1. จงเขียนโค้ด สำหรับโหลดข้อมูลจากไฟล์ csv ชื่อ mydata.csv เข้ามาเก็บไว้ใน Pandas Dataframe ชื่อ mydf

```
import numpy as np
import pandas as pd
import seaborn as sns
mydf = pd.read_csv('mydata.csv ', delimiter=',')
```

2. จงเขียนโค้ดเพื่อแสดงจำนวนแถว, จำนวนคอลัมน์, ชื่อคอลัมน์, จำนวนคอลัมน์ที่ไม่มีค่า Null, และชนิดข้อมูลของแต่ละคอลัมน์ ของดาต้าเฟรม mydf

```
mydf.info()
```

3. จงเขียนโค้ดเพื่อแสดงตารางสรุปค่าสถิติเชิงพรรณนา (descriptive statistics) ของดาต้าเฟรม mydf

```
mydf.describe()
```

4. จากตัวอย่างการสร้างโมเดลโดยใช้ชุดข้อมูล bike sharing จงตอบคำถามต่อไปนี้
   a. โมเดลที่ถูกนำมาใช้คือโมเดลอะไร

   Linear regression

   b. ค่าที่ต้องการให้โมเดลทำนาย (target feature) คือค่าของคอลัมน์ใด

   profit

   c. คอลัมน์ที่เป็นตัวแปรต้น ได้แก่ คอลัมน์ใดบ้าง

   population

d. จงสรุปขั้นตอนการเตรียมข้อมูล

1.) แยกข้อมูลที่ใช้ train หรือ test

2.) set ค่าของข้อมูลในส่วน train หรือ test

3.) สร้างตัวแปร theta (กำหนดค่าเป็น 0)

5. ฟังก์ชัน $J(\theta)$ ในตัวอย่าง bike sharing คืออะไร และนำมาใช้ทำอะไรในการสร้างโมเดล

คือ ฟังก์ชันแสดงความคลาดเคลื่อนของ model เทียบกับข้อมูลจริง
นำมาใช้หาค่า theta (ความชัน และ y เริ่มต้น) ของโมเดลให้มีความแม่นยำ

6. อัลกอริทึมที่ใช้ในการสร้างโมเดลตัวอย่างมีชื่อว่าอะไร จงอธิบายการทำงานของอัลกอริทึม ดังกล่าวโดยเขียนเป็น Pseudo-code

Gradient Descent

```
gradient_descent(ค่าพิกัดในส่วน train, อัตราการเปลี่ยนแปลง, จำนวนรอบ)
  กำหนดค่า theta

  loop ตามจำนวนรอบ
  {     หาค่า error ของ theta
        นำค่า error และ อัตราการเปลี่ยนแปลง มาคำนวณกับ theta เดิมเพื่อหา theta ใหม่
  }
```

คืนค่า theta ที่ได้

7. สมมติว่า ค่าของพารามิเตอร์ θ_0 , θ_1 ที่ได้จากการเทรนโมเดลตามตัวอย่าง bike sharing มีค่าเท่ากับ 4.8 และ 0.15 ตามลำดับ จงคำนวณคำทำนายของโมเดลเมื่อ จำนวนประชากรมีค่าเท่ากับ 50,000 คน

4.8 + X * 0.15 = 4.8 + 0.5 * 0.15 = 4.875 * 10^5 บาท

8. จงสร้างโมเดลสำหรับทำนายระยะทางที่ใช้ในการหยุดรถ จากความเร็ว โดยใช้ชุดข้อมูล cars.csv

```
import numpy as np
import pandas as pd
import seaborn as sns
col_list = ["speed","dist"]
data = pd.read_csv('cars.csv', usecols=col_list)
data = data.dropna()
data.describe()
def prepare_X_y(data, y_column, m):
   X_columns = [c for c in data.columns if c != y_column]
   X = data.loc[:, X_columns]
   X = np.append(np.ones((m, 1)), X, axis=1)
   y = data.loc[:, y_column].values.reshape(m, 1)
   return X, y

def split_train_test_datasets(data, y_column, test_frac, random_state):
   test_data  = data.sample(frac=test_frac, random_state=random_state)
   train_data = data.loc[~data.index.isin(test_data.index)]
   X_train, y_train = prepare_X_y(train_data, y_column, m=train_data.shape[0])
   X_test, y_test = prepare_X_y(test_data, y_column, m=test_data.shape[0])
   return X_train, y_train, X_test, y_test

def initialize_theta(num_features):
```

```
    theta = np.zeros((num_features, 1))
    return theta

y_col = 'dist'
num_features = data.columns.size
X_train, y_train, X_test, y_test = split_train_test_datasets(data, y_col, test_frac=0.2,
random_state=30)
theta = initialize_theta(num_features)

def cost_function(X, y, theta):
    m = len(y)
    y_pred = X.dot(theta)
    error = (y_pred - y) ** 2
    return 1 / (2 * m) * np.sum(error)

def gradient_descent(X, y, alpha, iterations):
    m = len(y)
    costs = []
    theta = initialize_theta(X.shape[1])
    for i in range(iterations):
        y_pred = X.dot(theta)
        propagated_error = np.dot(X.transpose(), (y_pred - y))
        theta = theta - alpha * (1/m) * propagated_error
        costs.append(cost_function(X, y, theta))
        if i % 200 == 0:
            print("Iteration %4d: MSE = %.5f" % (i+1, costs[-1]))

    print("Iteration %4d: MSE = %.5f" % (i+1, costs[-1]) )
    return theta, costs
theta, costs = gradient_descent(X_train, y_train, alpha=0.0072, iterations=2000)
print("h(x) = {} + {}x".format(str(round(theta[0, 0], 2)),
                               str(round(theta[1, 0], 2))))

sns.scatterplot(x='speed', y='dist', data=data)

x_value = [x for x in range(0, 25)]
y_value = [(x * theta[1, 0] + theta[0, 0]) for x in x_value]
sns.lineplot(x=x_value, y=y_value, color='b')

plt.xlabel('speed')
plt.ylabel('dist')
plt.title('Linear Regression Fit')
```