Bài 5
Data Visualization

Khóa học: Phân tích dữ liệu với Python

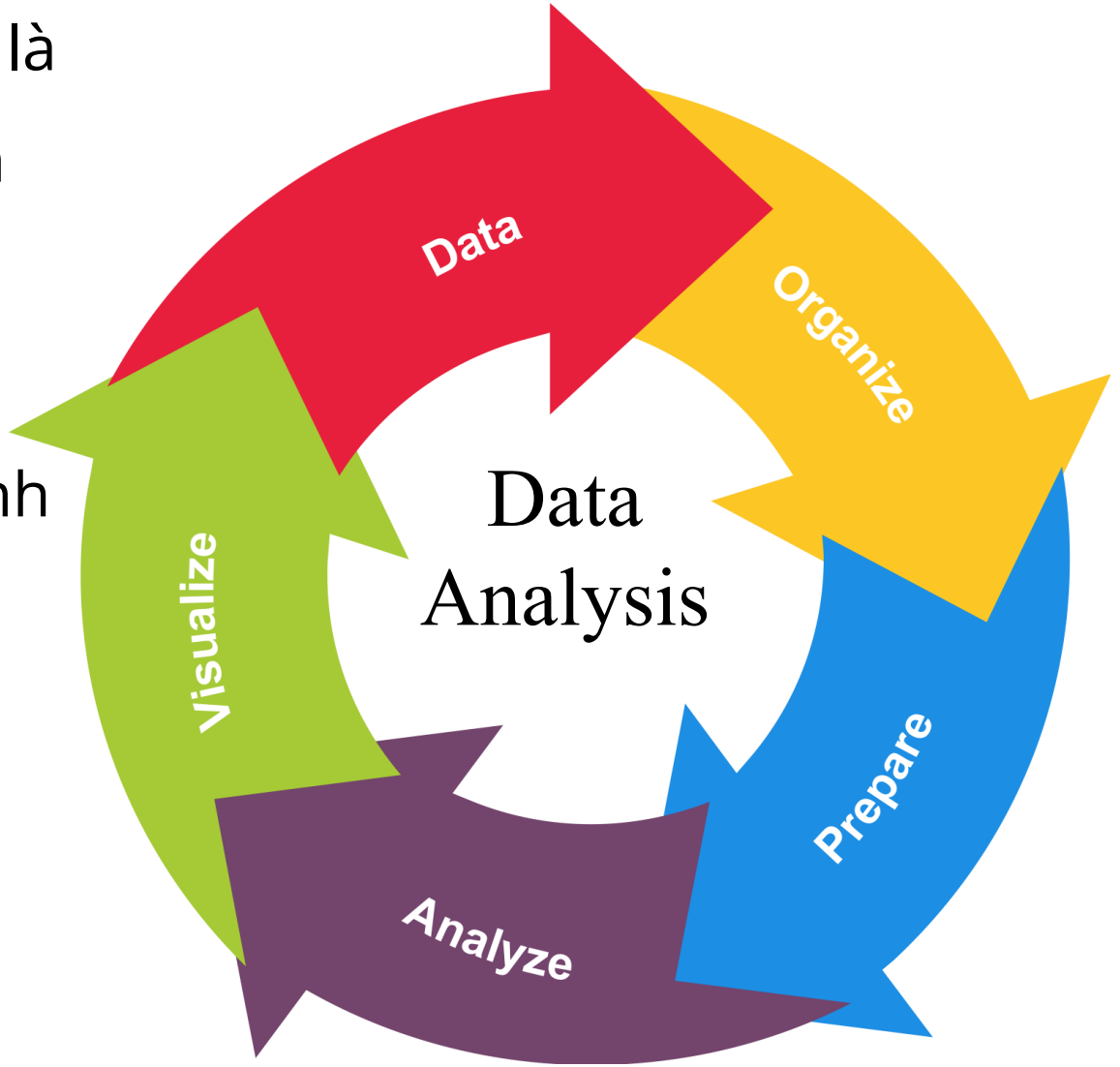
Nội dung

1. Giới thiệu
   - Khái niệm Data Visualization
   - Thư viện Matplotlib
2. Biểu đồ mô tả một thuộc tính
   - Biểu đồ Histogram
   - Biểu đồ Cột
   - Biểu đồ Tròn
3. Biểu đồ mô tả mối liên hệ giữa các thuộc tính
   - Biểu đồ Đường
   - Biểu đồ Scatter

1. Giới thiệu

Data Visualization (trực quan hóa dữ liệu) là một phần không thể thiếu trong quá trình phân tích dữ liệu:

- Có nguồn gốc từ Thống kê mô tả.
- Hiển thị dữ liệu và các thông tin qua hình ảnh biểu đồ, đồ thị.
- Thông tin từ dữ liệu lớn, phức tạp được hiển thị đơn giản, dễ hiểu và sinh động hơn.



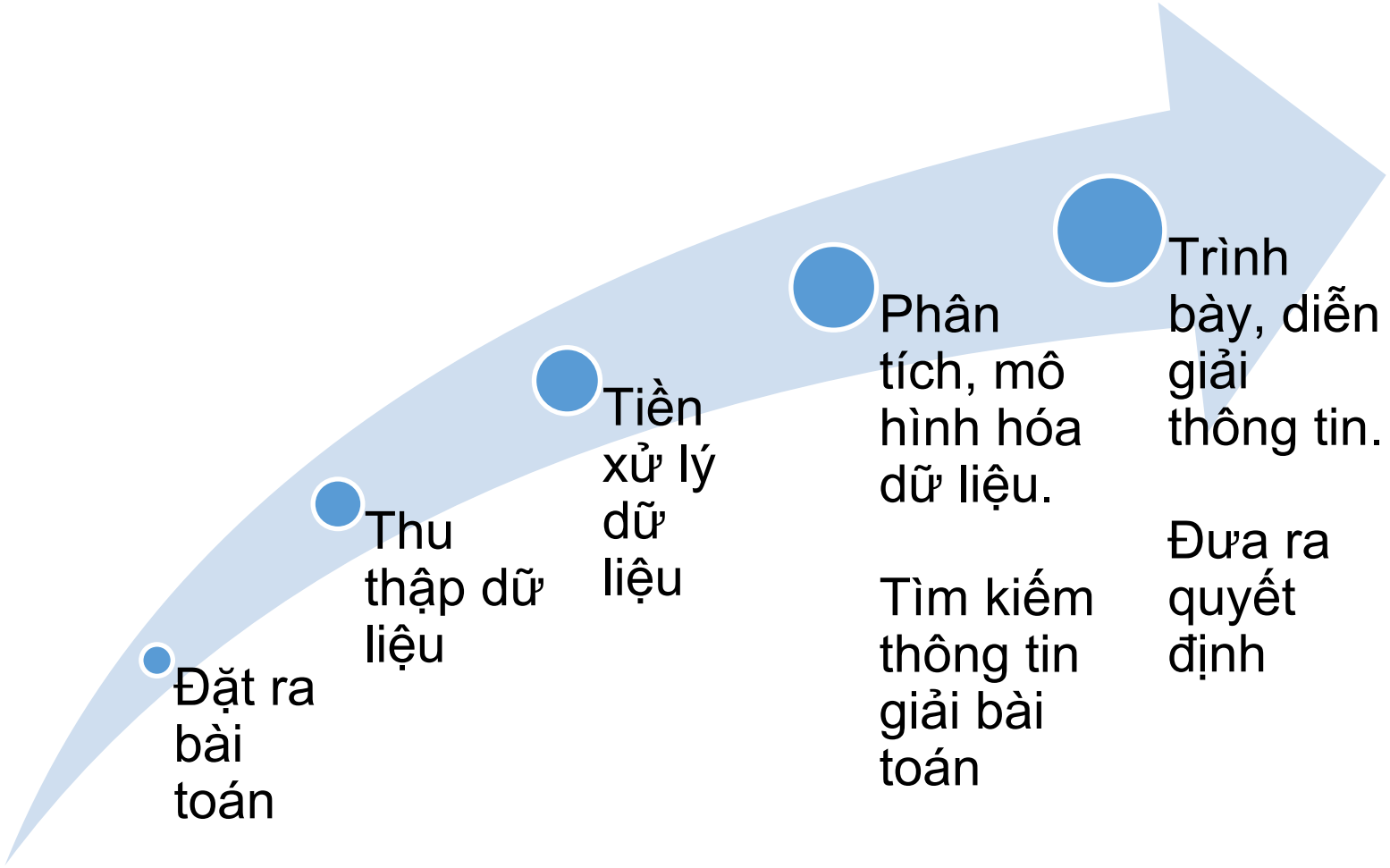
Tại sao cần Data Visualization?

- Tính khoa học
  - Con người phản ứng với hình ảnh tốt hơn rất nhiều so với các dạng trình bày khác như ký hiệu, chữ viết, con số.
  - Một biểu đồ cũng có thể biểu diễn thay thế cho rất nhiều dữ liệu một cách đơn giản, dễ hiểu, sinh động hơn so với bảng.

Tại sao cần Data Visualization?

- Ý nghĩa sử dụng
  - Xác định được bức tranh toàn cảnh.
  - Xác định xu hướng hiện tại, dự báo xu hướng tương lai.
  - Xác định mối liên hệ giữa các thuộc tính hay giữa các tập hợp dữ liệu.
  - Tăng khả năng truyền tải thông tin, đặc trưng của dữ liệu.
  - Đánh giá, so sánh thông tin dễ dàng thông qua các chỉ số được biểu diễn.
  - Hỗ trợ đưa ra các quyết định dựa trên dữ liệu.
  - Biểu diễn dữ liệu theo ý đồ của chúng ta

Giai đoạn nào cần Data Visualization?



Giai đoạn nào cần Data Visualization?

- **Giai đoạn tiền xử lý dữ liệu**:
  - Cung cấp các hiểu biết cơ bản về dữ liệu đầu vào như sự phân bố, sự khuyết thiếu, độ nhiễu và các giá trị ngoại lai.
- **Giai đoạn trích chọn đặc trưng**: kết hợp với các thuật toán ranking/score/selection để
  - Đánh giá, so sánh score cho từng feature.
  - Tìm hiểu sự phù hợp của feature.
  - Chọn lựa số lượng feature tối ưu.
- **Giai đoạn đánh giá mô hình**:
  - Thể hiện kết quả của mô hình học máy.
  - So sánh, đánh giá các giai đoạn của quá trình phân tích dữ liệu.

Các bước cơ bản để Data Visualization

- Xác định mục tiêu.
- Thu thập dữ liệu và xác định dữ liệu, thuộc tính cần thiết.
- Chọn phương pháp biểu diễn phù hợp.
- Biểu diễn

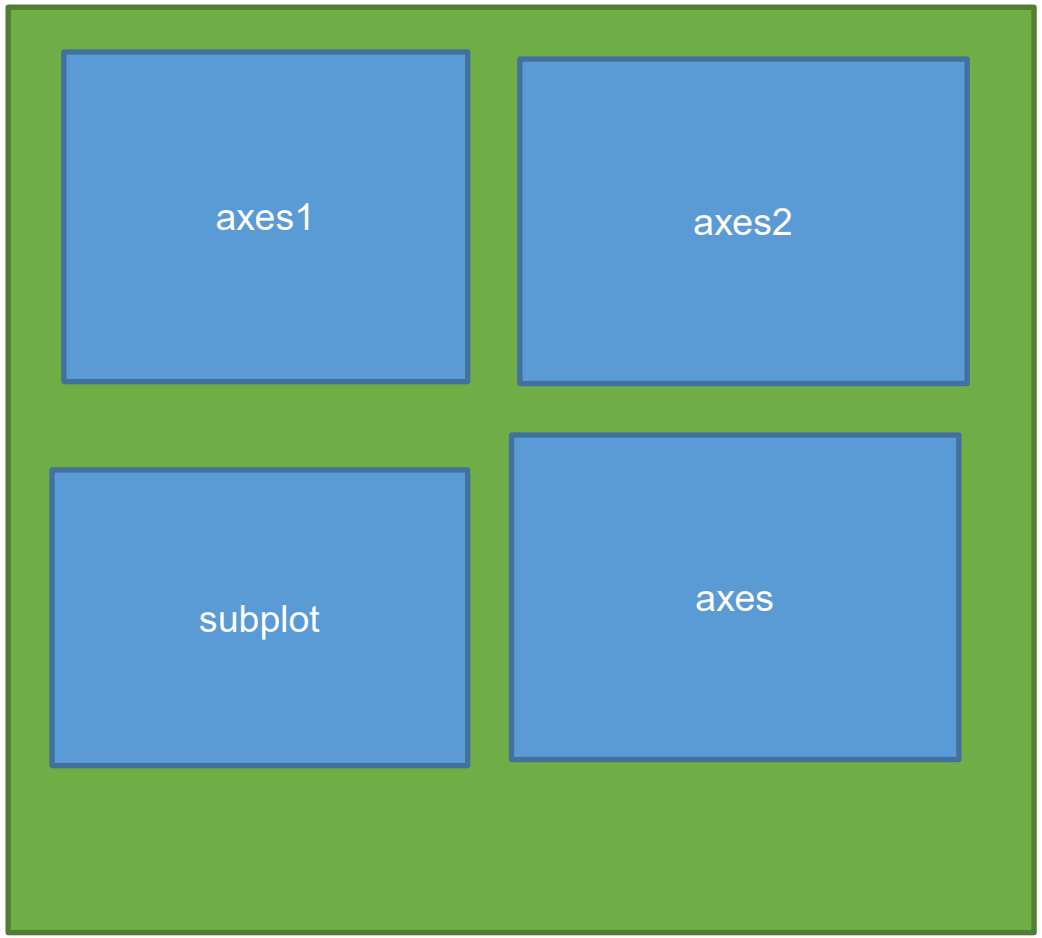
Thư viện Matplotlib

- Thư viện hỗ trợ trực quan hóa dữ liệu và đồ thị phổ biến nhất cho nhiều ngôn ngữ, trong đó có Python.
- Thư viện có nhiều tiện ích được tích hợp sẵn, linh hoạt và dễ dàng sử dụng để tạo và tinh chỉnh nhiều loại biểu đồ.

Thư viện Matplotlib

- Một Matplotlib Figure có thể được phân loại thành nhiều phần:
  - *Figure*: Cửa sổ chứa các biểu đồ.
  - *Axes*: Thành phần chính của một figure là các axes (những vùng nhỏ hơn với không gian dữ liệu).
  - *Axis*: Axes chứa 2 Axis (hoặc 3 trong trường hợp 3D). Mỗi Axis là một chiều (trục).
  - *Artist*: Mọi thứ nhìn thấy trên figure là một artist như Text objects, Line2D objects, collection objects.

Thư viện Matplotlib



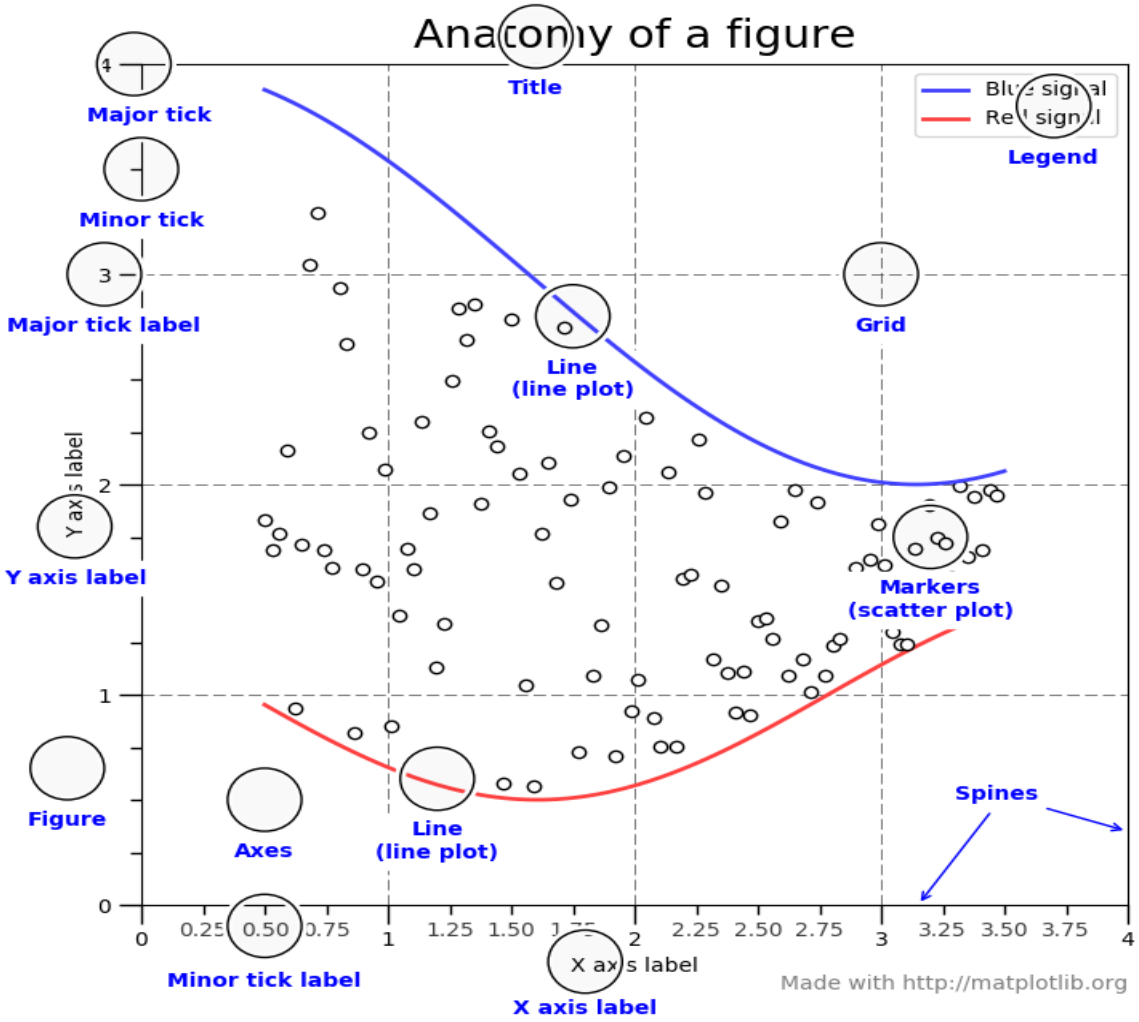


Figure 1: Đây là figure 1. Nó có nghía là bla bla bal

Thư viện Matplotlib

- Hướng dẫn cài đặt thư viện Matplotlib:
  - Điều kiện: đã cài đặt Python
  - Hệ điều hành Window: Windows + S => “cmd” => Command Prompt

  => “pip install matplotlib”
  - Hệ điều hành khác
    - Debian / Ubuntu: sudo apt-get install python3-matplotlib
    - Fedora: sudo dnf install python3-matplotlib
    - Red Hat: sudo yum install python3-matplotlib
    - Arch: sudo pacman -S python-matplotlib
  - Matplotlib đã được tích hợp sẵn trong Anaconda.

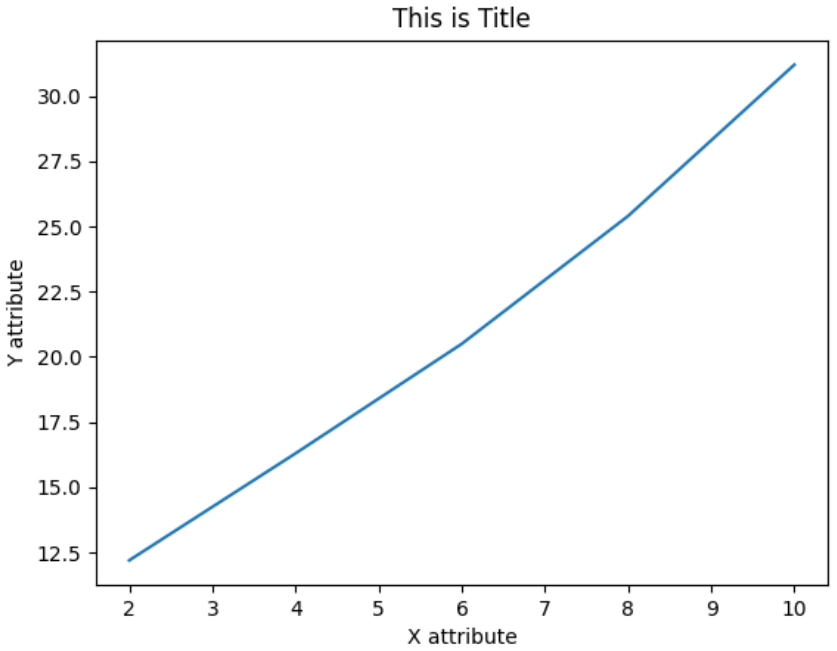
Tạo và tinh chỉnh biểu đồ

- Import thư viện Matplotlib: import matplotlib.pyplot as plt
- From matplotlob import pyplot as plt
- Tạo dữ liệu:

  x = [2, 4, 6, 8, 10]

  y = [12.2, 16.3, 20.5, 25.4, 31.2]
- Vẽ biểu đồ: plt.plot(x, y)
- Thêm tiêu đề: plt.title('This is Title')
- Gán nhãn các trục:

  plt.xlabel('Age')

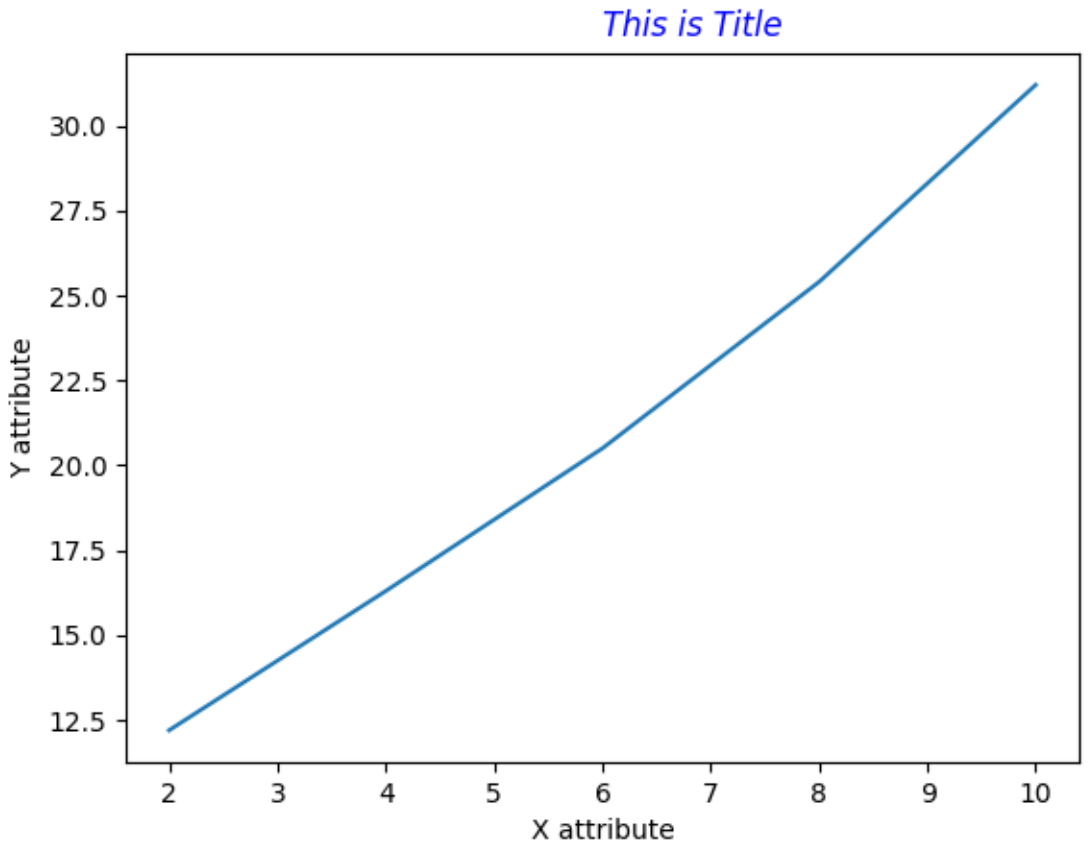


Output

Tạo và tinh chỉnh biểu đồ

- Thay đổi thuộc tính tiêu đề

```
plt.title('This is Title’,
fontdict = {'fontsize': 12,
'fontweight' : 8,
'fontweight' : ‘italic’,
‘color' : ‘blue’,
'verticalalignment': 'baseline',
'horizontalalignment': 'left'})
```

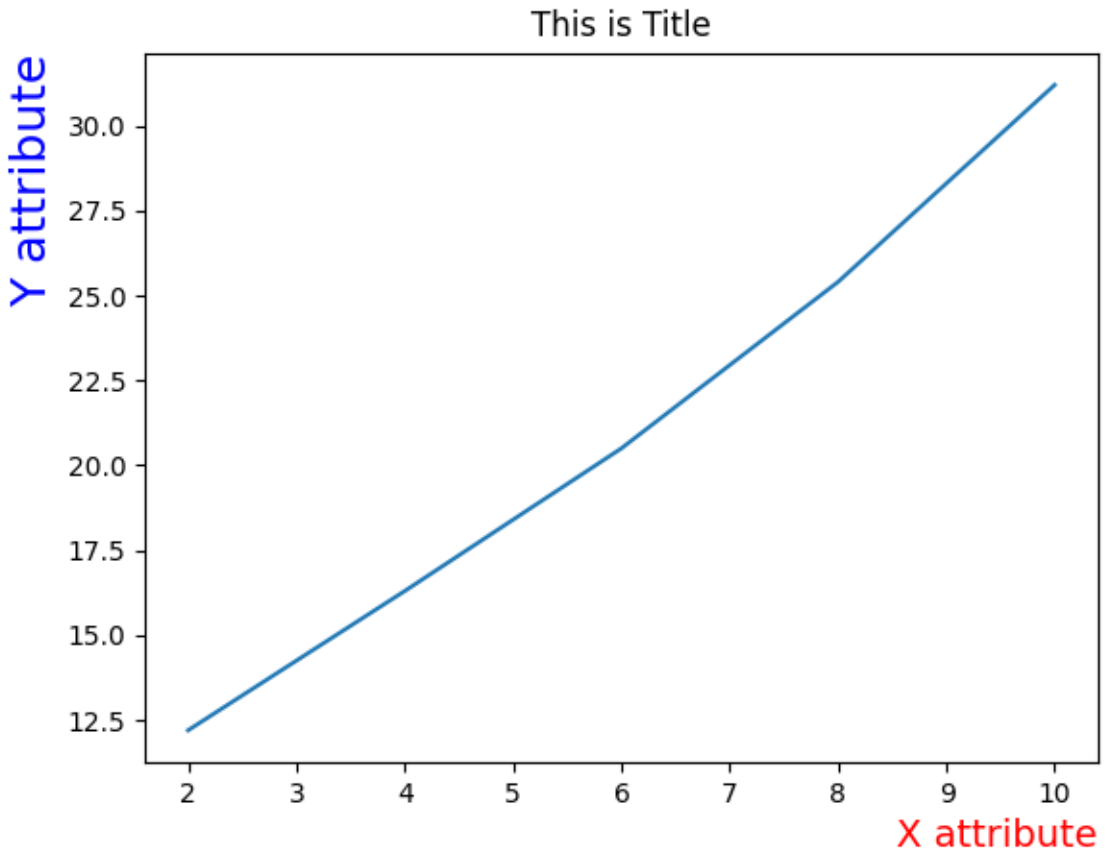


Output

Tạo và tinh chỉnh biểu đồ

- Thay đổi thuộc tính nhãn trục

```
plt.xlabel('X attribute’,
    fontsize=14, color='red’,
    loc=‘right')
plt.ylabel('Y attribute’,
    fontsize=18, color='blue’,
    loc=‘top’)
(có thể thay đổi thông qua
cài đặt axes)
```



Output

Tạo và tinh chỉnh biểu đồ

- Thay đổi cài đặt Axes, Axis

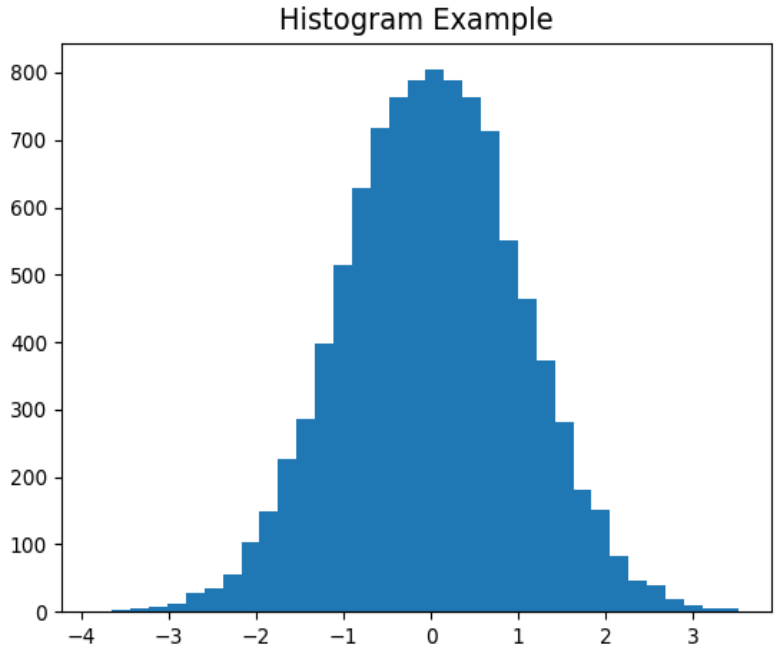
```
axes = plt.gca()                              #get the current axes
axes.set_xlabel('X - axis',
    color = 'red', fontsize = 14);            #setting up X-axis label
axes.set_ylabel('Y - axis',
    color = 'green', fontsize = 18);          #setting up Y-axis label
axes.tick_params(axis='x',
    colors='blue', size = 12)                 #setting up X-axis tick
axes.tick_params(axis='y',
    colors='black', size = 12)                #setting up Y-axis tick
axes.spines['left'].set_color('yellow') #setting up left axis color
```

2. Biểu đồ mô tả một thuộc tính

- Data Visualization cho các thuộc tính có thể được thực hiện độc lập hoặc kết hợp với nhau.
- Các biểu đồ đơn biến hiển thị dữ liệu độc lập nhằm mô tả một thuộc tính.
- Các biểu đồ đơn biến có thể giúp chúng ta:
  - Minh họa dữ liệu qua một khoảng thời gian.
  - Minh họa mối quan hệ so sánh dữ liệu giữa các danh mục với nhau.
  - So sánh một dữ liệu so với toàn thể.

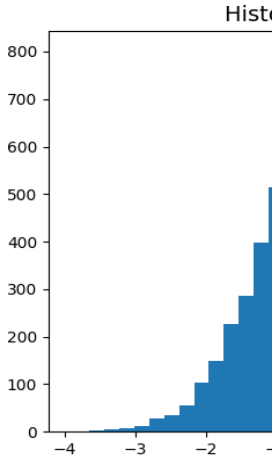
Biểu đồ Histogram

- Khái niệm:
  - Histogram (biểu đồ phân bố) là một dạng biểu đồ thể hiện tần suất dạng cột.
  - Histogram cho thấy hình thái phân bố của dữ liệu.
  - Hình dạng, "độ trơn" của biểu đồ được dùng để đánh giá vấn đề của dữ liệu.



Biểu đồ Histogram

- Thành phần chính:
  - Trục X: thể hiện các lớp (classes). Mỗi lớp là các giá trị dữ liệu trong một khoảng.
  - Mỗi cột trong biểu đồ thể hiện cho một lớp. Độ rộng của lớp chính là độ rộng của cột trong biểu đồ.
  - Trục Y: thể hiện chiều cao của mỗi cột hình chữ nhật trong biểu đồ, diện tích của cột biểu diễn tần số của lớp tương ứng.
  - Độ rộng của dải dữ liệu là sai khác giữa giá trị lớn nhất và nhỏ nhất.



Biểu đồ Histogram

- Ưu điểm:
  - Trực quan hóa dữ liệu tốt.
  - Có thể dùng để so sánh với đường cong chuông (Bell Curve) của phân phối chuẩn.
  - Thể hiện mode, max, min, average, tần suất tương ứng.
  - Hình dạng của phân bố và các vấn đề liên quan (lệch trái, lệch phải, hai đỉnh, bình nguyên, ...).
  - Mối quan hệ giữa các dữ liệu và các giới hạn yêu cầu (max, min. average).

Biểu đồ Histogram

- Hạn chế:
  - Chỉ hiển thị tần suất của dữ liệu, không hiển thị được chính xác giá trị của dữ liệu do dữ liệu đã bị phân vào các lớp.
  - Khó dùng để so sánh nhiều danh mục.

Biểu đồ Histogram

- Vẽ biểu đồ Histogram bằng thư viện Matplotlib:

```
plt.hist(x, bins = None, '...')
```

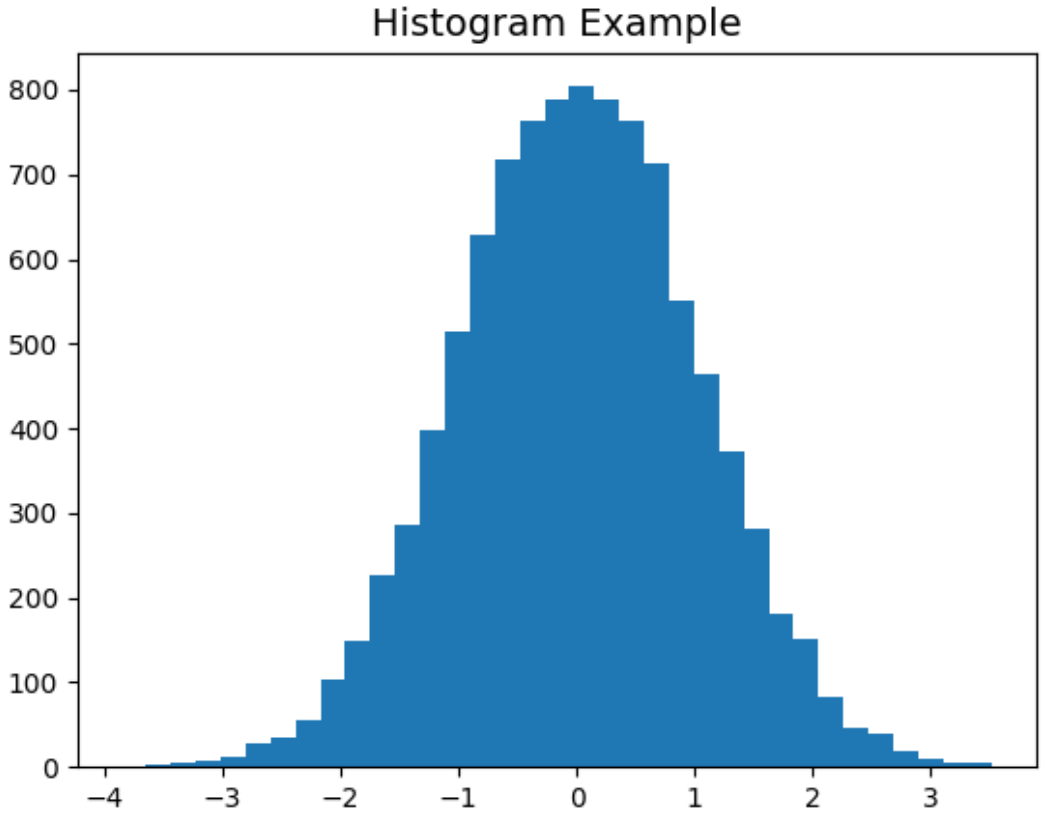
Trong đó:

- x: danh sách dữ liệu.
- bins: số lớp (số cột) dữ liệu.
- '...': các tham số cài đặt thuộc tính biểu đồ.

Biểu đồ Histogram

- Ví dụ:

```
import matplotlib.pyplot as plt
import numpy as np
data = np.random.normal(size=10000)
plt.hist(data, bins = 35)
plt.title('Histogram Example', fontsize = 14)
plt.show()
```



Output

Biểu đồ Histogram

- Các tham số:
  - Phạm vi sử dụng dữ liệu:

```
plt.hist(data, bins = 35, range=(-1,1))
```

  - Vẽ hàm mật độ xác suất:

```
plt.hist(data, bins = 35, density = True)
```

  - Vẽ hàm phân phối tích lũy:

```
plt.hist(gaussian_numbers, bins = 35, cumulative=True)
```

  - Cài đặt kiểu vẽ biểu đồ:

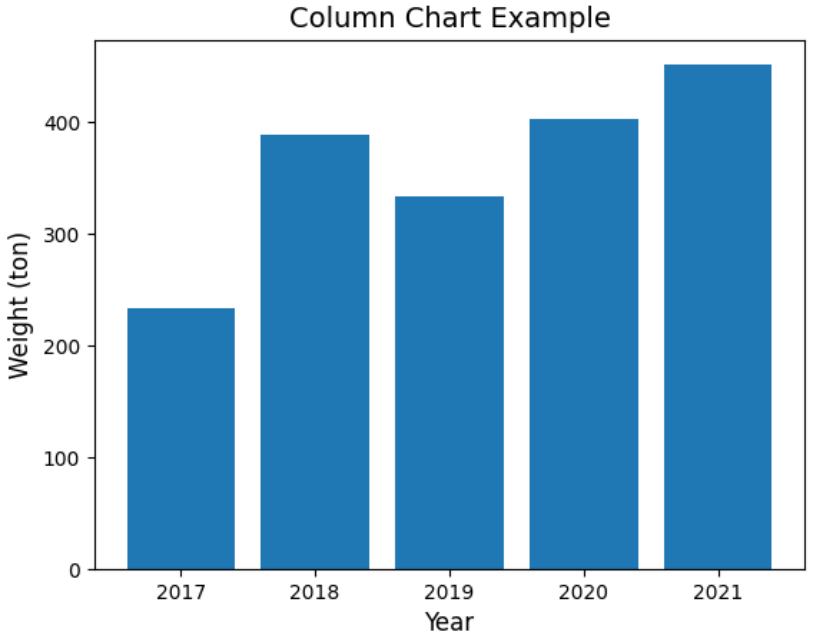
```
plt.hist(gaussian_numbers, bins = 35, histtype='step’)
```

  - Cài đặt hướng hiển thị của biểu đồ:

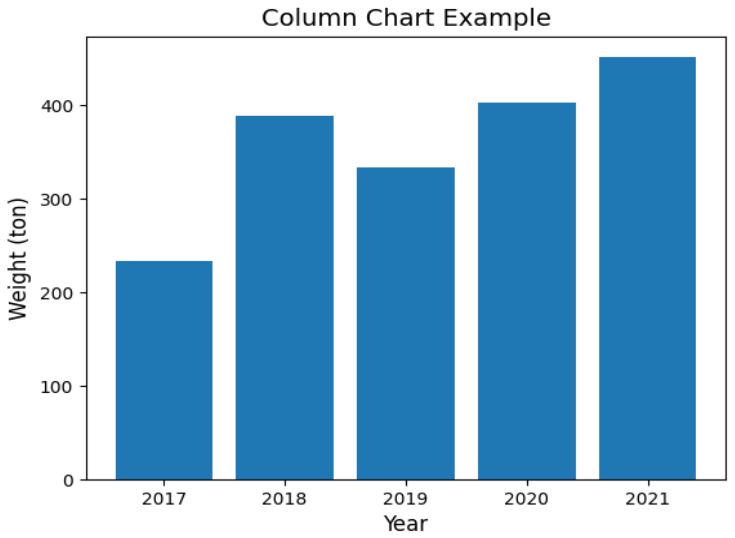
```
plt.hist(gaussian_numbers, bins = 35, orientation='horizontal')
```

Biểu đồ Cột

- Khái niệm:
  - Biểu đồ cột dùng để so sánh sự khác biệt dữ liệu giữa các mốc thời gian rời rạc hay các danh mục cố định.
  - Dữ liệu: tập dữ liệu được phân chia thành các loại/danh mục và không bị ràng buộc thứ tự.
  - Trực quan: dễ hình dung và so sánh dữ liệu của một thuộc tính.



Biểu đồ Cột



Thành phần chính:

- Trục X: một khoảng thời gian hoặc tập các danh mục.
- Trục Y: các giá trị số của danh mục.
- Các cột: đại diện cho một danh mục cụ thể.
- Tất cả các cột phải biểu diễn chung một thuộc tính của dữ liệu.
- Mỗi cột trong biểu đồ có chiều rộng bằng nhau.
- Khoảng cách giữa mỗi cột phải giống nhau.
- Các cột biểu diễn dữ liệu có thể ở dạng đứng hoặc nằm ngang.

Biểu đồ Cột

- Ưu điểm:
  - Dễ đọc, dễ hiểu, trực quan đơn giản.
  - Minh họa rõ xu hướng của dữ liệu tốt hơn so với bảng.
  - Giúp ước tính các giá trị quan trọng như max, min, mức độ chênh lệch dữ liệu giữa các danh mục một cách nhanh chóng.

Biểu đồ Cột

- Hạn chế:
  - Không thể hiện được quy luật các mẫu, nguyên nhân, tác động, v.v.
  - Dễ dàng bị thao túng để mang lại thông tin theo cách mình muốn.

  Ví dụ: nếu chọn tỉ lệ trên trục Y quá lớn thì sự chênh lệch dữ liệu có thể xuất hiện không đáng kể, trong khi thực tế nó có thể rất quan trọng.

Biểu đồ Cột

- Vẽ biểu đồ Cột bằng thư viện Matplotlib:

```
plt. bar(x, y, '...')
```

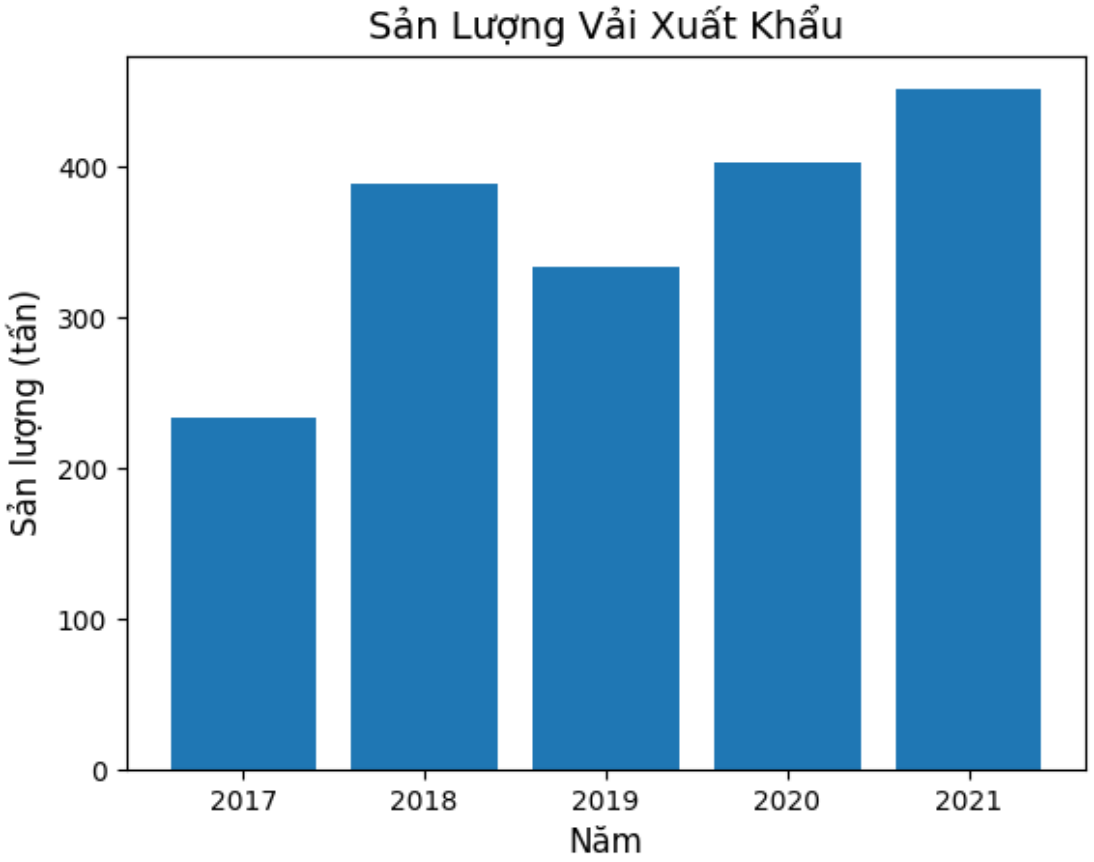
Trong đó:

- x: danh sách hoành độ (danh sách các danh mục hoặc các mốc thời gian).
- y: danh sách tung độ (giá trị số tương ứng với các danh mục).
- '...': các tham số cài đặt thuộc tính biểu đồ.

Biểu đồ Cột

- Ví dụ:

```
import matplotlib.pyplot as plt
x = ['2017', '2018', '2019', '2020', '2021']
y = [234, 389, 333, 402, 451]
plt.bar(x, y)
plt.title('Sản Lượng Vải Xuất Khẩu', fontsize
plt.xlabel('Năm', fontsize = 12)
plt.ylabel('Sản lượng (tấn)', fontsize = 12)
plt.show()
```



Output

Biểu đồ Cột

- Các tham số:
  - Thay đổi chiều rộng các cột:

```
plt.bar(x, y, width=0.5)
```

  - Cài đặt giá trị cơ sở của trục Y :

```
plt.bar(x, y, bottom= 100)
```

  - Căn chỉnh vị trí của cột so với tọa độ x :

```
plt.bar(x, y, bottom= 100)
```

  - Cài đặt một màu hoặc danh sách các màu cho các cột:

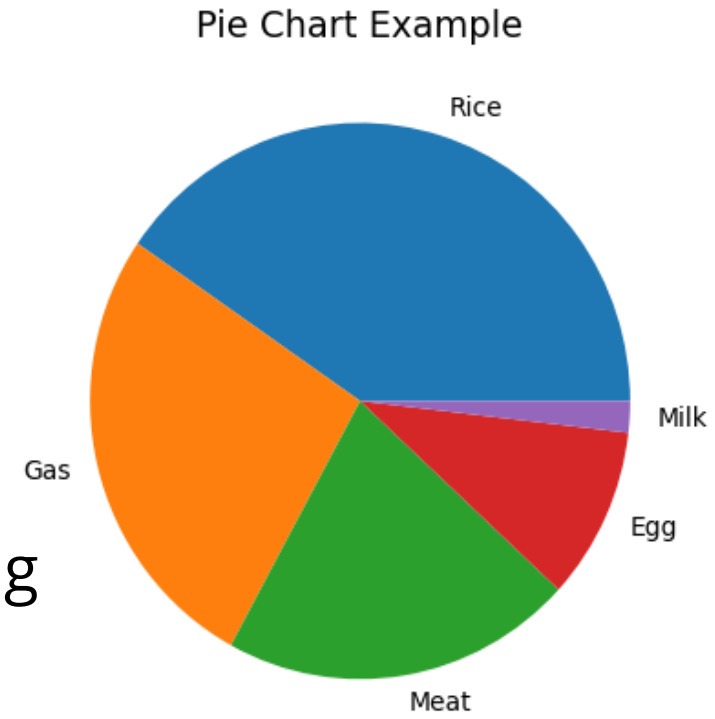
```
plt.bar(x, y, color = ['red', 'green', 'blue'])
```

  - Cài đặt nhãn cho các cột:

```
plt.bar(x, y, tick_label = ['Năm 1', 'Năm 2', 'Năm 3', 'Năm 4', 'Năm 5'])
```

Biểu đồ Tròn

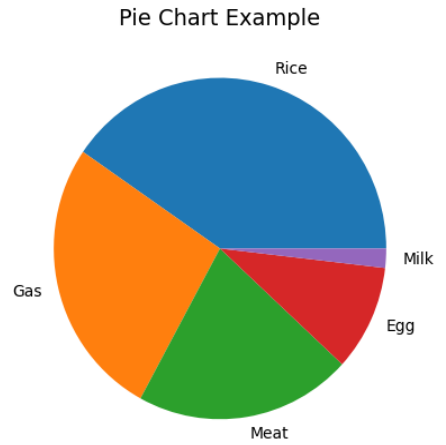
- Khái niệm:
  - Biểu diễn tỉ lệ phần trăm của các thành phần so với tổng thể.
  - Thể hiện thị phần của mỗi giá trị trong tập dữ liệu (tần số một biến rời rạc).
  - Minh họa cấu thành của bộ dữ liệu và tương quan giữa các thành phần so với tổng thể.



Biểu đồ Tròn

Thành phần chính:

- Tổng các thành phần là 100%, không thành phần nào có giá trị âm.
- Chỉ dùng biểu đồ tròn khi số lượng thể loại ít hơn 6.
- Không dùng biểu đồ tròn nếu tỉ lệ giữa các thể loại gần tương đương nhau.
- Nên sắp xếp giá trị các thể loại để dễ hiểu hơn (ví dụ từ lớn đến nhỏ).
- Tránh sử dụng dạng 3D hoặc nghiêng (bóp méo biểu đồ).



Biểu đồ Tròn

- Ưu điểm:
  - Tóm tắt một tập dữ liệu lớn ở dạng trực quan.
  - Giúp dễ dàng hình dung được giá trị % mà các thành phần đóng góp vào tổng thể.

Biểu đồ Tròn

- Hạn chế:
  - Không thể hiện được giá trị chính xác.
  - Có thể cần nhiều biểu đồ hình tròn để hiển thị các thay đổi theo thời gian.
  - Không thể hiện được các quy luật, nguyên nhân, tác động, v.v.
  - Dễ dàng bị thao túng để mang lại thông tin thiếu khách quan.

Biểu đồ Tròn

- Vẽ biểu đồ Tròn bằng thư viện Matplotlib:

```
plt.pie(sizes, labels = danh-sách, '...')
```

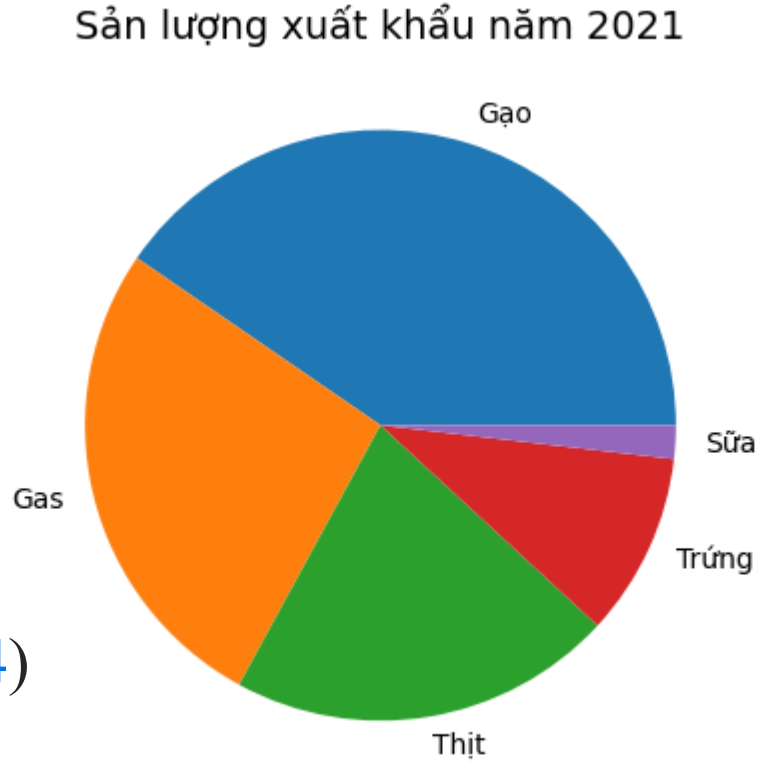
Trong đó:

- sizes: danh sách giá trị các danh mục.
- labels: danh sách nhãn các danh mục.
- '...': các tham số.

Biểu đồ Tròn

- Ví dụ:

```
import matplotlib.pyplot as plt
labels = ['Gạo', 'Gas', 'Thịt', 'Trứng', 'Sữa']
sizes = [443, 292, 231, 110, 20]
plt.pie(sizes, labels = labels)
plt.title('Sản lượng xuất khẩu năm 2021', fontsize = 14)
plt.show()
```



Output

Biểu đồ Tròn

- Các tham số:
  - Cài đặt màu sắc cho các thành phần:

```
plt.pie(sizes, labels = labels, colors = ['r', 'g', 'b', 'y', 'c'])
```

  - Dùng để gán nhãn % cho từng thành phần:

```
plt.pie(sizes, labels = labels, autopct='%1.2f%%')
```

  - Tỷ lệ giữa tâm của mỗi phần và nhãn (autopct != None):

```
plt.pie(sizes, labels = labels, autopct='%1.2f%%', pctdistance=0.4)
```

  - Vẽ bóng mờ cho các thành phần:

```
plt.pie(sizes, labels = labels, shadow= True)
```

  - Bán kính của hình tròn:

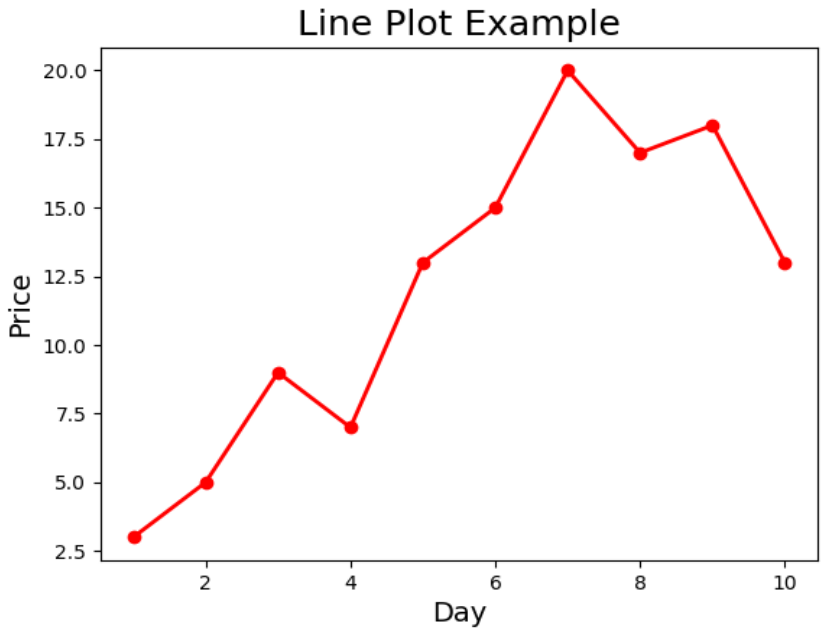
```
plt.pie(sizes, labels = labels, radius=1.3)
```

3. Biểu đồ mô tả mối liên hệ giữa các thuộc tính

- Khi chúng ta có nhiều thuộc tính, chúng ta thường quan tâm đến việc mô tả hoặc nắm bắt các mối quan hệ giữa các thuộc tính như:
  - Quan hệ tuyến tính hay phi tuyến.
  - Mức độ tương quan nhiều hay ít.
  - Xu hướng dữ liệu.

Biểu đồ Đường

- Khái niệm:
  - Hiển thị sự thay đổi theo thời gian dưới dạng một chuỗi các điểm dữ liệu được nối bởi các đoạn thẳng.
  - Giúp xác định mối quan hệ giữa hai thuộc tính như tỉ lệ thuận hay nghịch.
  - Dự đoán về kết quả của dữ liệu chưa được ghi lại.



Biểu đồ Đường

- Thành phần chính:
  - Trục X: thể hiện các giá trị của thuộc tính thời gian.
  - Trục Y: thể hiện giá trị của dữ liệu tương ứng điểm thời gian.
  - Mỗi điểm tương ứng với một giá trị thuộc tính Y tại thời điểm X.
  - Các điểm liên tiếp được nối với nhau bởi các đoạn thẳng.
  - Không vẽ quá nhiều đường trong cùng một biểu đồ.
  - Sử dụng tỉ lệ trục Y phù hợp, luôn bắt đầu tại 0.

Biểu đồ Đường

- Ưu điểm:
  - Trình bày đơn giản; dễ đọc và tạo.
  - Có thể trình bày trực quan nhiều chuỗi dữ liệu khác nhau.
  - Phù hợp để biểu thị một xu hướng trong một khoảng thời gian, có thể sử dụng để dự đoán dữ liệu bị thiếu, dữ liệu chưa xảy ra.
  - Xử lý được những dữ liệu tích cực và tiêu cực

Biểu đồ Đường

- Hạn chế:
  - Khó đọc hơn khi những dữ liệu chồng lấp lên nhau
  - Có thể khó xác định các giá trị chính xác tại một điểm nhất định của biểu đồ.
  - Không phù hợp khi khoảng thời gian hay phạm vi dữ liệu quá lớn.

Biểu đồ Đường

- Vẽ biểu đồ Đường bằng thư viện Matplotlib:

plt. plot(x, y, '...')

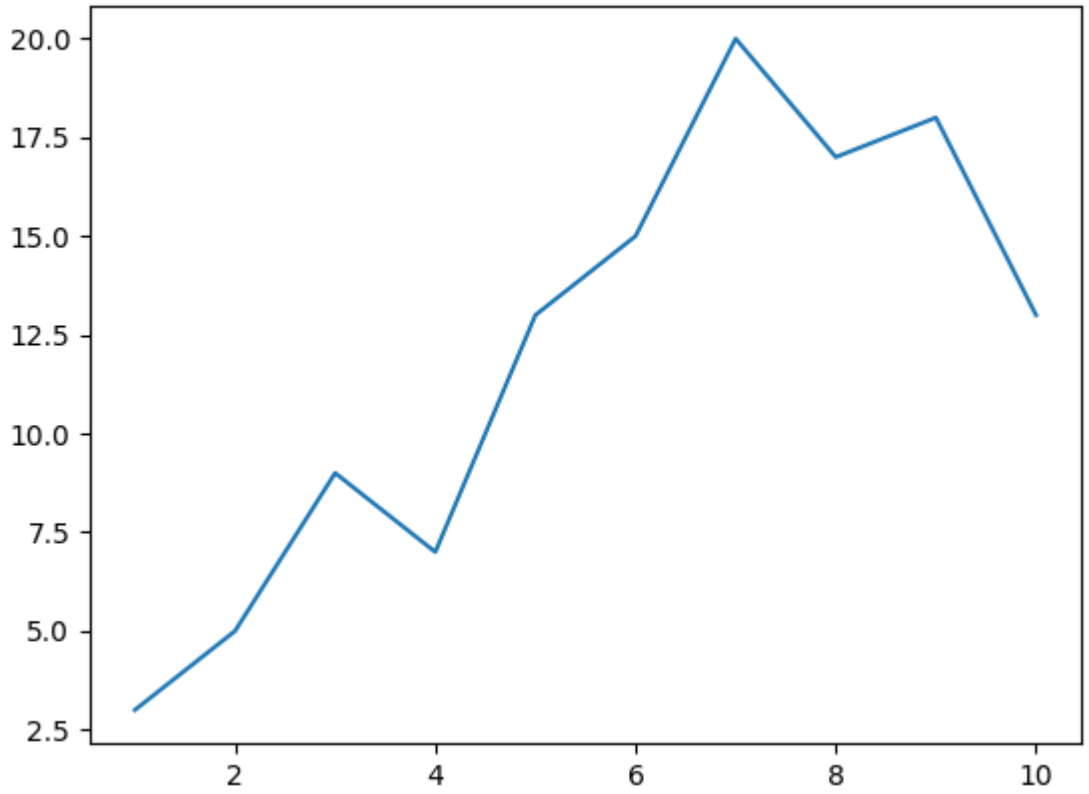
Trong đó:

- x: danh sách hoành độ.
- y: danh sách tung độ.
- '...': các tham số.

Biểu đồ Đường

- Ví dụ:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [3, 5, 9, 7, 13, 15, 20, 17, 18, 13]
plt.plot(x, y)
plt.show()
```



Output

Biểu đồ Đường

- Các tham số:
  - Cài đặt độ rộng của đường kẻ:

```
plt.plot(x, y, linewidth=3)
```

  - Kiểu đường vẽ: '-' or 'solid' or '--' or 'dashed' or '-.' or 'dashdot' or':' or 'dotted' or …

```
plt.plot(x, y, '-')
```

  - Đánh dấu đường kẻ:

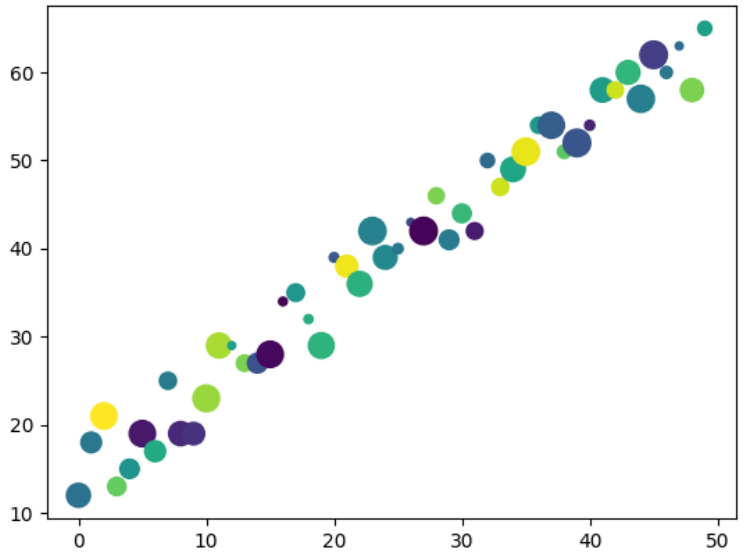
```
plt.plot(x, y, marker='s', markersize=10, markerfacecolor='red', markeredgecolor = 'blue', markeredgewidth=2)
```

  - Cài đặt màu cho đường kẻ: 'b' (blue) 'g' (green) 'r' (red) 'c' (cyan) 'm' (magenta) 'y' (yellow) 'k' (black) 'w' (white) …

```
plt.plot(x, y, 'r')
```

Biểu đồ Scatter

- Khái niệm:
  - Biểu đồ sử dụng các dấu chấm (điểm) để thể hiên giá trị (điểm giao nhau) của hai thuộc tính khác nhau.
  - Thường dùng để biểu diễn mối liên hệ giữa hai thuộc tính.
  - Mối liên hệ có thể được diễn tả thông qua độ dốc (slope), độ mạnh (strength), và độ tuyến tính (linearity).



Biểu đồ Scatter

- Thành phần chính:
  - Trục Y: hiển thị các giá trị của biến phụ thuộc (biến cần dự đoán).
  - Trục X: hiển thị giá trị của các biến độc lập (biến dùng để dự đoán).
  - Mỗi điểm thể hiện một quan sát từ một tập dữ liệu
  - Độ tuyến tính (linearity).
  - Độ dốc (slope): dương (tỉ lệ thuận), âm (tỉ lệ nghịch).
  - Độ mạnh (strength): mức độ tập trung hay phân tán của các điểm (mối tương quan mạnh hay yếu).

Biểu đồ Scatter

- Ưu điểm:
  - Phát hiện mối quan hệ giữa hai thuộc tính.
  - Phát hiện các điểm ngoại lệ.
  - Cung cấp một vài mô tả ban đầu về mối quan hệ giữa hai thuộc tính như: tỉ lệ thuận/nghịch, mức độ tương quan,...

Biểu đồ Scatter

- Hạn chế:
  - Không dễ để phân tích như các loại biểu đồ khác.
  - Chỉ thích hợp cho các loại dữ liệu cần đi tìm mối tương quan.

Biểu đồ Scatter

- Vẽ biểu đồ Scatter bằng thư viện Matplotlib:

```
plt. scatter(x, y, '...');
```

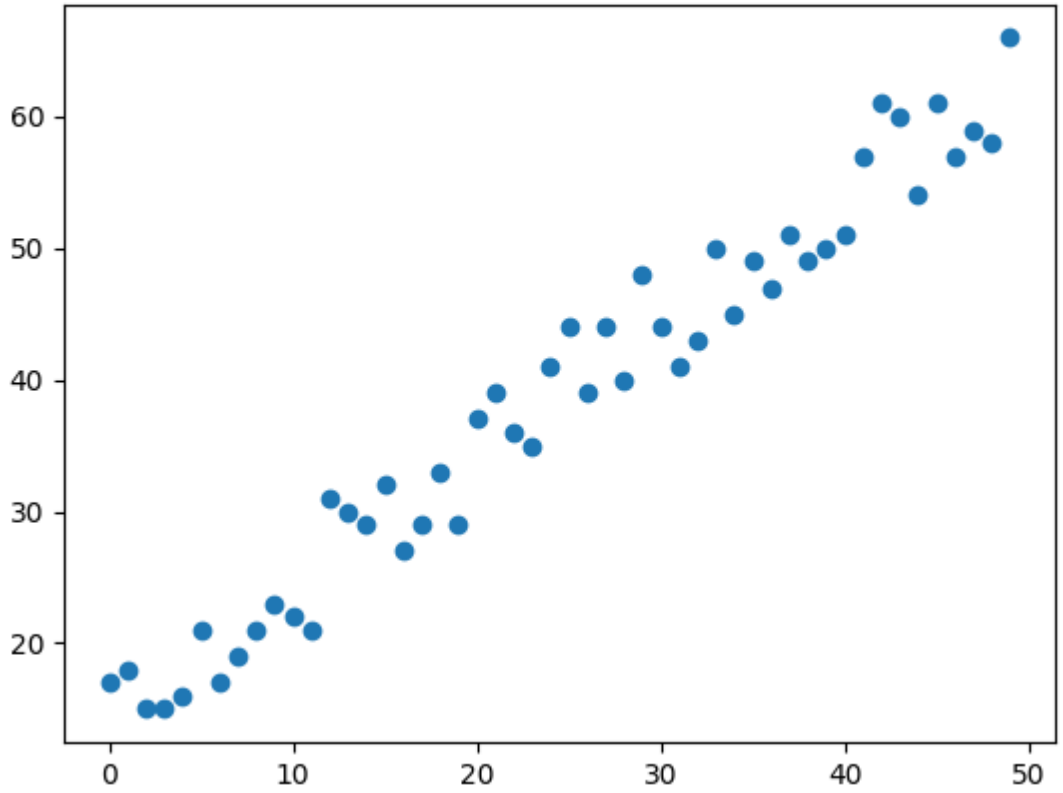
Trong đó:

- x: danh sách giá trị thuộc tính độc lập.
- y: danh sách giá trị thuộc tính phụ thuộc.
- '...': các tham số.

Biểu đồ Scatter

- Ví dụ:

```
import matplotlib.pyplot as plt
import numpy as np
x = range(50)
y = range(50) + np.random.randint(10,20,50)
plt.scatter(x, y)
plt.show()
```



Output

Biểu đồ Scatter

- Các tham số:
  - Thay đổi kích thước điểm : sizes = np.random.randint(5, 200, 50)

```
plt.scatter(x, y, s=sizes)
```

  - Thay đổi màu sắc điểm: colors = np.random.randint(0, 100, 50)

```
plt.scatter(x, y, s=sizes, c = colors)
```

  - Thay đổi kiểu đánh dấu điểm: "o" (circle) "1" (tri_down) "2" (tri_up) "3" (tri_left) "4" (tri_right) "s" (square) "p" (pentagon) "P" (filled plus) "*" (star)

```
plt.scatter(x, y, s=sizes, c = colors, marker='*')
```

  - Hòa trộn giữa độ trong suốt (0) và rõ ràng (1):

```
plt.scatter(x, y, s=sizes, c = colors, alpha=0.5)
```

Tóm tắt

Qua bài học này, chúng ta đã tìm hiểu những kiến thức sau:

- Khái niệm Data Visualization
- Thư viện Matplotlib và cách sử dụng để vẽ và tinh chỉnh biểu đồ
  - Biểu đồ Histogram.
  - Biểu đồ Cột.
  - Biểu đồ Tròn.
  - Biểu đồ Đường.
  - Biểu đồ Scatter.