Lab2: Lab1 nâng cao

1. Numpy

In [ ]:
```
import numpy as np

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])

# Lấy số "3"
print(x[2])

# lấy giá trị 123
print(x[:3])

# lấy từ 3 đến 8
print(x[2:])

# lấy từ 3 đến 7
print(x[2:-1])

# lấy ra số 7 và 8
print(x[-2:])

# lấy phần tử 2 và 4
print(x[2], x[4])
print(x[[2, 4]])
```

```
3
[1 2 3]
[3 4 5 6 7 8]
[3 4 5 6 7]
[7 8]
3 5
[3 5]
```

In [ ]:
```
# tạo mảng 2 chiều

data = [[1, 2], [3,4], [5,6]]
mang = np.array(data)
print(mang)

print(mang[0,1])

print(mang[-1,-1])

print(mang[[1,2]])

print(mang[1:, :])

print(mang[1], mang[2])

# cách khác

print(mang[1:])
print(mang[1:3])

# cách khác dùng -
print(mang[-2:])
```

```
# lấy 1,3,5
print(mang[:, 0])

# lấy 1,5
print(mang[[0,2], 0])

# lấy 1 và 5 theo fomat start:stop:step
print(mang[::2, 0])

# cách khác
print(mang[[0,2], 0])
```

```
[[1 2]
 [3 4]
 [5 6]]
2
6
[[3 4]
 [5 6]]
[[3 4]
 [5 6]]
[3 4] [5 6]
[[3 4]
 [5 6]]
[[3 4]
 [5 6]]
[[3 4]
 [5 6]]
[1 3 5]
[1 5]
[1 5]
[1 5]
```

In [ ]:
```
X = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10,11,12],
    [13,14,15,16]
]

x = np.array(X)
print(x)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

In [ ]:
```
# lấy 2,6,10,14
print(x[:, 1])

# lấy 7,11
print(x[1:3, 2])

# lấy 4,7,10
print(x[0, 3], x[1, 2], x[2, 1])

# lấy 16 15 14 13
print(x[-1, -1], x[-1, -2], x[-1, -3], x[-1, -4])
print(x[3,::-1])
```

```
print(x[-1, -1::-1])
```

```
[ 2  6 10 14]
[ 7 11]
4 7 10
16 15 14 13
[16 15 14 13]
[16 15 14 13]
```

In [ ]:

```
# Bài tập:
# Cho ma trận sau:
# [[ 99  99  99 ]
#  [ 99  99  99 ]
#  [ 99  99  99 ]]

#  Giả sử: 0 = O và 1 là X
#  Nhận đầu vào từ phía X và O luân phiên
#  cho các bạn 1 cặp chỉ số, nếu phía nhập (0,0) thì ma trận trở thành

# [[ X  99  99 ]
#  [ 99  99  99 ]
#  [ 99  99  99 ]]

#  Nếu phía O nhập (0,0) (trùng với X) thì yêu cầu nhập lại, nếu không thì điền vào ma trận
import numpy as np

def is_valid_move(mat, row, col):
    return mat[row, col] == 99

def make_move(mat, row, col, player):
    mat[row, col] = 'X' if player == 1 else 'O'

def print_matrix(mat):
    for row in mat:
        print(' '.join(str(cell) for cell in row))
    print()  # Thêm dòng trống để dễ nhìn

def check_winner(mat, player_symbol):
    # Kiểm tra hàng và cột
    for i in range(3):
        if all(mat[i, j] == player_symbol for j in range(3)) or all(mat[j, i] == player_symbol
            return True
    # Kiểm tra 2 đường chéo
    if all(mat[i, i] == player_symbol for i in range(3)) or all(mat[i, 2 - i] == player_symbol
        return True
    return False

def play_game():
    matrix = np.full((3, 3), 99, dtype=object)  # Sửa dtype thành object
    current_player = 1
    moves = 0
    while moves < 9:
        print_matrix(matrix)
        try:
            move = input(f"Player {'X' if current_player == 1 else 'O'}, enter your move (row,
            row, col = map(int, move.replace('(', '').replace(')', '').split(','))
            if row not in range(3) or col not in range(3):
                print("Invalid input, row and column must be between 0 and 2.")
                continue
            if is_valid_move(matrix, row, col):
                make_move(matrix, row, col, current_player)
                player_symbol = 'X' if current_player == 1 else 'O'
```

```
                if check_winner(matrix, player_symbol):
                    print_matrix(matrix)
                    print(f"Player {player_symbol} wins!")
                    return
                current_player = 1 - current_player
                moves += 1
            else:
                print("Cell already taken, try again.")
        except (ValueError, IndexError):
            print("Invalid input, please enter the move in the format (row,col).")
    print_matrix(matrix)
    print("It's a draw!")

play_game()
```

```
99 99 99
99 99 99
99 99 99

X 99 99
99 99 99
99 99 99

X O 99
99 99 99
99 99 99

X O 99
X 99 99
99 99 99

Cell already taken, try again.
X O 99
X 99 99
99 99 99

X O 99
X O 99
99 99 99

X O 99
X O 99
X 99 99

Player X wins!
```

In [ ]:
```
# BT:
# Cho ma trận
# [
#     [1, 2, 3],
#     [4, 5, 6],
#     [7, 8, 9]
# ]

x = np.array([[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]])
# 1. Lấy ra số 4 5 6
print(x[1])

# 2. Lấy ra số 2 5
```

```
print(x[:2, 1])
# 3. Lấy ra số 3 4
print(x[0, 2], x[1, 0])
# cach khac
print(x[0, -1], x[1, 0])
# cach khac
print(x[[0,1], [2, 0]])
# 4. Lấy ra số 9 6 3
print(x[2, 2], x[1, 2], x[0, 2])

print(x[::-1, 2])
```

```
[4 5 6]
[2 5]
3 4
3 4
[3 4]
9 6 3
[9 6 3]
```

In [ ]:
```
x = np.array([1, 2, 3,4,5,6,7,8,9,10])

mang_chan = [ _ for _ in x if _ % 2 == 0]
print(mang_chan)

# tạo 1 mảng toàn số 1
mang_1 = np.ones((3,3))

print(mang_1)

m3 = np.arange(3)
print(mang_1 + m3)
```

```
[2, 4, 6, 8, 10]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[1. 2. 3.]
 [1. 2. 3.]
 [1. 2. 3.]]
```

In [ ]:
```
x = np.arange(3)
print(len(x))
print(x.shape)
print(x.size)

x = x.reshape((3,1))
print(x)

y = np.arange(3)

print(x+y)
```

```
3
(3,)
3
[[0]
 [1]
 [2]]
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

In [ ]:

```
# Bài tập về nhà
# Tạo mảng np có khích thước 150x5 hãy tưởng tượng mảng này chứa 150 mẫu về chiều cao, cân nặn
# Chia mảng 4 cột đầu tiên thành 1 biến có tên là x và cột cuối cùng bằng y.
# Chia x thành  X_train và X_test chứa lần lượt là 70% và 30% dữ liệu, trong đó Y_train, Y_tes


# Tạo mảng np có kích thước 150x5
data = np.random.uniform(0, 10,(150, 5)) # tạo mảng 150x5 với giá trị ngẫu nhiên từ 0 đến 10

# Chia mảng 4 cột đầu tiên thành 1 biến có tên là x và cột cuối cùng bằng y
x = data[:, :-1]  # Lấy x là 4 cột đầu tiên (Chiều cao, Cân nặng, Tuổi, Lương)
y = data[:, -1]  # Lấy y là cột cuối cùng (GPA)

dataset_size_x = x.shape[0]  # Lấy số lượng mẫu (số dòng) của dữ liệu đầu vào X
train_size_x = int(dataset_size_x * 0.7)  # Tính số lượng mẫu thuộc tập huấn luyện (70% tổng s

dataset_size_y = y.shape[0]  # Lấy số lượng mẫu (số dòng) của dữ liệu đầu ra Y
train_size_y = int(dataset_size_y * 0.7)  # Tính số lượng mẫu thuộc tập huấn luyện (70% tổng s


X_train = x[:train_size_x]  # Lấy 70% đầu tiên của X làm tập huấn luyện
x_test = x[train_size_x:]   # Lấy 30% còn lại của X làm tập kiểm tra


Y_train = y[:train_size_y]  # Lấy 70% đầu tiên của Y làm tập huấn luyện
y_test = y[train_size_y:]   # Lấy 30% còn lại của Y làm tập kiểm tra

print(X_train.shape, x_test.shape, Y_train.shape, y_test.shape)

# Tạo 10 tập dữ kiệu k chồng chéo của X_train .

split = X_train.shape[0] // 10  # Chia số lượng mẫu trong X_train thành 10 phần bằng nhau

print(split)

for counter in range(0, X_train.shape[0], split):  # Lặp qua X_train với bước nhảy là split
    print(X_train[counter:counter+split].shape)  # In kích thước của từng tập con

# BTVN: input, output, nhận xét bài trên
# input là 1 mảng chứa 150 mẫu dữ liệu, mỗi mẫu có 5 đặc trưng (feature), gồm:
# Chiều cao (Height)
# Cân nặng (Weight)
# Tuổi (Age)
# Lương (Salary)
# GPA (Điểm trung bình)
# Dữ liệu được tạo ngẫu nhiên
# Chia dữ liệu thành 2 phần:
# X: 4 cột đầu tiên (Chiều cao, Cân nặng, Tuổi, Lương).
# y: Cột cuối cùng (GPA)
#output:
# X_train chứa 105 mẫu với 4 đặc trưng.
# X_test chứa 45 mẫu với 4 đặc trưng.
# Y_train chứa 105 mẫu với 1 đặc trưng.
# Y_test chứa 45 mẫu với 1 đặc trưng.
# Và 10 tập con, mỗi tập con có 10 mẫu và 4 đặc trưng.
```

```
(105, 4) (45, 4) (105,) (45,)
10
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(10, 4)
(5, 4)
```

2. Pandas

In [ ]:
```python
import numpy as np
import pandas as pd

data = {
    "ID" : [101, 102, 103, 104, 105],
    "Name" : ["Quynh Nhu", "Hoang Hai", None, "Phuong Tuan", "Thien An"],
    "Age" : [26,18,20,None,19],
    "Salary" : [5000,35000,65000,100000,None]
}
df= pd.DataFrame(data)
print(df)
```

```
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
2  103         None  20.0   65000.0
3  104  Phuong Tuan   NaN  100000.0
4  105     Thien An  19.0       NaN
```

In [ ]:
```python
# Điền giá trị Thiếu trong cột Age = giá trị trung bình
# Loại bỏ các dòng chứa giá trị thiếu
# Điền giá trị thiếu trong name = "Unknown"ArithmeticError
# Điền giá trị thiếu trong Salary = phương pháp Interpolation
```

In [ ]:
```python
# 1. Điền giá trị thiếu trong cột 'Age' bằng giá trị trung bình
df["Age"].fillna(int(df["Age"].mean()), inplace=True)
print(df)
# 2. Loại bỏ các dòng chứa giá trị thiếu
df.dropna(inplace=True)  # Không cần vì ID không có thiếu, nhưng để làm mẫu
print(df)
# 3. Điền giá trị thiếu trong cột 'Name' bằng "Unknown"
df["Name"].fillna("Unknown", inplace=True)
print(df)

# 4. Điền giá trị thiếu trong cột 'Salary' bằng phương pháp nội suy (interpolation)
df["Salary"].interpolate(method='linear', inplace=True)
print(df)
```

```
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
2  103         None  20.0   65000.0
3  104  Phuong Tuan  20.0  100000.0
4  105     Thien An  19.0       NaN
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
    ID         Name   Age    Salary
0  101    Quynh Nhu  26.0    5000.0
1  102    Hoang Hai  18.0   35000.0
3  104  Phuong Tuan  20.0  100000.0
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Age"].fillna(int(df["Age"].mean()), inplace=True)
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:8: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Name"].fillna("Unknown", inplace=True)
C:\Users\DELL\AppData\Local\Temp\ipykernel_988\98559109.py:12: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work because the interme
diate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace
on the original object.


  df["Salary"].interpolate(method='linear', inplace=True)
```

In [ ]:
```python
# cho data

df1= pd.DataFrame({
    "ID" : [1,2,3],
    "Score_A" : [70,90,85]
})
```

```
df2= pd.DataFrame({
    "ID" : [3,4,5],
    "Score_B" : [62,91,75]
})
```

In [ ]:
```
# Thực hiện Merge trên cột id (inner join, outer join, left join)

#Nối DataFrame theo chiều dọc

#Gộp df1 và df2 để điền giá trị thiếu
df3 = pd.merge(df1, df2, on='ID', how='inner')
print(df3)
df4 = pd.merge(df1, df2, on='ID', how='outer')
print(df4)
df5 = pd.merge(df1, df2, on='ID', how='left')
print(df5)

df6 = pd.concat([df1, df2], axis=0)
print(df6)

df7 = df1.set_index("ID").combine_first(df2.set_index("ID")).reset_index()
print(df7)
```

```
   ID  Score_A  Score_B
0   3       85       62
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0     62.0
3   4      NaN     91.0
4   5      NaN     75.0
   ID  Score_A  Score_B
0   1       70      NaN
1   2       90      NaN
2   3       85     62.0
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0      NaN
0   3      NaN     62.0
1   4      NaN     91.0
2   5      NaN     75.0
   ID  Score_A  Score_B
0   1     70.0      NaN
1   2     90.0      NaN
2   3     85.0     62.0
3   4      NaN     91.0
4   5      NaN     75.0
```

In [ ]:
```
# Cho dữ liệu sau
data = pd.DataFrame({
    "ID" : range(1, 100001),
    "Value" : np.random.randint(1, 100, 100000)
})
# data

#Dùng .astype để tối ưu hóa bộ nhớ
data["ID"] = data["ID"].astype("int32")
data["Value"] = data["Value"].astype("int8")

# Tìm 5 giá trị phổ biến trong cột value
data["Value"].value_counts().head(5)
```

```
# Sử dụng query để lọc dữ liệu nhanh hơn df[df["Value"] > 90]
data.query("Value > 90")
```

Out[ ]:

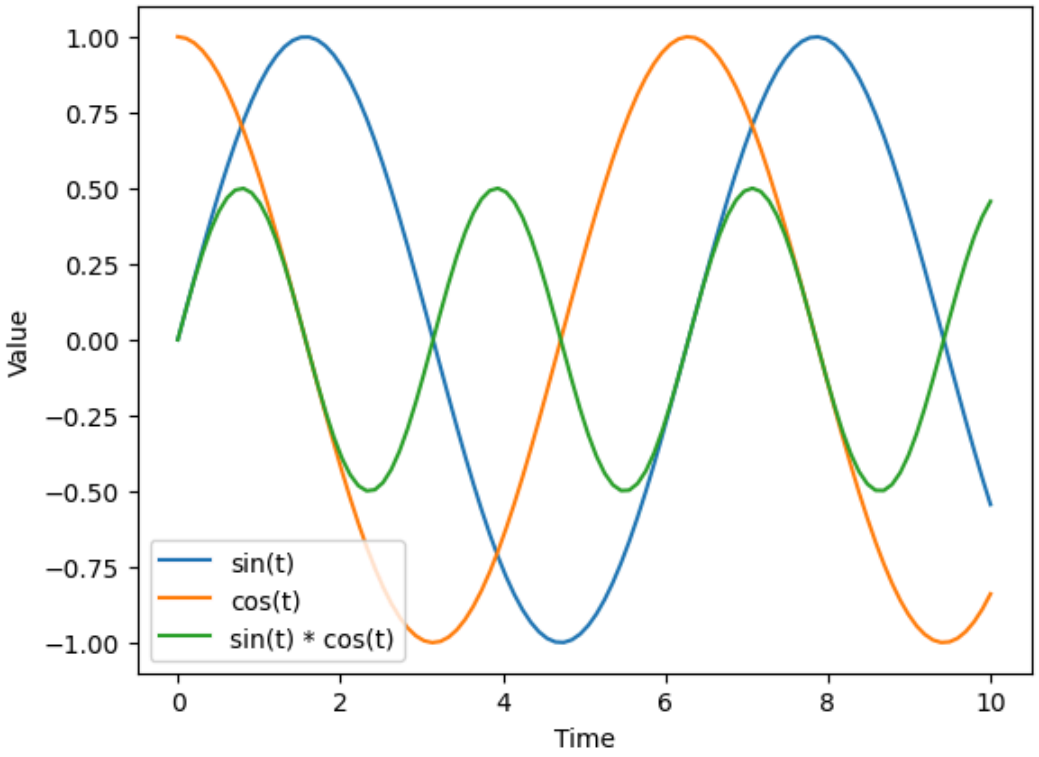
	ID	Value
1	2	93
5	6	98
8	9	91
26	27	96
63	64	97
...	...	...
99968	99969	96
99972	99973	95
99979	99980	95
99980	99981	96
99996	99997	94

9169 rows × 2 columns

3. Matplotlib

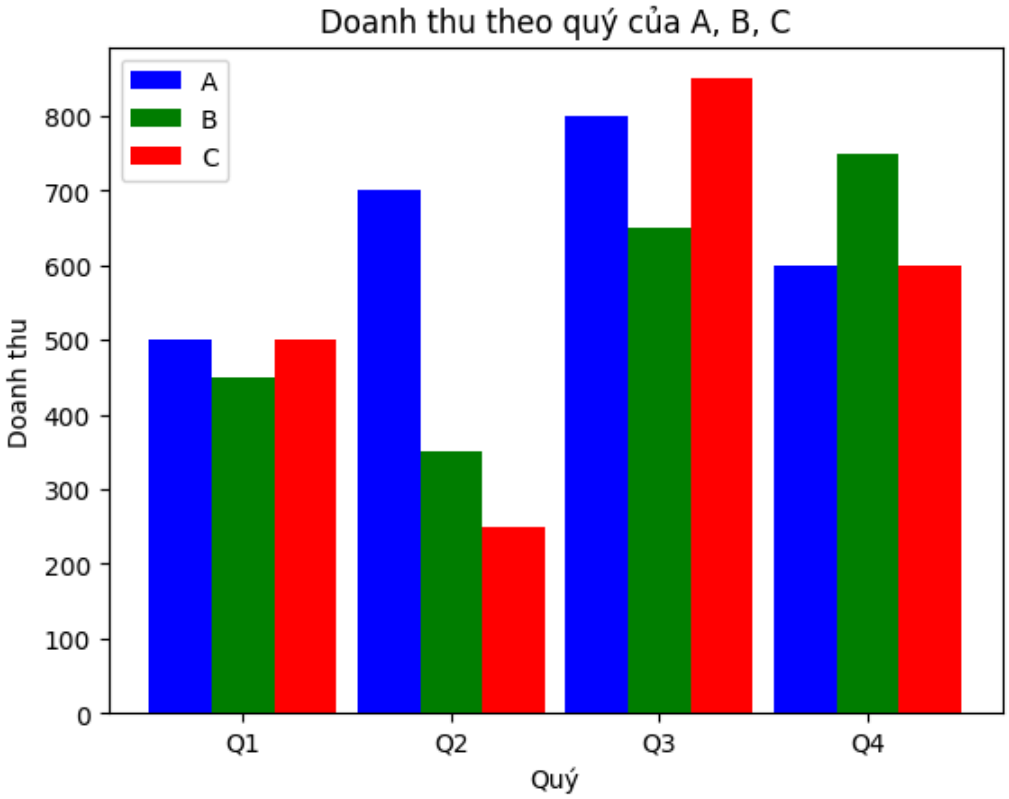
In [ ]:

```
# Biểu đồ nhiều đường

# cho dữ liệu sau

import matplotlib.pyplot as plt

import numpy as np

t = np.linspace(0, 10, 100)
y1 = np.sin(t)
y2 = np.cos(t)
y3 = np.sin(t) * np.cos(t)

# tạo biểu đồ các đường theo thời gian

plt.plot(t, y1, label='sin(t)')
plt.plot(t, y2, label='cos(t)')
plt.plot(t, y3, label='sin(t) * cos(t)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()
plt.show()
```

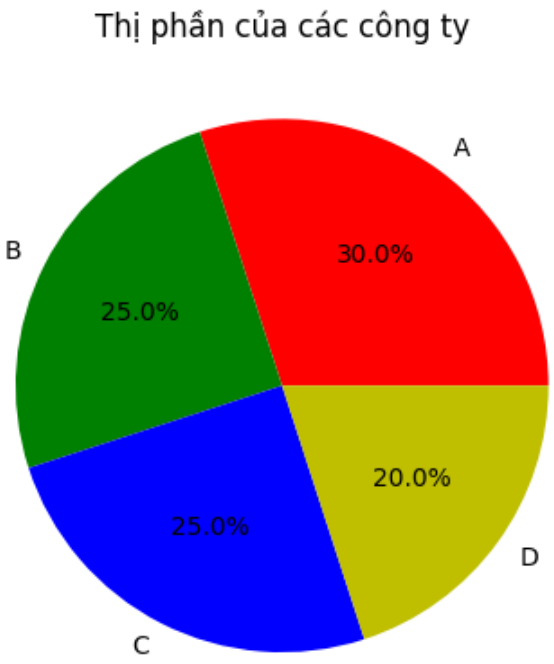


In [ ]:
```python
# Biểu đồ thanh nhóm

# cho dữ liệu sau
labels = ['Q1', 'Q2', 'Q3', 'Q4']
A = [500,700,800,600]
B = [450, 350, 650, 750]
C = [500, 250, 850, 600]

# Vẽ biểu đồ thanh nhóm thể hiện danh thu của A, B, C trong 4 quý
x = np.arange(len(labels))
width = 0.3
plt.bar(x - width, A, width, label='A', color='b')  # Dịch trái width đơn vị
plt.bar(x, B, width, label='B', color='g')          # Vẽ tại vị trí gốc
plt.bar(x + width, C, width, label='C', color='r')  # Dịch phải width đơn vị

# Thêm nhãn trục và tiêu đề
plt.xlabel("Quý")
plt.ylabel("Doanh thu")
plt.title("Doanh thu theo quý của A, B, C")

# Đặt nhãn trục x
plt.xticks(x, labels)

# Hiển thị chú thích
plt.legend()
```

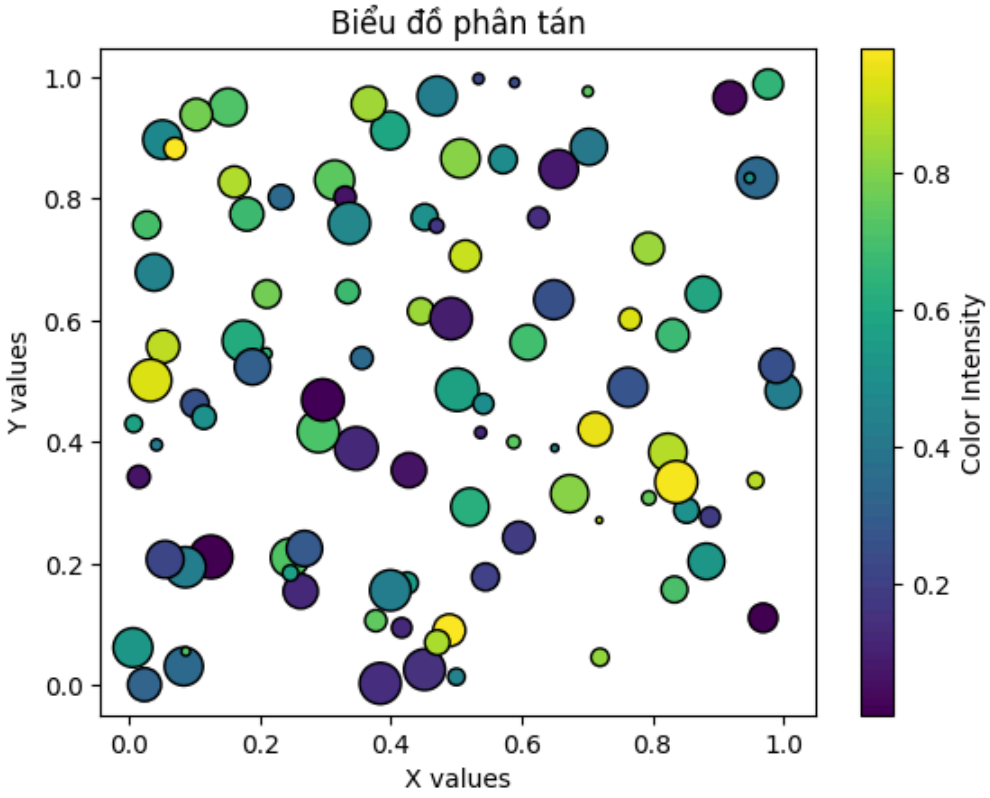
Out[ ]:
```
<matplotlib.legend.Legend at 0x15752d262d0>
```



In [ ]:
```
# Biểu đồ tròn

Cty = ['A', 'B', 'C', 'D']
thiphan = [30, 25, 25, 20]
color = ['r', 'g', 'b', 'y']

# Tạo biểu đồ tròn biểu hiện tỉ lệ thị phần của các công ty

plt.pie(thiphan, labels=Cty, colors=color, autopct='%1.1f%%')
plt.title("Thị phần của các công ty")
plt.show()
```



In [ ]:
```python
# Biểu đồ phân tán

# cho dữ liệu sau

x = np.random.rand(100)
y = np.random.rand(100)
sizes = np.random.rand(100) * 300
colors = np.random.rand(100)

# tạo biểu đồ phân tán của 2 biến ngẫu nhiên
plt.scatter(x, y, s=sizes, c=colors, cmap='viridis', edgecolors='k')

# Thêm nhãn trục và tiêu đề
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Biểu đồ phân tán")

# Hiển thị thang màu
plt.colorbar(label="Color Intensity")

# Hiển thị đồ thị
plt.show()
```



In [ ]:
```
# biểu đồ nhiệt

data = np.random.rand(10, 10)

# vẽ 1 heatmap thể hiện sự phân bố của dữ liệu trên ma trận
plt.imshow(data, cmap='hot', interpolation='nearest')
plt.colorbar()
plt.show()
```