**Five different NumPy programs that showcase various operations**

1. **Creating an Array and Performing Basic Operations:**

**import numpy as np**

**# Create a NumPy array**

**arr = np.array([1, 2, 3, 4, 5])**

**# Perform basic operations**

**print("Original Array:", arr)**

**print("Sum:", np.sum(arr))**

**print("Mean:", np.mean(arr))**

**print("Max Value:", np.max(arr))**

**print("Squared Values:", np.square(arr))**

1. **Matrix Multiplication:**

**import numpy as np**

**# Create two matrices**

**A = np.array([[1, 2], [3, 4]])**

**B = np.array([[5, 6], [7, 8]])**

**# Perform matrix multiplication**

**result = np.dot(A, B)**

**print("Matrix A:")**

**print(A)**

**print("Matrix B:")**

**print(B)**

**print("Result of Matrix Multiplication:")**

print(result)

1. **Reshaping an Array:**

**import numpy as np**

**# Create a 1D array**

**arr = np.arange(1, 13)**

**# Reshape it into a 3x4 matrix**

**reshaped\_arr = arr.reshape(3, 4)**

**print("Original Array:")**

**print(arr)**

**print("Reshaped Array:")**

**print(reshaped\_arr)**

1. **Element-wise Operations:**

**import numpy as np**

**# Create two arrays**

**arr1 = np.array([1, 2, 3, 4])**

**arr2 = np.array([5, 6, 7, 8])**

**# Perform element-wise operations**

**sum\_result = arr1 + arr2**

**product\_result = arr1 \* arr2**

**print("Array 1:", arr1)**

**print("Array 2:", arr2)**

**print("Element-wise Sum:", sum\_result)**

**print("Element-wise Product:", product\_result)**

1. **Indexing and Slicing:**

**import numpy as np**

**# Create a NumPy array**

**arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])**

**# Access elements and perform slicing**

**print("Original Array:", arr)**

**print("Element at index 3:", arr[3])**

**print("Elements from index 2 to 7:", arr[2:8])**

**print("Every second element:", arr[::2])**

**Using Pandas read dataset and print it**

To read a dataset using Pandas and print its contents, you'll need to follow these steps:

1. **Import Pandas**: First, make sure you have Pandas installed. You can install it using pip if you haven't:

**pip install pandas**

1. **Import Pandas and Read the Dataset:** Import the Pandas library and use one of its functions (e.g., **read\_csv**, **read\_excel**, **read\_json**, etc.) to read the dataset into a Pandas DataFrame.

**import pandas as pd**

**# Replace "your\_dataset.csv" with the actual file path or URL of your dataset**

**dataset\_path = "your\_dataset.csv"**

**# Read the dataset into a Pandas DataFrame**

**df = pd.read\_csv(dataset\_path) # You can use other read functions for different file types**

1. **Print the DataFrame:** Use the **print()** function or simply enter the DataFrame variable name to print the contents of the DataFrame.

**# Print the contents of the DataFrame**

**print(df)**

**Display two Matplot and Searborn chart using any dataset**

Following are the examples of two different types of charts using Matplotlib and Seaborn libraries with a sample dataset. For this example, we'll use the built-in Seaborn dataset called "tips," which contains information about restaurant tips.

**Matplotlib Example (Bar Chart):** In this example, we'll create a bar chart to visualize the total bill amounts for different days of the week.

import seaborn as sns

import matplotlib.pyplot as plt

# Load the "tips" dataset

tips = sns.load\_dataset("tips")

# Calculate the total bill amounts for each day of the week

total\_bill\_by\_day = tips.groupby("day")["total\_bill"].sum()

# Create a bar chart using Matplotlib

plt.figure(figsize=(8, 6))

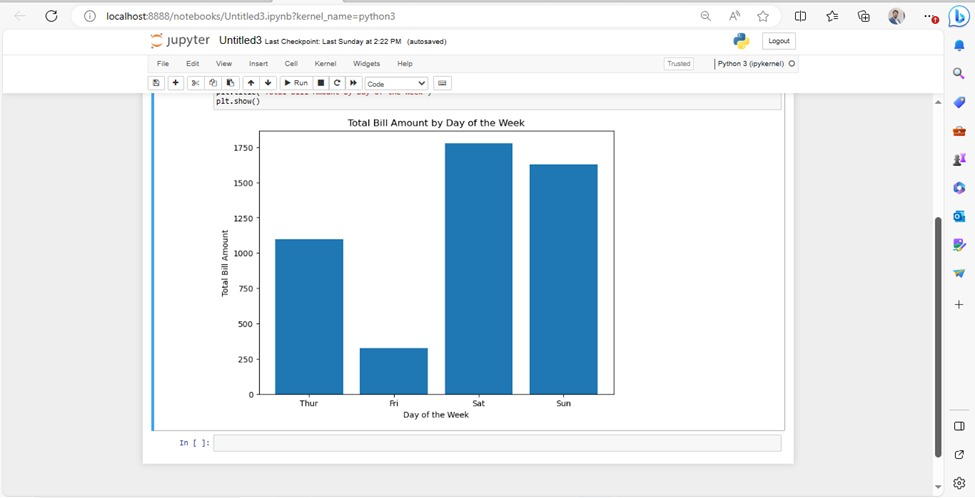
plt.bar(total\_bill\_by\_day.index, total\_bill\_by\_day.values)

plt.xlabel("Day of the Week")

plt.ylabel("Total Bill Amount")

plt.title("Total Bill Amount by Day of the Week")

plt.show()



**Seaborn Example (Scatter Plot):** In this example, we'll create a scatter plot to visualize the relationship between total bill amounts and tips.

import seaborn as sns

import matplotlib.pyplot as plt

# Load the "tips" dataset

tips = sns.load\_dataset("tips")

# Create a scatter plot using Seaborn

plt.figure(figsize=(8, 6))

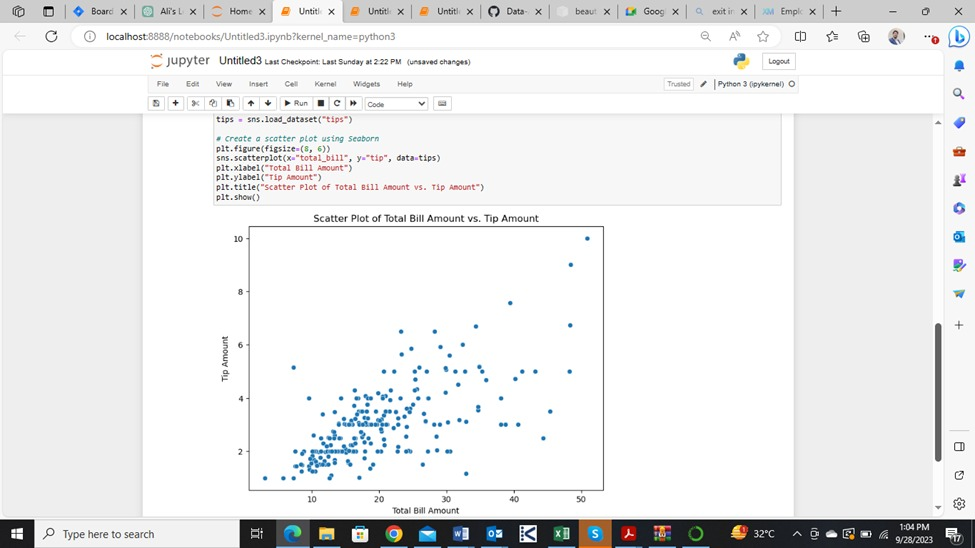
sns.scatterplot(x="total\_bill", y="tip", data=tips)

plt.xlabel("Total Bill Amount")

plt.ylabel("Tip Amount")

plt.title("Scatter Plot of Total Bill Amount vs. Tip Amount")

plt.show()

****

**Scrap Data from Amazon website**

Data scraped from Apify.com:

[**https://d.docs.live.net/b86f4b15342bfdde/Desktop/anaam\_dataset\_facebook-posts-scraper\_2023-09-24\_05-30-58-774.csv**](https://d.docs.live.net/b86f4b15342bfdde/Desktop/anaam_dataset_facebook-posts-scraper_2023-09-24_05-30-58-774.csv)

[**https://d.docs.live.net/b86f4b15342bfdde/Desktop/dataset\_amazon-bestsellers\_2023-09-24\_05-39-28-900.csv**](https://d.docs.live.net/b86f4b15342bfdde/Desktop/dataset_amazon-bestsellers_2023-09-24_05-39-28-900.csv)

[**https://d.docs.live.net/b86f4b15342bfdde/Desktop/dataset\_facebook-posts-scraper\_2023-09-24\_05-20-45-202.csv**](https://d.docs.live.net/b86f4b15342bfdde/Desktop/dataset_facebook-posts-scraper_2023-09-24_05-20-45-202.csv)

*Data provided in class:*

**https://www.kaggle.com/datasets/yasserh/housing-prices-dataset**

**Write 5 Scipy Programs**

Following are five different programs that showcase various capabilities of the SciPy library:

1. **Solving Linear Equations:**

**import numpy as np**

**from scipy.linalg import solve**

**# Define the coefficient matrix and the right-hand side vector**

**A = np.array([[2, 1], [1, 3]])**

**b = np.array([5, 8])**

**# Solve the linear system of equations Ax = b**

**x = solve(A, b)**

**print("Solution for x:", x)**

1. **Finding Eigenvalues and Eigenvectors:**

**import numpy as np**

**from scipy.linalg import eig**

**# Define a square matrix**

**A = np.array([[4, -2], [1, 1]])**

**# Calculate eigenvalues and eigenvectors**

**eigenvalues, eigenvectors = eig(A)**

**print("Eigenvalues:", eigenvalues)**

**print("Eigenvectors:", eigenvectors)**

1. **Interpolation:**

**import numpy as np**

**from scipy.interpolate import interp1d**

**import matplotlib.pyplot as plt**

**# Sample data points**

**x = np.array([0, 1, 2, 3, 4])**

**y = np.array([0, 1, 4, 9, 16])**

**# Create an interpolation function**

**f = interp1d(x, y, kind='cubic')**

**# Generate new data points for interpolation**

**x\_new = np.linspace(0, 4, num=100)**

**y\_new = f(x\_new)**

**# Plot the original data and the interpolated curve**

**plt.scatter(x, y, label='Original Data')**

**plt.plot(x\_new, y\_new, label='Interpolated Curve', linestyle='--')**

**plt.legend()**

**plt.show()**

1. **Numerical Integration:**

**import numpy as np**

**from scipy.integrate import quad**

**# Define the function to be integrated**

**def f(x):**

**return x\*\*2**

**# Perform numerical integration**

**result, error = quad(f, 0, 2)**

**print("Result of Integration:", result)**

**print("Error Estimate:", error)**

1. **Nonlinear Optimization:**

**from scipy.optimize import minimize**

**# Define an objective function to minimize**

**def objective\_function(x):**

**return x[0]\*\*2 + x[1]\*\*2**

**# Initial guess**

**initial\_guess = [1, 1]**

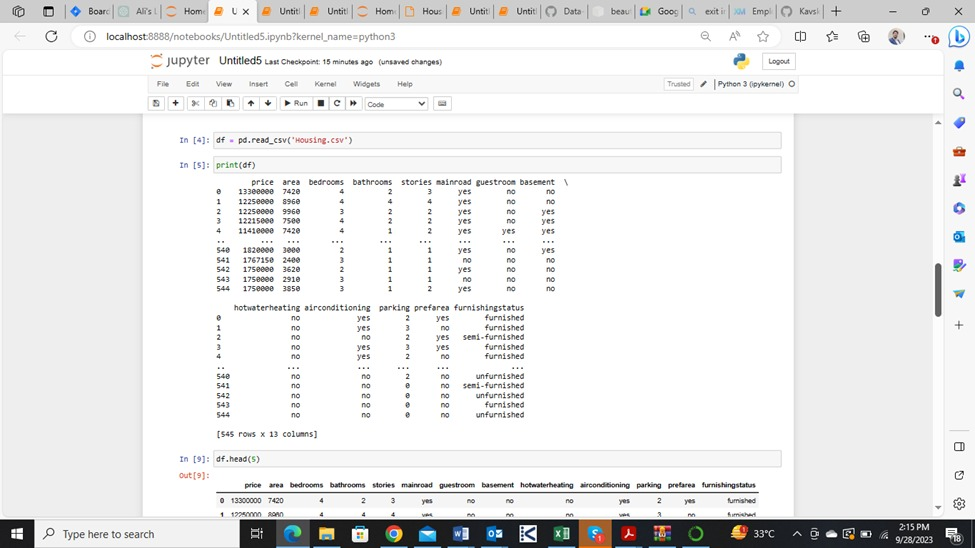
**# Perform optimization**

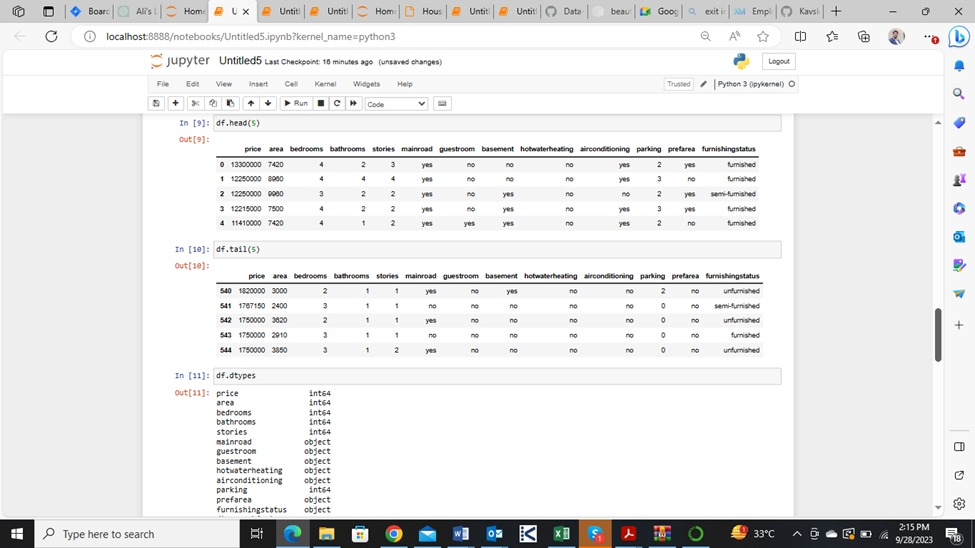
**result = minimize(objective\_function, initial\_guess, method='BFGS')**

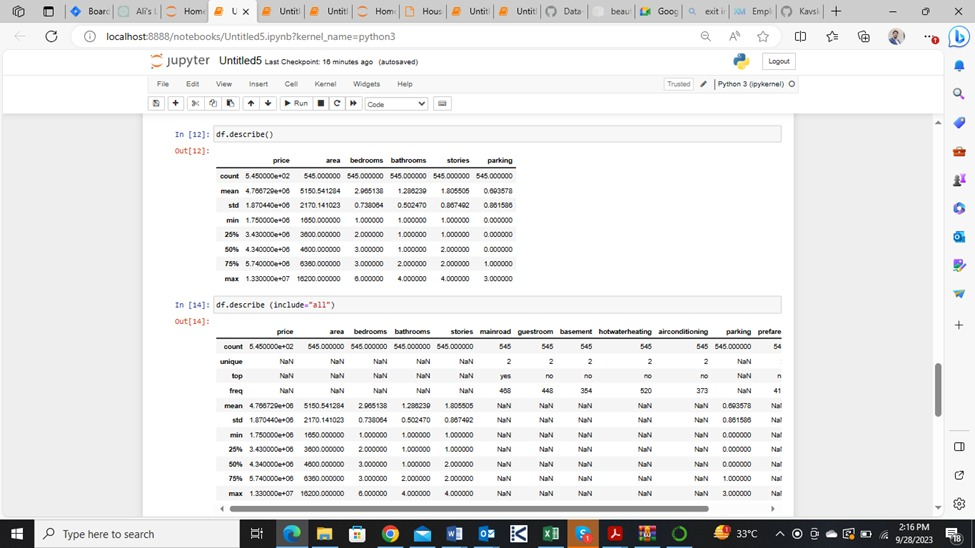
**print("Optimal Solution:", result.x)**

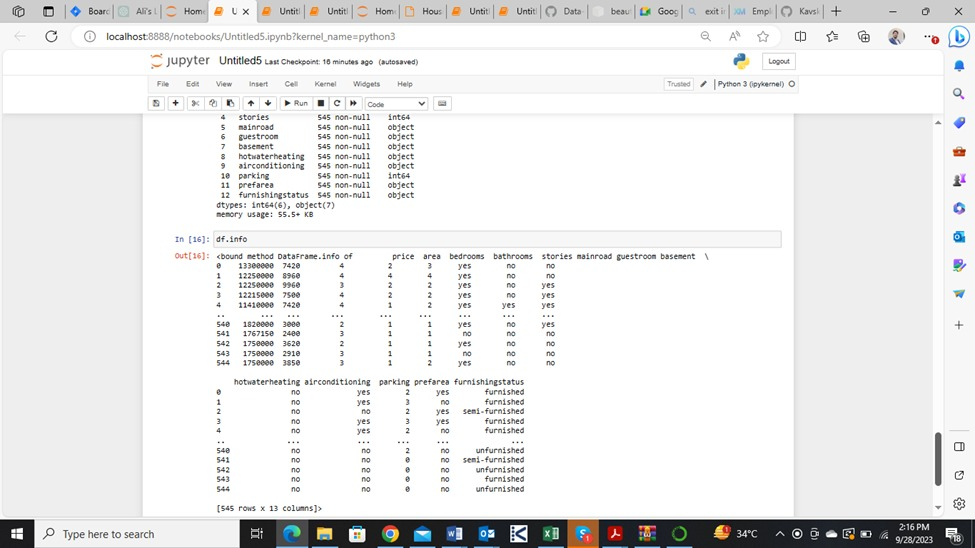
**print("Optimal Objective Value:", result.fun)**

**Perform Data Analysis using house price prediction dataset**

****

****

****

****