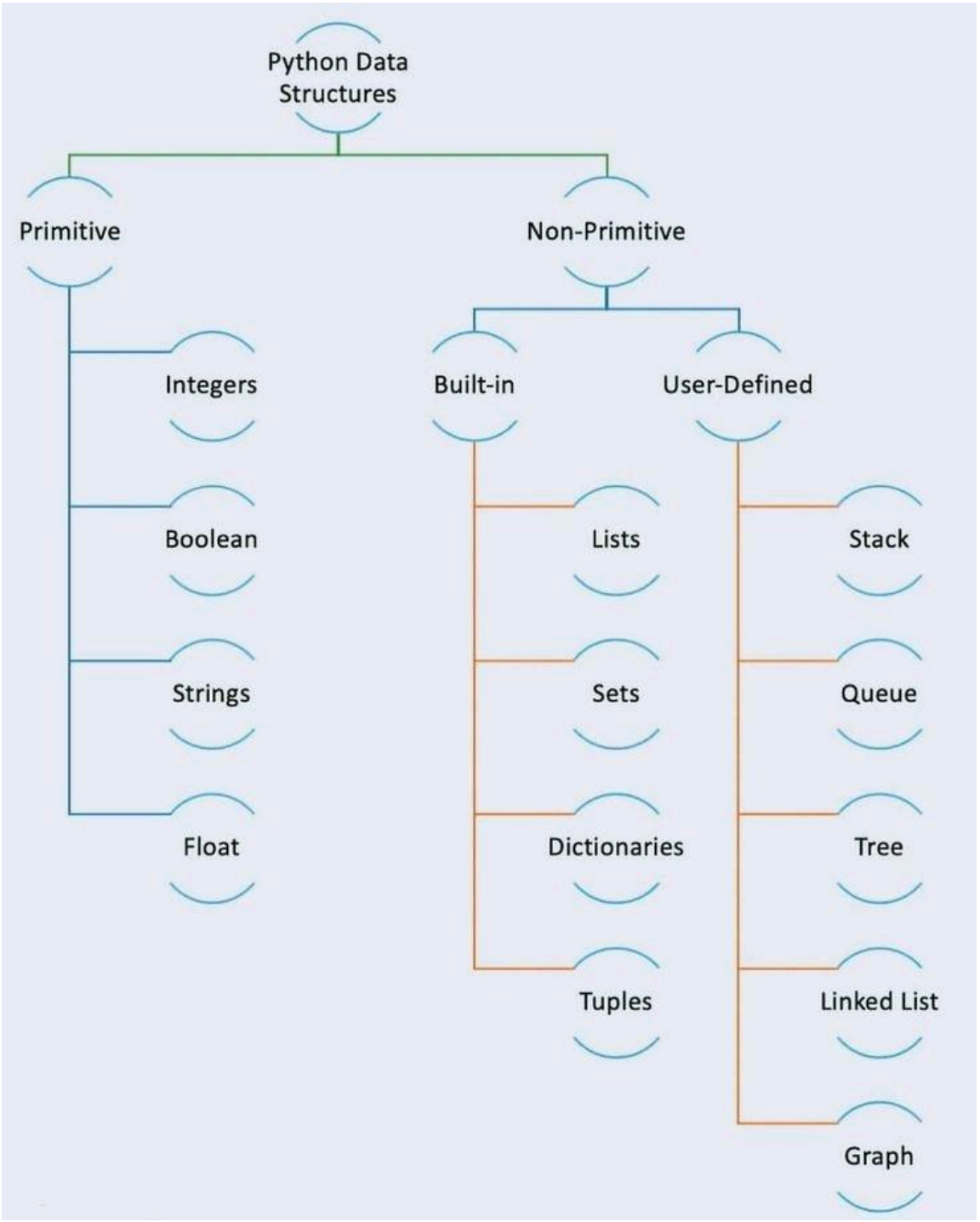
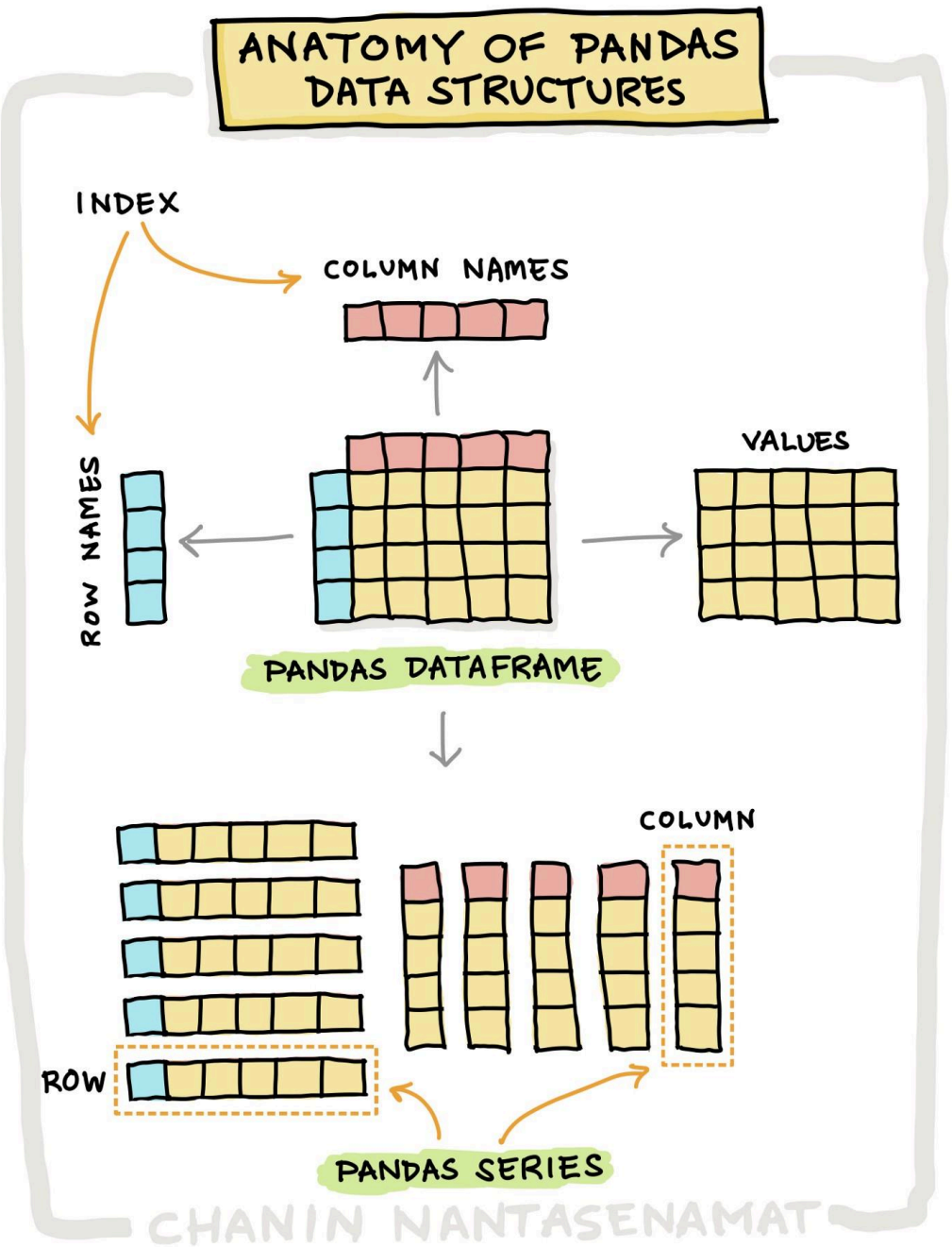
Draft Version

Chương I. XỬ LÝ DỮ LIỆU VỚI PANDAS

Python Data Structures



Pandas



Data Importing:

1. pd.read_csv()
2. pd.read_table()
3. pd.read_excel()
4. pd.read_sql()
5. pd.read_json()
6. pd.read_html()
7. pd.DataFrame()
8. pd.concat()
9. pd.series()
10. pd.date_range()

Data Cleaning:

1. pd.fillna()
2. pd.dropna()
3. pd.sort_values()
4. pd.apply()
5. pd.groupby()
6. pd.append()
7. pd.join()
8. pd.rename()
9. pd.to_csv()
10. pd.set_index()

Data Statistic:

1. pd.head()
2. pd.tail()
3. pd.describe()
4. pd.info()
5. pd.mean()
6. pd.median()
7. pd.count()
8. pd.std()
9. pd.max()
10. pd.min()

Chương II. TRỰC QUAN HOÁ DỮ LIỆU

1. Prepare The Data

Mục tiêu:

- Đọc và xử lý dữ liệu từ file CSV.
- Chuyển đổi dữ liệu (ví dụ: chuyển cột ngày thành kiểu `datetime`) và sắp xếp theo thứ tự thời gian để đảm bảo đồ thị hiển thị đúng trình tự.

Các bước thực hiện:

- Sử dụng thư viện **pandas** để đọc file CSV.
- Chuyển đổi cột `Date` sang kiểu `datetime`.
- Sắp xếp dữ liệu theo cột `Date` nếu cần.

Ví dụ mã nguồn python:

```
import pandas as pd

# Đọc file CSV
df = pd.read_csv('Gia_uniswap.csv')

# Chuyển đổi cột 'Date' sang kiểu datetime
df['Date'] = pd.to_datetime(df['Date'])

# Sắp xếp dữ liệu theo ngày (nếu dữ liệu chưa được sắp xếp)
df.sort_values('Date', inplace=True)
```

---

2. Create Plot

Mục tiêu:

- Tạo ra một figure và axes dùng để vẽ đồ thị.
- Figure là khung chứa toàn bộ đồ thị, còn axes là vùng mà các đường đồ thị được vẽ.

Các bước thực hiện:

- Sử dụng `plt.subplots()` từ thư viện **matplotlib.pyplot** để khởi tạo figure và axes.
- Thiết lập kích thước của figure nếu cần.

Ví dụ mã nguồn python:

```
import matplotlib.pyplot as plt

# Tạo figure và axes với kích thước 10x6 inch
fig, ax = plt.subplots(figsize=(10, 6))
```

---

3. Plotting Routines

Mục tiêu:

- Vẽ các đường biểu diễn dữ liệu lên axes.
- Sử dụng các hàm như `ax.plot()` để tạo các đường đồ thị.

Các bước thực hiện:

- Vẽ dữ liệu từ cột `Close` và `Adj Close` theo trục thời gian.
- Tùy chọn các tham số như kiểu đường, marker, và nhãn (label) để phân biệt các đường dữ liệu.

Ví dụ mã nguồn python:

```
# Vẽ đường biểu diễn giá 'Close'
ax.plot(df['Date'], df['Close'], label='Close Price', marker='o', linestyle='-')

# Vẽ đường biểu diễn giá 'Adj Close'
ax.plot(df['Date'], df['Adj Close'], label='Adjusted Close', marker='s',
linestyle='--')
```

---

4. Customize Plot

Mục tiêu:

- Tùy chỉnh đồ thị cho đẹp mắt và dễ hiểu.
- Thêm tiêu đề, nhãn trục, chú thích (legend) và lưới (grid) để cải thiện khả năng đọc.

Các bước thực hiện:

- Sử dụng `ax.set_title()` để đặt tiêu đề cho đồ thị.
- Sử dụng `ax.set_xlabel()` và `ax.set_ylabel()` để đặt tên cho trục X và Y.
- Kích hoạt chú thích bằng `ax.legend()` và hiển thị lưới bằng `ax.grid(True)`.

Ví dụ mã nguồn python:

```
# Thêm tiêu đề cho đồ thị
ax.set_title('Historical Price Data for Uniswap Asset')

# Đặt nhãn cho trục X và trục Y
ax.set_xlabel('Date')
ax.set_ylabel('Price')

# Hiển thị chú thích để phân biệt các đường
ax.legend()

# Bật lưới cho đồ thị
ax.grid(True)
```

---

5. Save Plot

Mục tiêu:

- Lưu đồ thị vừa tạo ra dưới dạng file ảnh (ví dụ PNG) để sử dụng sau này hoặc chia sẻ.

Các bước thực hiện:

- Sử dụng hàm `plt.savefig()` để lưu đồ thị.
- Cấu hình thông số như dpi (độ phân giải) và định dạng file.

Ví dụ mã nguồn python:

```
# Lưu đồ thị thành file 'uniswap_price_plot.png' với dpi=300
plt.savefig('uniswap_price_plot.png', dpi=300)
```

---

6. Show Plot

Mục tiêu:

- Hiển thị đồ thị lên màn hình để xem kết quả.

Các bước thực hiện:

- Sử dụng `plt.show()` để mở cửa sổ hiển thị đồ thị.

Ví dụ mã nguồn python:

```
# Hiển thị đồ thị
plt.show()
```

Tích hợp tất cả các bước

Dưới đây là ví dụ hoàn chỉnh tích hợp tất cả các bước từ 1 đến 6:

```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Prepare The Data
df = pd.read_csv('Gia_uniswap.csv')
df['Date'] = pd.to_datetime(df['Date'])
df.sort_values('Date', inplace=True)

# 2. Create Plot
fig, ax = plt.subplots(figsize=(10, 6))

# 3. Plotting Routines
ax.plot(df['Date'], df['Close'], label='Close Price', marker='o', linestyle='-')
```

```
ax.plot(df['Date'], df['Adj Close'], label='Adjusted Close', marker='s',
linestyle='--')

# 4. Customize Plot
ax.set_title('Historical Price Data for Uniswap Asset')
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()
ax.grid(True)

# 5. Save Plot
plt.savefig('uniswap_price_plot.png', dpi=300)

# 6. Show Plot
plt.show()
```

---

2. Create Plot

Nhóm hàm/tính năng này giúp bạn khởi tạo vùng vẽ (figure) và các trục (axes) trước khi đặt biểu đồ vào. Một số hàm quan trọng:

1. **`plt.figure()`**
    - Tạo một Figure trống (cửa sổ/khung vẽ chính).
    - Ví dụ: `fig = plt.figure(figsize=(8, 6))`
2. **`plt.subplots()`**
    - Tạo một Figure và một (hoặc nhiều) Axes (tập các trục) cùng lúc.

Thường dùng nhất vì nó tiện lợi:
python
CopyEdit

```
fig, ax = plt.subplots()            # 1 cặp Axes
fig, axes = plt.subplots(2, 2)      # 4 Axes (2x2)
```

    -
3. **`plt.subplot(nrows, ncols, index)`**
    - Tạo một Axes trong figure tại vị trí chỉ định (dùng chung figure).
    - Thường dùng cho cách bố trí lưới thủ công.
4. **`matplotlib.figure.Figure()`** (lớp Figure) & **`fig.add_subplot()`**
    - Khi bạn đã có đối tượng Figure (từ `plt.figure()` chẳng hạn), bạn có thể thêm Axes bằng hàm `add_subplot()`.

---

3. Plotting Routines

Đây là các hàm "vẽ" biểu đồ cụ thể lên đối tượng Axes (hoặc lên "current Axes" của `pyplot`). Trong `matplotlib.pyplot`, có rất nhiều hàm vẽ thường dùng như:

1. **Dạng tuyến tính & cơ bản**
    - `plt.plot()` – Vẽ đường (line plot) hoặc đánh dấu điểm theo kiểu tuyến tính.
    - `plt.scatter()` – Vẽ biểu đồ phân tán (scatter plot).
    - `plt.step()` – Vẽ dạng bậc thang.
    - `plt.stem()` – Vẽ dạng cột xung (stem plot).
2. **Biểu đồ cột & biểu đồ phân bố**
    - `plt.bar()` – Vẽ biểu đồ cột đứng.
    - `plt.barh()` – Vẽ biểu đồ cột ngang.
    - `plt.hist()` – Vẽ biểu đồ histogram (phân bố tần suất).
3. **Biểu đồ thống kê khác**
    - `plt.boxplot()` – Vẽ box plot (giá trị tứ phân vị).
    - `plt.violinplot()` – Vẽ violin plot (tương tự box plot nhưng thể hiện phân bố "mềm").
    - `plt.errorbar()` – Vẽ đường kèm thanh sai số (error bars).
4. **Biểu đồ hình tròn**
    - `plt.pie()` – Vẽ biểu đồ hình tròn.
5. **Biểu đồ ảnh, bản đồ màu**
    - `plt.imshow()` – Hiển thị hình ảnh/ma trận 2D.
    - `plt.contour()` / `plt.contourf()` – Vẽ đường đồng mức (contour) hoặc vùng màu.
    - `plt.pcolor()`, `plt.pcolormesh()` – Vẽ lưới màu theo giá trị.
    - `plt.hexbin()` – Vẽ biểu đồ phân bố dạng lưới lục giác 2D.
6. **Biểu đồ nhiều lớp hoặc vùng phủ**
    - `plt.fill()` – Vẽ vùng được tô màu giữa các điểm.
    - `plt.fill_between()` – Vẽ vùng tô màu giữa hai đường (hoặc giữa một đường và trục).
    - `plt.stackplot()` – Vẽ các miền xếp chồng (phân lớp).

Có thêm các hàm hiếm dùng khác như `plt.polar()`, `plt.plot_date()`, v.v. tùy yêu cầu cụ thể.

---

4. Customize Plot

Nhóm hàm này giúp tùy chỉnh nhãn trục, tiêu đề, chú thích, lưới, phong cách (style) ...
Bạn có thể gọi trực tiếp trên **pyplot** (ví dụ `plt.title()`) hoặc gọi qua **đối tượng Axes** (ví dụ `ax.set_title()`). Một số hàm chính:

1. **Tiêu đề & Nhãn**
    - `plt.title("Tiêu đề")`
    - `plt.xlabel("Trục X")`, `plt.ylabel("Trục Y")`
    - Với đối tượng Axes: `ax.set_title()`, `ax.set_xlabel()`, `ax.set_ylabel()`
2. **Phạm vi trục & Thang chia**
    - `plt.xlim(min, max)`, `plt.ylim(min, max)` – Giới hạn miền hiển thị trên trục x/y.
    - `plt.xticks([...])`, `plt.yticks([...])` – Tùy chỉnh vị trí & nhãn ticks cho trục.
3. **Chú thích & Ghi chú**
    - `plt.legend()` – Hiển thị chú thích (legend) cho các đường/biểu đồ (khi có gán nhãn `label`).
    - `plt.annotate(text, xy=(x,y), ...)` – Thêm chú thích vào vị trí (x,y) cụ thể trên biểu đồ.
    - `plt.text(x, y, s)` – Thêm chữ tùy ý tại vị trí (x,y).
4. **Lưới (grid)**
    - `plt.grid(True)` – Bật/tắt lưới trên biểu đồ.
5. **Thiết lập phong cách & tham số mặc định**
    - `plt.style.use("tên_style")` – Chọn phong cách vẽ có sẵn (vd: “ggplot”, “classic”, …).
    - `plt.rcParams[...] = ...` – Tùy chỉnh tham số mặc định (cỡ chữ, font, màu, v.v.).
6. **Lấy Axes/Figure hiện hành**
    - `plt.gca()` – Lấy đối tượng Axes hiện hành.
    - `plt.gcf()` – Lấy đối tượng Figure hiện hành.

---

5. Save Plot

Dùng để lưu hình ảnh biểu đồ ra file. Có thể gọi từ **pyplot** hoặc trên **đối tượng Figure**:

1. `plt.savefig("ten_file.png")`
    - Lưu figure hiện hành thành file ảnh (png, jpg, pdf, svg,...).
    - Thường đi kèm các tham số như `dpi=300`, `bbox_inches="tight"`, v.v.
2. `fig.savefig("ten_file.png")`
    - Lưu figure cụ thể (nếu bạn đang quản lý nhiều figure).

---

6. Show Plot

Dùng để hiển thị cửa sổ biểu đồ (khi chạy tương tác hoặc trên môi trường cần pop-up):

1. `plt.show()`
    - Hiển thị tất cả figure đang mở.
    - Chú ý: Trong một số môi trường như Jupyter Notebook, bạn có thể không cần `plt.show()` mà biểu đồ vẫn hiện; tuy nhiên khi code Python thuần, bạn gần như luôn gọi `plt.show()` để "xuất hiện" cửa sổ vẽ.

Tóm tắt

- **Chuẩn bị dữ liệu (Prepare The Data)**: Hầu như không có hàm tích hợp sẵn trong Matplotlib; thường dùng NumPy/pandas.
- **Tạo vùng vẽ (Create Plot)**: `plt.figure()`, `plt.subplots()`, `plt.subplot()`, `fig.add_subplot()`.
- **Vẽ biểu đồ (Plotting Routines)**: `plt.plot()`, `plt.scatter()`, `plt.bar()`, `plt.hist()`, `plt.boxplot()`, `plt.pie()`, `plt.fill_between()`, `plt.contour()`, ...
- **Tùy biến biểu đồ (Customize Plot)**: `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.legend()`, `plt.grid()`, `plt.xticks()`, `plt.ylim()`, `plt.rcParams`, ...
- **Lưu biểu đồ (Save Plot)**: `plt.savefig()`, `fig.savefig()`.
- **Hiển thị biểu đồ (Show Plot)**: `plt.show()`.

Data Visualization with Pandas

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('data.csv')
```

```
df.head(3)
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0

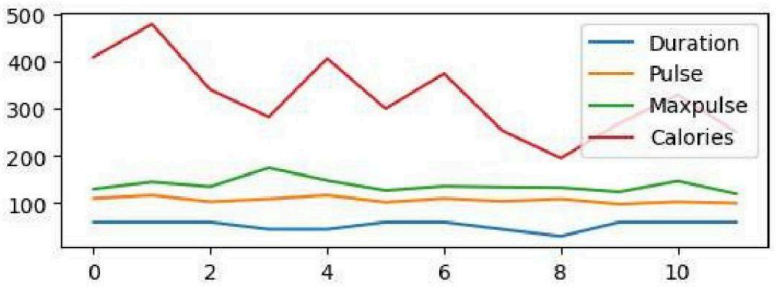
plot()

- It makes charts – defaults is line plot
- More graphs with kind='line','bar','scatter','hist',etc

Line Plot

syntax: plot() or plot.line()

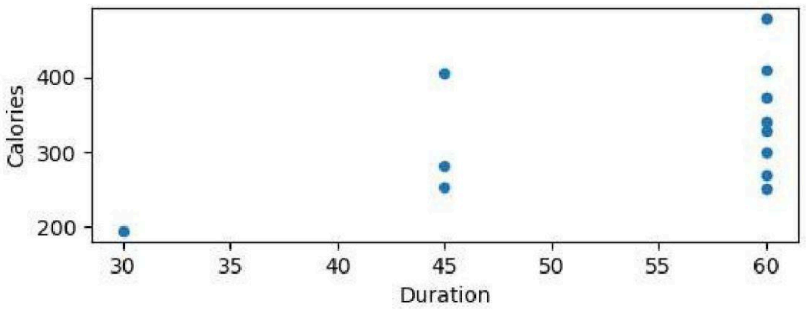
```
df.plot()
```



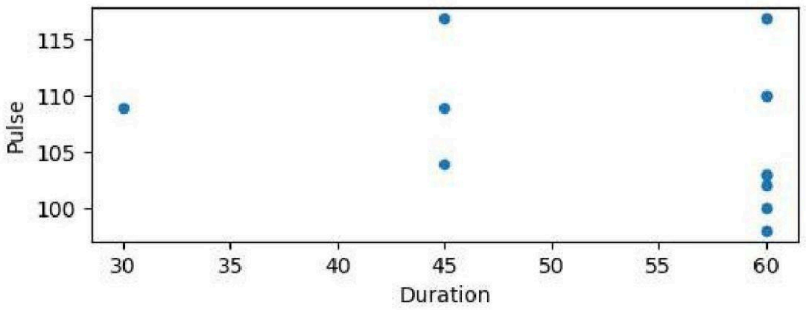
Scatter Plot

syntax: plot(kind='scatter',x,y) or plot.scatter(x,y)

```
df.plot(kind = 'scatter',
     x='Duration',y='Calories')
```



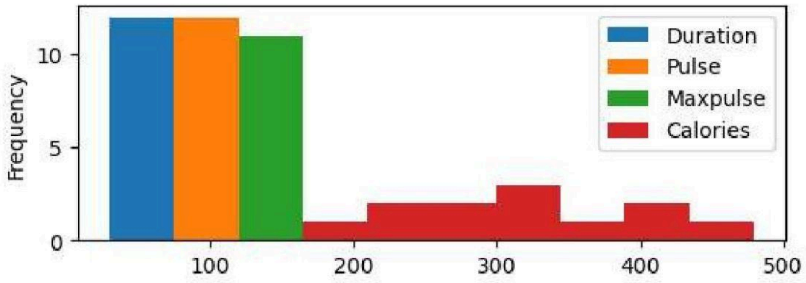

```
df.plot.scatter(x='Duration',y='Pulse')
```



Histogram

syntax: plot(kind='hist') or plot.hist()

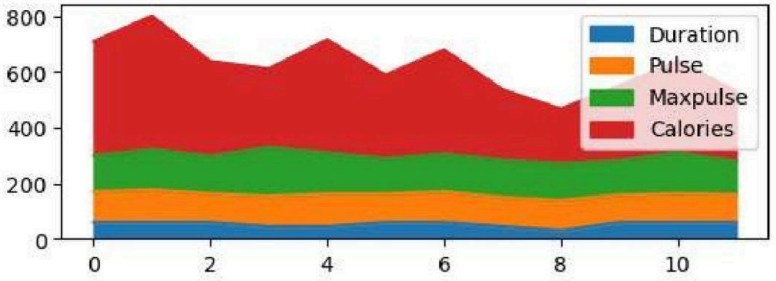
```
df.plot(kind = 'hist')

# df.plot.hist()
```



Area Plot

syntax: plot(kind='area') or plot.area()

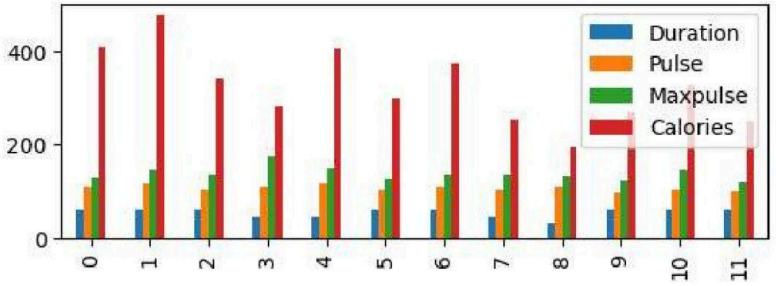
```
df.plot.area()
```



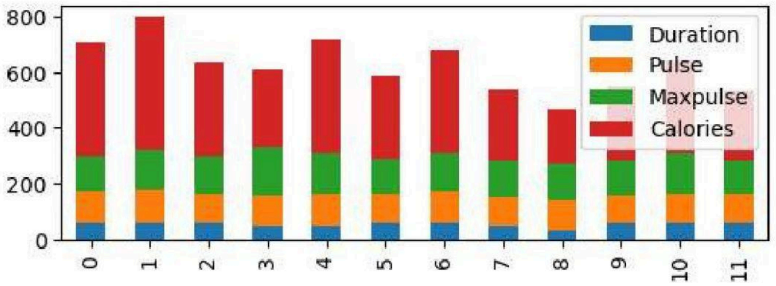
Bar Chart

syntax: plot(kind='bar') or plot.bar()
barh() for horizontal bar

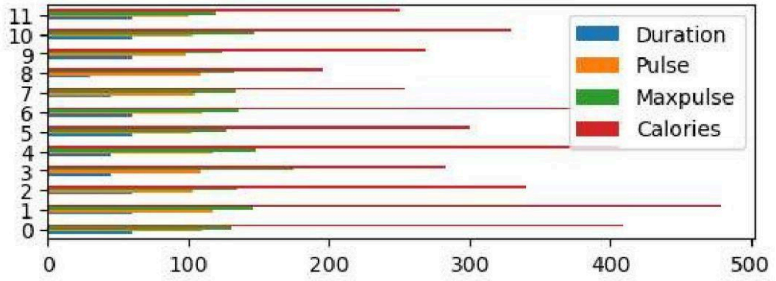
```
df.plot.bar()
```



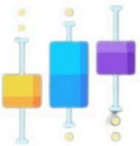
```
df.plot.bar(stacked=True)
```




```
# horizontal bar
df.plot.barh()
```

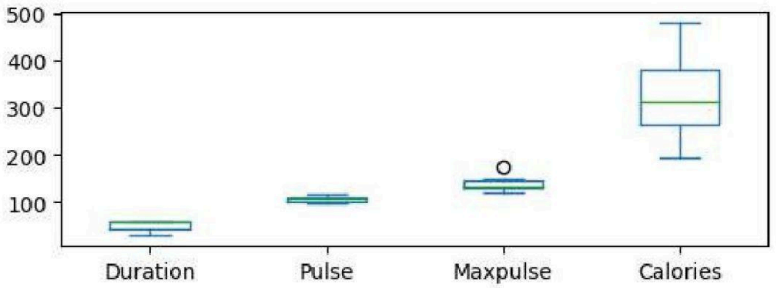


Box plot



syntax: plot(kind='box') or plot.box()

```
df.plot.box()
```

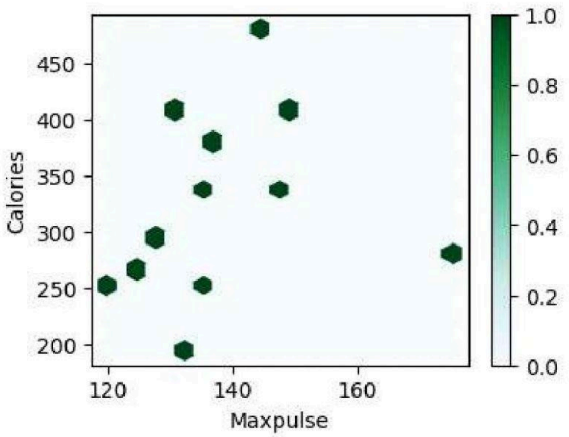


hexagonal binning plot

plot.hexbin()

```
df.plot.hexbin(x='Maxpulse',y='Calories',
               gridsize=18)
```

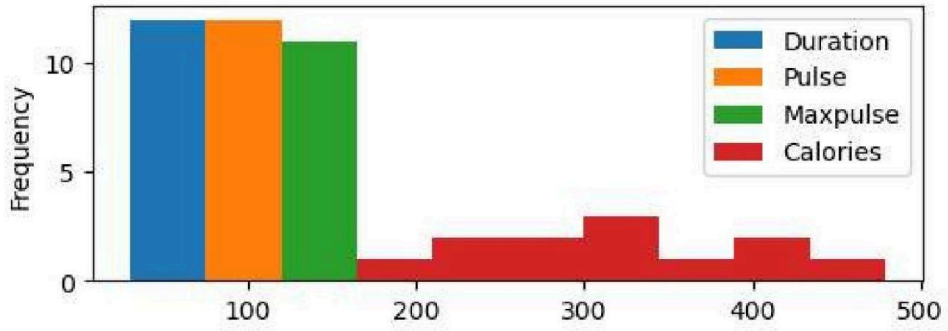
```
<Axes: xlabel='Maxpulse', ylabel='Calories'>
```



histogram plot

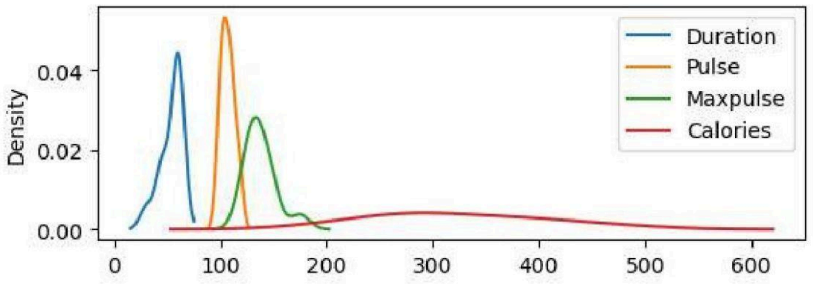
plot.hist()

```
df.plot.hist()
```



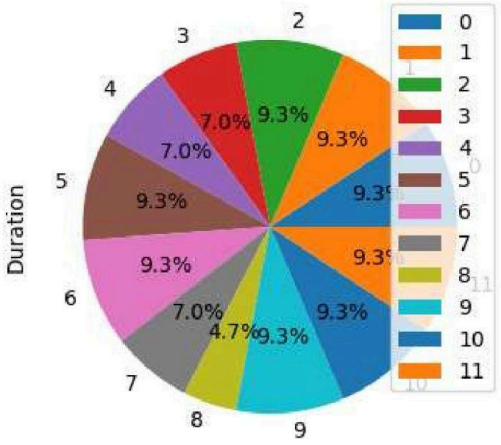
Density Estimate plot } plot.kde() or plot.density() kernel density estimate charts

```
df.plot.kde()
```



Pie plot pie.plot()

```
df.plot.pie(y='Duration',autopct='%1.1f%%')
```



Save the plot as an image | savefig()

```
df.plot()
plt.savefig('lineplot.png')
```

Chương III. HỌC MÁY CƠ BẢN

Chương IV. XỬ LÝ DỮ LIỆU CHUỖI THỜI GIAN VÀ DỮ LIỆU BẢNG