**THƯ VIỆN MATPLOTLIB - PYTHON**

**I. Giới thiệu thư viện Matplotlib**

- Vẽ biểu đồ là một trong những kỹ năng cần thiết và rất quan trọng, đặc biệt là những bạn muốn phát triển sự nghiệp trong lĩnh vực Data Analyst hay Data Science.

- **Matplotlib** là một thư viện Python dùng để trực quan hoá dữ liệu thông qua các biểu đồ và đồ thị. Thư viện hỗ trợ nhiều loại biểu đồ như: đồ thị đường, biểu đồ cột, biểu đồ tròn, đồ thị phân tán,… Người dùng có thể dễ dàng tuỳ chỉnh màu sắc, nhãn, tiêu đề và xuất đồ thị ra nhiều định dạng hình ảnh khác nhau. Matplotlib thường được sử dụng trong khoa học dữ liệu, học máy và phân tích số liệu.

- Mặc dù hiện này có khá nhiều thư viện cũng được đánh giá tốt như Seaborn, Hiplot (của Facebook) trong Python hay ggplot trong R, Matplotlib tuy cũ nhưng vẫn được sử dụng rộng rãi hơn cả cho mục đích Data Visualization.

**II. Cài đặt và khai báo thư viện Matplotlib**

- Để sử dụng thư viện Matplotlib, bước đầu tiên sau khi thiết lập môi trường Python là chúng ta phải cài đặt package. Sử dụng pip install matplotlib để cài đặt version mới nhất.

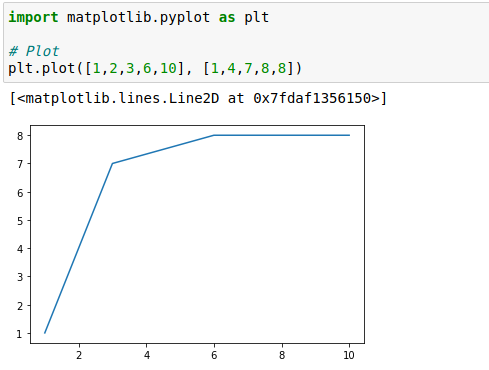
- Sau đó ta import thư viện bằng dòng code:

import matplotlib.pyplot as plt

**III. Các thao tác cơ bản với thư viện Matplotlib**

**1. Tạo biểu đồ đơn giản:**

Sau đây ta vẽ một biểu đồ đơn giản với phương thức **plot()**, mặc định nó sẽ vẽ biểu đồ đường với các tham số tối thiểu là list các toạ độ x, list toạ độ y (lưu ý 2 list này phải có độ dài bằng nhau). Ví dụ:



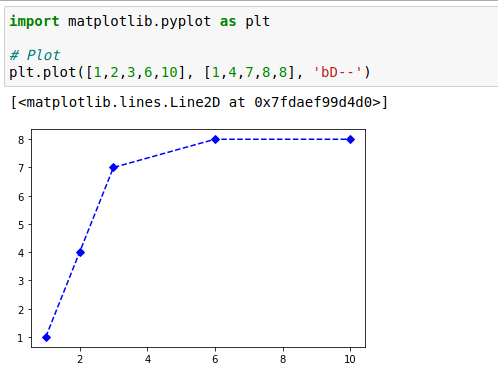
Để thay đổi một chút, ta có thể thêm tham số về định dạng vào, chẳng hạn như:

- Tham số màu: r (red), g (green), b (blue), y (yellow),…

- Định dạng đường: - (đường thẳng), -- (đường nét đứt), nếu không truyền tham số là không vẽ đường.

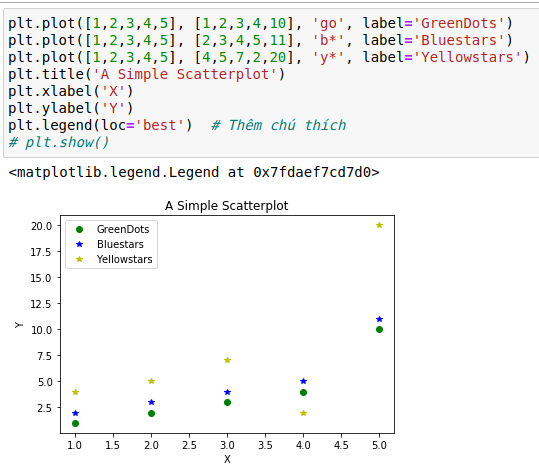
- Định dạng đỉnh: \* (đỉnh hình bông tuyết), D (đỉnh hình thoi), s (đỉnh hình vuông), go (chấm tròn),…

Ví dụ thêm tham số b tức là màu blue, D tức là đỉnh hình thoi và -- tức là cạnh nét đứt:



**2. Vẽ nhiều đối tượng trong cùng một biểu đồ:**

Ta sẽ thêm title cho biểu đồ, title cho trục toạ độ và thêm chú thích, ví dụ:



Đoạn code trong hình ta thấy:

- Tham số label trong hàm **plt.plot()** là để đặt tên cho một điểm, tên này sẽ xuất hiện trong chú thích.

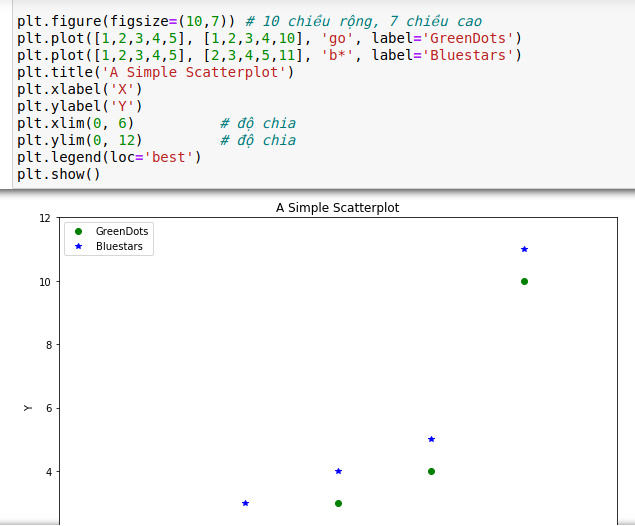
- **plt.xlabel** hay **plt.ylabel** là để đặt tên cho trục x và trục y.

- **plt.title** là để đặt tên cho biểu đồ.

- **plt.legend** sẽ đi tìm các thành phần chứa tham số label và đưa vào chú thích.

**3. Thay đổi kích thước của biểu đồ:**

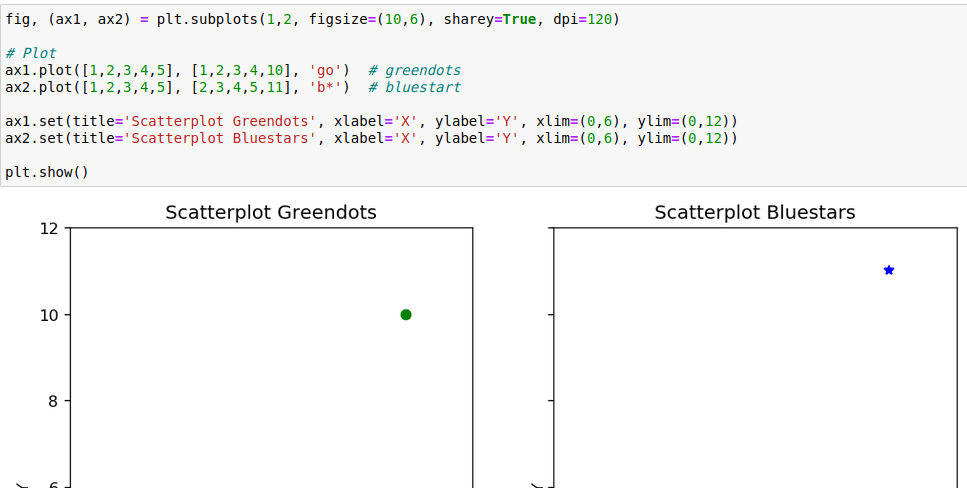
Ta sử dụng phương thức **figure()** để thay đổi kích thước của biểu đồ, tham số truyền vào là **figsize**. Ví dụ:



Ở ví dụ trên tham số truyền vào là figsize = (10, 7) trong đó 10 là tỉ lệ độ dài trục x và 7 là tỉ lệ độ dài trục y.

Ta cũng thấy có **plt.xlim** và **plt.ylim**. Hai phương thức này dùng để xác định hệ số chia trên trục tương ứng.

**4. Vẽ hai biểu đồ cạnh nhau:**



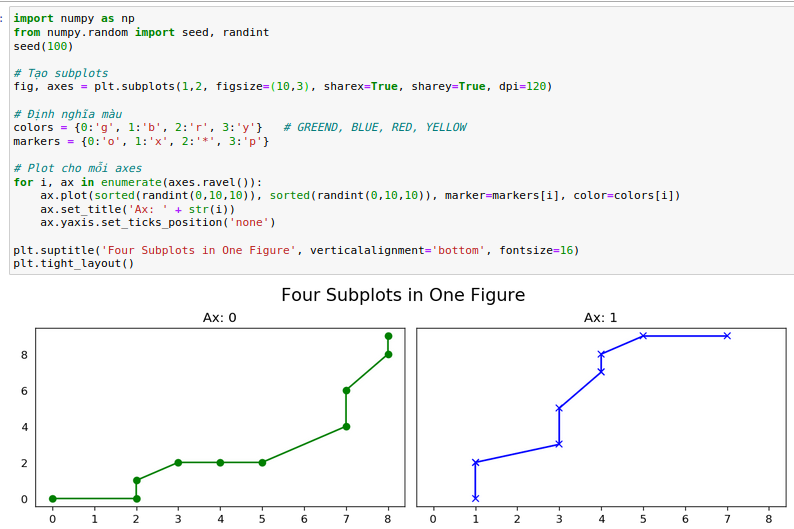
Đầu tiên ta sử dụng phương thức **subplots()** để tạo ra 2 biểu đồ đó. Sử dụng subplots() các đối tượng **axes** sẽ được trả về. Sau đó bạn có thể set các thuộc tính của từng đối tượng này sử dụng **ax1.set** và **ax2.set** như trên (mỗi đối tượng axes định nghĩa cho một biểu đồ).

Tham số (1, 2) trong hàm subplots có thể hiểu là 1 hàng, 2 cột, tương ứng bạn sẽ vẽ ra 2 biểu đồ cạnh nhau. Nếu chẳng hạn bạn thay (1, 2) bằng (2, 2),, bạn sẽ được 4 biểu đồ.

Tham số **dpi** là độ phân giải của hình vẽ.

Tham số **sharey = True** cho phép chia sẻ thuộc tính giữa 2 trục x và y.

**5. Vẽ trên nhiều biểu đồ sử dụng vòng lặp:**



Đây là một kỹ thuật khá hữu ích, nhất là khi bạn muốn vẽ biểu đồ từ một cơ sở dữ liệu hay file dữ liệu được.

Đầu tiên bạn tạo một subplot với số biểu đồ mà bạn muốn vẽ lên, sau đó các đối tượng axes được tạo ra và bạn sẽ sử dụng các đối tượng này trong vòng lặp for để đưa dữ liệu vào các biểu đồ tương ứng.

**IV. Các thao tác nâng cao với thư viện Matplotlib**

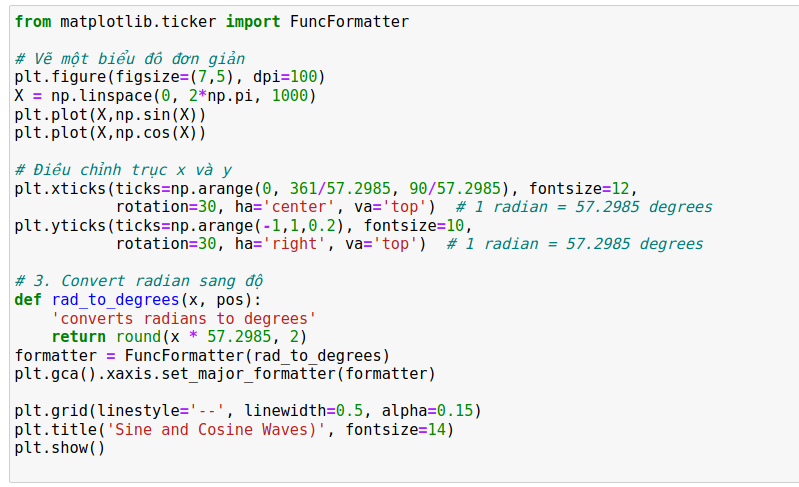
import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

**1. Điều chỉnh view trục x và y:**

Sử dụng phương thức **plt.xticks()** để thực hiện thay đổi trục x và **plt.yticks()** để thực hiện những thay đổi trên trục y. Ví dụ:



Ta sử dụng **plt.figure()** để đặt lại kích thước, độ phân giải của biểu đồ và sử dụng **plt.plot()** để vẽ lên biểu đồ.  
Phương thức **xticks()** và **yticks()** là tương tự nhau nên mình chỉ hướng dẫn xticks() như là một ví dụ.

Các tham số: Phương thức này có 2 tham số chính là **ticks** và **label**:

**- ticks:** tham số này truyền vào một list các giá trị, các giá trị này chính là các điểm được ghi số trên trục x, hay gọi tắt là number label. Trong ví dụ trên, với 1 radian = 57.2985 độ, như vậy ta sẽ được list các giá trị tương ứng bằng [0.0, 1.57, 3.14, 4.71, 6.28], hay nó cũng bằng với [0, pi/2, pi, 3\*pi/2, 2\*pi].

**- label:** tham số này là list các nhãn mà bạn muốn đặt thay cho giá trị của ticks ở trên, nếu không có nhãn mặc định là các giá trị của ticks.

Ngoài ra, các tham số khác có thể thêm vào là các tham số của đối tượng **text** mà bạn có thể thêm vào như:

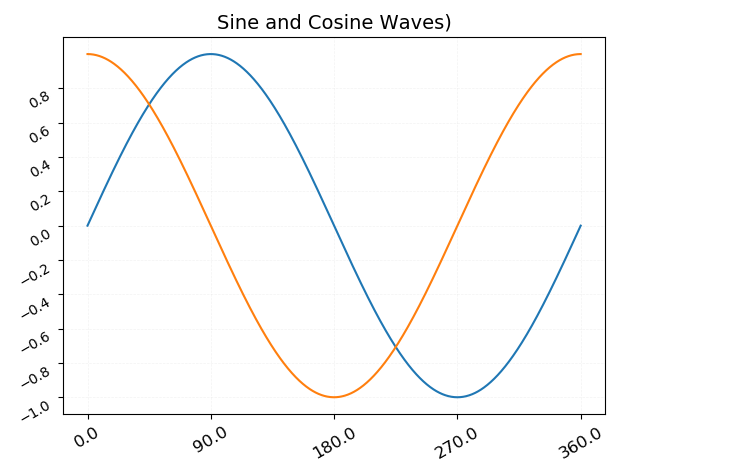
**- fontsize:** tham số này là kích thước font của number label.

**- rotation:** góc lệch của text so với trục thẳng y.

Chuyển radian sang độ: Sử dụng **FuncFormatter** để thêm vào một hàm tự định nghĩa để chuyển radian sang độ.

Phương thức **plt.gca()** trả về đối tượng **axes** hiện tại, tức là biểu đồ hiện tại.

Để vẽ lưới hiển thị dưới nền, sử dụng phương thức **plt.grid()**. Và dưới đây là kết quả:



**2. Các tham số mặc định rcParams và Styles:**

Để kiểm tra các tham số mặc định ta sử dụng phương thức **mpl.rc\_params()**.

Muốn thay đổi các tham số ta sử dụng phương thức **update()**, chẳng hạn:

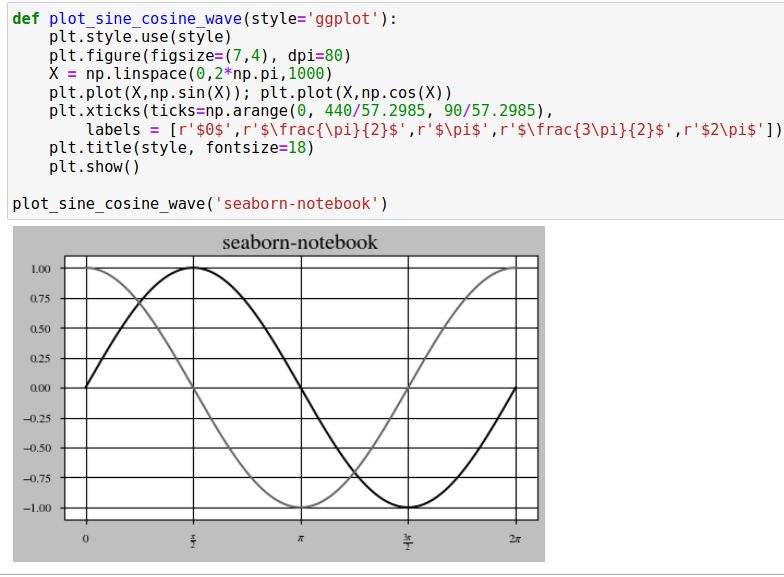
mpl.rcParams.update({'font.size': 18, 'font.family': 'STIXGeneral', 'mathtext.fontset': 'stix'})

Để kiểm tra các style có sẵn ta sử dụng **plt.style.available**.

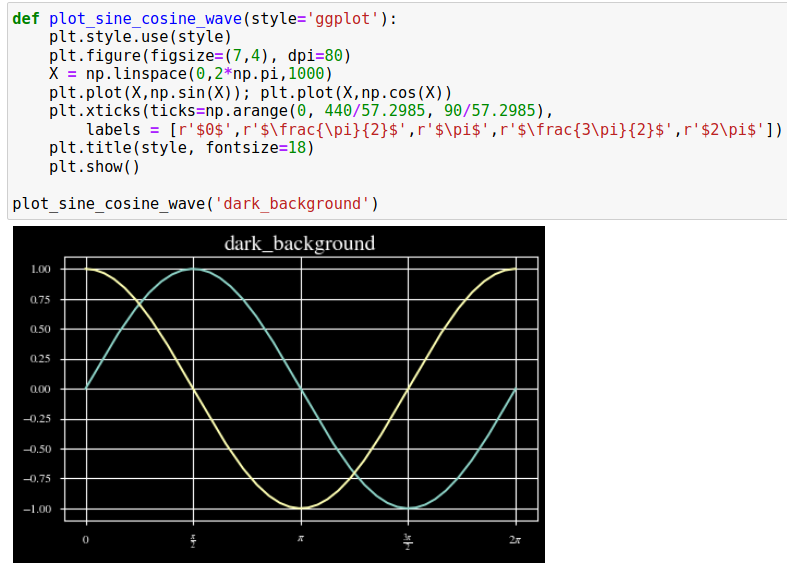
Thay đổi style bằng phương thức **plt.style.use(style\_name)**.

Dưới đây là ví dụ về các style khác nhau:

**- seaborn-notebook:**

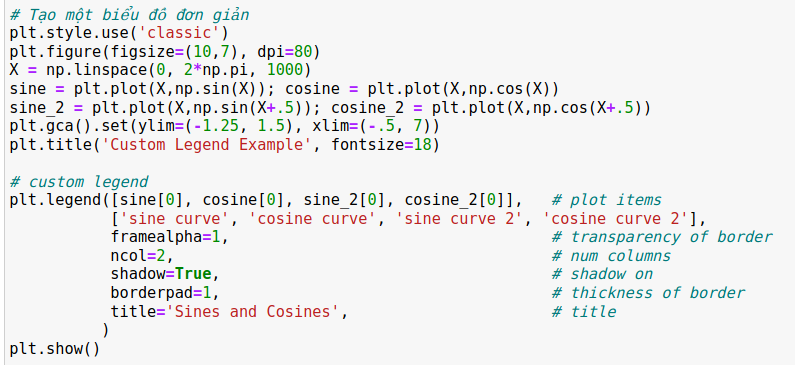


**- dark\_background:**



**3. Điều chỉnh chú thích (legend):**

Nếu phương thức **plt.legend()** giúp các bạn chú thích những thành phần được vẽ trong biểu đồ thì ở phần này, mình sẽ hướng dẫn các bạn custom khung chú thích này.



Nếu bạn truyền vào title, mặc định legend sẽ tìm các đối tượng được vẽ và chú thích label của chúng. Nếu bạn không muốn chú thích toàn bộ hãy chỉ vào những đối tượng mà bạn muốn chú thích, như trong ví dụ trên là sine, cosine, sine\_2, cosine\_2 và list các label tương ứng của chúng là [‘sine curve’, ‘cosine curve’, ‘sine curve 2’, ‘cosine curve 2’].

Ngoài ra các tham số khác như:

**- framealpha:** Độ trong suốt của khung chú thích, có thể kiểm tra giá trị này bằng **mpl.rcParams["legend.framealpha"]**.

**- loc:** Vị trí của khung chú thích trong biểu đồ, các giá trị như ‘best’, ‘center’, ‘left center’, ‘right centre’, ‘lower center’, ‘lower left’,…

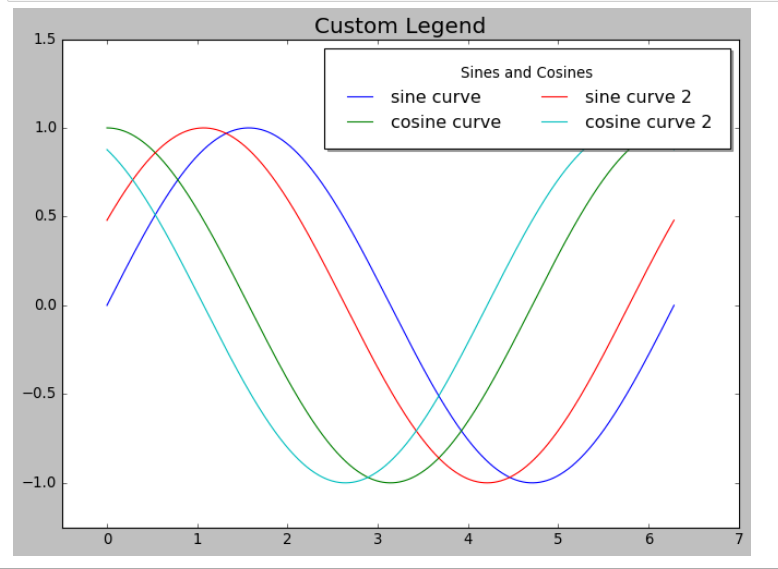
**- ncol:** Số cột (trong ví dụ chú thích thành 2 cột).

**- shadow:** True hay False. Viền có đổ bóng hay không.

**- title:** Title của khung chú thích.

**- borderpad:** Khoảng cách từ chú thích đến viền.

Và đây là kết quả của ví dụ trên:



**4. Chú thích trực tiếp trong hình:**

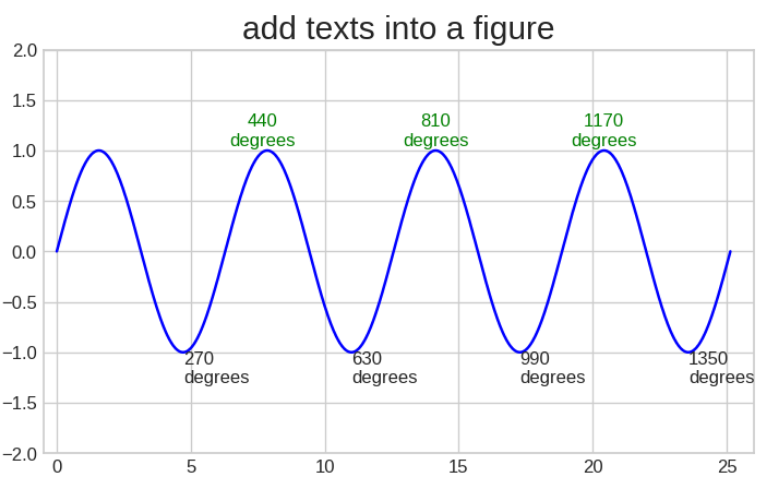
Việc thêm những dòng chữ chú thích vào trực tiếp trong hình sẽ giúp biểu đồ của bạn trở nên rõ ràng và dễ hiểu hơn. Bạn có thể làm được với **plt.text()**.

Hai tham số chính là toạ độ và nội dung đoạn text, ngoài ra bạn có thể thêm vào các tham số như màu sắc hay căn lề.



Trong ví dụ, ta sử dụng vòng lặp for để xác định vị trí và nội dung của đoạn text mà ta muốn vẽ lên biểu đồ.

Kết quả như sau:

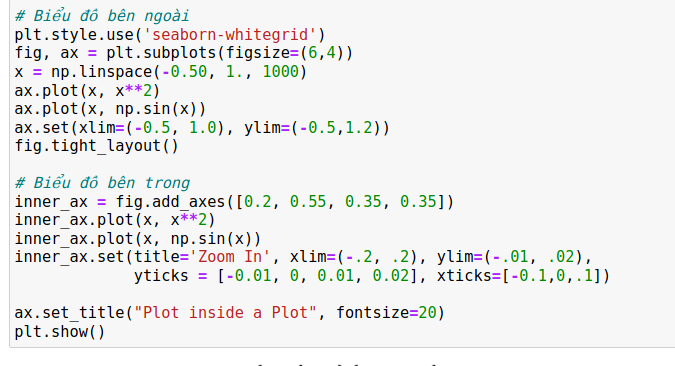


**5. Vẽ biểu đồ bên trong biểu đồ:**

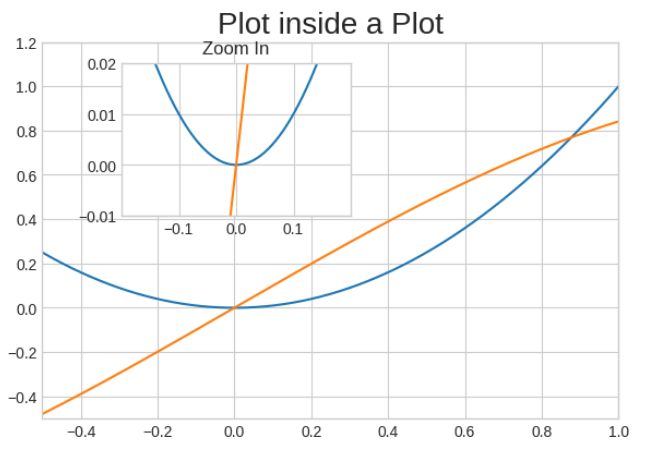
Sử dụng phương thức **add\_axes()** của đối tượng **figure**.

Đầu tiên ta tạo một figure và một axes sử dụng **plt.subplots()**.

Sau đó, bạn có thể vẽ một biểu đồ nhỏ trong biểu đồ này như trong ví dụ dưới đây:



Kết quả của ví dụ này như sau:



**6. Vẽ biểu đồ bên trong biểu đồ:**

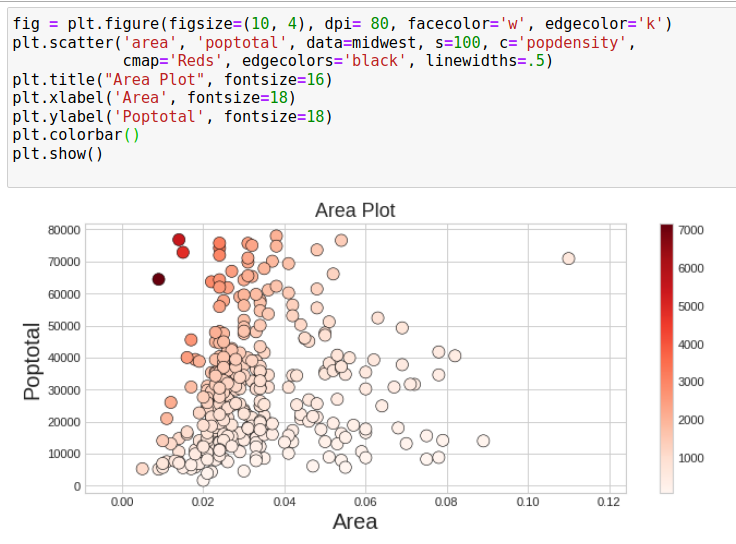
Ta sẽ sử dụng phương thức **plt.scatter()**. Một vài tham số được sử dụng như:

**- x, y:** list các toạ độ x và y của điểm dữ liệu (hoặc nếu bạn có một mảng dữ liệu hãy truyền nó qua tham số data như ví dụ bên dưới).

**- s:** kích thước của điểm dữ liệu.

**- c:** màu của điểm dữ liệu.

**- edgecolors:** màu viền của điểm dữ liệu.



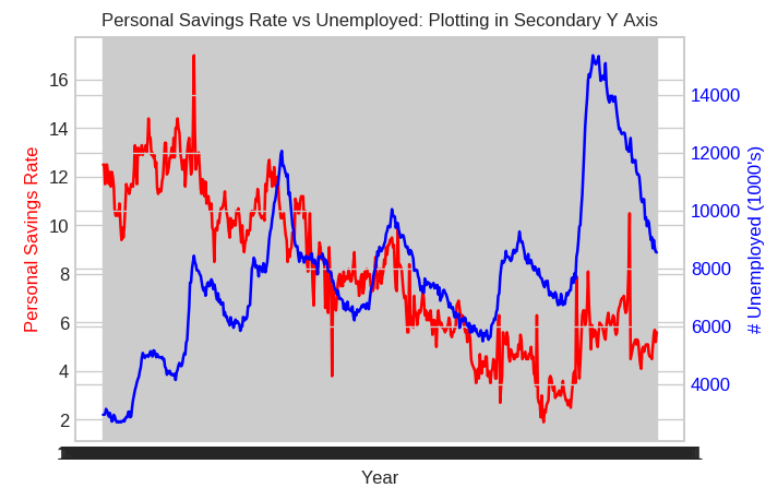
Ở ví dụ này ta sử dụng thêm phương thức **plt.colorbar()** để có được thanh bên phải trong kết quả, chú thích mức độ màu sắc.

**7. Vẽ một trục x và hai trục y ở hai bên:**

Để làm được điều này ta sử dụng phương thức **twinx()** để tạo ra một đối tượng **axes** chia sẻ chung trục toạ độ x. Lúc này, axes được tạo ra có trục y ở vị trí đối lập.



Kết quả ví dụ trên như sau:



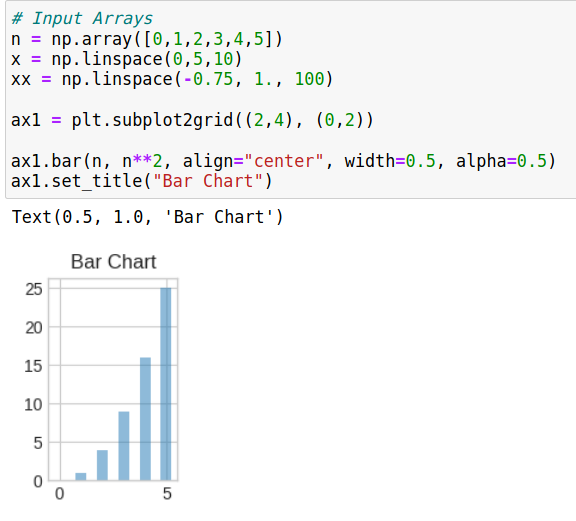
**8. Histograms, Boxplots và Time Series:**

Ở phần này mình sẽ trình bày một số kiểu vẽ biểu đồ.

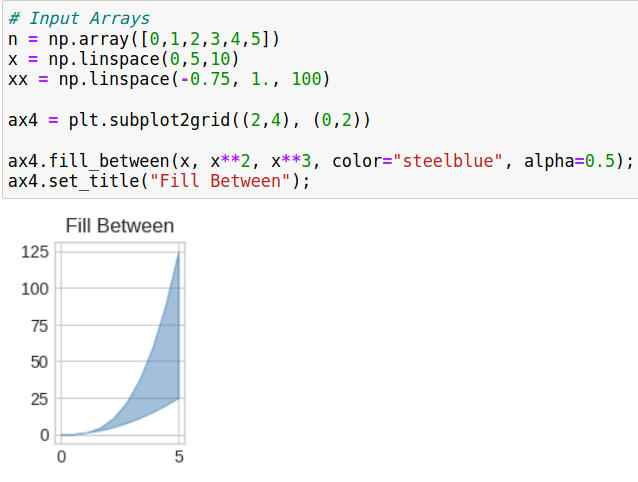
Chúng ta sẽ làm quen với **subplot2grid**, tương tự như subplot nhưng biểu đồ được tạo trên nền lưới sẽ giúp cho việc vẽ các cột hay đường.

Dưới đây là các dạng biểu đồ có thể vẽ với Matplotlib:

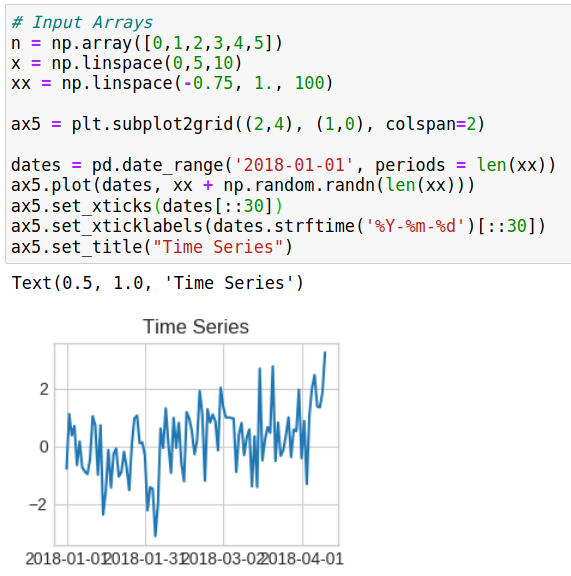
**- Bar Chart:**



**- Fill Between:**



**- Time Series:**



**- Histogram:**

