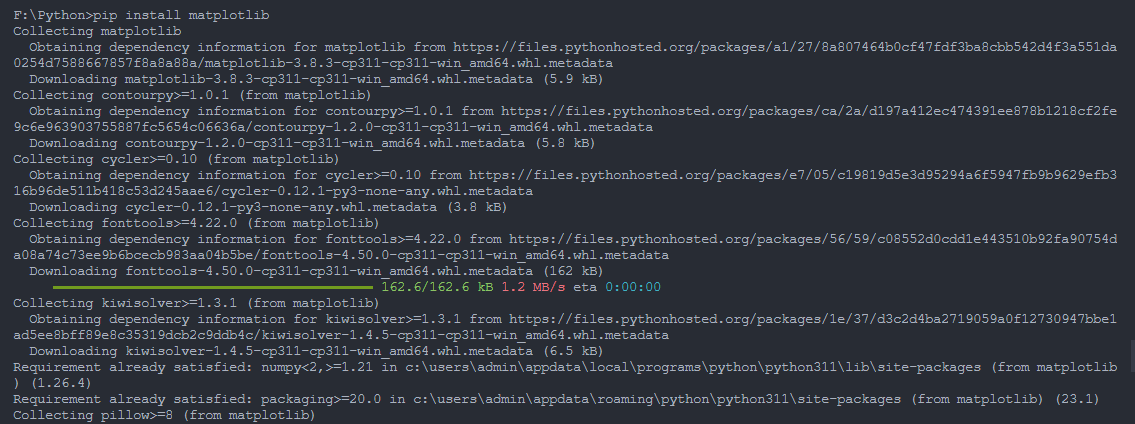
**Nguyễn Nguyên Hảo**

**Matplotlib**

Chúng ta đã đề cập rằng việc trực quan hóa dữ liệu là rất quan trọng trong data sience. Nó cung cấp một cái nhìn tổng quan và giúp chúng ta phân tích dữ liệu và đưa ra kết luận. Do đó, chúng ta sẽ học về Matplotlib, thư viện mà chúng ta sẽ sử dụng để vẽ đồ thị và trực quan hóa dữ liệu

+ INSTALLING MATPLOTLIB



+ PLOTING MATHEMATICAL FUNCTIONS

Import module



Giống với module numpy chúng ta cũng sử dụng một alias cho pylot là plt

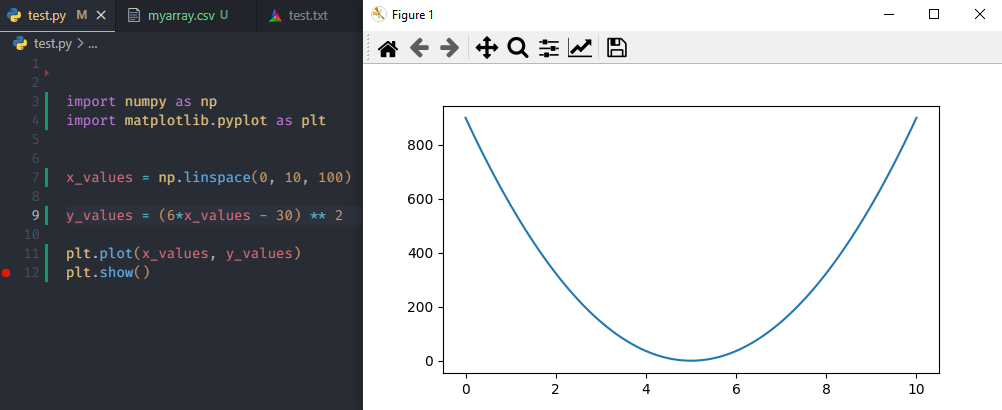
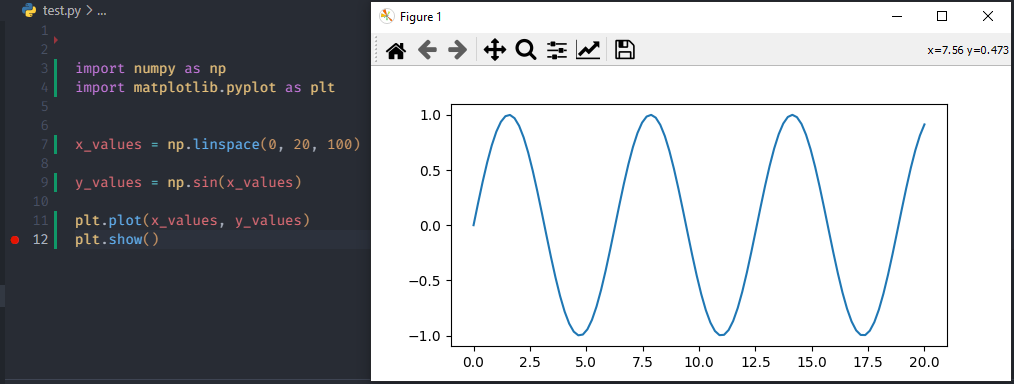
Xem xét đến việc vẽ đồ thị của một hàm số, chúng ta cần x\_value làm đầu vào và y\_value lam đầu ra:



Chúng ta sử dụng hàm đã biết là linspace để tạo ra 100 giá trị giữa 0 và 20. Bây giờ chúng ta sẽ tạo giá trị đầu ra y\_values. Chúng ta sẽ ví dụ với hàm sin

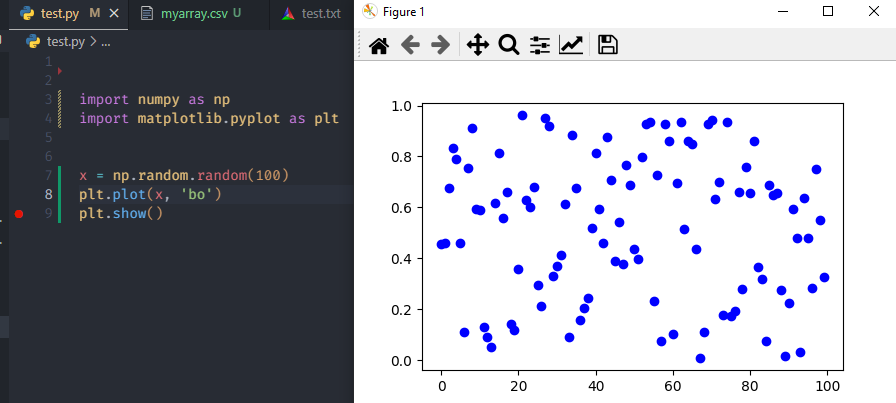


x\_value sẽ là 1 mảng 100 phần tử đơn lẻ và chúng ta sẽ apply hàm sin cho từng phần tử đơn lẻ đó. Sau đó chúng ta sử dụng hàm plot để vẽ đồ thị:



+ VISUALIZING VALUES

Thay vì việc vẽ đồ thị hàm số, chúng ta có thẻ trực quan hóa dưới dạng các giá trị đơn lẻ

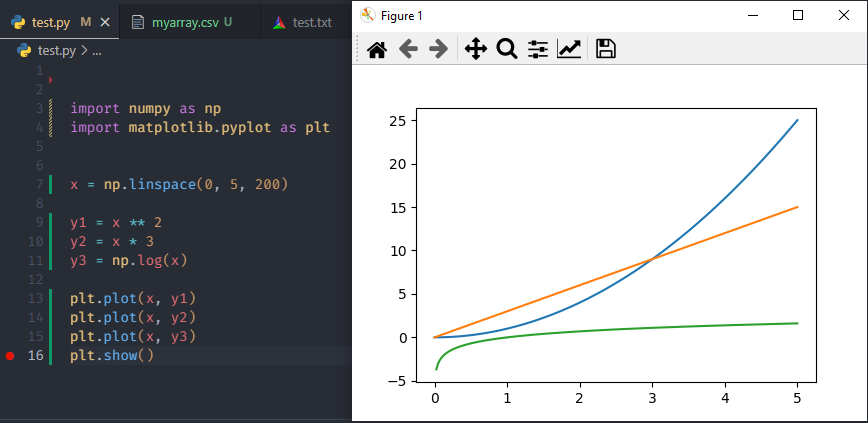


Chúng ta generating random 100 giá trị trong khoảng từ 0 đến 10. Chúng ta vẽ các giá trị này bằng các điểm mày xanh.

Trong hàm plot chúng ta định nghĩa tham số thứ 2 ‘bo’. b: blue, o: dots

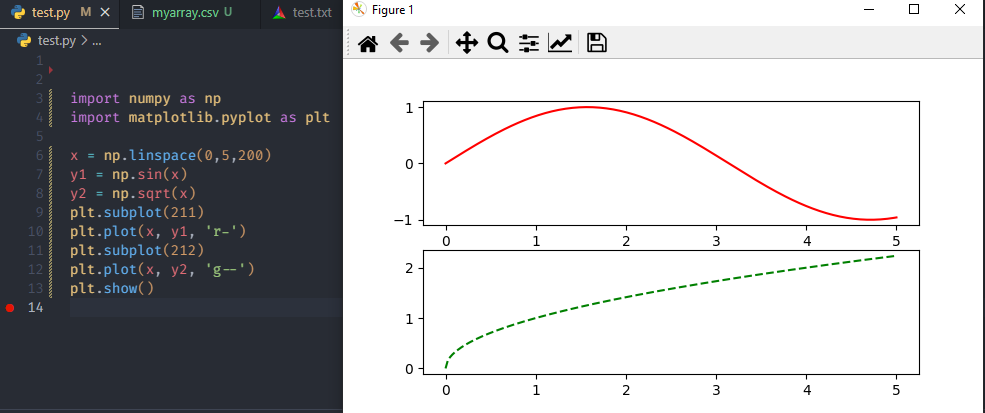
+ MULTIPLE GRAPHS

Đồ thị của chúng ta không giới hạn ở một đồ thị mà chúng ta có thể vẽ nhiều đồ thị, nhiều hàm khác nhau về màu sắc và hình dạng



+ SUBPLOT

Đôi khi chúng ta không muốn vẽ nhiều hàm số trên cùng một đồ thị. Vì lý do này, chúng ta gọi là subplots. Những đồ thị này sẽ được show trên cùng một cửa số tuy nhiên nó sẽ phân biệt với nhau



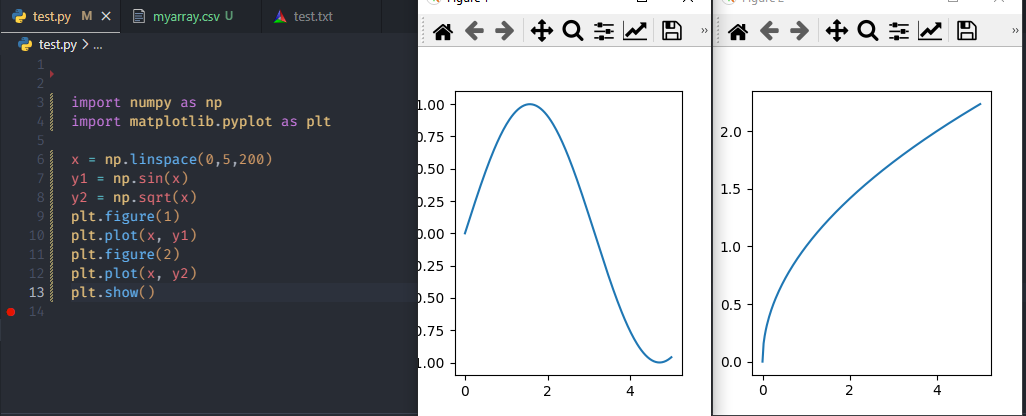
Hàm subplot(x), với tham số x = 211 có nghĩa là cửa số sẽ chia thành 2 hàng 1 cột và đồ thị sẽ nằm ở vị trí thứ nhất

‘r-’ : màu đỏ, ‘-‘ tượng trưng cho đường thẳng liên tục

‘g--’: màu xanh green, ‘--’ tượng trưng cho đường thẳng đứt quãng

+ MULTIPLE PLOTTING WINDOWS

Thay vì hiển thị trên 1 cửa số, chúng ta sử dụng hàm figures để hiện thị trên nhiều cửa sổ



+ PLOTTING STYLES

Matplotlib cung cấp cho chúng ta nhiều kiểu vẽ khác nhau để lựa chọn. Nếu bạn quan tâm đến cách chúng trông như thế nào khi chúng được áp dụng.

<https://matplotlib.org/3.1.0/gallery/style_sheets/style_sheets_reference.html>

from matplotlib import style

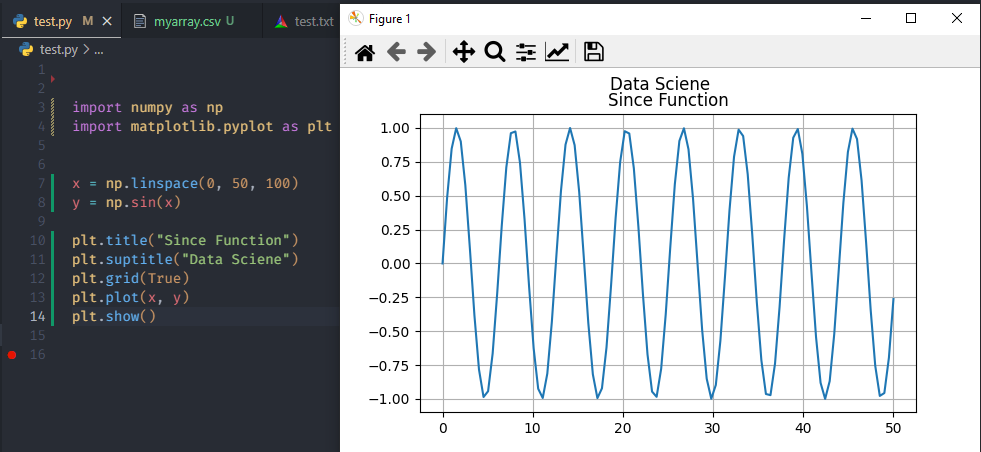
style.use(‘ggplot’)

Ở đây chúng ta áp dụng kiểu vẽ của ggplot.

+ LABELING DIAGRAMS

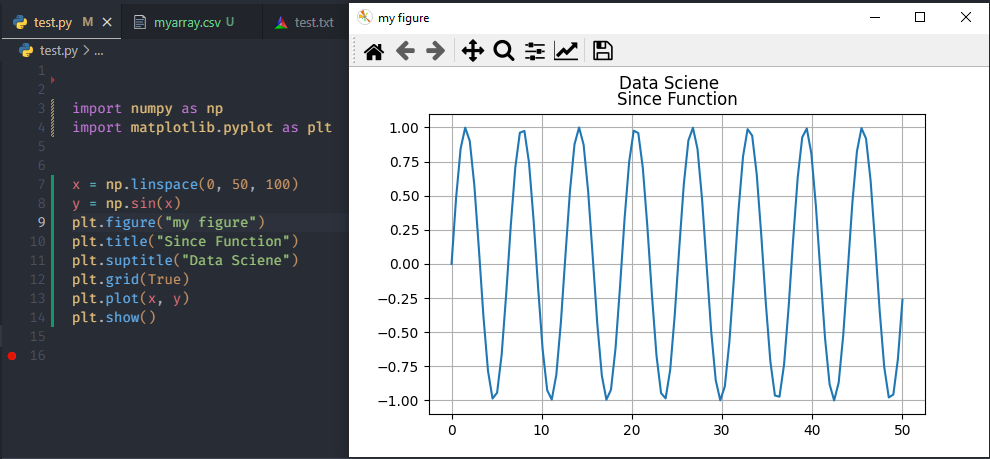
Để làm cho các đồ thị dễ hiểu, chúng ta cần gán nhẵn cho chúng một cách đúng đắn, chúng ta nên gán nhẵn cho các trục.

+ SETTING TITLES



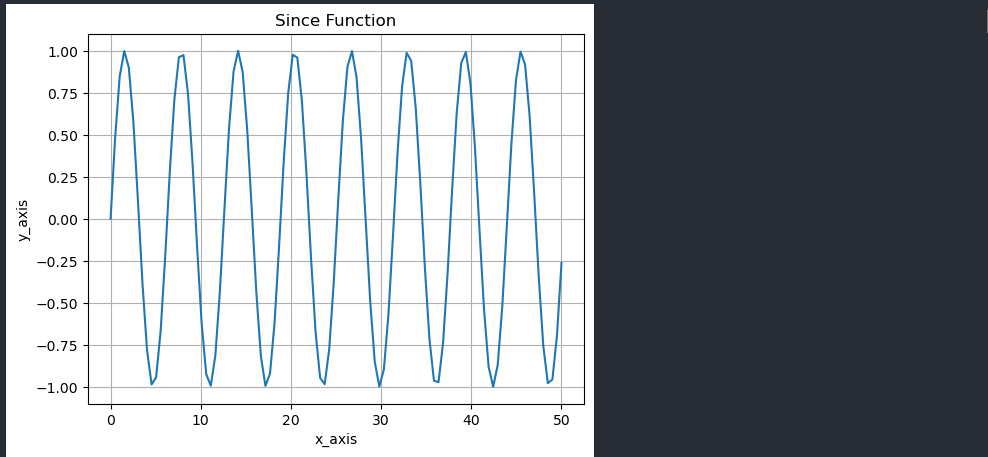
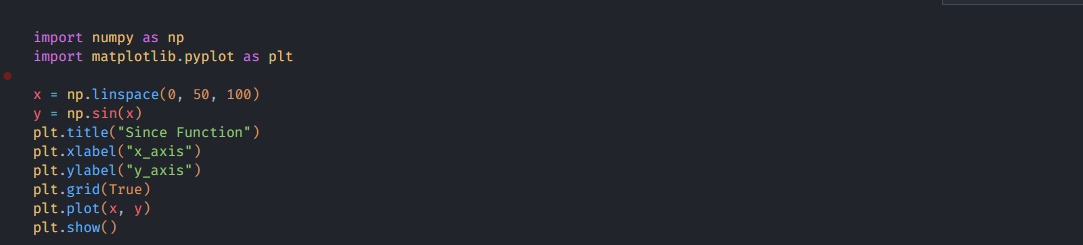
Hàm đầu tiên thêm một tiêu đề đơn giản vào đồ thị, và hàm thứ 2 thêm một tiêu đề phụ ở trung tâm phía trên. Hàm thứ 2 thêm một tiêu đề phụ ở trung tâm phía trên nó.

+ Hàm grid bật lưới của đồ thị



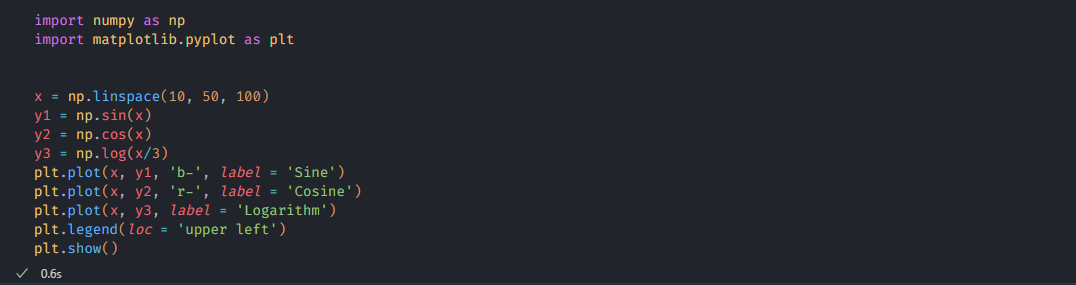
+ LABLE AXES

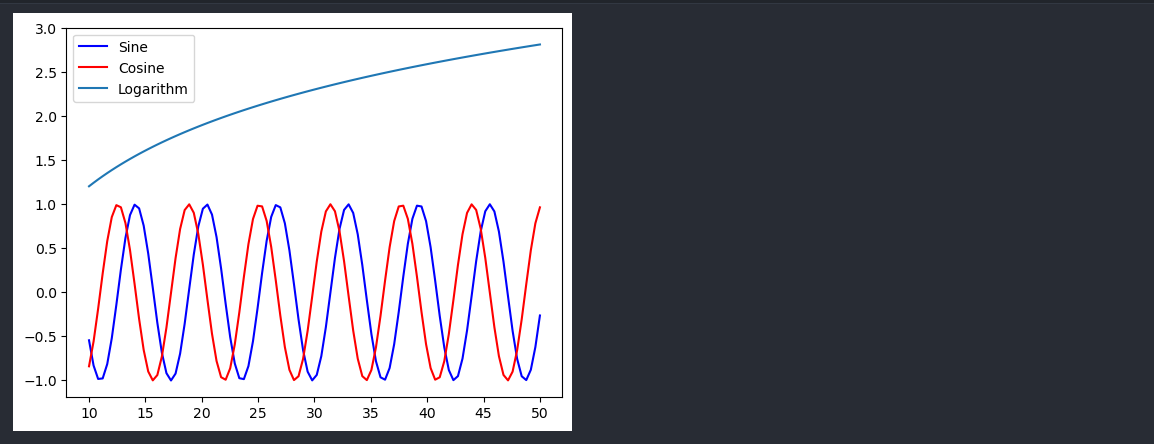
Để gán nhãn cho từng trục ta sử dụng hàm xlable để gán nhãn cho trục x và hàm ylable để gán nhãn cho trục y



+ LEGENDS

Khi đồ thị có nhiều biểu đồ và đối tượng, việc sử dụng chú thích là cần thiết để ghi nhãn cho từng phần tử, giúp làm cho mọi thứ thêm dễ đọc và dễ hiểu hơn



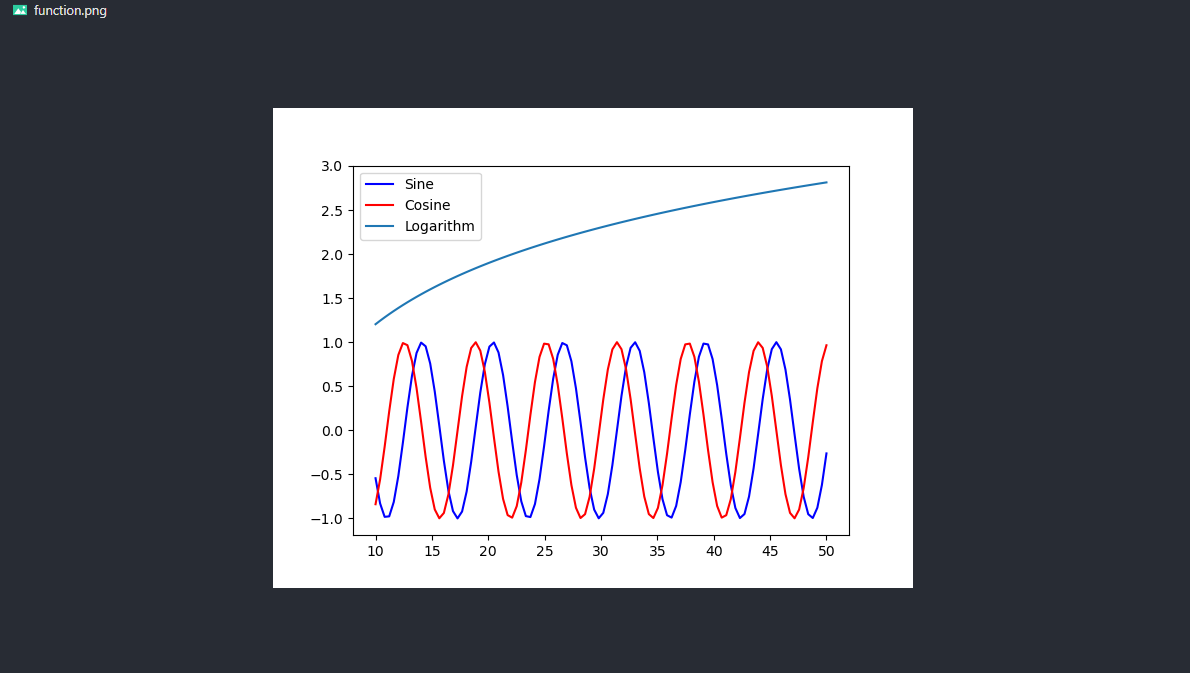


Vẽ tất cả đồ thị vào plot và thêm 1 nhãn cho nó. Chúng ta sử dụng hàm legend và để đặt chú thích và có thể đặt các chú thích vào chỗ mà mình mong muốn

Ở đây chúng ta sử dụng upper left ngoài ra có thể sử dụng upper right, lower left, lower right

+ SAVING DIAGRAMS

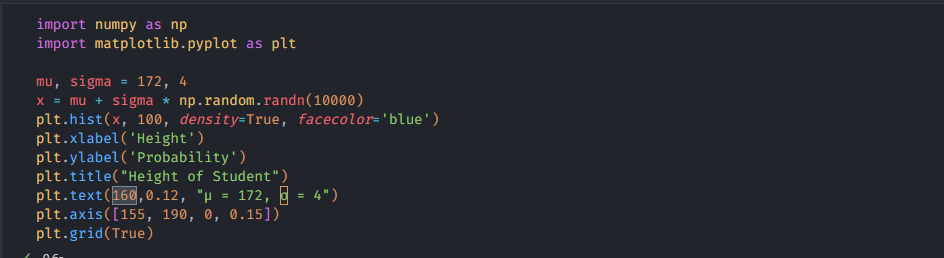
Để lưu lại đồ thị ta sử dụng hàm savefig

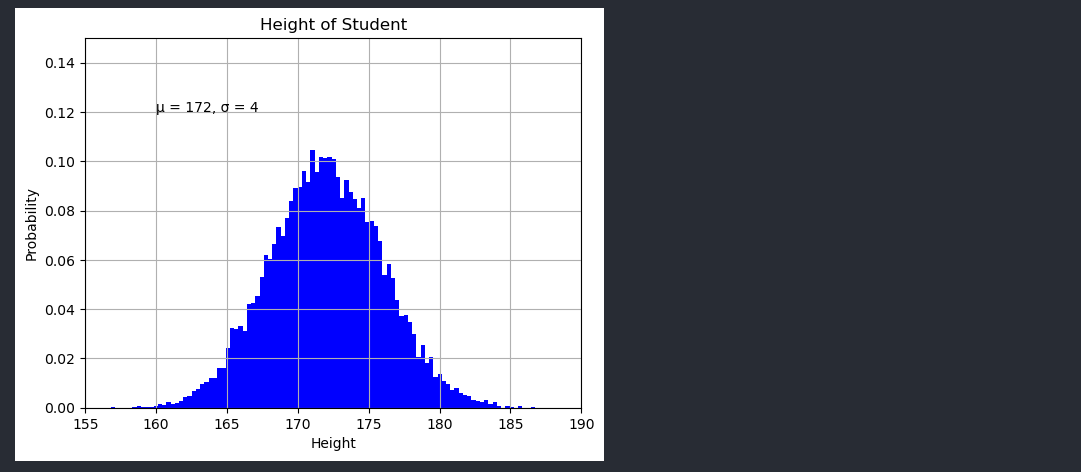


+ MATPLOTLIB PLOT TYPE

HISTOGRAMS

Histogram được gọi là biểu đồ biểu diễn phân phối của các giá trị số





Hàm hist được sử dụng để vẽ biểu đồ histogram. Tham số thứ nhất của hist là hàm số x, tham số thứ 2 là số ô cho histogram (100 Ô). Tham số density được đặt thành True để các giá trị được chuẩn hóa. Tức là tổng các giá trị y = 1 và chúng ta có thể coi chúng như phần trăm, tham số facecolor ở đây chúng ta đặt thành màu xanh

Trong biểu đồ histogram chúng ta sử dụng hàm plt.text() chúng ta đặt text này tại x\_value tại 160 và y\_value = 0.12

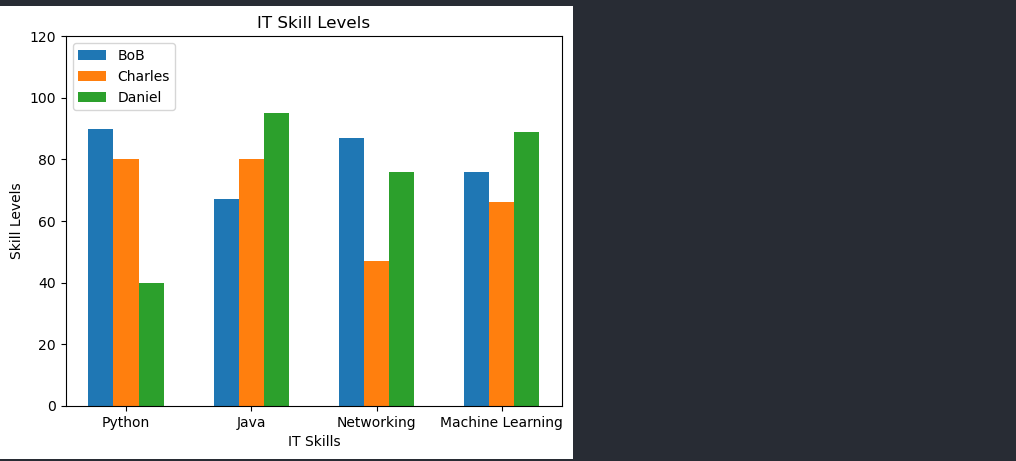
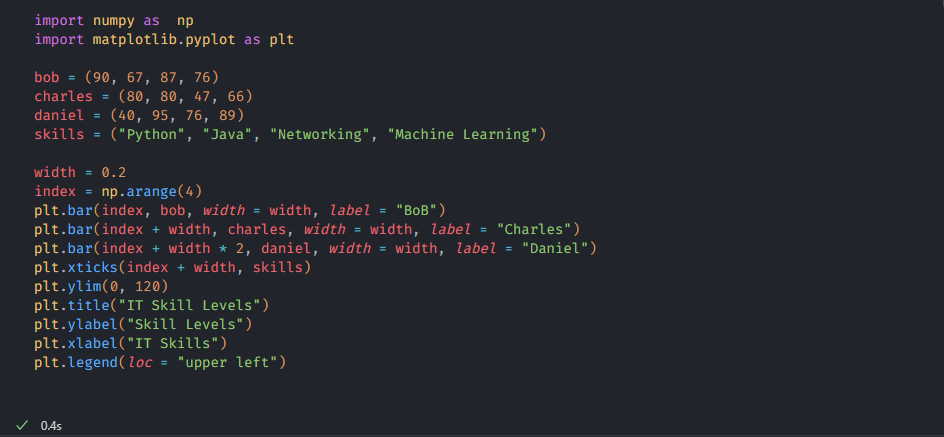
Sử dụng hàm plt.axis để set limit cho trục x và trục y. Tại đây trục x có giá trị từ [155, 190], trục y có giá trị từ [0, 0.15]

Hàm plt.grid(True) để đặt lưới cho đồ thị

+ BAR CHART

Để phục vụ cho việc thống kê, bar charts thường rất hữu ích. Trong ví dụ này chúng ta sẽ vẽ skill levels của 3 người khác nhau

Có 3 người là bob, charles, daniel. Các skills của họ được minh họa bằng một tuple gồm 4 phần tử tương ứng với 4 skills: python, java, networking, Machine Learning



Hàm plt.bar(x, height, width,…): Sử dụng để vẽ biểu đồ bar chart:

+ Tham số x: là vị trí của các cột trong biểu đồ, có thể là một list hoặc 1 tupple

+ Tham số height: là chiều cao của các cột cần vẽ

+ Tham số width: là độ rộng của mỗi cột

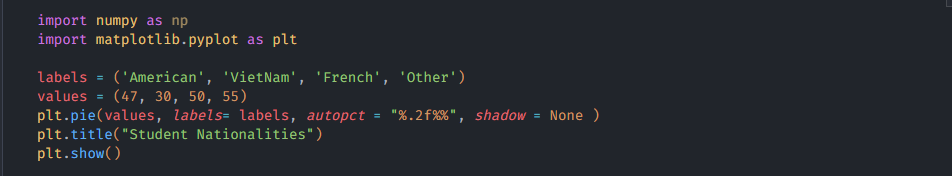
Hàm plt.xticks(x, y): Sử dụng để gán nhãn cho trục x

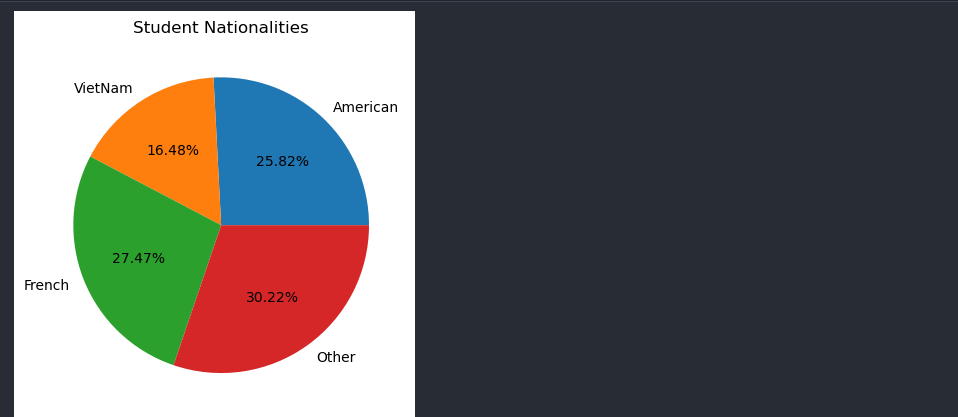
+ Tham số x: là vị trí của các phần tử cần gán nhãn trong trục x, có thể sử dụng list, tupple

+ Tham số y: Là tên nhãn, có thể sử dụng list, tupple

+ PIE CHART

Pie chart giống như hình chiếc bánh, sử dụng để vẽ những biểu đồ thống kê phần trăm

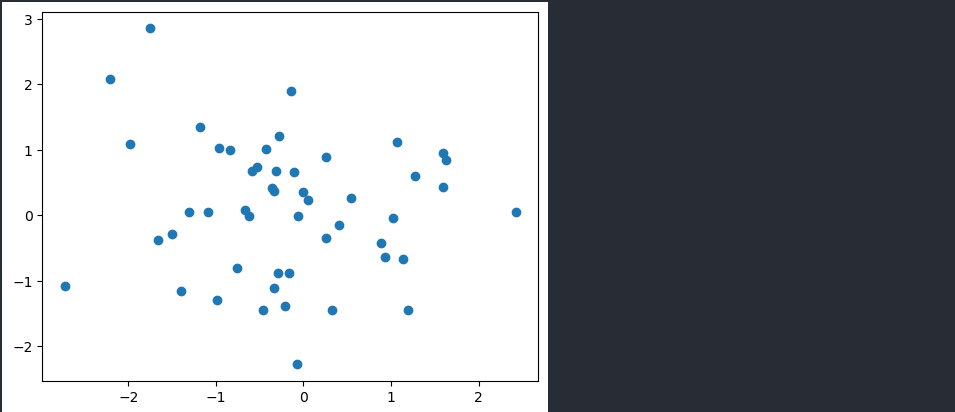
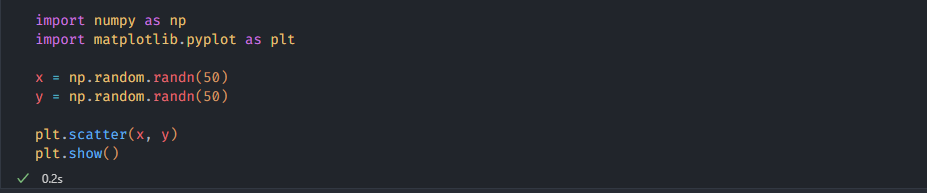




Sử dụng hàm plt.pie() để vẽ biểu đồ hình pie. Setup tham số autocpt = “%.2f%%” để tính toán giá trị phần trăm ra 2 chữ số sau dấu phẩy

+ SCATTER PLOTS

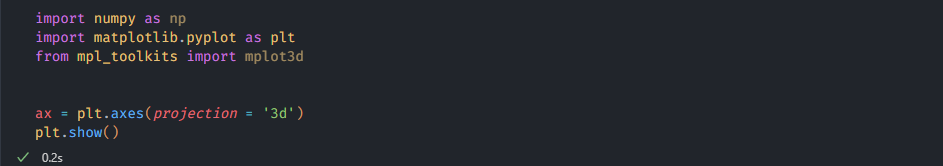
Scatter plots được sử dụng để trình bày dữ liệu 2 chiều sử dụng các dấu chấm

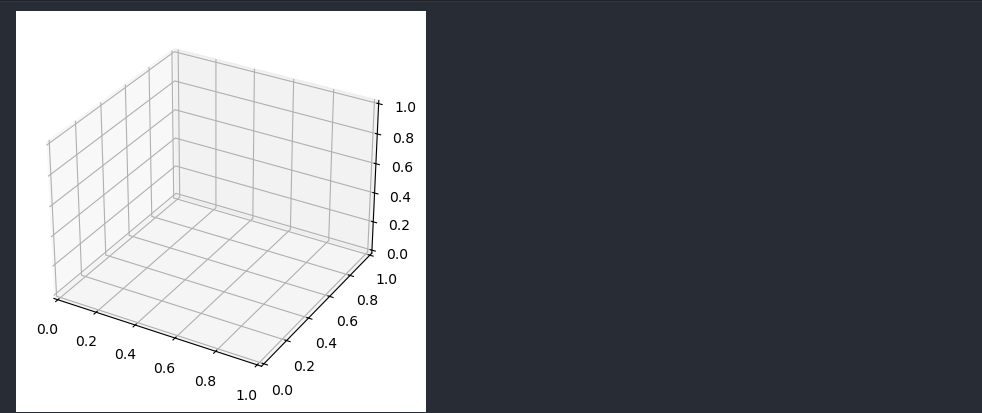


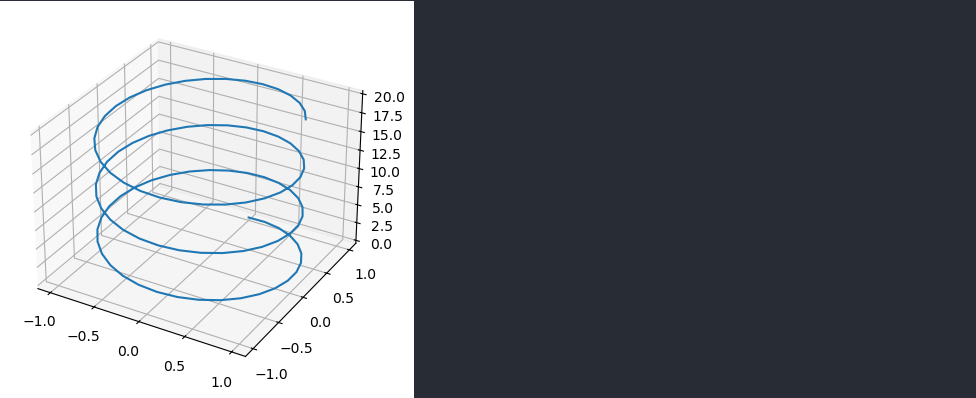
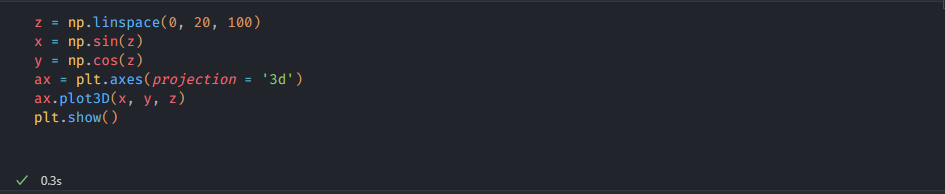
+ 3D PLOT

Với việc vẽ đồ thị 3D chúng ta cần import thêm module mpl\_toolkots là 1 phần của Matplitlib stack

Khi import module mplot3d. Chúng ta có thể sử dụng từ khóa 3d như một tham số khi chúng ta định nghĩa các trục

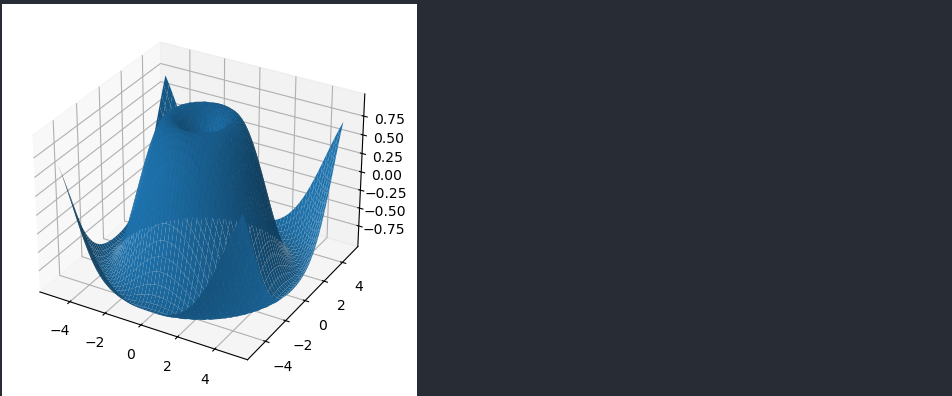
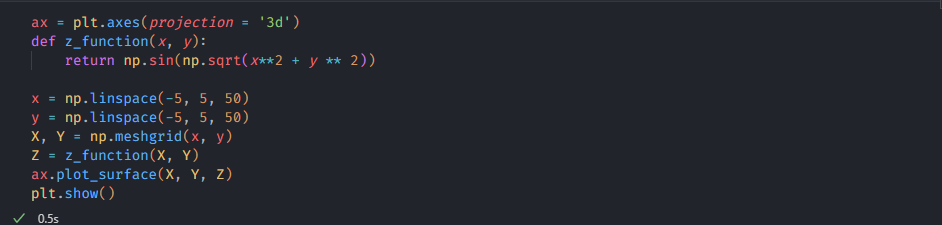


****

****

**+** SURFACE PLOT

Xem xét đến việc một hàm số ở dạng mặt phẳng, chúng ta cần tính toán tất cả các điểm nằm trên mặt phẳng đó.



Để vẽ được biểu đồ 3D chúng ta cần tạo ra các lưới điểm trong không gian 2 chiều, một lưới điểm cho x và 1 lưới điểm cho y. và mỗi điểm trên lưới tương ứng với 1 cặp giá trị (x, y)

Hàm np.meshgrid() là công cụ cần thiết để tạo ra lưới điểm này. Nó chuyển đổi các mảng một chiều x và y thành các mảng 2 chiều và có cùng hình dạng, mỗi phần tử trong mảng được lặp lại nhiều lần theo chiều tương ứng. Điều này giúp chúng ta dễ dàng tính toán các giá trị trên lưới

Khi có lưới các điểm chúng ta có thể tính toán giá trị của hàm Z để tạo ra một ma trận z với z[i,j] là kết quả khi tính toán giá trị của hàm tại điểm x[i,j] và y[i,j]