.show()
```





# Matplotlib

## ■ Dạng ghi chú cho nhiều thành phần, thêm vị trí

```
from matplotlib import pyplot as plt

x = [5,8,10]
y = [12,16,6]

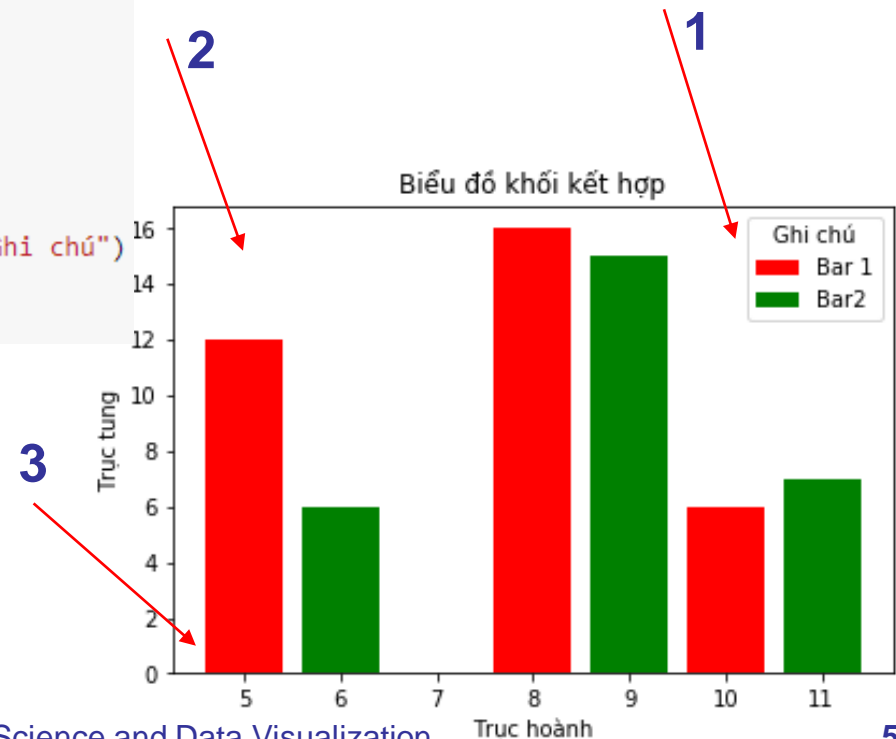
x2 = [6,9,11]
y2 = [6,15,7]

bar1 = plt.bar(x, y, color='r', align='center')
bar2 = plt.bar(x2, y2, color='g', align='center')

plt.title('Biểu đồ khối')
plt.ylabel('Trục tung')
plt.xlabel('Trục hoành')
plt.title('Biểu đồ khối kết hợp')

plt.legend([bar1, bar2], ['Bar 1', 'Bar2'], loc = 1, title="Ghi chú")

plt.show()
```

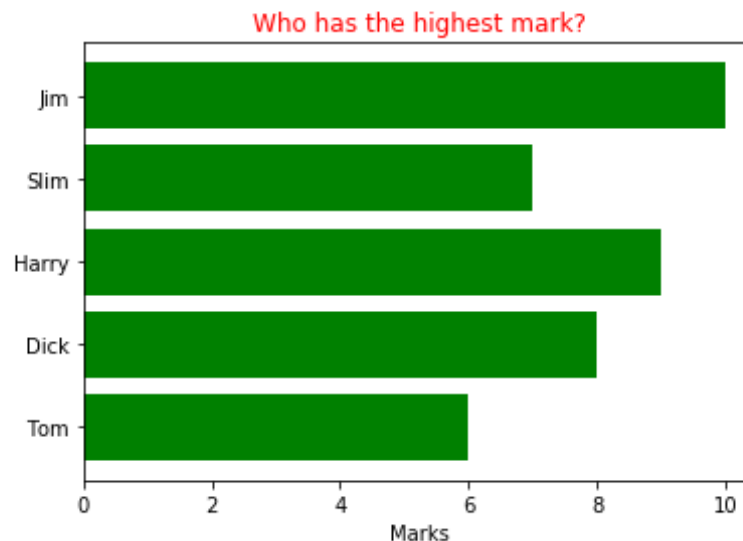


# Matplotlib

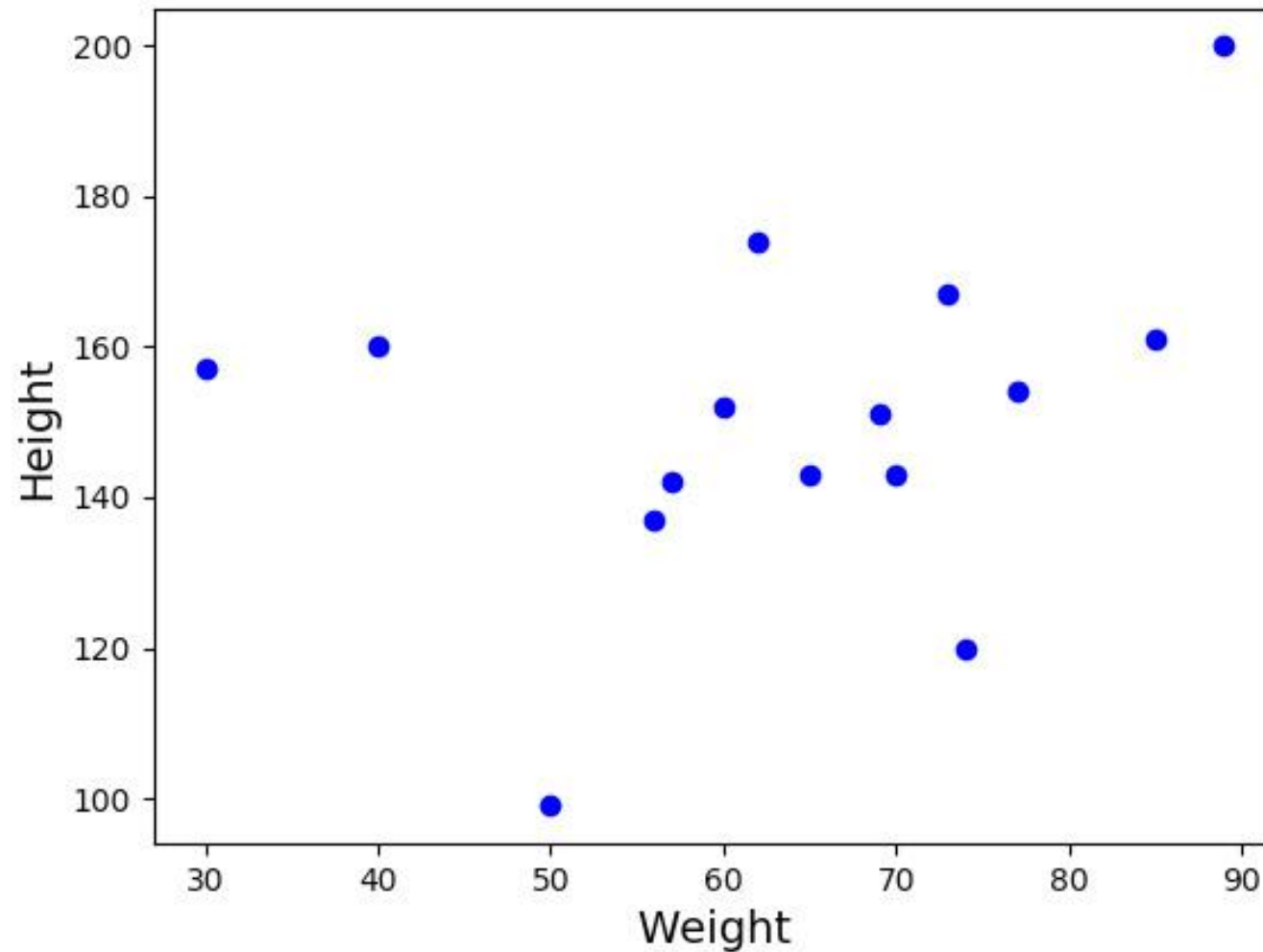
## ❑ Biểu đồ khối ngang: matplotlib.pyplot.barh()

```
# create data
people = ['Tom', 'Dick', 'Harry', 'Slim', 'Jim']
marks = [6, 8, 9, 7, 10]
y_pos = np.arange(len(people))

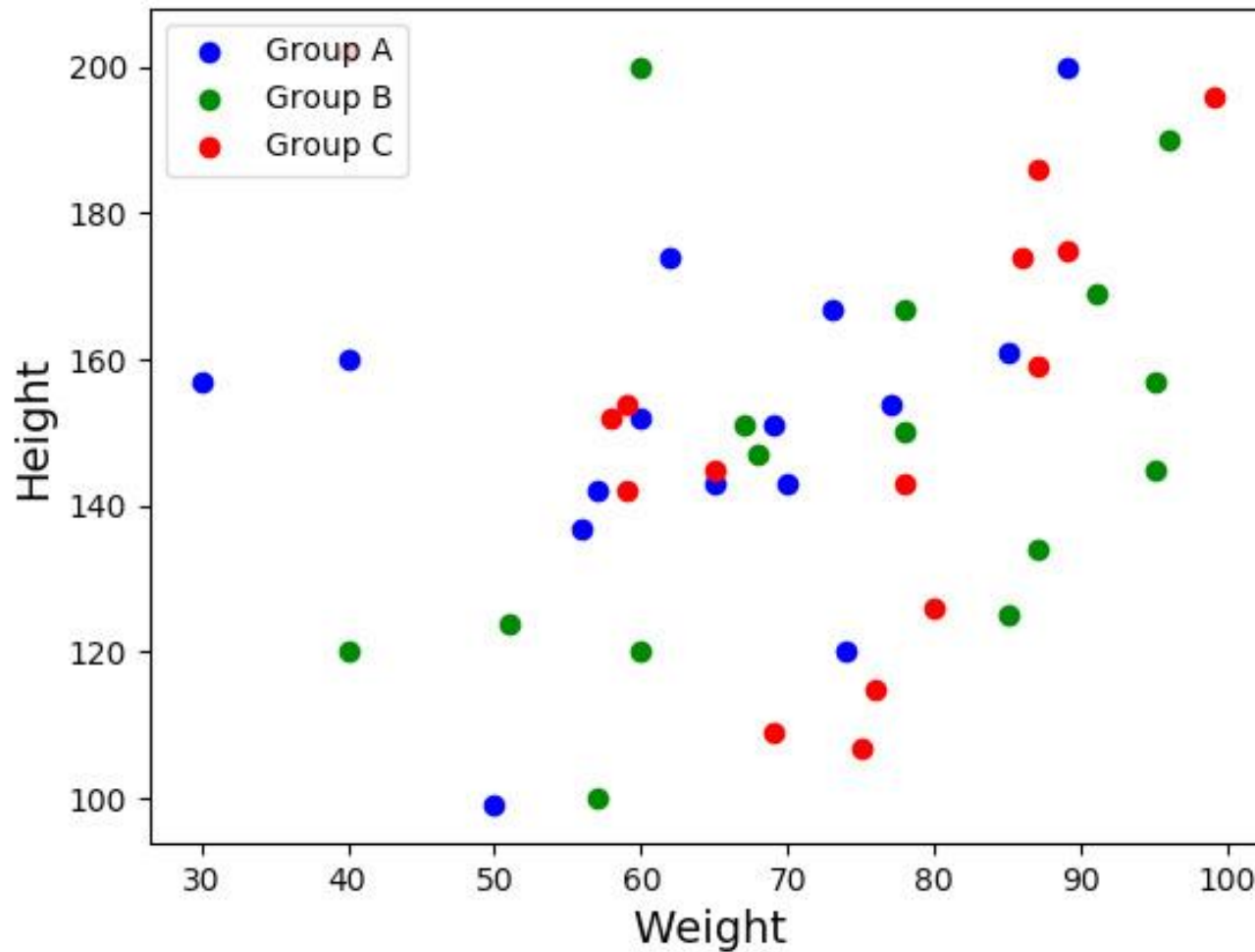
# Horizontal bar chart
plt.barh(y_pos, marks, align='center',
         color='green', ecolor='black')
plt.yticks(y_pos, people)
plt.xlabel('Marks')
plt.title('Who has the highest mark?', color = 'red')
plt.show()
```



# SCATTER PLOT



# SCATTER PLOT



# Matplotlib

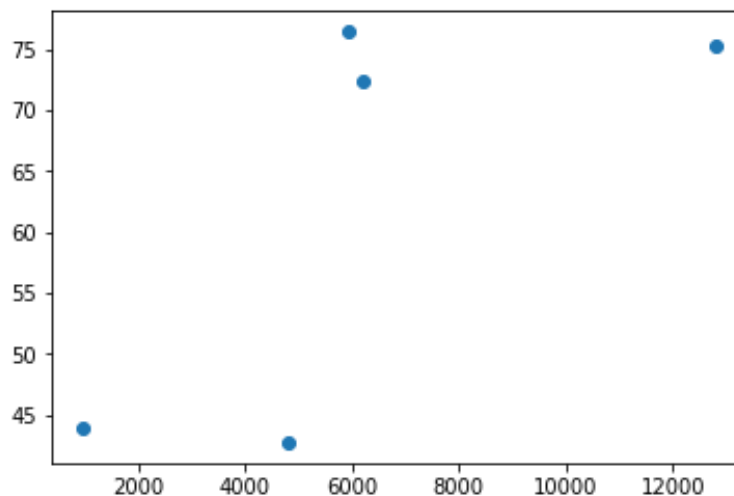
❑ Scatter plot: `plt.scatter(x, y, [s=size], [c=color])`

- Vẽ biểu đồ phân tán của y và x

- Dạng mặc định

```
gdp_cap = [974.5803384, 5937.029525999998, 6223.367465, 4797.231267, 12779.37964]  
life_exp = [43.828, 76.423, 72.301, 42.731, 75.32]
```

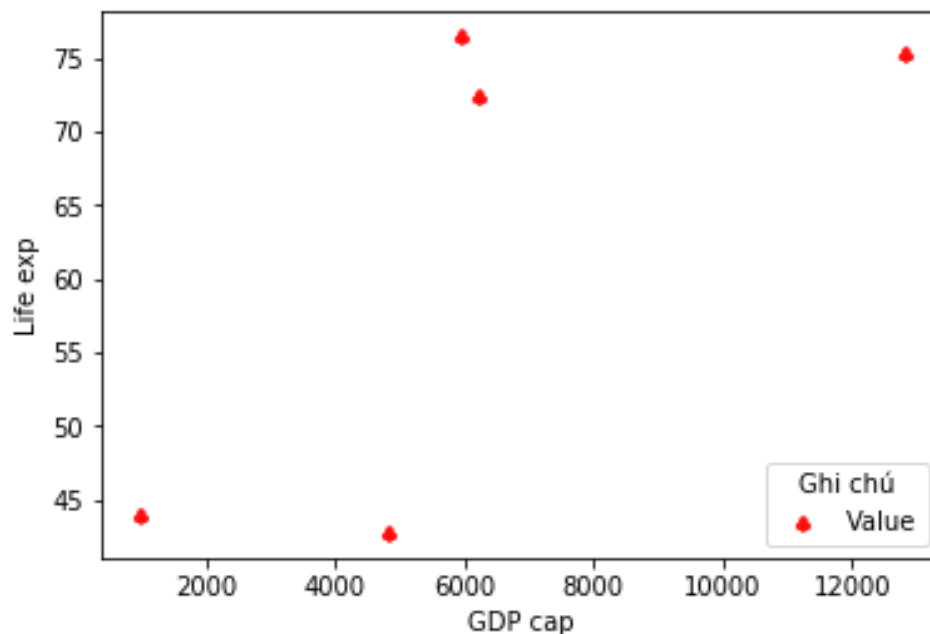
```
plt.scatter(gdp_cap, life_exp)  
plt.show()
```



# Matplotlib

## ■ Dạng có location của ghi chú, marker

```
plt.scatter(gdp_cap, life_exp, c="r", alpha=0.9, marker=r'$\clubsuit$', label="Value")  
plt.xlabel("GDP cap")  
plt.ylabel("Life exp")  
plt.legend(title="Ghi chú", loc=4)  
plt.show()
```



# BUBBLE PLOT

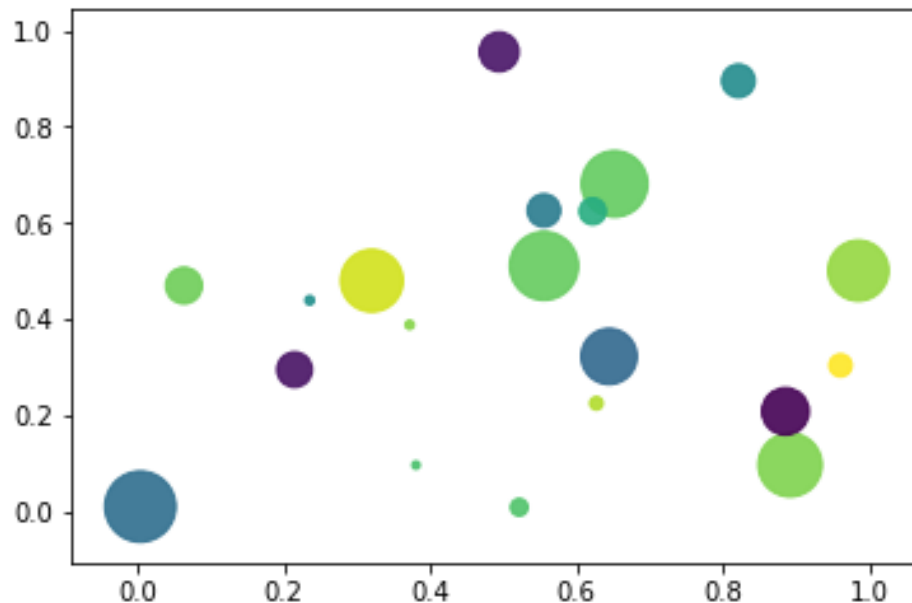


# Matplotlib

- Dạng hiển thị điểm theo diện tích (bubble chart)

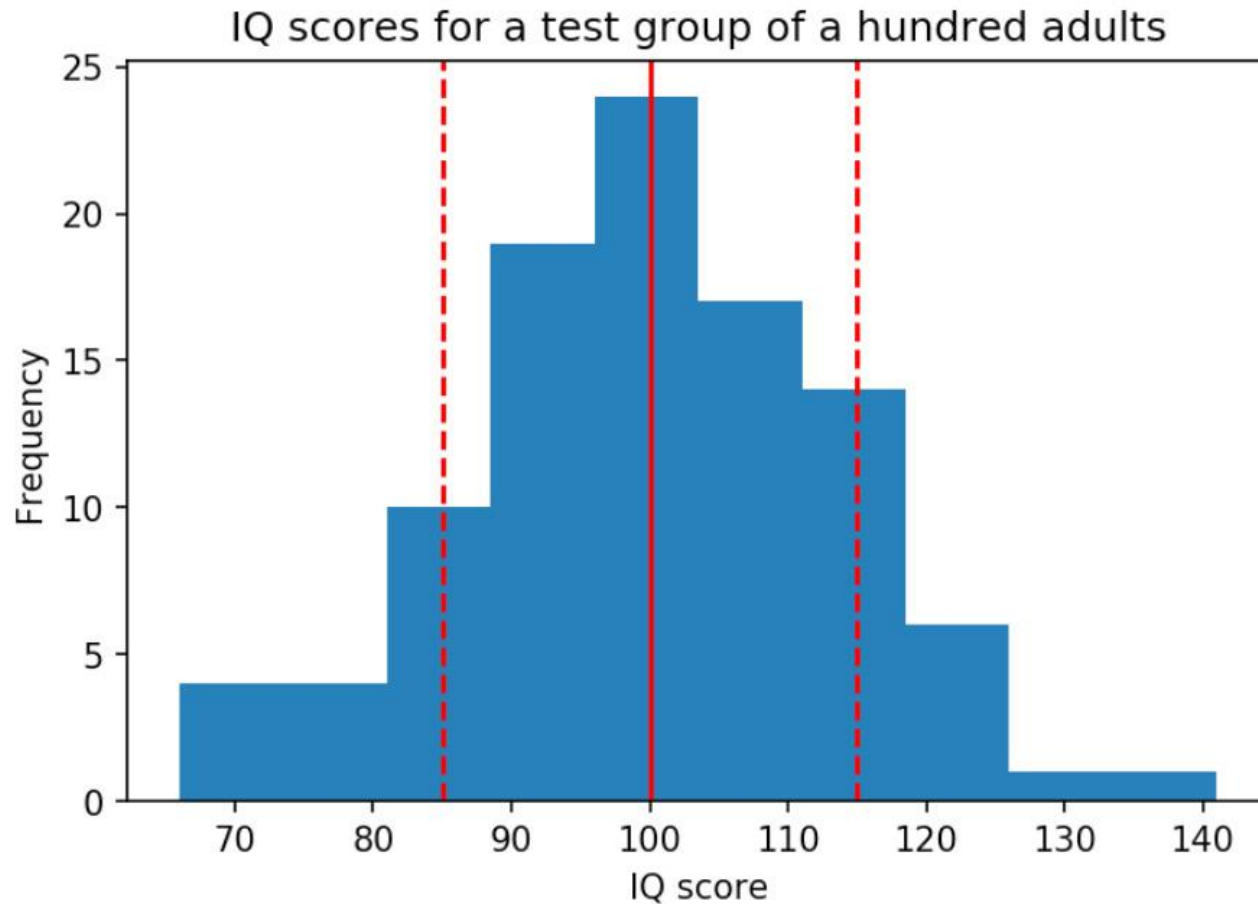
```
N = 20
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2 # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.9)
plt.show()
```

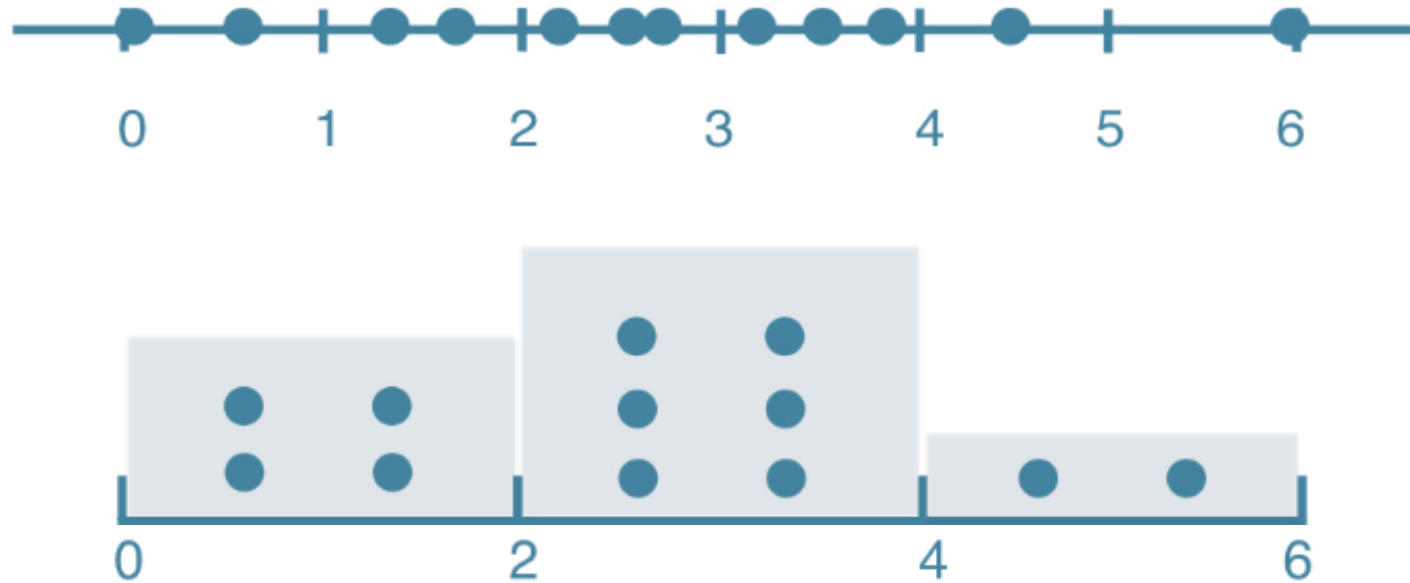




# HISTOGRAM



# HISTOGRAM



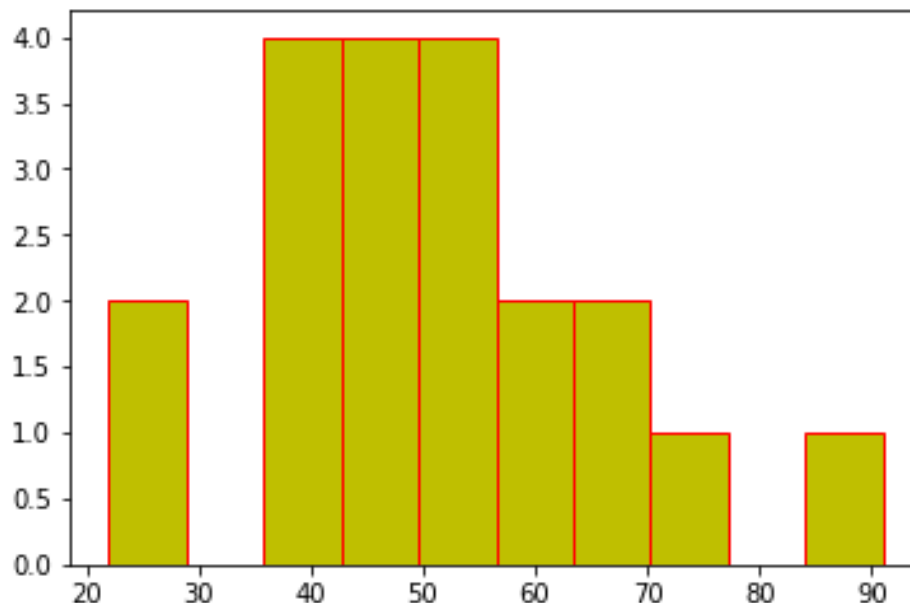
# Matplotlib

## □ Histogram:

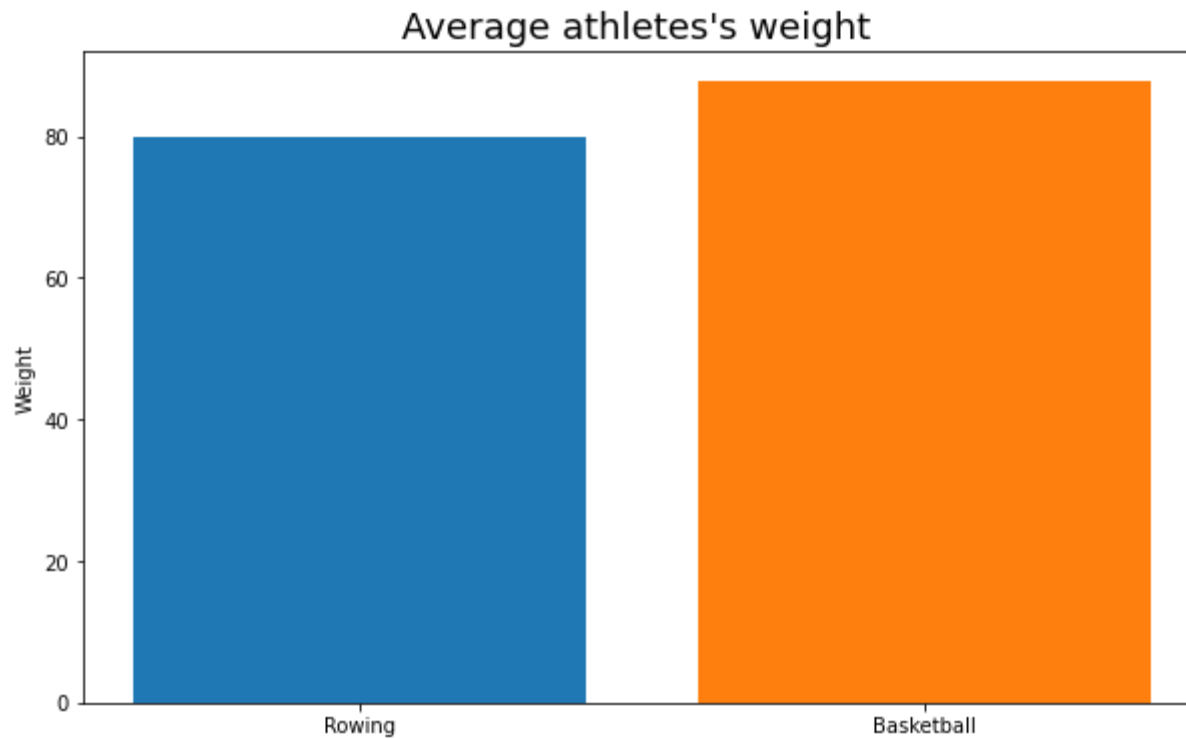
```
plt.hist(x, [bins], [range], [density])
```

### ● Vẽ biểu đồ histogram

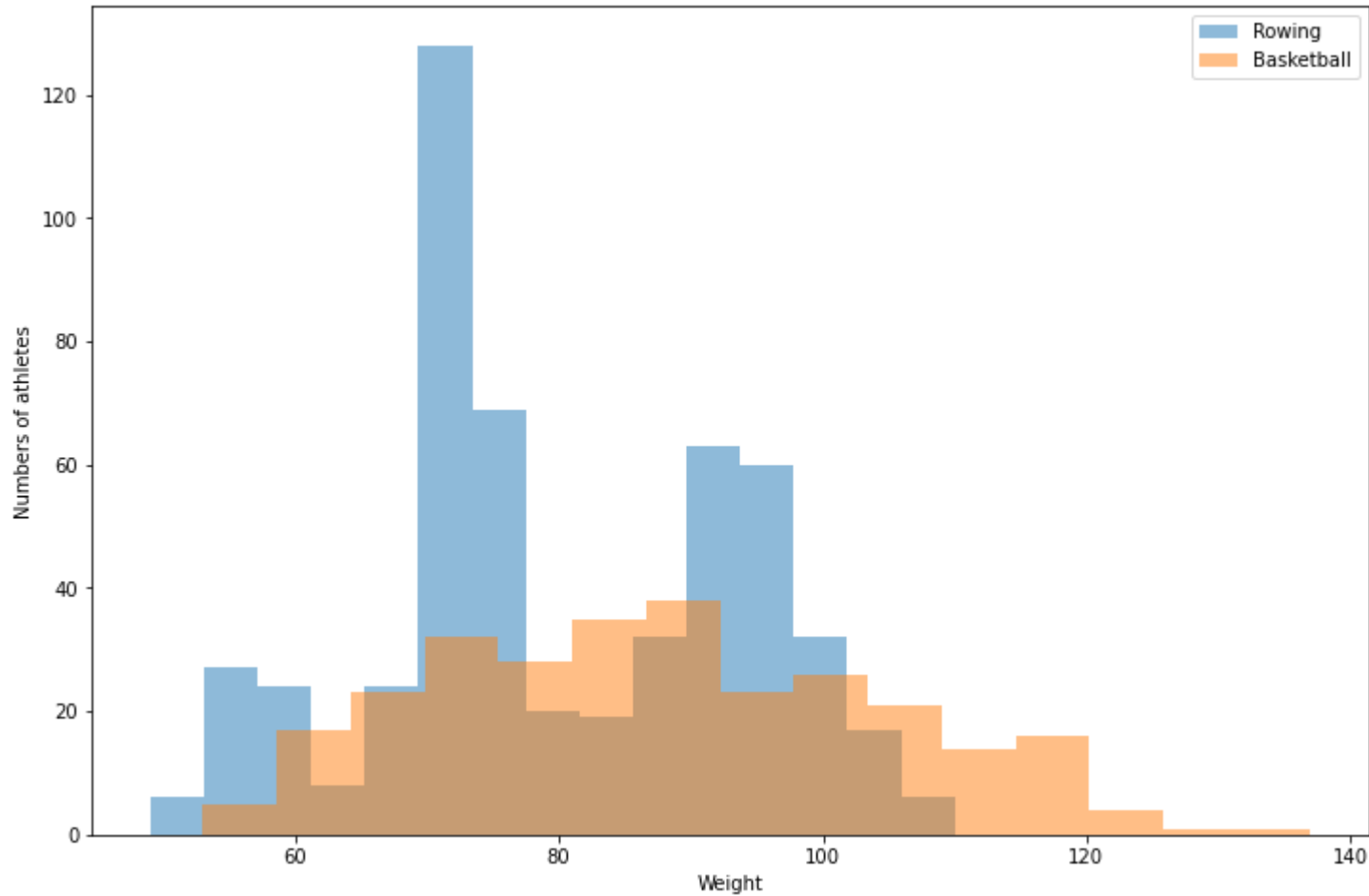
```
■ plt.hist(age, color = 'y', edgecolor='r')  
plt.show()
```



# Matplotlib



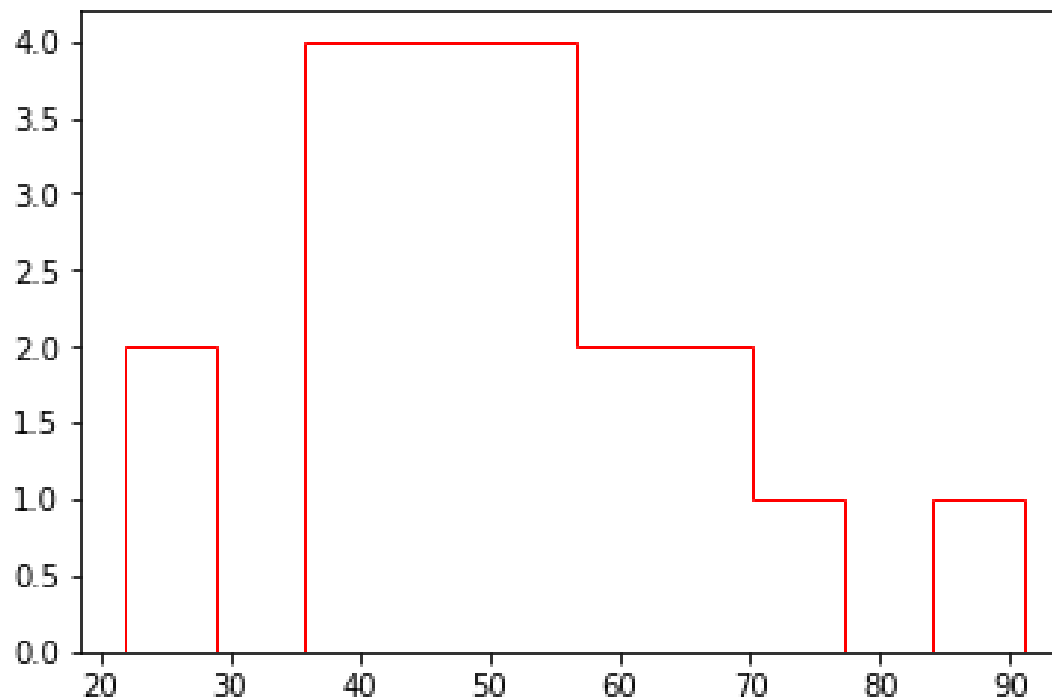
# Matplotlib



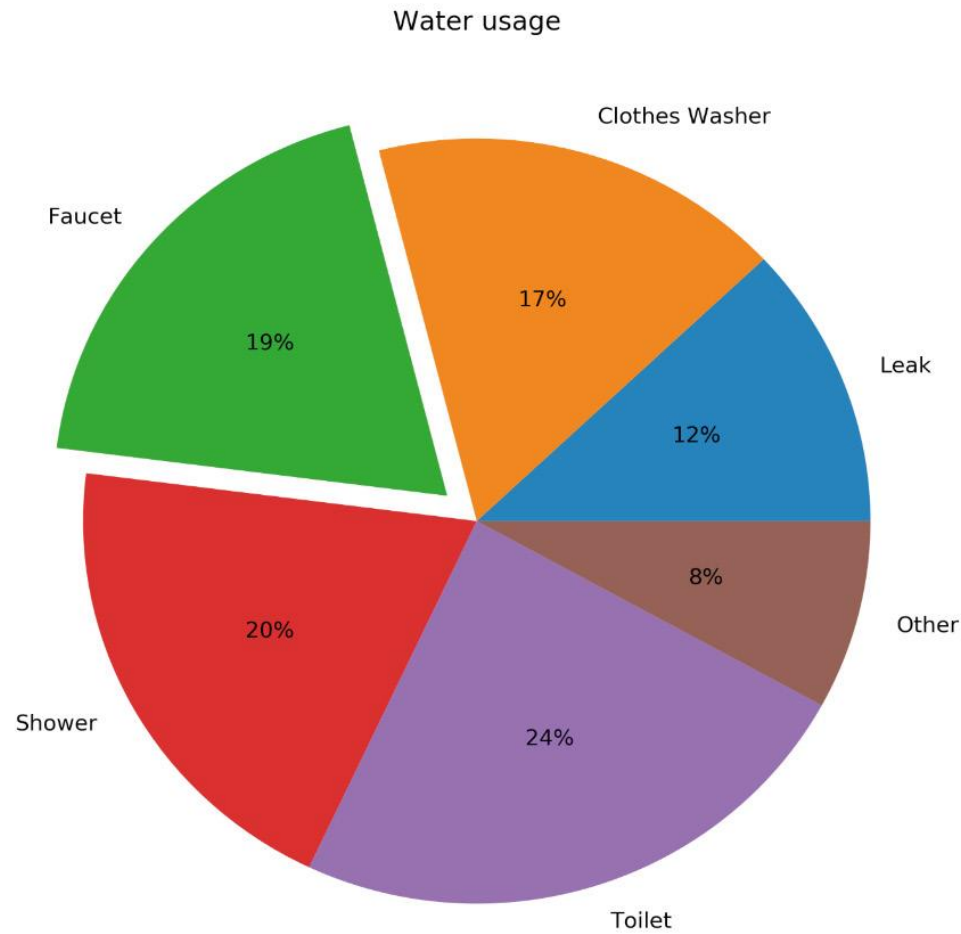
# Matplotlib

- Dạng có **histtype** : 'bar', 'barstacked', 'step', 'stepfilled'

```
plt.hist(age, color = 'r', histtype='step')  
plt.show()
```



# PIE CHART



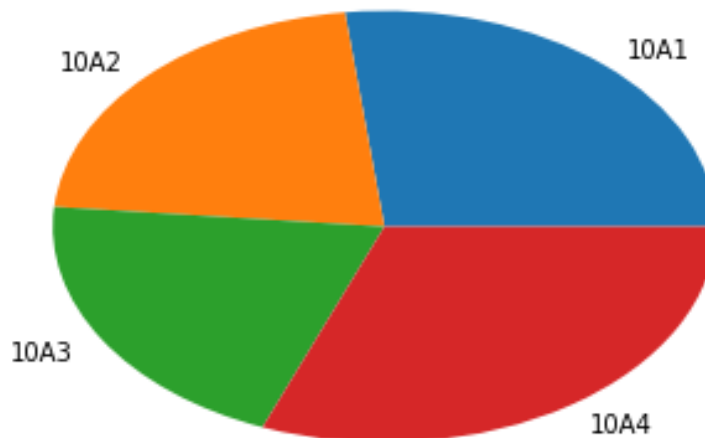
# Matplotlib

❑ Pie chart: `plt.pie(x, [explode], [labels], [autopct])`

- Vẽ pie chart từ array

- Dạng mặc định

```
plt.pie(x, labels = labels)  
plt.show()
```

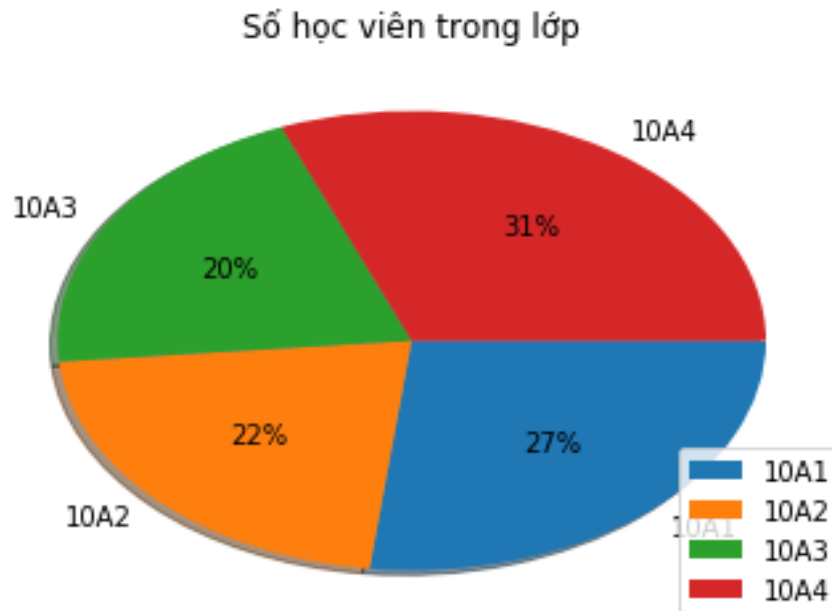




# Matplotlib

- Dạng có counterclockwise, shadow, autopct (tỷ lệ phần trăm)

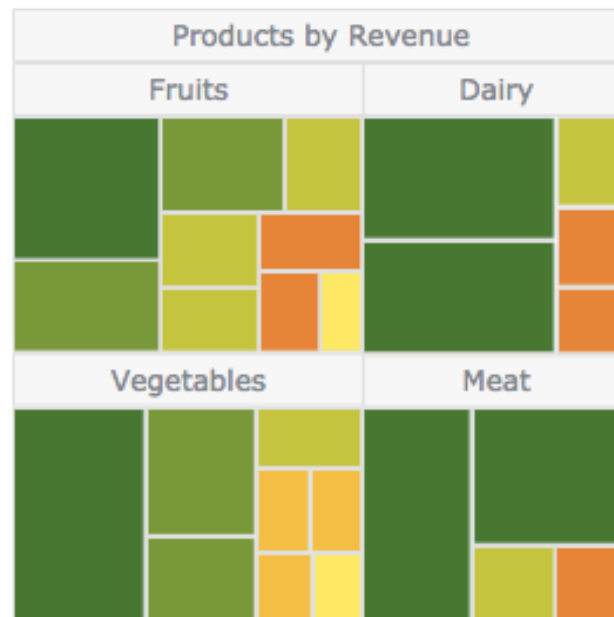
```
plt.pie(x, labels = labels, counterclock=False, shadow=True, autopct='%.0f%%')  
plt.title("Số học viên trong lớp")  
plt.legend(loc=4)  
plt.show()
```



# Matplotlib

## □ Tree map

- Tương tự như pie chart nhưng nó hoạt động tốt hơn vì tạo ra trực quan rõ ràng, tránh được việc hiểu lầm về sự đóng góp của các nhóm.



# Matplotlib

---

## ❑ Sử dụng thư viện squarify để vẽ tree map

- Cài đặt: `pip install squarify`
- Dùng `squarify.plot()` để vẽ

# Matplotlib

- Dạng mặc định

```
# with 2 lists  
plt.figure(figsize=(8,6))  
squarify.plot(sizes=[7,18,20,25], label=['Excellent', 'Very Good', 'Good', 'Average'])  
plt.axis('off')  
plt.show()
```



# Matplotlib

- Dạng mặc định

`import squarify`

	student_type	number_per_type
0	Excellent	7
1	Very Good	18
2	Good	20
3	Average	15



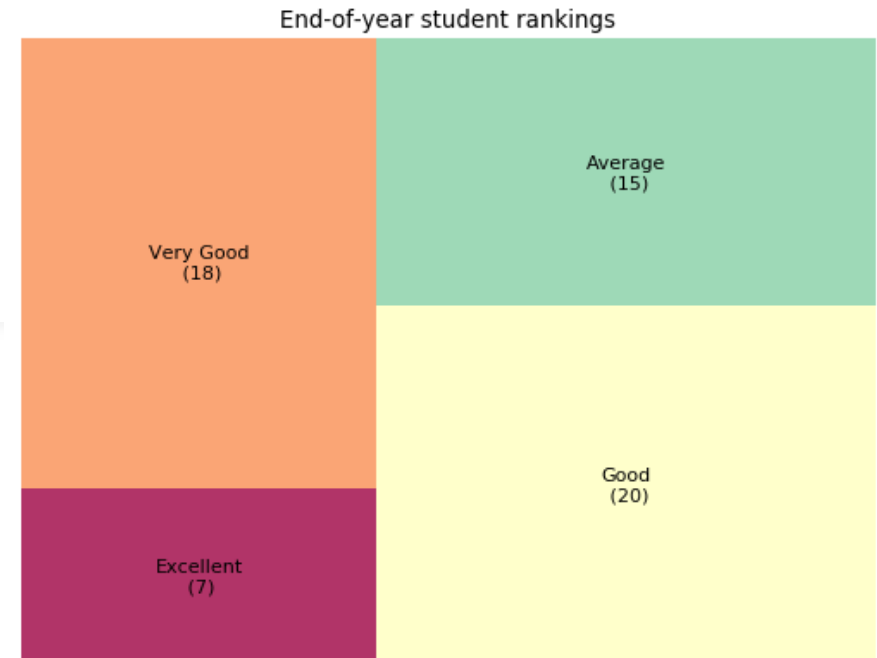
```
# with dataframe
plt.figure(figsize=(8,6))
squarify.plot(sizes=df['number_per_type'], label=df['student_type'])
plt.axis('off')
plt.show()
```

# Matplotlib

## • Tùy chỉnh color, label

### import squarify

	student_type	number_per_type
0	Excellent	7
1	Very Good	18
2	Good	20
3	Average	15

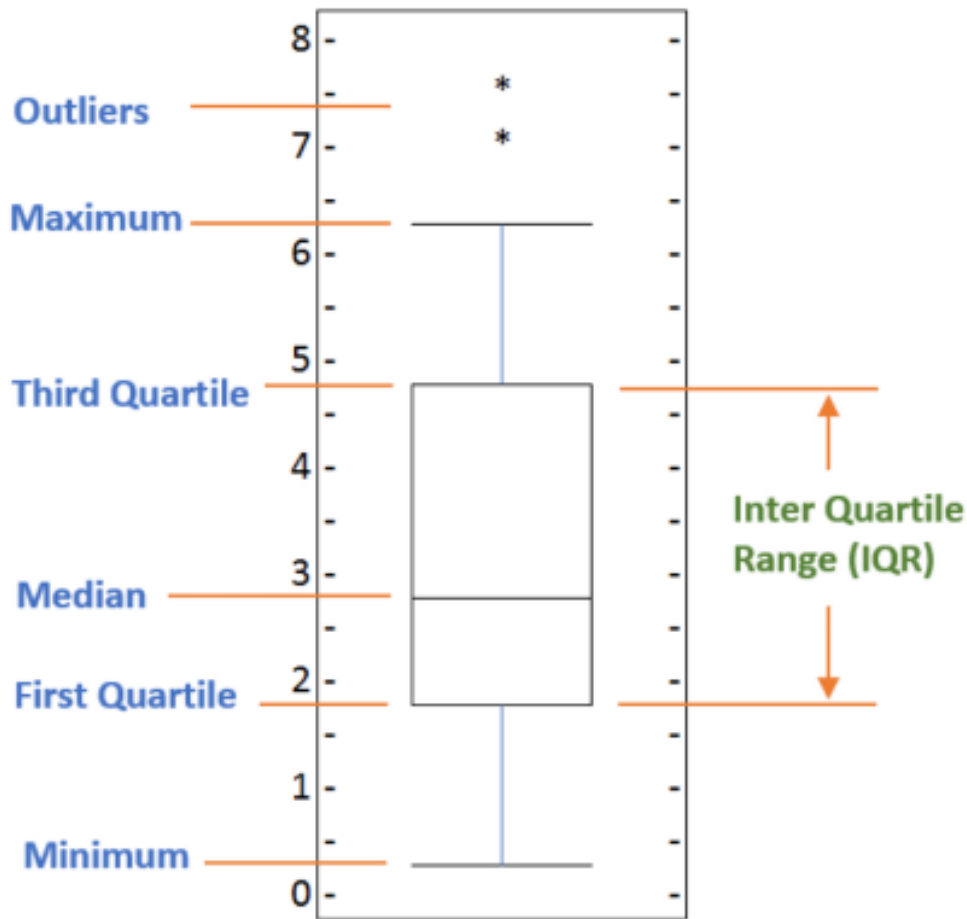


```
# create data
labels = df.apply(lambda x: str(x[0]) + "\n (" + str(x[1]) + ")", axis=1)
sizes = df['number_per_type'].values
colors = [plt.cm.Spectral(i/float(len(labels))) for i in range(len(labels))]
# Draw Plot
plt.figure(figsize=(8,6), dpi= 80)
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=.8)

# Decorate
plt.title('End-of-year student rankings')
plt.axis('off')
plt.show()
```

# Matplotlib

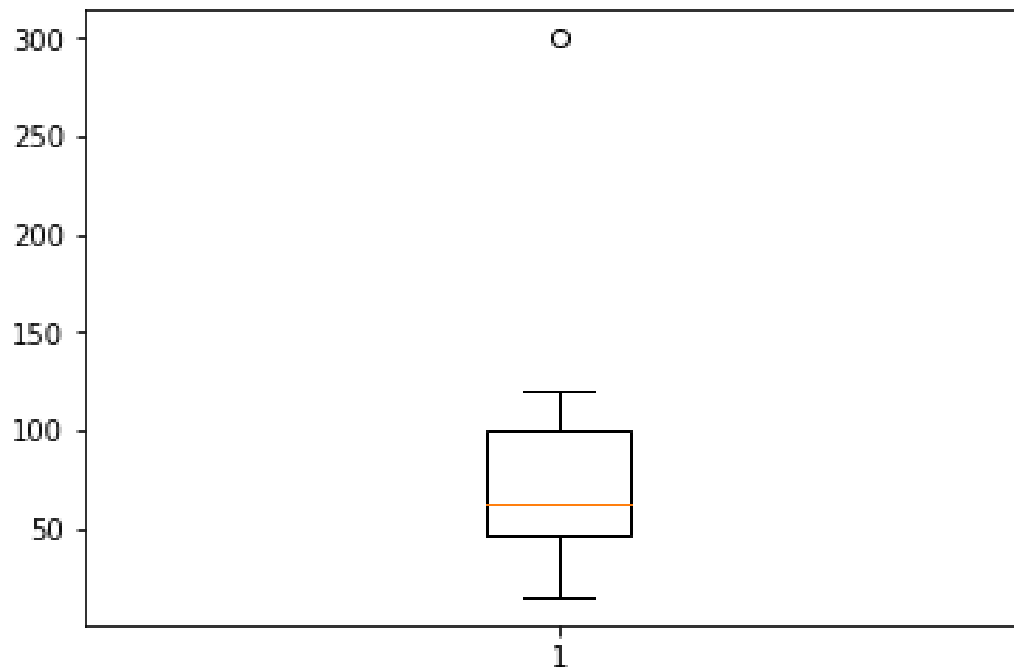
□ **Boxplot:** `plt.boxplot(x)`



# Matplotlib

- Dạng mặc định

```
# create data
price = np.array([15, 45, 50, 120, 300, 34, 55, 70, 105, 85])
# box plot
plt.boxplot(price)
plt.show()
```

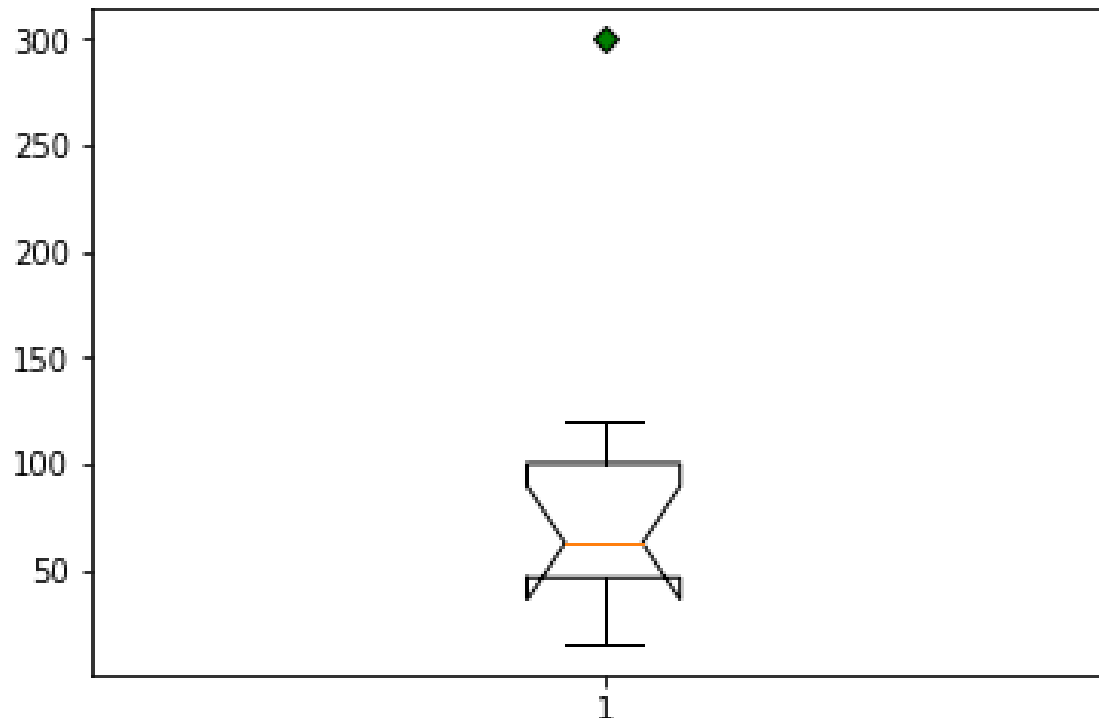




# Matplotlib

- Có notch tại median, thay đổi outlier symbol

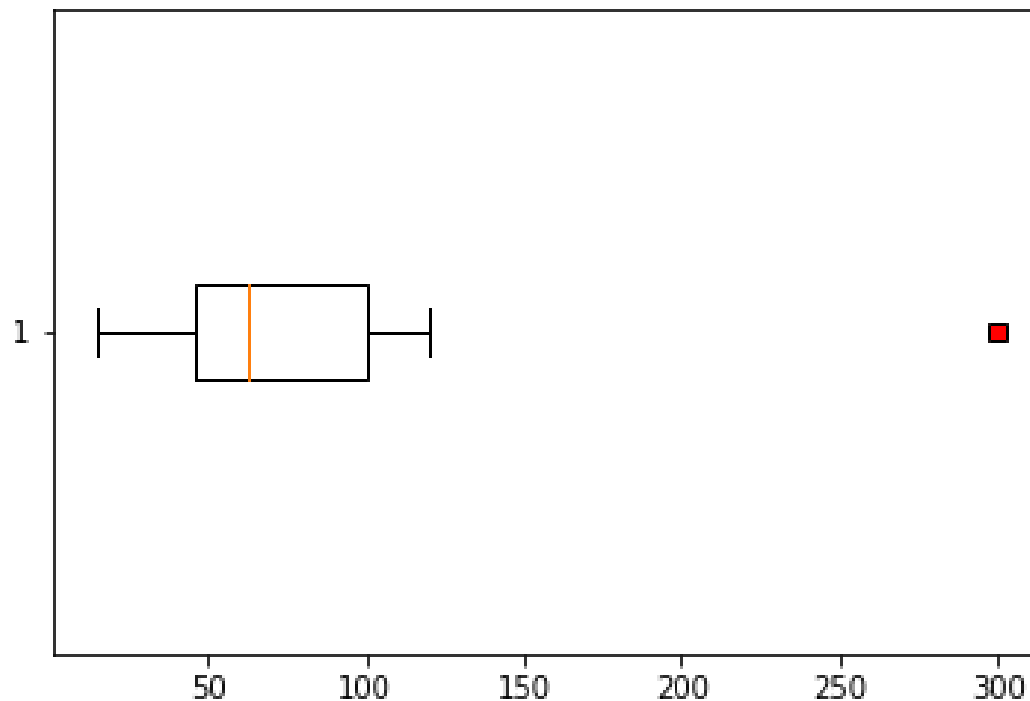
```
# Have notch at median, change outlier symbol  
green_diamond = dict(markerfacecolor='g', marker='D')  
plt.boxplot(price, notch=True, flierprops = green_diamond)  
plt.show()
```



# Matplotlib

- Hiển thị box theo horizontal

```
# Horizontal Boxes  
red_square = dict(markerfacecolor='r', marker='s')  
plt.boxplot(price, vert=False, flierprops=red_square)  
plt.show()
```



# Matplotlib

---

- Cách tính giá trị ngoại lai

$$\text{Lower Outlier} = Q1 - (1.5 \times IQR)$$

$$\text{Higher Outlier} = Q3 + (1.5 \times IQR)$$

# Matplotlib

## ❑ Waffle chart

- Là một hình ảnh trực quan thú vị thường được tạo ra để hiển thị tiến trình hướng tới mục tiêu. Đây là một lựa chọn hiệu quả khi ta cố gắng thêm các tính năng trực quan vào một hình ảnh chủ yếu bao gồm các ô, giống như Excel dashboard.



# Matplotlib

---

## ❑ Sử dụng thư viện pyWaffle để vẽ Waffle Chart

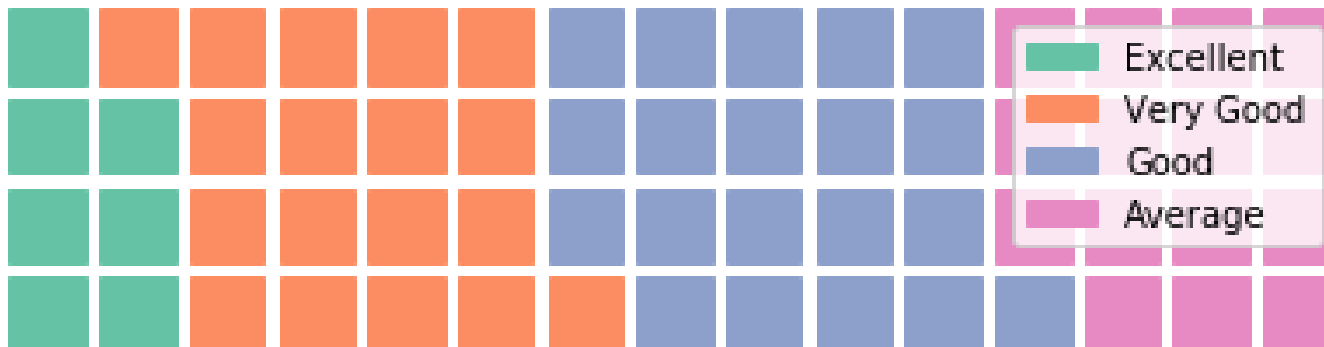
- Cài đặt: `pip install pywaffle`
- Dùng `plt.figure(FigureClass=Waffle, ...)` để vẽ

# Matplotlib

## • Dạng mặc định

	student_type	number_per_type
0	Excellent	7
1	Very Good	18
2	Good	20
3	Average	15

```
fig = plt.figure(
    FigureClass=Waffle,
    rows=df.shape[0],
    values=df.number_per_type,
    labels=list(df.student_type)
)
```

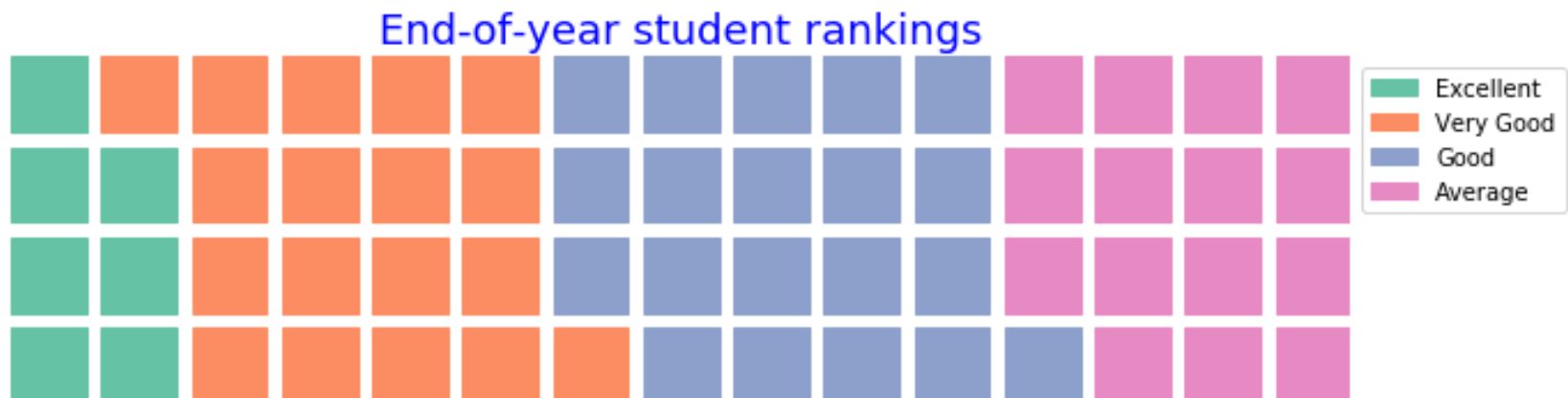


# Matplotlib

## •CÓ figsize, legend, title

```
# Legend, figsize
fig = plt.figure(
    FigureClass=Waffle,
    rows=df.shape[0],
    values=df.number_per_type,
    labels=list(df.student_type),
    figsize=(10, 5),
    legend={'loc': 'upper left', 'bbox_to_anchor': (1, 1)}
)
plt.title("End-of-year student rankings", fontsize=18, color = 'b')
```

```
Text(0.5, 1.0, 'End-of-year student rankings')
```



# Matplotlib

---

## □ Word Clouds

- Word clouds (còn được gọi là text clouds hay tag clouds) được tạo ra theo cách đơn giản như sau: càng nhiều từ cụ thể xuất hiện trong nguồn dữ liệu văn bản (như bài phát biểu - speech, bài đăng trên blog - blog post, hoặc cơ sở dữ liệu), thì nó càng lớn và trọng tâm hơn khi thể hiện trong word clouds. Nó hiển thị một danh sách các từ, tầm quan trọng của mỗi từ được hiển thị với kích thước phông chữ hoặc màu sắc. Định dạng này rất hữu ích khi giúp người xem nhanh chóng nhận thức các thuật ngữ nổi bật nhất.



# Matplotlib

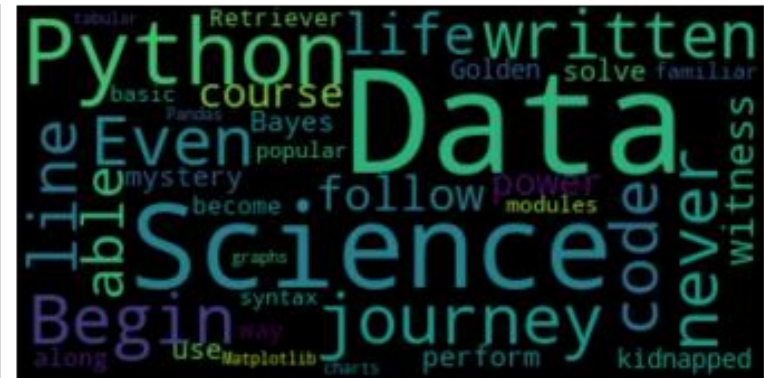
---

## ❑ Sử dụng thư viện wordcloud để vẽ Word Clouds

- Cài đặt: `pip install wordcloud`
- Dùng `WordCloud()` và `plt.imshow()` để vẽ
- Các bước thực hiện:
  - Thu thập dữ liệu – text data
  - Tạo wordcloud image
  - Hiển thị bằng cách sử dụng matplotlib

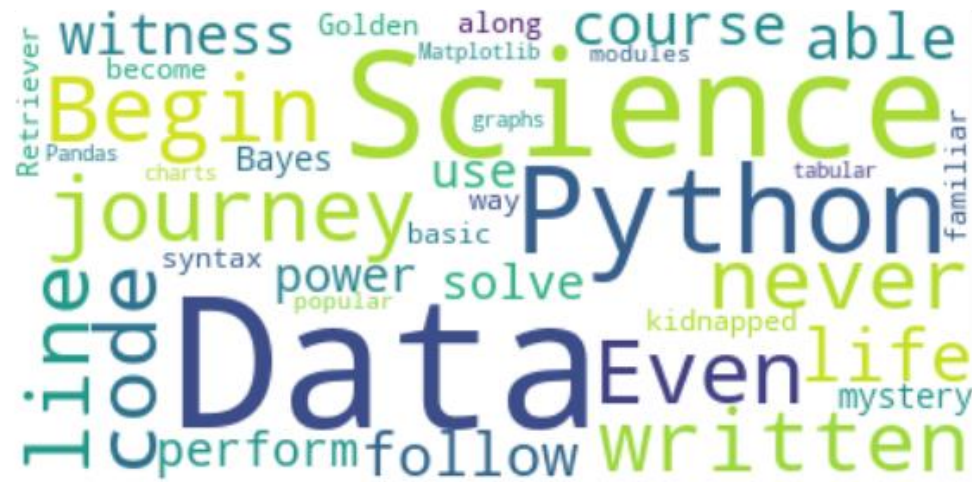
```
text = '''Begin your journey into Data Science! \
Even if you've never written a line of code in your life, \
you'll be able to follow this course and witness the power \
of Python to perform Data Science. You'll use data to solve the mystery of Bayes, \
the kidnapped Golden Retriever, \
and along the way you'll become familiar with basic Python syntax \
and popular Data Science modules like Matplotlib (for charts and graphs) \
and Pandas (for tabular data).'''
```

```
plt.show()
```



```
from wordcloud import WordCloud, STOPWORDS
```

```
plt.figure(figsize=(10, 12))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```



# Matplotlib

- Loại bỏ một số word không quan trọng

```
list_of_words = ['solve', 'Retriever', 'Begin', 'along', 'able', 'use', 'kidnapped', 'never', 'Even']  
for word in list_of_words:  
    stopwords.add(word) # add the less important word to stopwords  
# re-generate the word cloud  
wc.generate(text)  
# display the cloud  
plt.figure(figsize=(10, 12))  
plt.imshow(wc, interpolation='bilinear')  
plt.axis('off')  
plt.show()
```



# Matplotlib

- Tạo mask cho word clouds

- Tạo mask

```
import numpy as np  
from PIL import Image
```

```
# save mask to wc_mask  
wc_mask = np.array(Image.open('heart.png'))
```

```
plt.imshow(wc_mask, interpolation='bilinear')  
plt.axis('off')  
plt.show()
```



# Matplotlib

- Tạo mask cho word clouds
  - Đưa mask vào word clouds

```
# instantiate a word cloud object
wc1 = WordCloud(background_color='white', max_words=1000, mask=wc_mask, stopwords=stopwords)
# generate the word cloud
wc1.generate(text)
# display the word cloud
plt.figure(figsize=(10, 12))
plt.imshow(wc1, interpolation='bilinear')
plt.axis('off')
plt.show()
```



# Matplotlib

---

## □ Thiết lập font chữ

```
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Tahoma', 'DejaVu
Sans', 'Lucida Grande', 'Verdana']
```

# Matplotlib

## ❑ Tạo chữ watermark

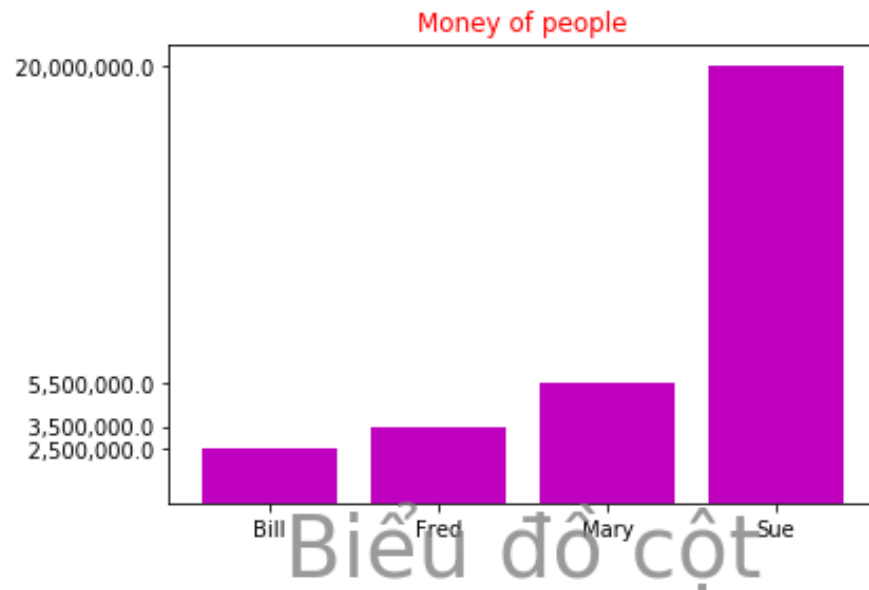
```
import matplotlib.pyplot as plt

import pandas as pd
name = ['Bill', 'Fred', 'Mary', 'Sue']
money = pd.Series([2.5e6, 3.5e6, 5.5e6, 2.0e7])

plt.bar(name, money, color="m")
plt.xticks(name)
plt.yticks(money, money.map(lambda x: '{:15,.1f}'.format(x)))
plt.title("Money of people", color="red")

# Làm chữ mờ (Watermark)
# position top center
plt.text(1.5, 0.05, 'Biểu đồ cột',
        fontsize=40, color='gray',
        ha='center', va='top', alpha=0.8)

plt.show()
```





# Matplotlib

---

## □ Tùy chỉnh bố cục sử dụng

- Dùng để thực hiện nhiều biểu đồ cùng một lượt
- Sử dụng: `subplots()` và `GridSpec()`

# Matplotlib

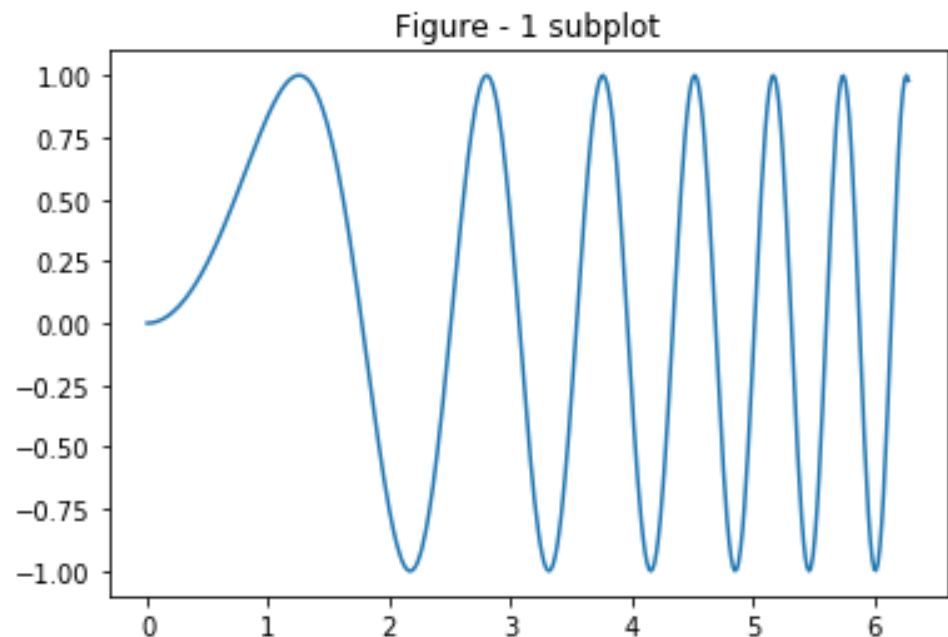
- Cách tạo sử dụng subplots()

- Tạo một figure và một tập hợp các subplot.

- Ví dụ: 1 figure có 1 subplot

```
x = np.linspace(0, 2*np.pi, 400)  
y = np.sin(x**2)
```

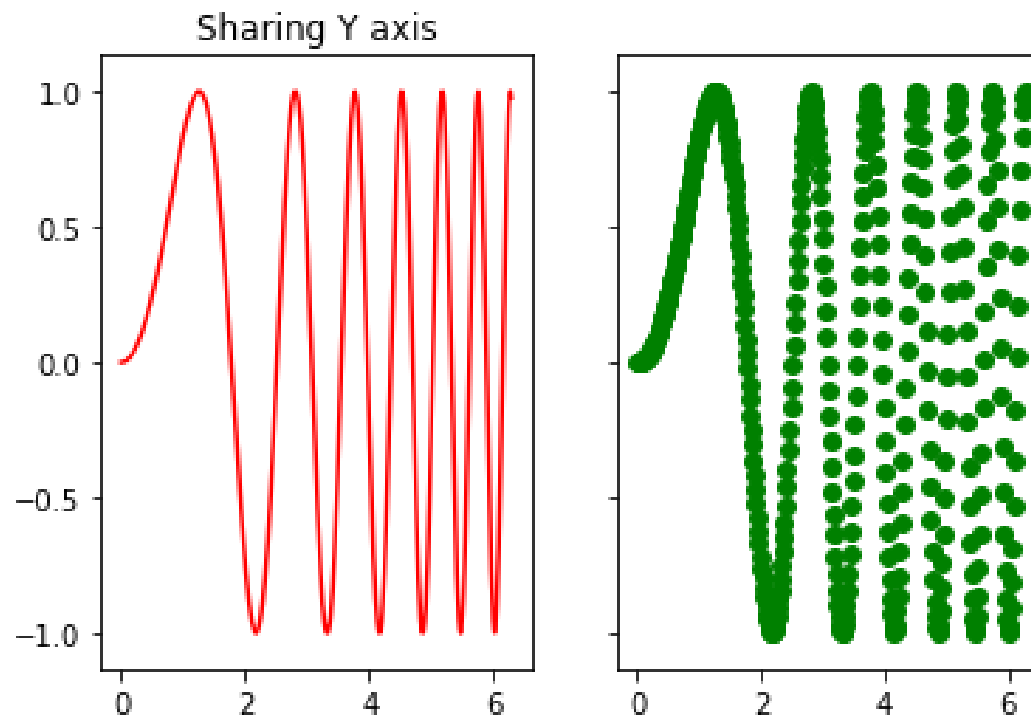
```
# tạo 1 figure có 1 subplot  
fig, ax = plt.subplots()  
ax.plot(x, y)  
ax.set_title('Figure - 1 subplot')  
plt.show()
```



# Matplotlib

## • Ví dụ: 1 figure có 2 subplot

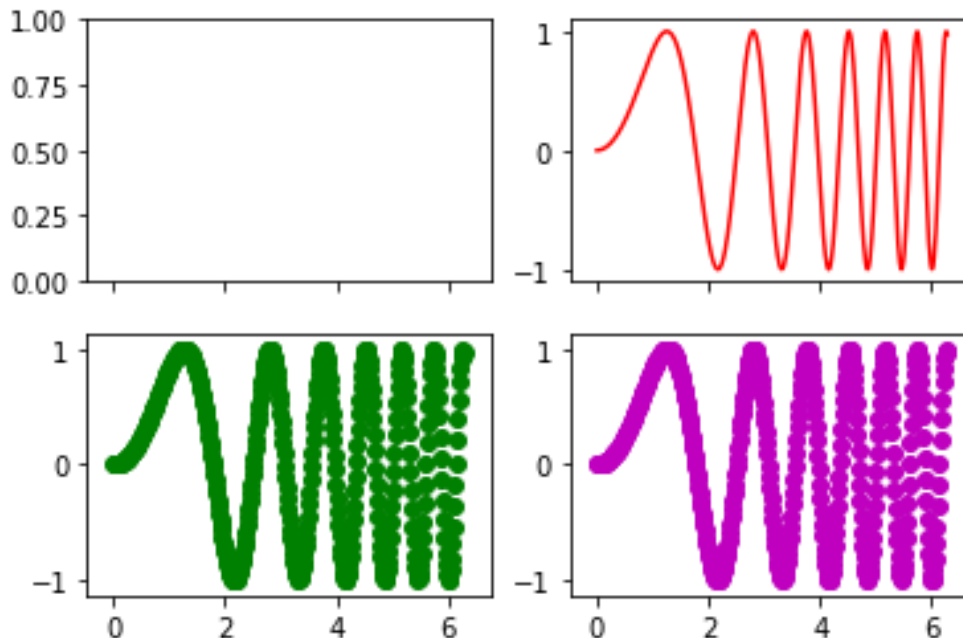
```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
ax1.plot(x, y, color='r')
ax1.set_title('Sharing Y axis')
ax2.scatter(x, y, color='g')
plt.show()
```



# Matplotlib

## • Ví dụ: 1 figure có 4 subplot

```
fig, axes = plt.subplots(2, 2, sharex=True)
axes[0, 1].plot(x, y, color='r')
axes[1, 0].scatter(x, y, color='g')
axes[1, 1].scatter(x, y, color='m')
plt.show()
```



# Matplotlib

---

- **Cách tạo sử dụng GridSpec()**
  - Ta phải tạo một figure và một thực thể GridSpec riêng lẻ, sau đó đưa các element của thực thể gridspec vào `add_subplot()` để tạo axes object. Các element của gridspec được truy cập như truy cập numpy array.

# Matplotlib

## • Ví dụ: Tạo lưới 4x4

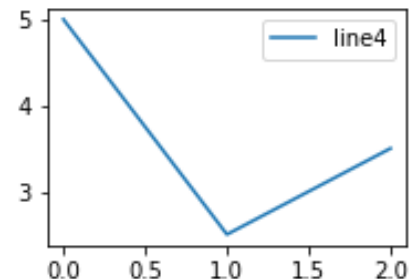
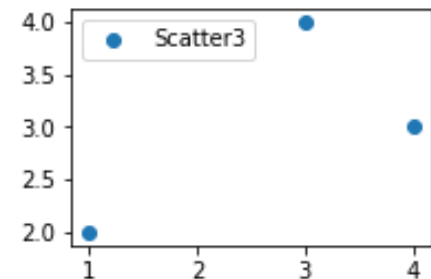
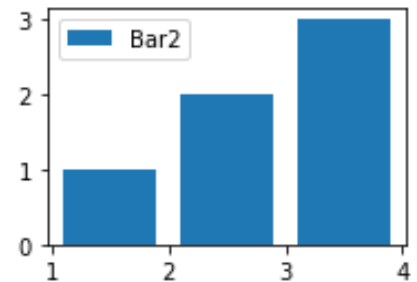
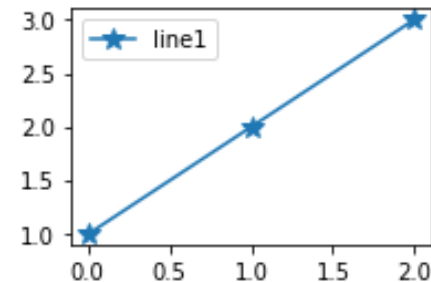
```
fig2 = plt.figure()

spec2 = gridspec.GridSpec(ncols=2, nrows=2)
f2_ax1 = fig2.add_subplot(spec2[0, 0])
f2_ax2 = fig2.add_subplot(spec2[0, 1])
f2_ax3 = fig2.add_subplot(spec2[1, 0])
f2_ax4 = fig2.add_subplot(spec2[1, 1])
fig2.tight_layout()

f2_ax1.plot([1, 2, 3], label='line1', marker='*', markersize=10)
f2_ax2.bar([1.5, 2.5, 3.5], [1, 2, 3], label='Bar2')
f2_ax3.scatter([1, 4, 3], [2, 3, 4], label='Scatter3')
f2_ax4.plot([5, 2.5, 3.5], label='line4')

f2_ax1.legend()
f2_ax2.legend()
f2_ax3.legend()
f2_ax4.legend()

plt.show()
```



# Matplotlib

- Ví dụ: Tạo lưới 2 dòng, dòng 1 ghép cột có 1 subplot, dòng 2 có 2 subplot

```
fig = plt.figure(tight_layout=False)
gs = gridspec.GridSpec(2, 2)

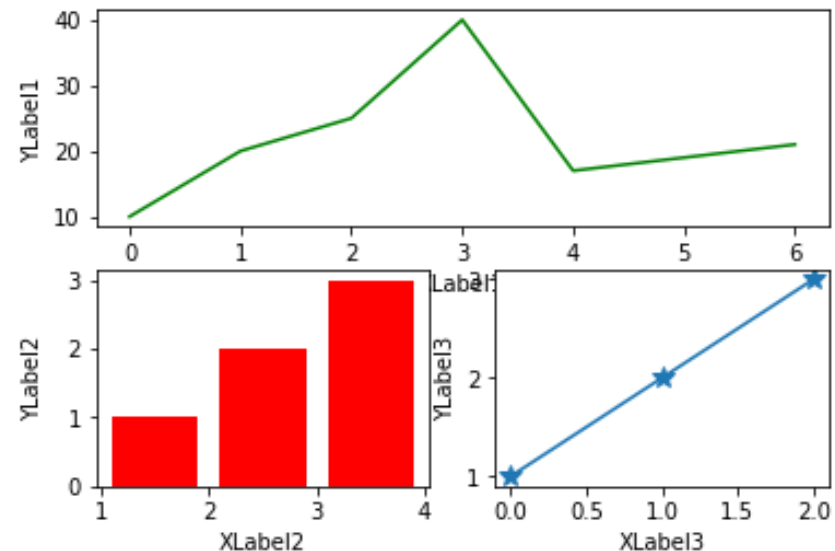
ax = fig.add_subplot(gs[0, :])
ax.plot([10, 20, 25, 40, 17, 19, 21], color='g')
ax.set_ylabel('YLabel1')
ax.set_xlabel('XLabel1')

ax1 = fig.add_subplot(gs[1, 0])
ax1.bar([1.5, 2.5, 3.5], [1, 2, 3], color='r', label='Bar2')
ax1.set_ylabel('YLabel2')
ax1.set_xlabel('XLabel2')

ax2 = fig.add_subplot(gs[1, 1])
ax2.plot([1, 2, 3], label='line1', marker='*', markersize=10)
ax2.set_ylabel('YLabel3')
ax2.set_xlabel('XLabel3')

fig.align_labels()

plt.show()
```



# Nội dung

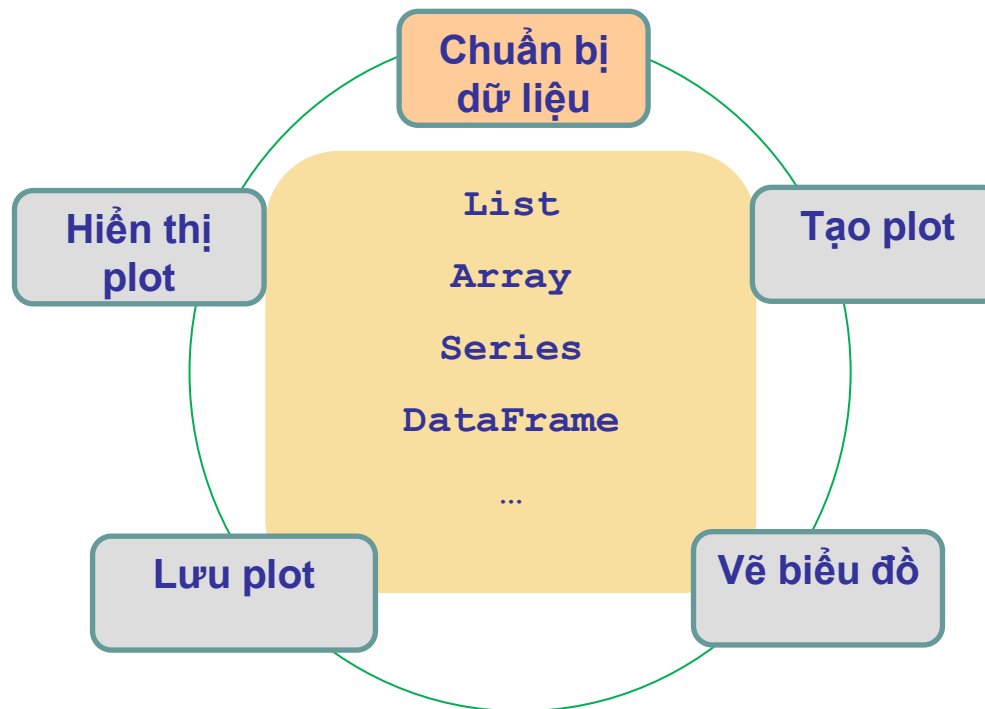
---

1. Vai trò của trực quan hóa dữ liệu
2. Matplotlib
3. Quy trình tạo biểu đồ



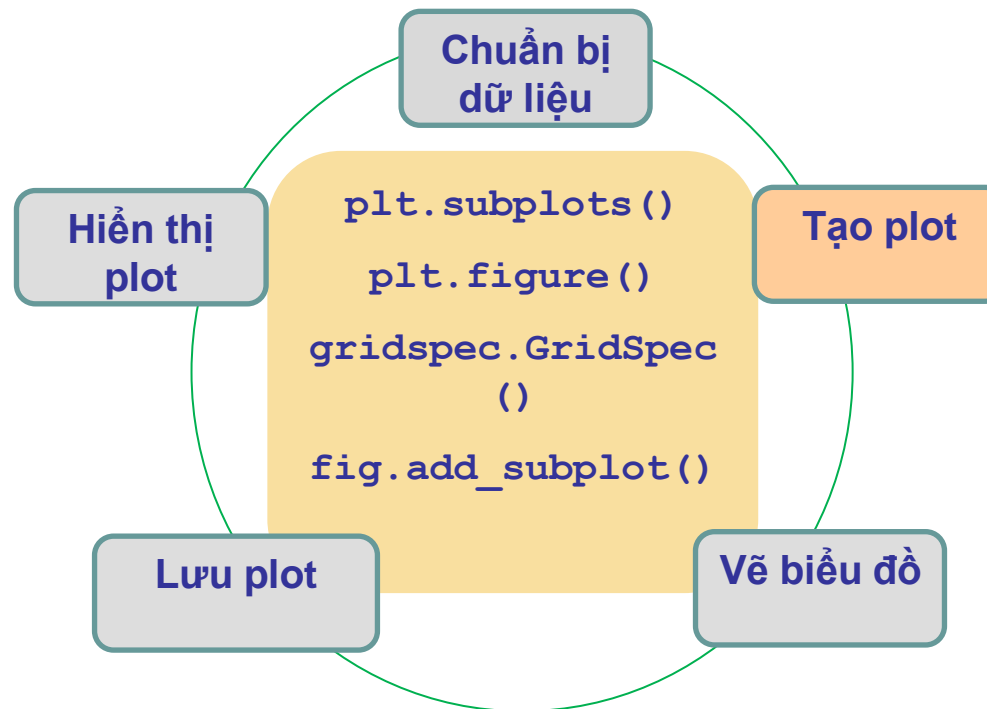
# Tổng kết

## □ Các bước tạo biểu đồ



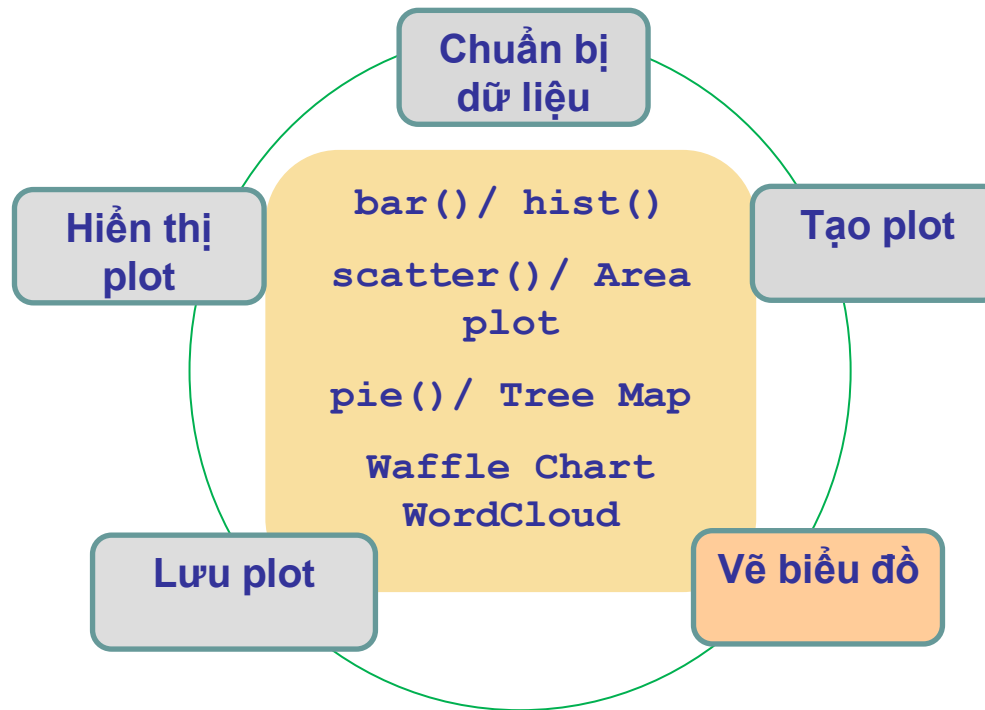
# Tổng kết

## □ Các bước tạo biểu đồ



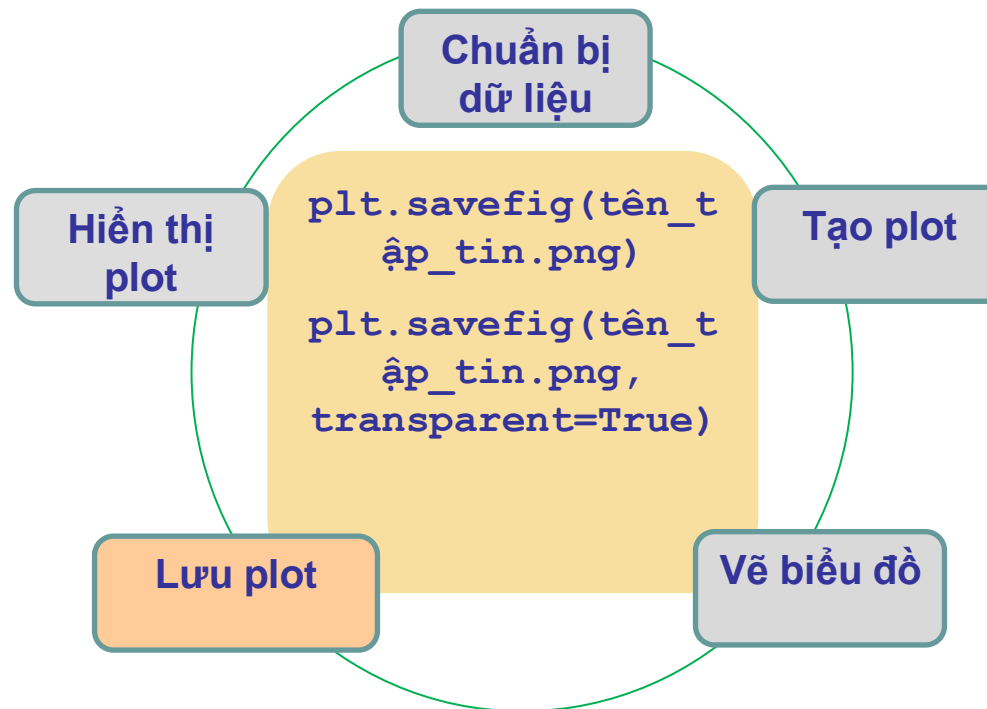
# Tổng kết

## □ Các bước tạo biểu đồ



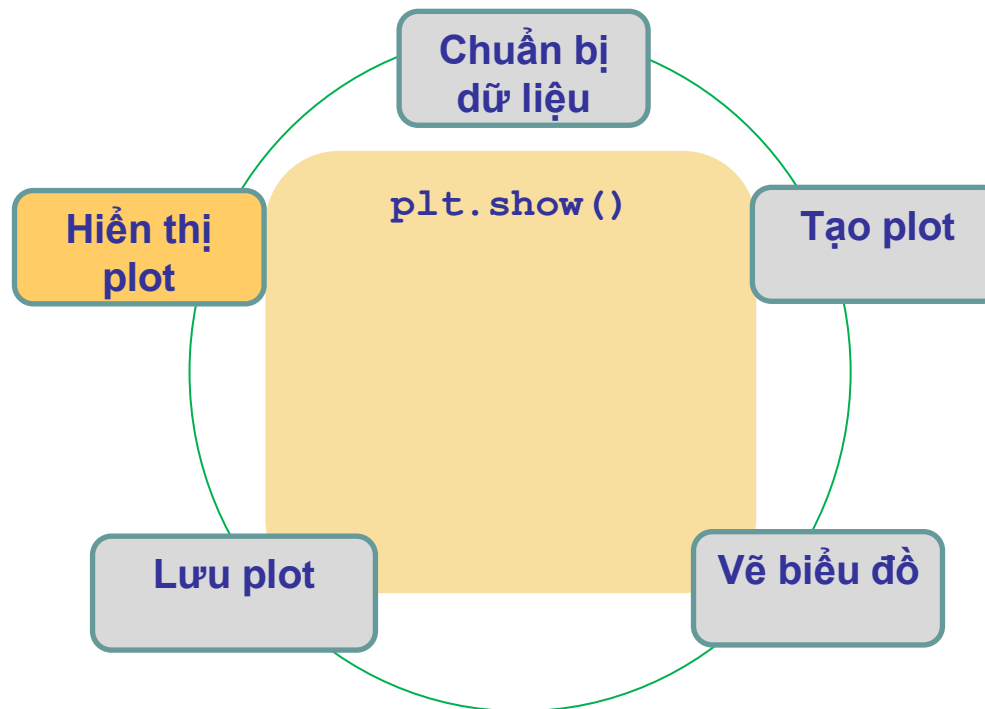
# Tổng kết

## □ Các bước tạo biểu đồ



# Tổng kết

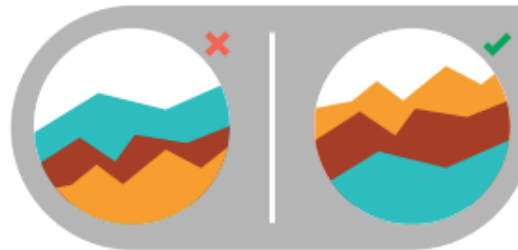
## □ Các bước tạo biểu đồ





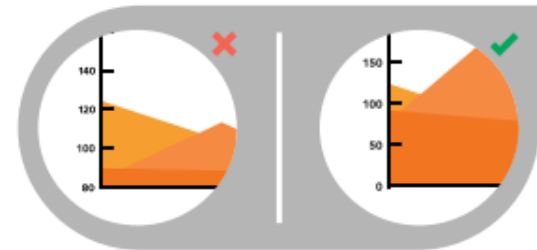
# AREA CHART

## DESIGN BEST PRACTICES



### MAKE IT EASY TO READ

In stacked area charts, arrange data to position categories with highly variable data on the top of the chart and low variability on the bottom.



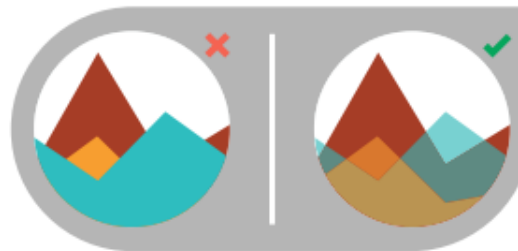
### START Y-AXIS VALUE AT 0

Starting the axis above zero truncates the visualization of values.



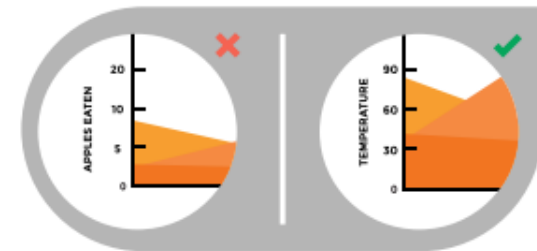
### DON'T DISPLAY MORE THAN 4 DATA CATEGORIES

Too many will result in a cluttered visual that is difficult to decipher.



### USE TRANSPARENT COLORS

In standard area charts, ensure data isn't obscured in the background by ordering thoughtfully and using transparency.

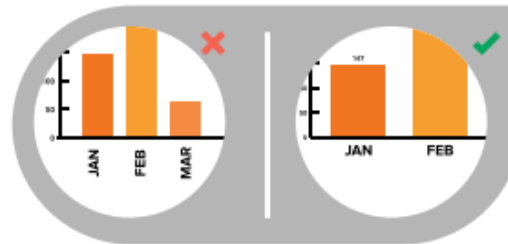


### DON'T USE AREA CHARTS TO DISPLAY DISCRETE DATA

The connected lines imply intermediate values, which only exist with continuous data.

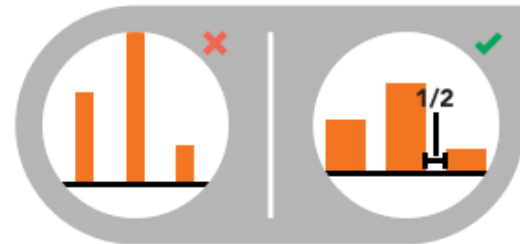
# BAR CHART

## DESIGN BEST PRACTICES



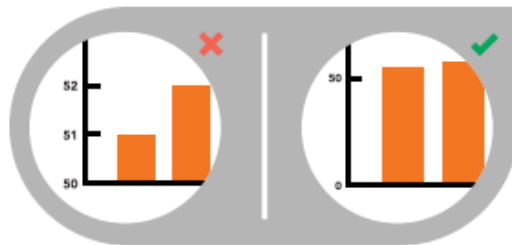
### USE HORIZONTAL LABELS

Avoid steep diagonal or vertical type, as it can be difficult to read.



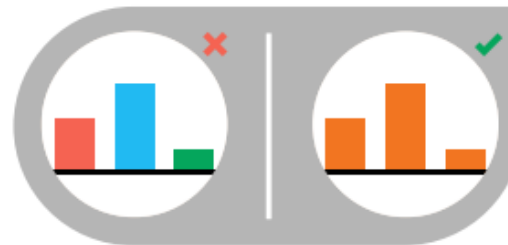
### SPACE BARS APPROPRIATELY

Space between bars should be  $\frac{1}{2}$  bar width.



### START THE Y-AXIS VALUE AT 0

Starting at a value above zero truncates the bars and doesn't accurately reflect the full value.



### USE CONSISTENT COLORS

Use one color for bar charts. You may use an accent color to highlight a significant data point.

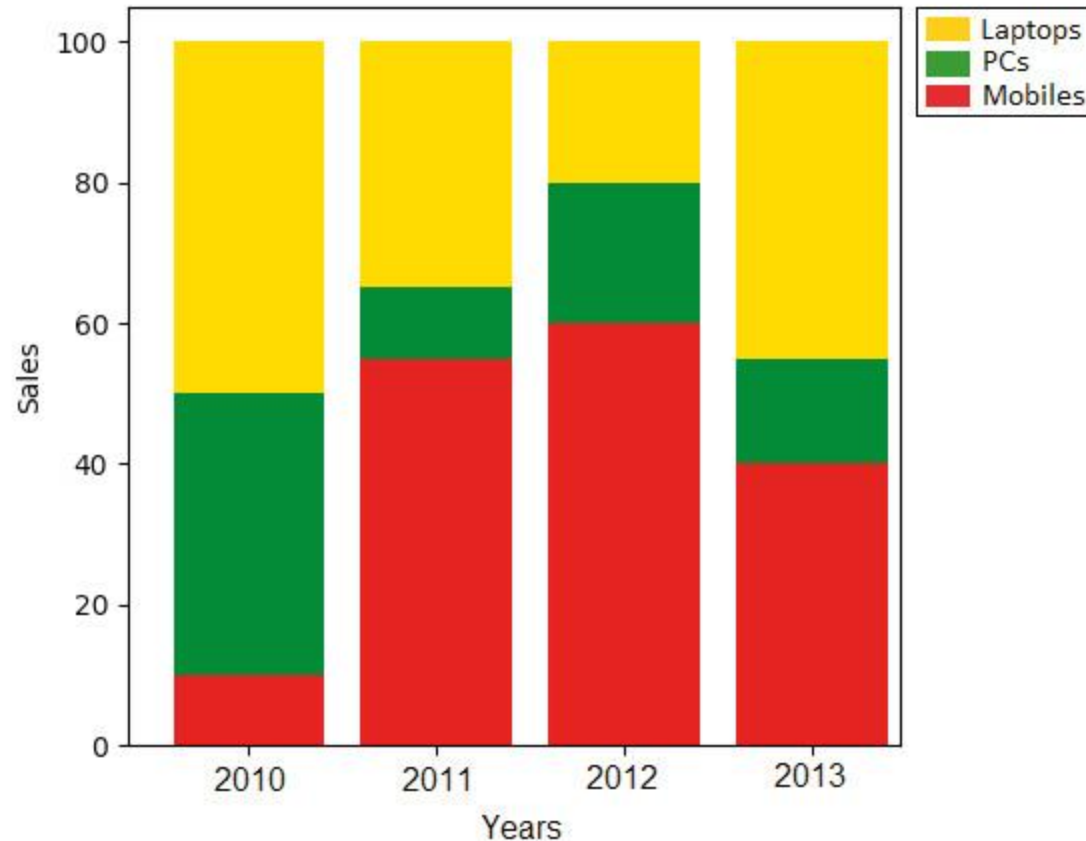


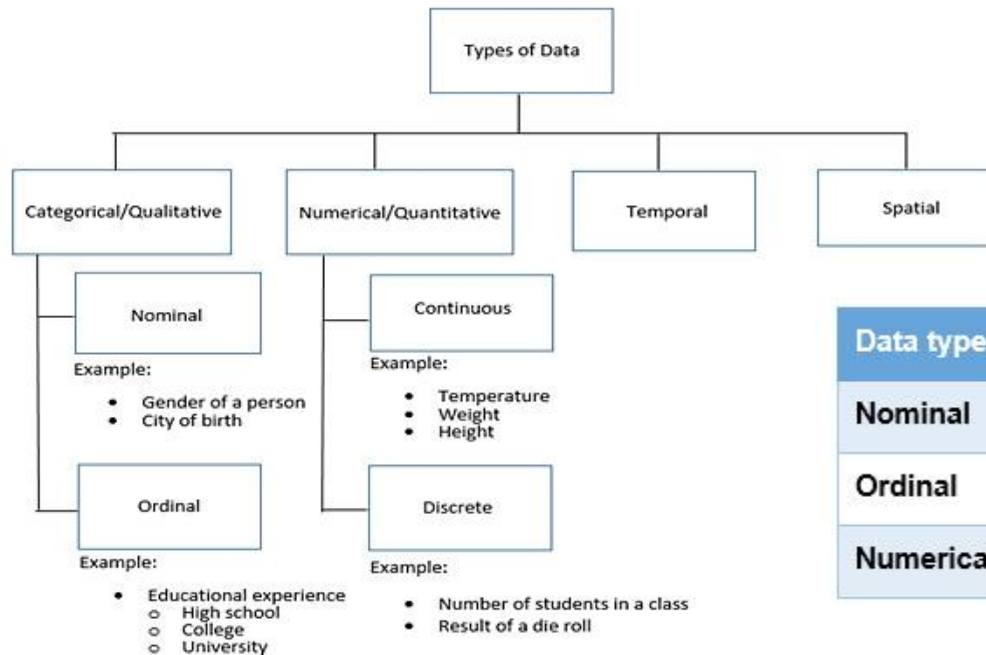
### ORDER DATA APPROPRIATELY

Order categories alphabetically, sequentially, or by value.



The following diagram shows a 100% stacked bar chart with the same data that was used in the preceding diagr

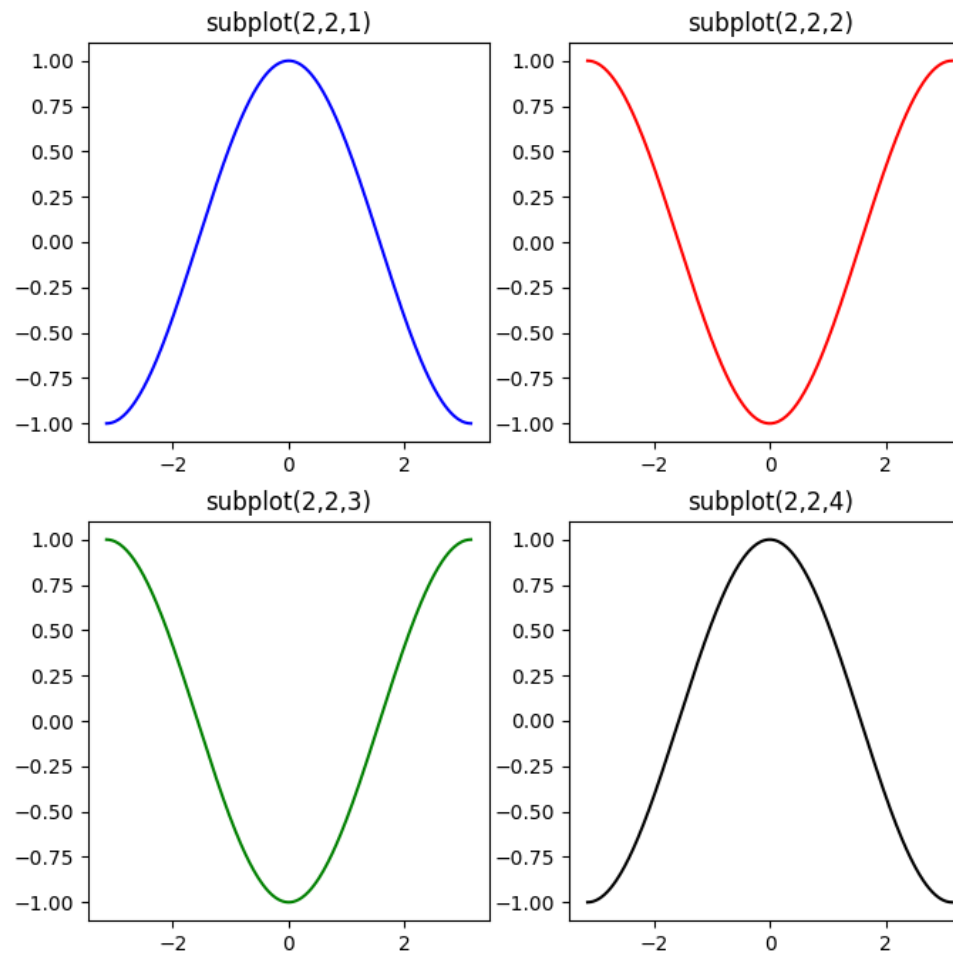




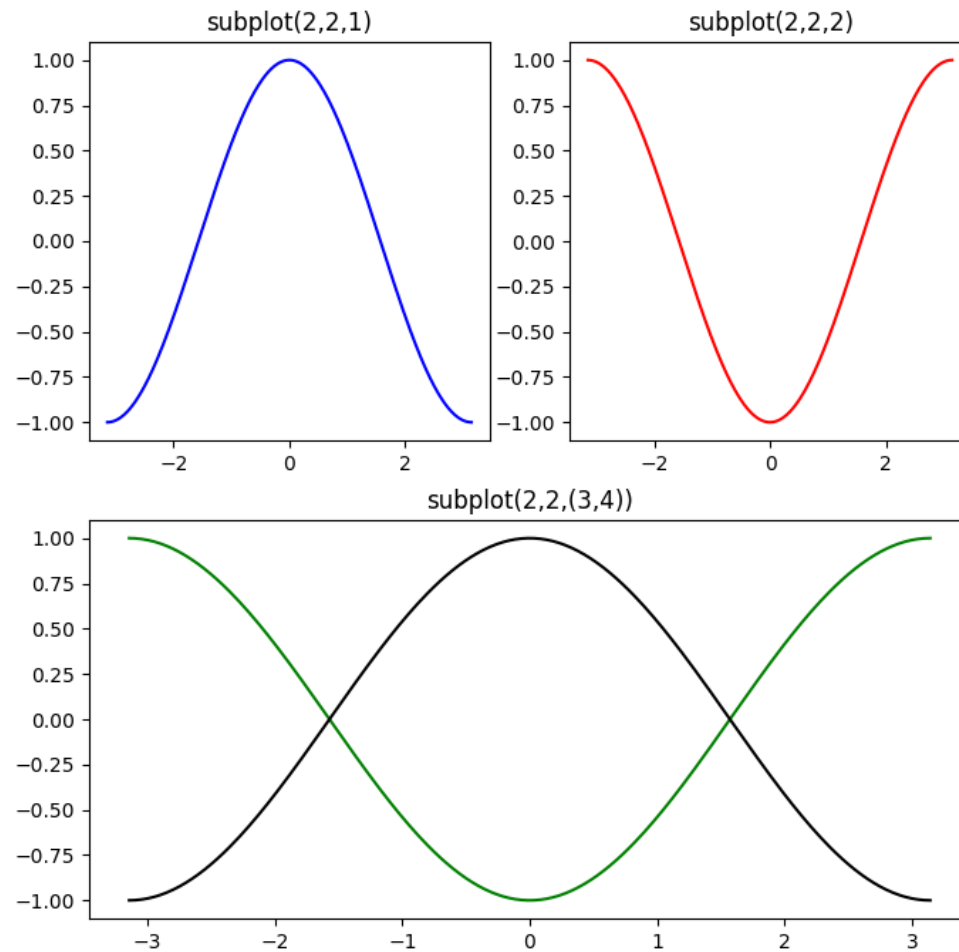
Data type	Ideal measure of central tendency
Nominal	Mode
Ordinal	Median
Numerical	Mean/Median

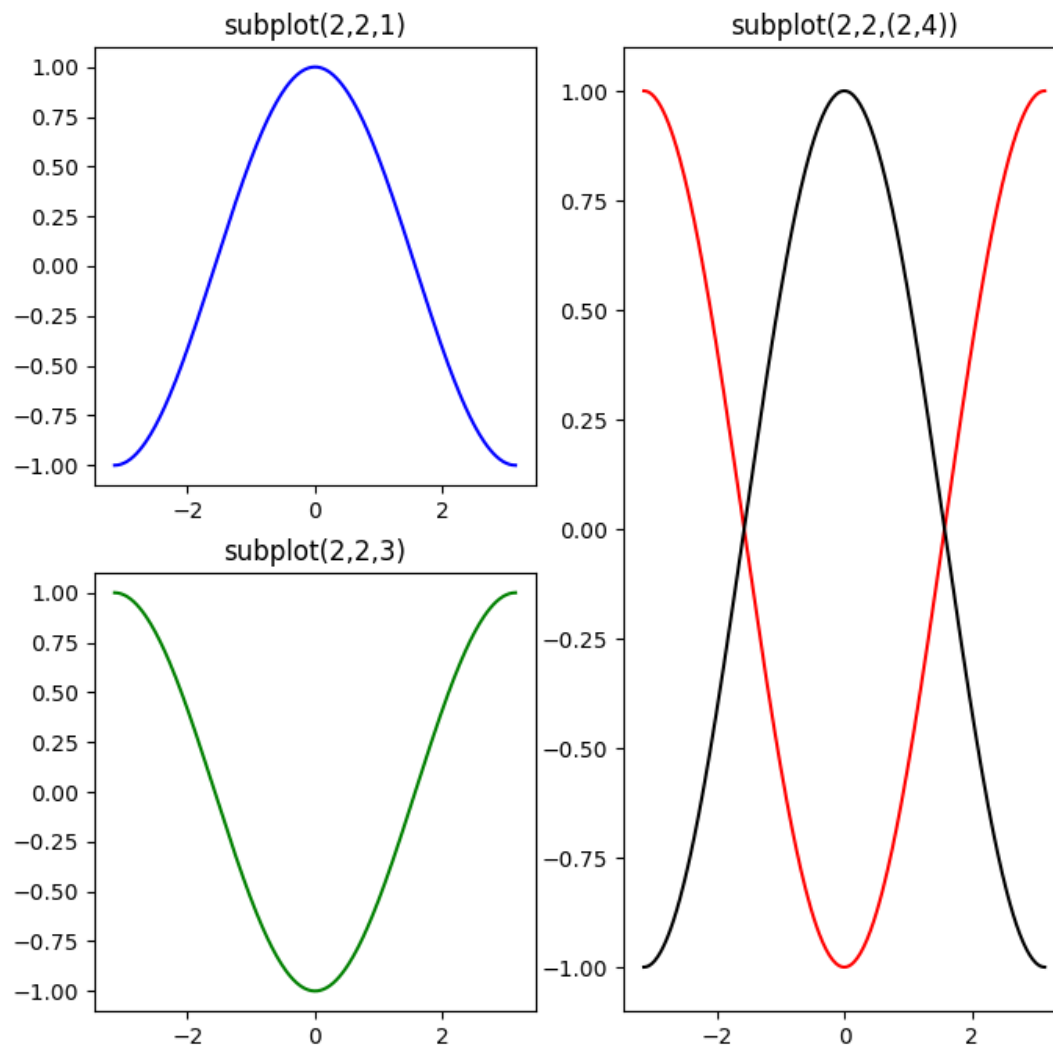
Loại dữ liệu

**Figure 1.6: Best-suited measures of central tendency for different types of data**



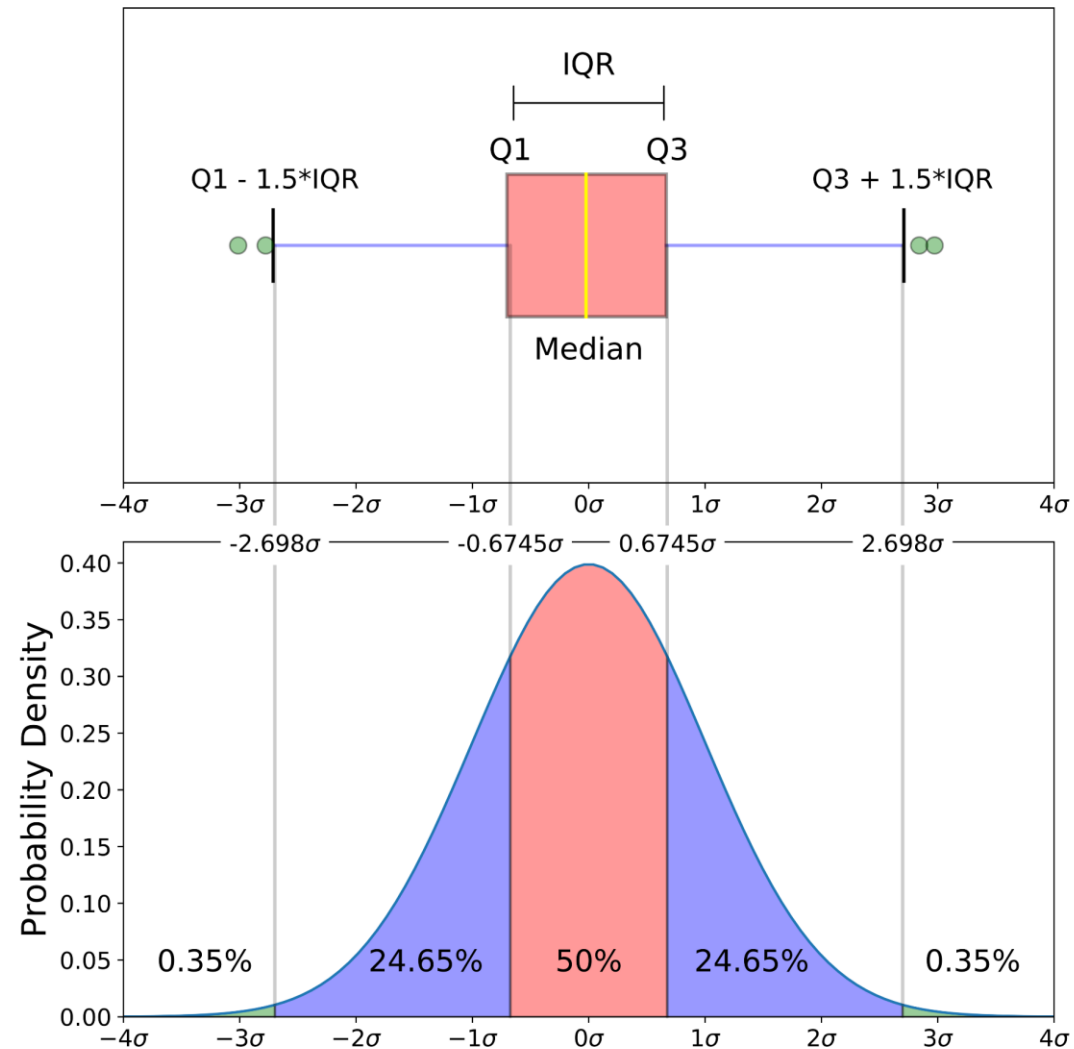
```
plt.subplot(rows, columns, subplot_id)
```



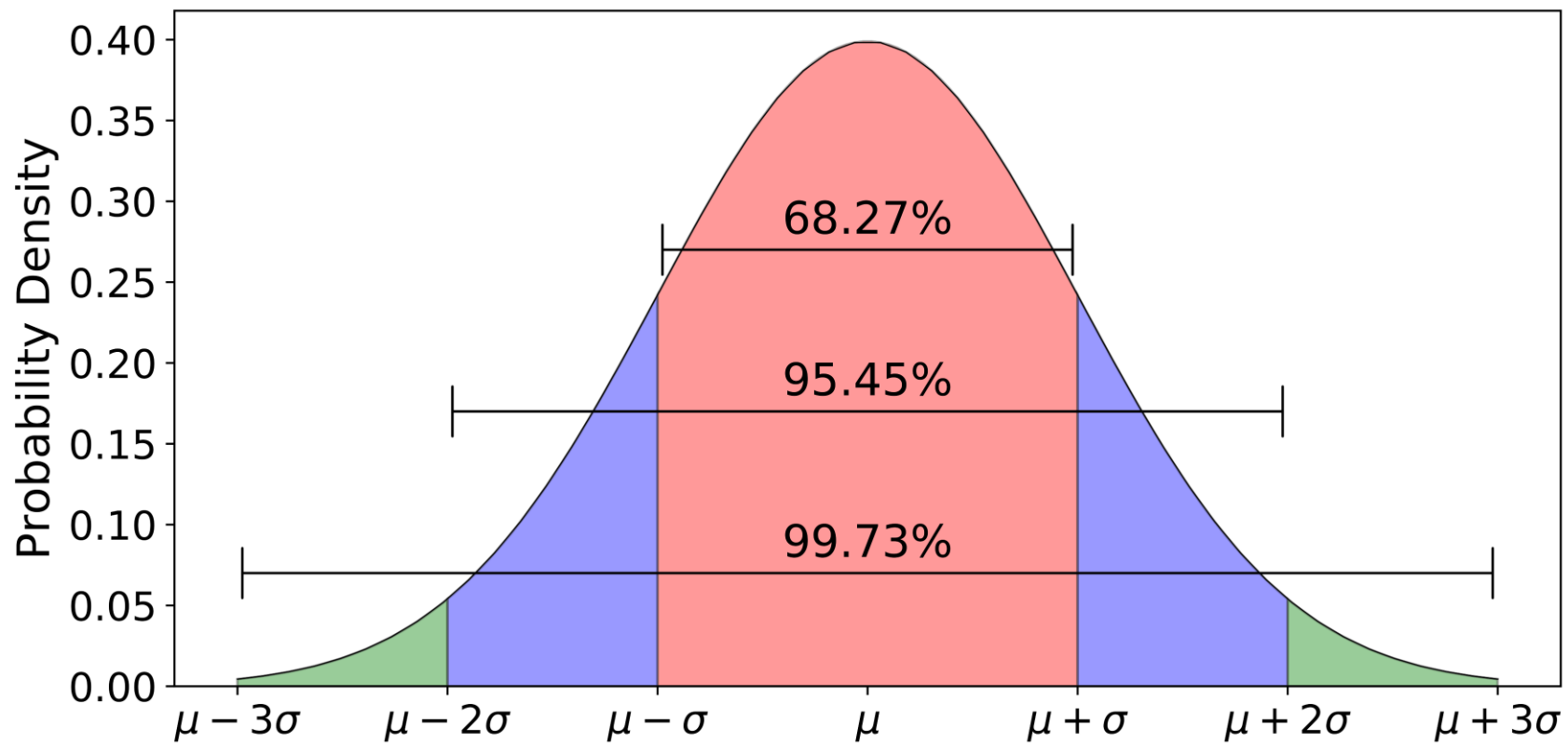


# Matplotlib

## Boxplot:



## 68-95-99.7 Rule



# Ôn lại thống kê

## ● Ví dụ:

Bạn muốn tìm một căn hộ khá khá để thuê mà giá không quá cao so với các căn hộ khác. Giả sử bạn tìm thấy các căn hộ trên các website cho thuê có giá như sau: \$ 700, \$ 850, \$ 1.500 và \$ 750 mỗi tháng

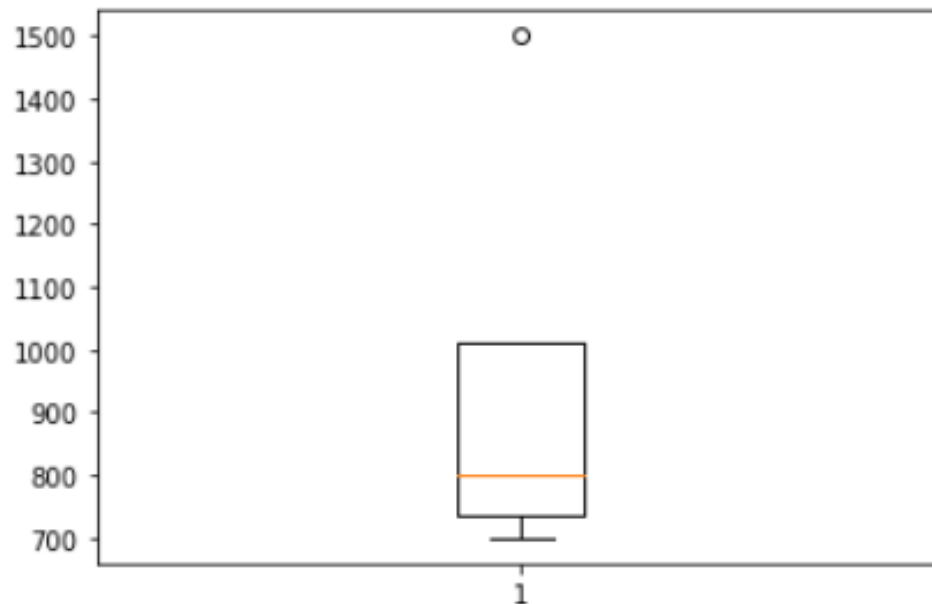
- Giá thuê trung bình:  $\frac{\$700 + \$850 + \$1500 + \$750}{4} = \$950$
- Giá thuê trung bình giữa (trung vị)  $\frac{\$750 + \$850}{2} = \$800.$
- Độ lệch chuẩn  $\sqrt{\frac{(\$700 - \$950)^2 + (\$850 - \$950)^2 + (\$1500 - \$950)^2 + (\$750 - \$950)^2}{4}} = \$322.10$
- Khoảng giá thuê  $\$1500 - \$700 = \$800$



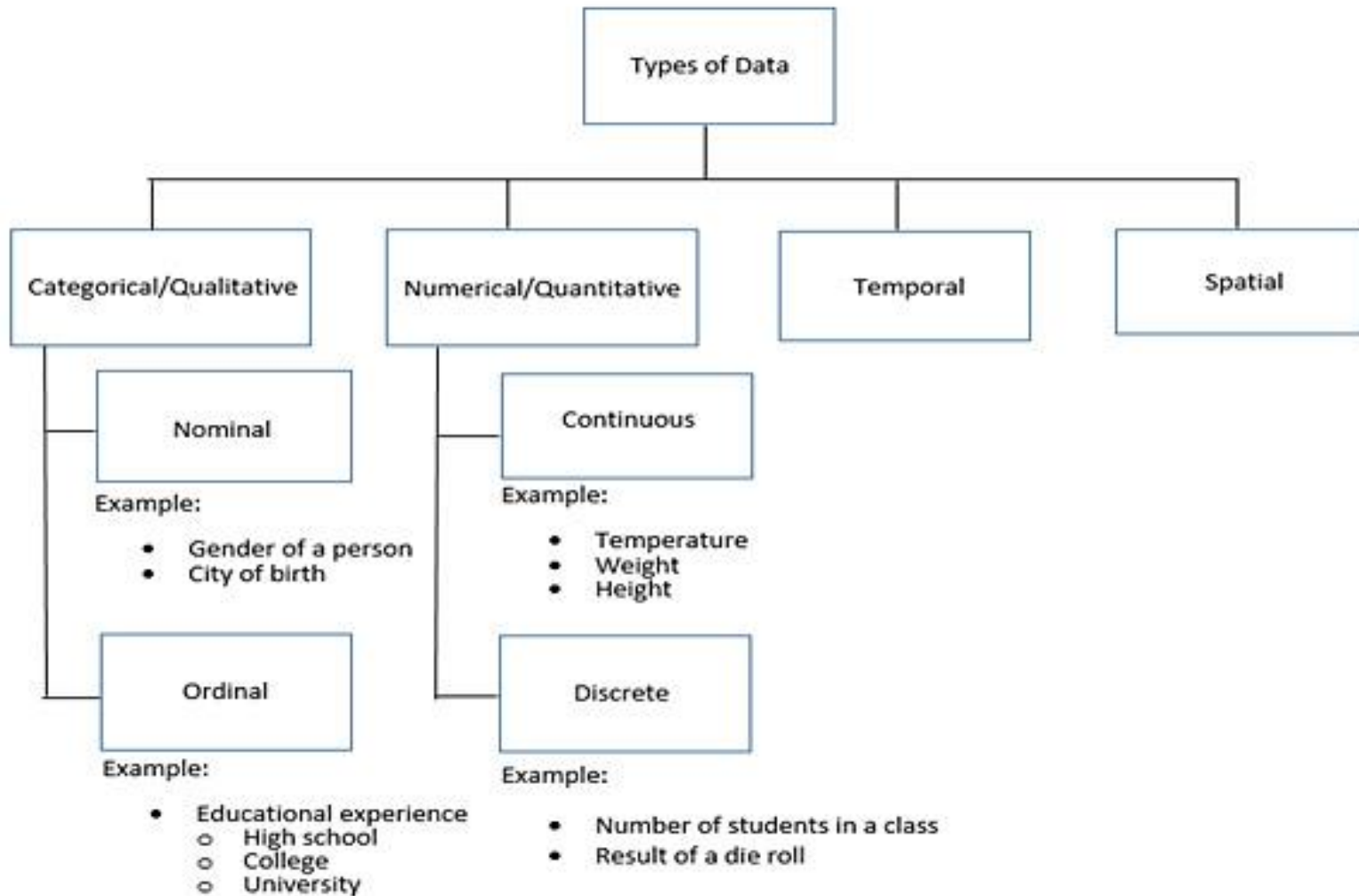
# Ôn lại thống kê

- Ví dụ:

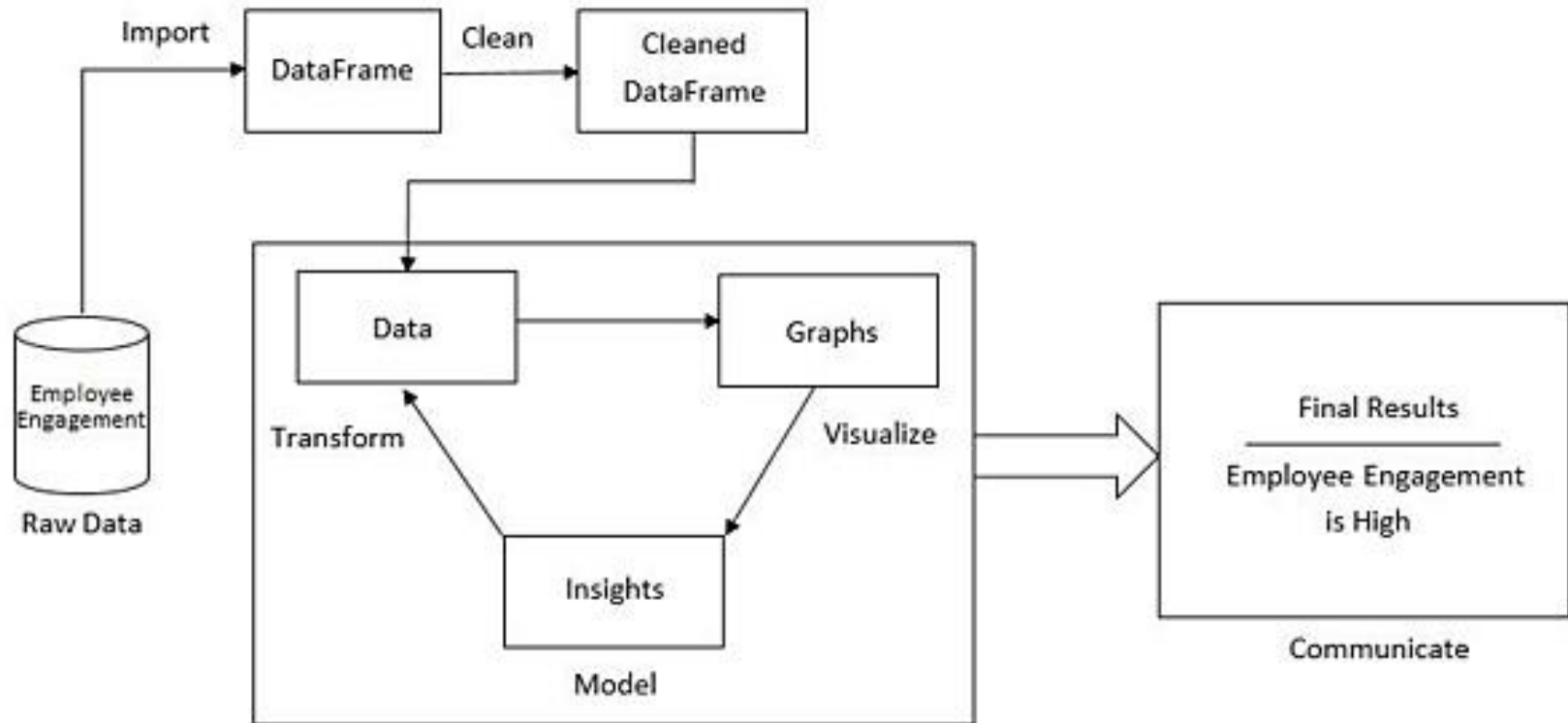
Bạn muốn tìm một căn hộ khá khá để thuê mà giá không quá cao so với các căn hộ khác. Giả sử bạn tìm thấy các căn hộ trên các website cho thuê có giá như sau: \$ 700, \$ 850, \$ 1.500 và \$ 750 mỗi tháng



# Loại dữ liệu



# Vai trò của trực quan hóa dữ liệu

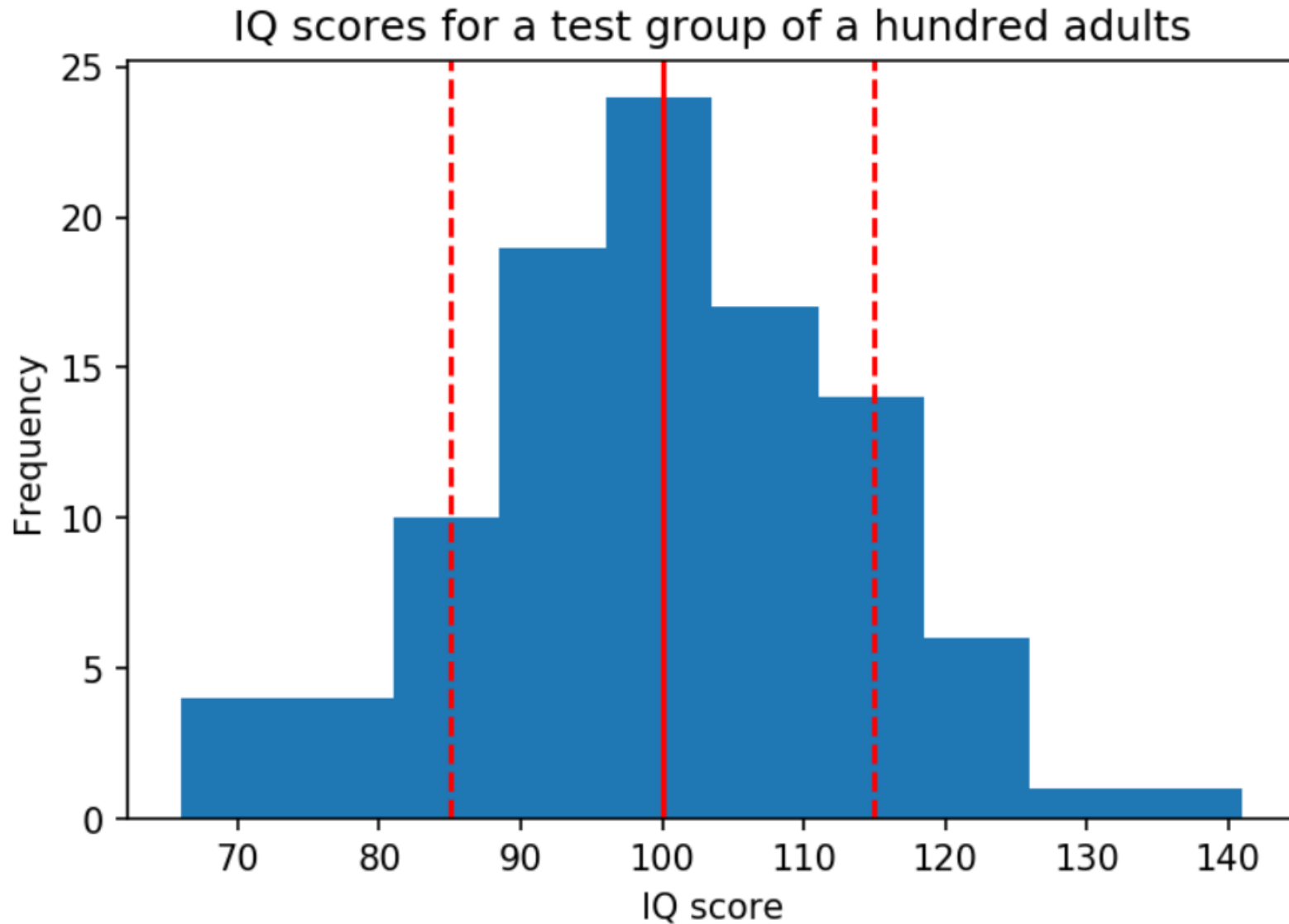


# Phân loại biểu đồ

---

- ☐ Comparison plots
- ☐ Relation plots
- ☐ Composition plots
- ☐ Geo plots

# Matplotlib



# Các thành phần trong biểu đồ

