Experiment 1

Aim: Create a program that prompts the user for their name and age and prints a personalized message.

Software Used: Visual Studio Code.

Code:

```python
# Program to prompt the user for their name and age, then display a personalized message

# Taking user input for name and age
name = input("Enter your name: ")  # Accepts a string input for the name
age = input("Enter your age: ")    # Accepts a string input for the age

# Type conversion: Convert age from string to integer
try:
    age = int(age)
except ValueError:
    print("Invalid age! Please enter a numeric value.")
    exit()

# Processing and output
# Displaying a personalized message using string formatting
print(f"Hello, {name}! You are {age} years old.")

# Simple operation: Calculate the age in 5 years
future_age = age + 5
print(f"In 5 years, you will be {future_age} years old.")

# Thanking the user
print("Thank you for using this program!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±1 (8.395s)
python3 -m exp1
Enter your name: John Doe
Enter your age: 25
Hello, John Doe! You are 25 years old.
In 5 years, you will be 30 years old.
Thank you for using this program!
```

Experiment 2

Aim: Create a program that prompts user for their age and tells them if they can vote in the next election.

Software Used: Visual Studio Code.

Code:

```python
# Program to check if a user is eligible to vote

# Taking user input for age
age = input("Enter your age: ")

# Type conversion and validation
try:
    age = int(age)
except ValueError:
    print("Invalid input! Please enter a numeric value.")
    exit()

# Checking voting eligibility
if age >= 18:
    print("You are eligible to vote in the next election.")
else:
    print("You are not eligible to vote yet.")

# Thanking the user
print("Thank you for using the voting eligibility checker!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±1 (6.06s)
 python3 -m exp2
Enter your age: 16
You are not eligible to vote yet.
Thank you for using the voting eligibility checker!
```

```
base ~/Desktop/PIPlab git:(main)±2 (4.683s)
python3 -m exp2
Enter your age: 18
You are eligible to vote in the next election.
Thank you for using the voting eligibility checker!
```

Experiment 3

Aim: Create a program that calculates the factorial of a number entered by user using a loop.

Software Used: Visual Studio Code.

Code:

```python
# Program to calculate the factorial of a number using a loop

# Taking user input
num = input("Enter a number to calculate its factorial: ")

# Type conversion and validation
try:
    num = int(num)
    if num < 0:
        print("Factorial is not defined for negative numbers.")
        exit()
except ValueError:
    print("Invalid input! Please enter a numeric value.")
    exit()

# Initializing factorial to 1
factorial = 1

# Calculating factorial using a loop
for i in range(1, num + 1):
    factorial *= i

# Displaying the result
print(f"The factorial of {num} is {factorial}.")

# Thanking the user
print("Thank you for using the factorial calculator!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±2 (5.244s)
python3 -m exp3
Enter a number to calculate its factorial: 7
The factorial of 7 is 5040.
Thank you for using the factorial calculator!
```

Experiment 4

Aim: Create a program that prompts the user for a list of numbers and then sorts them in ascending order.

Software Used: Visual Studio Code.

Code:

```python
# Program to sort a list using a simple sorting algorithm and an in-built function

# Taking user input for the list size
n = int(input("Enter the number of elements required: "))
# Creating an empty list
l2 = []
# Taking list elements from the user
print("Enter the elements:")
for i in range(n):
    element = int(input(f"Element {i + 1}: "))
    l2.append(element)
# Displaying the original list
print("\nOriginal List:", l2)
# Sorting using a simple sorting algorithm (Bubble Sort)
for i in range(len(l2)):
    for j in range(i + 1, len(l2)):
        if l2[i] > l2[j]:
            # Swapping elements
            l2[i], l2[j] = l2[j], l2[i]
# Displaying the sorted list
print("Sorted List (using custom algorithm):", l2)
# --- Sorting using an in-built function ---
# Taking input again for comparison
l1 = []
print("\nUsing in-built sort function:")
for i in range(n):
    element = int(input(f"Element {i + 1}: "))
    l1.append(element)
# Displaying the original list
print("\nOriginal List:", l1)
# Sorting using Python's built-in `sort()` function
l1.sort()
# Displaying the sorted list
print("Sorted List (using in-built sort):", l1)
# Thanking the user
print("\nThank you for using the sorting program!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±3 (55.125s)
python3 -m exp4
Enter the number of elements required: 7
Enter the elements:
Element 1: 10
Element 2: 1
Element 3: 5
Element 4: 4
Element 5: 3
Element 6: 7
Element 7: 6

Original List: [10, 1, 5, 4, 3, 7, 6]
Sorted List (using custom algorithm): [1, 3, 4, 5, 6, 7, 10]

Using in-built sort function:
Element 1: 10
Element 2: 1
Element 3: 5
Element 4: 4
Element 5: 3
Element 6: 7
Element 7: 6

Original List: [10, 1, 5, 4, 3, 7, 6]
Sorted List (using in-built sort): [1, 3, 4, 5, 6, 7, 10]

Thank you for using the sorting program!
```

Experiment 5

Aim: Create a program that prompts the user for a string and then prints the string reversed.

Software Used: Visual Studio Code.

Code:

```python
# Program to reverse a string using a loop and a library function

# Function to reverse a string using a loop
def reverseString(string):
    reverseStr = ""
    for i in range(len(string) - 1, -1, -1):
        reverseStr += string[i]
    return reverseStr

# Function to reverse a string using a library function
def reverseString2(string):
    return "".join(reversed(string))

# Taking user input
string = input("Enter the string: ")

# Calling both functions
reverseStr = reverseString(string)
reverseStr2 = reverseString2(string)

# Displaying results
print("\nOriginal String:", string)
print("Reversed String (using loop):", reverseStr)
print("Reversed String (using library function):", reverseStr2)

# Thanking the user
print("\nThank you for using the string reversal program!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±4 (31.227s)
python3 -m exp5
Enter the string: The quick brown fox jumped over the lazy dog.

Original String: The quick brown fox jumped over the lazy dog.
Reversed String (using loop): .god yzal eht revo depmuj xof nworb kciuq ehT
Reversed String (using library function): .god yzal eht revo depmuj xof nworb kciuq ehT

Thank you for using the string reversal program!
```

Experiment 6

Aim: Create a program that defines a function to calculate the area of a circle based on the radius of the circle entered by the user.

Software Used: Visual Studio Code.

Code:

```python
# Importing the math library
import math as m

# Function to calculate the area of a circle
def areaOfCircle(radius):
    area = m.pi * radius**2
    return area

# Taking user input
try:
    radius = float(input("Enter the radius of the circle: "))
    if radius < 0:
        print("Radius cannot be negative. Please enter a valid value.")
    else:
        # Calling the function and displaying the result
        print(f"The area of the circle with radius {radius} is: {areaOfCircle(radius):.2f}")
except ValueError:
    print("Invalid input! Please enter a numeric value.")

# Thanking the user
print("\nThank you for using the circle area calculator!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±5 (29.908s)
python3 -m exp6
Enter the radius of the circle: 20
The area of the circle with radius 20.0 is: 1256.64

Thank you for using the circle area calculator!
```

Experiment 7

Aim: Create a program that defines a class to represent a car and then create an object of that class with specific attributes.

Software Used: Visual Studio Code.

Code:

```python
# Defining the Car class
class Car:
    # Class variable (common to all instances)
    wheels = 4

    # Constructor to initialize object attributes
    def __init__(self, make, model, color, year):
        self.make = make
        self.model = model
        self.color = color
        self.year = year

    # Method to display car details
    def displayCar(self):
        print(f"Car Make: {self.make}, Model: {self.model}, Wheels: {Car.wheels}, Color: {self.color}, Year: {self.year}")

# Creating objects of the Car class
car1 = Car("Toyota", "Corolla", "Black", "2022")
car2 = Car("Honda", "Civic", "Silver", "2023")

# Displaying car details
print("\nCar Details:")
car1.displayCar()
car2.displayCar()

# Thanking the user
print("\nThank you for using the car details program!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±6 (0.07s)
python3 -m exp7

Car Details:
Car Make: Toyota, Model: Corolla, Wheels: 4, Color: Black, Year: 2022
Car Make: Honda, Model: Civic, Wheels: 4, Color: Silver, Year: 2023

Thank you for using the car details program!
```

Experiment 8

Aim: Create a program that reads data from a file and writes it to another file in a format.

Software Used: Visual Studio Code.

Code:

```python
import os

# Displaying the list of files before copying
print("List of files before copying:")
print(os.listdir())
print("\n")
# Opening file1.txt in read mode
with open('file1.txt', 'r') as file1:
    # Opening file2.txt in write mode
    with open('file2.txt', 'w') as file2:
        # Copying content line by line
        for line in file1:
            file2.write(line)
# Reading and displaying the contents of file2.txt
with open('file2.txt', 'r') as file2:
    print("Contents of file2.txt:\n")
    print(file2.read())
# Displaying the list of files after copying
print("\nList of files after copying:")
print(os.listdir())
print("\n")
# Thanking the user
print("Thank you for using the file handling program!")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±8 (0.05s)
python3 -m exp8
List of files before copying:
['file1.txt', '.DS_Store', 'exp15Server.py', 'scraped_data.txt', 'exp10.py', 'exp14.py', 'exp4.py', 'exp5.py', '__pycache__', 'exp
11.py', 'exp1.py', 'exp15Client.py', 'exp6.py', 'exp12.py', '.gitignore', 'exp2.py', 'exp13.py', 'Toyota.csv', 'exp3.py', 'exp7.py
', 'myenv', '.git', 'exp8.py', 'exp9.py']


Contents of file2.txt:

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit
in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.

Curabitur pretium tincidunt lacus.
Nulla gravida orci a odio. Nullam varius, turpis et commodo pharetra,
est eros bibendum elit, nec luctus magna felis sollicitudin mauris.
Integer in mauris eu nibh euismod gravida. Duis ac tellus et risus
vulputate vehicula. Donec lobortis risus a elit. Etiam tempor.
Ut ullamcorper, ligula eu tempor congue, eros est euismod turpis,
id tincidunt sapien risus a quam. Maecenas fermentum consequat mi.
Donec fermentum. Pellentesque malesuada nulla a mi. Duis sapien sem,
aliquet nec, commodo eget, consequat quis, neque. Aliquam faucibus,
elit ut dictum aliquet, felis nisl adipiscing sapien, sed malesuada
diam lacus eget erat.

List of files after copying:
['file2.txt', 'file1.txt', '.DS_Store', 'exp15Server.py', 'scraped_data.txt', 'exp10.py', 'exp14.py', 'exp4.py', 'exp5.py', '__pyc
ache__', 'exp11.py', 'exp1.py', 'exp15Client.py', 'exp6.py', 'exp12.py', '.gitignore', 'exp2.py', 'exp13.py', 'Toyota.csv', 'exp3.
py', 'exp7.py', 'myenv', '.git', 'exp8.py', 'exp9.py']


Thank you for using the file handling program!
```

Experiment 9

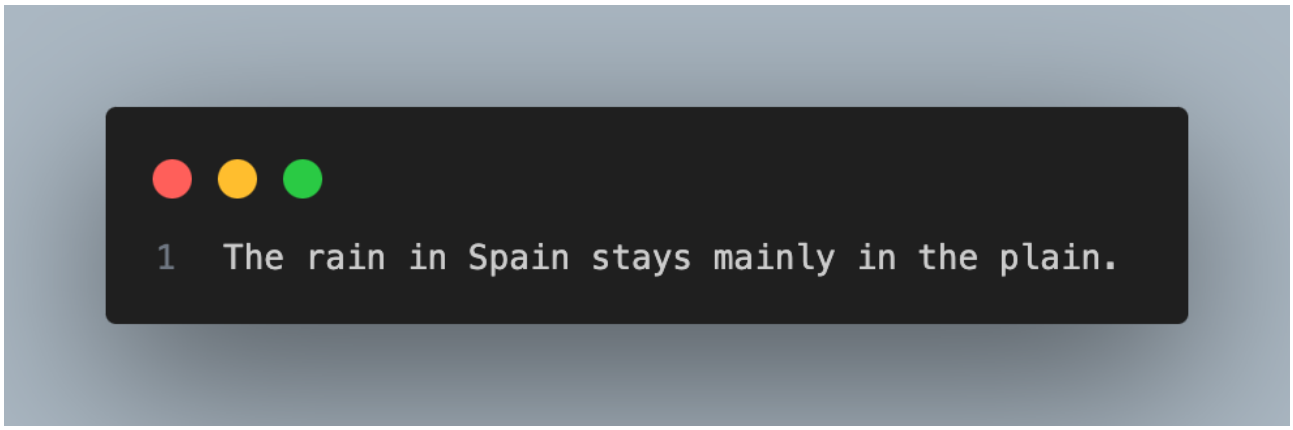
Aim: Create a program that uses regular expressions to find all instances of a specific pattern in a text file.

Software Used: Visual Studio Code.

Code:

```python
import re

# Function to find a pattern in a file using regex
def find_pattern_in_file(file_name, pattern):
    try:
        # Open the file and read its content
        with open(file_name, 'r') as file:
            content = file.read()

            # Find all occurrences of the pattern
            matches = re.findall(pattern, content)
            return matches
    except FileNotFoundError:
        print(f"Error: The file '{file_name}' was not found.")
        return []
    except Exception as e:
        print(f"An error occurred: {e}")
        return []
# Taking user input for file name and regex pattern
file_name = input("Enter the filename (including extension): ")
pattern = input("Enter the regex pattern to search for: ")
# Calling the function and storing the result
matches = find_pattern_in_file(file_name, pattern)
# Displaying the results
if matches:
    print(f"\nFound {len(matches)} matches:")
    for match in matches:
        print(match)
else:
    print("\nNo matches found.")
# Thanking the user
print("\nThank you for using the regex pattern matching program!")
```

Sample.txt File



Output:

```
base ~/Desktop/PIPlab git:(main)±11 (10.577s)
python3 -m exp9
Enter the filename (including extension): sample.txt
Enter the regex pattern to search for: ain

Found 4 matches:
ain
ain
ain
ain

Thank you for using the regex pattern matching program!
```

Experiment 10

Aim: Create a program that prompts users for two numbers and then divides them, handling any exception that may arise.

Software Used: Visual Studio Code.

Code:

```python
# Function to divide two numbers with exception handling
def divide_numbers():
    try:
        # Taking user input for two numbers
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        # Performing division
        result = num1 / num2
        # Displaying the result
        print(f"\nThe result of {num1} divided by {num2} is: {result:.2f}")
    # Handling invalid numeric input
    except ValueError:
        print("\nError: Please enter valid numeric values.")
    # Handling division by zero
    except ZeroDivisionError:
        print("\nError: Division by zero is not allowed.")
    # Handling other unexpected errors
    except Exception as e:
        print(f"\nAn unexpected error occurred: {e}")
# Calling the function
divide_numbers()
```

Output:

```
base ~/Desktop/PIPlab git:(main)±12 (8.124s)
python3 -m exp10
Enter the first number: 10
Enter the second number: 0

Error: Division by zero is not allowed.
```

```
base ~/Desktop/PIPlab git:(main)±13 (3.607s)
python3 -m exp10
Enter the first number: 10
Enter the second number: 2

The result of 10.0 divided by 2.0 is: 5.00
```

Experiment 11

Aim: Create a program that uses a graphical user interface (GUI) to allow users to perform simple calculations.

Software Used: Visual Studio Code.

Code:

```python
import tkinter as tk

# Function to handle button clicks
def on_click(button_text):
    current_text = entry_var.get()
    if button_text == "=":
        try:
            # Evaluate the expression and update the entry
            result = str(eval(current_text))
            entry_var.set(result)
        except Exception:
            entry_var.set("Error")
    elif button_text == "C":
        # Clear the entry field
        entry_var.set("")
    else:
        # Append the clicked button text to the entry field
        entry_var.set(current_text + button_text)
# Create the main window
root = tk.Tk()
root.title("Simple Calculator")
root.configure(bg="#f0f0f0")
# Entry field for displaying input and results
entry_var = tk.StringVar()
entry = tk.Entry(root, textvariable=entry_var, font=("Arial", 24), justify='right', bd=8, relief=tk.GROOVE)
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)
```

```python
# Define button labels
buttons = [
    ('7', '8', '9', '/'),
    ('4', '5', '6', '*'),
    ('1', '2', '3', '-'),
    ('C', '0', '=', '+')
]
# Create buttons and add them to the grid
for r, row in enumerate(buttons):
    for c, text in enumerate(row):
        button = tk.Button(root, text=text, font=("Arial", 20), width=5, height=2,
                           bg="#e0e0e0", fg="#333",
                           command=lambda t=text: on_click(t))
        button.grid(row=r + 1, column=c, padx=5, pady=5)
# Run the Tkinter event loop
root.mainloop()
```

Output:

```
base ~/Desktop/PIPlab git:(main)±13 (59.654s)
python3 -m exp11
2025-03-19 22:32:41.721 python3[10581:370104] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-19 22:32:41.721 python3[10581:370104] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

Experiment 12

Aim: Create a program that uses a web-scrapping library to extract data from a website and then stores it in a database.

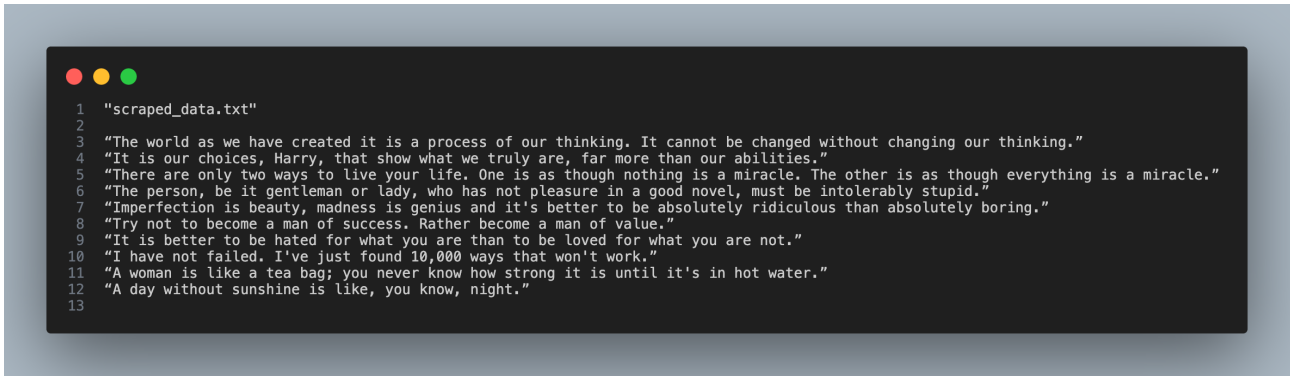
Software Used: Visual Studio Code.

Code:

```python
import requests
from bs4 import BeautifulSoup

# URL to scrape
url = "http://quotes.toscrape.com"

# Send GET request to the URL
response = requests.get(url)

# Check if the request was successful (Status code 200)
if response.status_code == 200:
    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(response.text, "html.parser")

    # Extract quotes using the specific CSS class
    quotes = [quote.text.strip() for quote in soup.find_all("span", class_="text")]

    # Save the extracted quotes to a text file
    with open("scraped_data.txt", "w", encoding="utf-8") as file:
        for quote in quotes:
            file.write(quote + "\n")

    print("✅ Data scraped and stored successfully in 'scraped_data.txt'!")
else:
    print(f"❌ Failed to fetch page, status code: {response.status_code}")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±14 (3.047s)
python3 -m exp12
✅ Data scraped and stored successfully in 'scraped_data.txt'!
```



```
"scraped_data.txt"

“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”
“It is our choices, Harry, that show what we truly are, far more than our abilities.”
“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”
“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”
“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”
“Try not to become a man of success. Rather become a man of value.”
“It is better to be hated for what you are than to be loved for what you are not.”
“I have not failed. I've just found 10,000 ways that won't work.”
“A woman is like a tea bag; you never know how strong it is until it's in hot water.”
“A day without sunshine is like, you know, night.”

```

Experiment 13

Aim: Create a program that reads data from a file and then creates a visualization of the data using a data visualization library.

Software Used: Visual Studio Code.

Code:

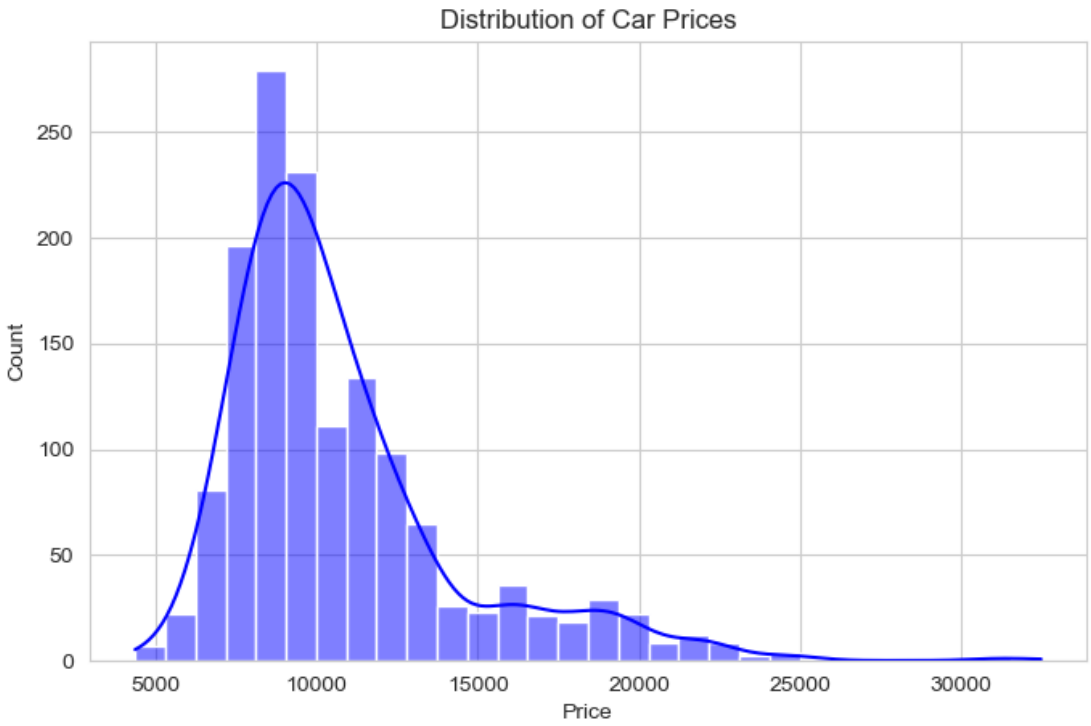
```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# File path to the dataset
file_path = "Toyota.csv"

# Load the dataset into a pandas DataFrame
df = pd.read_csv(file_path)

# Convert relevant columns to numeric, handling errors gracefully
df['KM'] = pd.to_numeric(df['KM'], errors='coerce')   # Convert KM column to numeric
df['HP'] = pd.to_numeric(df['HP'], errors='coerce')   # Convert HP column to numeric
df['Doors'] = pd.to_numeric(df['Doors'], errors='coerce') # Convert Doors column to numeric

# Set Seaborn style for better plot aesthetics
sns.set_style("whitegrid")
```

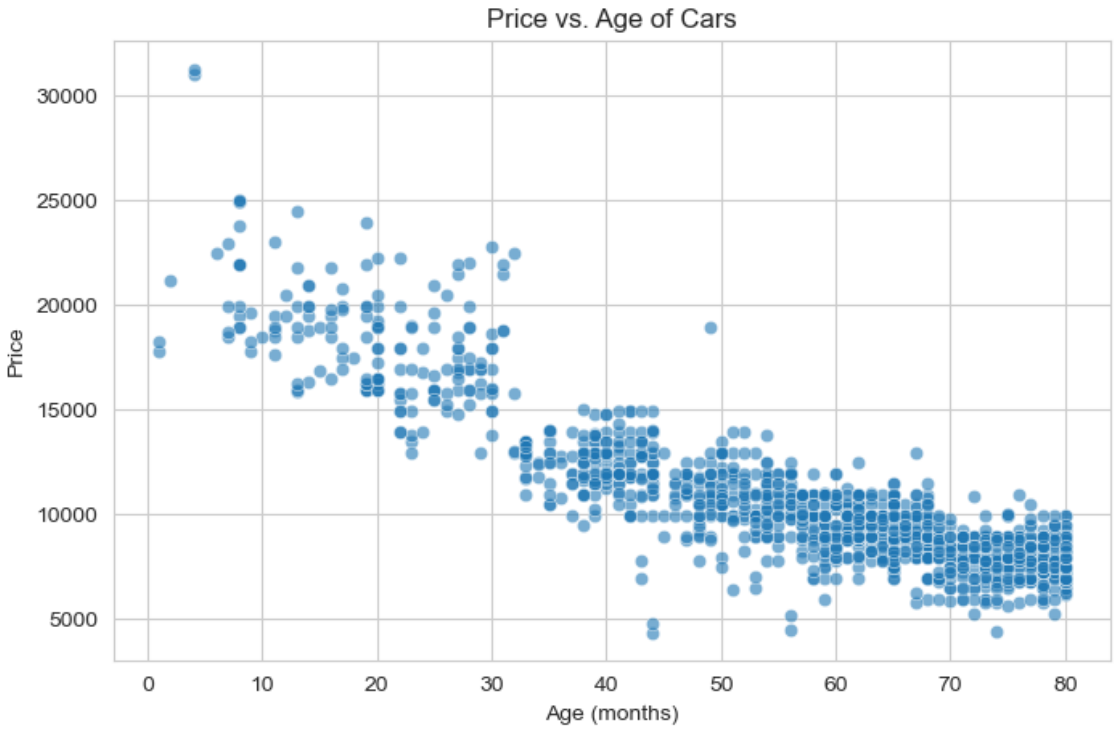
```python
# ============================
# 1. Distribution of Car Prices
# ============================
plt.figure(figsize=(8, 5))
sns.histplot(df['Price'], bins=30, kde=True, color='blue')
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()

# ============================
# 2. Price vs. Age of Cars
# ============================
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df['Age'], y=df['Price'], alpha=0.6)
plt.title('Price vs. Age of Cars')
plt.xlabel('Age (months)')
plt.ylabel('Price')
plt.show()

```

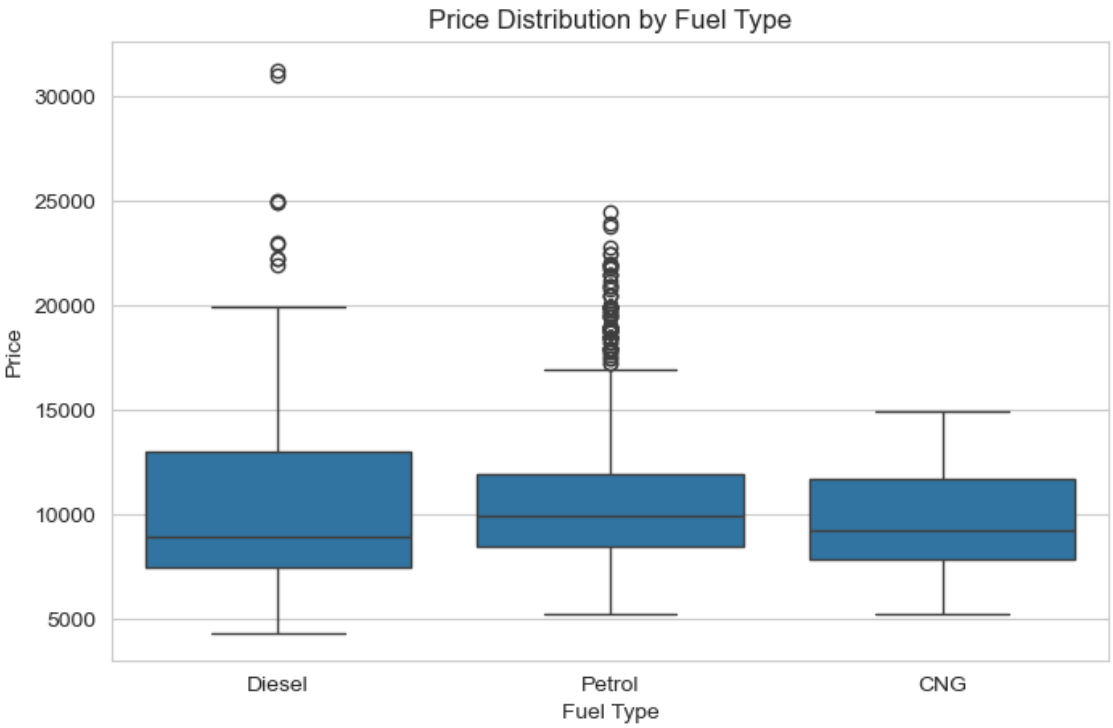
```python
# ============================
# 3. Price Distribution by Fuel Type
# ============================
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['FuelType'], y=df['Price'])
plt.title('Price Distribution by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Price')
plt.show()

# ============================
# 4. Count of Cars by Fuel Type
# ============================
plt.figure(figsize=(8, 5))
sns.countplot(x=df['FuelType'], palette='pastel')
plt.title('Count of Cars by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Count')
plt.show()
```

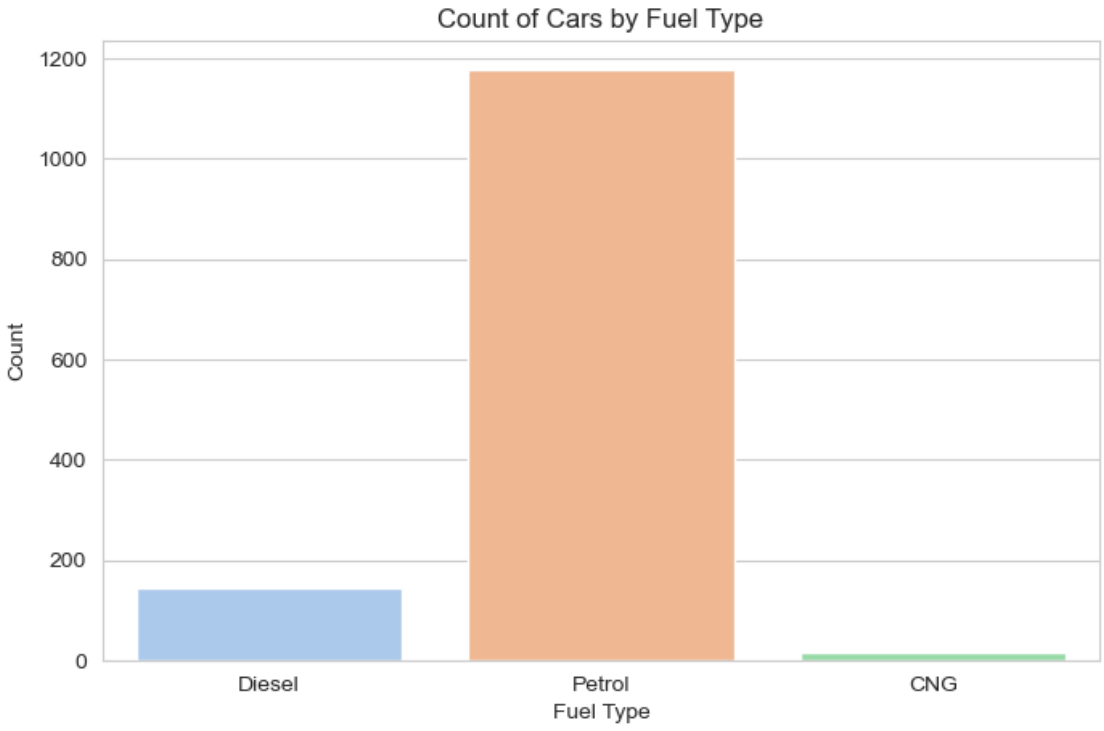
Output:

```
base ~/Desktop/PIPlab git:(main)±15 (53.336s)
python3 -m exp13
2025-03-19 22:57:35.595 python3[11116:388947] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-19 22:57:35.595 python3[11116:388947] +[IMKInputSession subclass]: chose IMKInputSession_Modern
2025-03-19 22:57:41.965 python3[11116:388947] The class 'NSSavePanel' overrides the method identifier.  This method is implemented
 by class 'NSWindow'
/Users/thehalfbldprinc3/Desktop/PIPlab/exp13.py:54: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `
legend=False` for the same effect.

  sns.countplot(x=df['FuelType'], palette='pastel')
```


Distribution of Car Prices
Count
Price
0
50
100
150
200
250
5000
10000
15000
20000
25000
30000


Price vs. Age of Cars
Price
Age (months)
5000
10000
15000
20000
25000
30000
0
10
20
30
40
50
60
70
80


Price Distribution by Fuel Type
Price
30000
25000
20000
15000
10000
5000
Diesel
Petrol
CNG
Fuel Type


Count of Cars by Fuel Type
Count
1200
1000
800
600
400
200
0
Diesel
Petrol
CNG
Fuel Type

Experiment 14

Aim: Create a program that uses a machine learning library to classify images based on their content.

Software Used: Visual Studio Code.

Code:

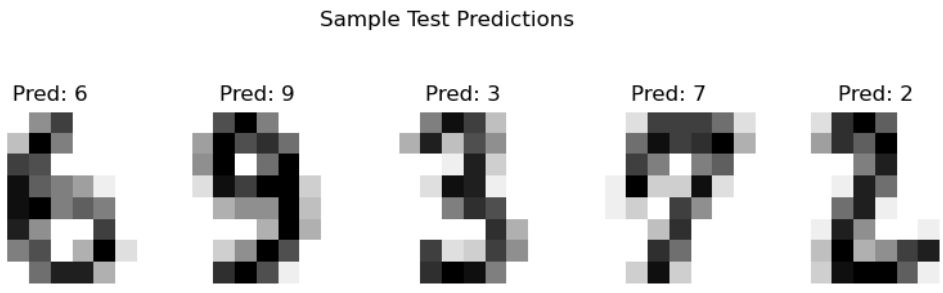
```python
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
# ============================
# 1. Load the Digits Dataset
# ============================
digits = datasets.load_digits()
# ============================
# 2. Split the Data into Training and Test Sets
# ============================
X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, test_size=0.2, random_state=42
)
```

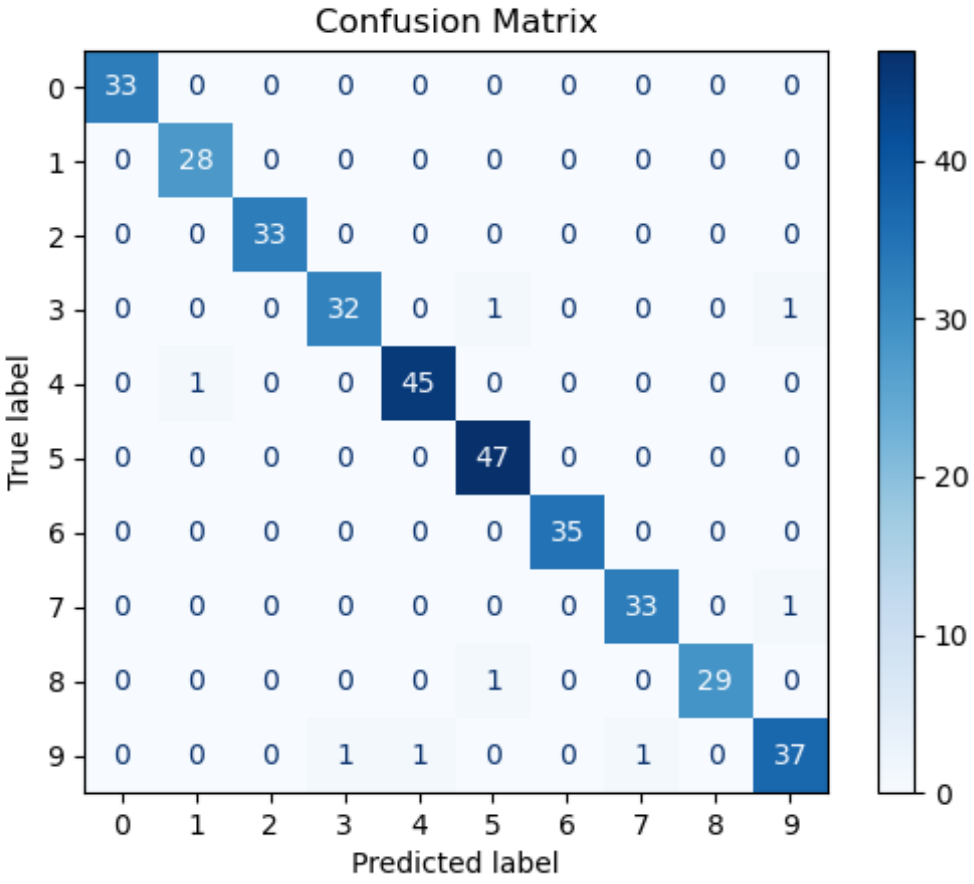
```python
# ============================
# 3. Create and Train the SVM Classifier
# ============================
clf = SVC(kernel='linear')   # Using a linear kernel
clf.fit(X_train, y_train)    # Train the model
# ============================
# 4. Make Predictions
# ============================
y_pred = clf.predict(X_test)
# ============================
# 5. Evaluate the Model
# ============================
# Accuracy Score
print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")
# Classification Report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```python
# ============================
# 6. Display the Confusion Matrix
# ============================
disp = ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
# ============================
# 7. Display Sample Predictions
# ============================
fig, axes = plt.subplots(1, 5, figsize=(10, 3))
for i, ax in enumerate(axes):
    ax.set_axis_off()
    ax.imshow(X_test[i].reshape(8, 8), cmap=plt.cm.gray_r)
    ax.set_title(f"Pred: {y_pred[i]}")
plt.suptitle("Sample Test Predictions")
plt.show()
```

Output:

```
base ~/Desktop/PIPlab git:(main)±17 (30.761s)
python3 -m exp14
Accuracy: 97.78%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.97      1.00      0.98        28
           2       1.00      1.00      1.00        33
           3       0.97      0.94      0.96        34
           4       0.98      0.98      0.98        46
           5       0.96      1.00      0.98        47
           6       1.00      1.00      1.00        35
           7       0.97      0.97      0.97        34
           8       1.00      0.97      0.98        30
           9       0.95      0.93      0.94        40

    accuracy                           0.98       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.98      0.98       360

2025-03-19 23:10:29.843 python3[11383:399447] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-19 23:10:29.843 python3[11383:399447] +[IMKInputSession subclass]: chose IMKInputSession_Modern
2025-03-19 23:10:33.205 python3[11383:399447] The class 'NSSavePanel' overrides the method identifier.  This method is implemented
 by class 'NSWindow'
```




Confusion Matrix
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
33 0 0 0 0 0 0 0 0 0
0 28 0 0 0 0 0 0 0 0
0 0 33 0 0 0 0 0 0 0
0 0 0 32 0 1 0 0 0 1
0 1 0 0 45 0 0 0 0 0
0 0 0 0 0 47 0 0 0 0
0 0 0 0 0 0 35 0 0 0
0 0 0 0 0 0 0 33 0 1
0 0 0 0 0 1 0 0 29 0
0 0 0 1 1 0 0 1 0 37
40
30
20
10
0

Experiment 15

Aim: Create a program that uses a networking library to communicate with a server and retrieves data from it.

Software Used: Visual Studio Code.

Code:

Server Side:

```python
import socket

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to a specific address and port
HOST = '127.0.0.1'  # Localhost
PORT = 12345
server_socket.bind((HOST, PORT))

# Start listening for connections
server_socket.listen()

print(f"Server is listening on {HOST}:{PORT}")

# Accept a connection
client_socket, addr = server_socket.accept()
print(f"Connected by {addr}")

# Receive data from the client
data = client_socket.recv(1024).decode('utf-8')
print(f"Received: {data}")

# Send a response
response = "Hello from the server!"
client_socket.send(response.encode('utf-8'))

# Close the connection
client_socket.close()
server_socket.close()
```

Client Side:

```python
import socket

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
HOST = '127.0.0.1'
PORT = 12345
client_socket.connect((HOST, PORT))

# Send a message
message = "Hello from the client!"
client_socket.send(message.encode('utf-8'))

# Receive a response
response = client_socket.recv(1024).decode('utf-8')
print(f"Server Response: {response}")

# Close the connection
client_socket.close()
```

Output:

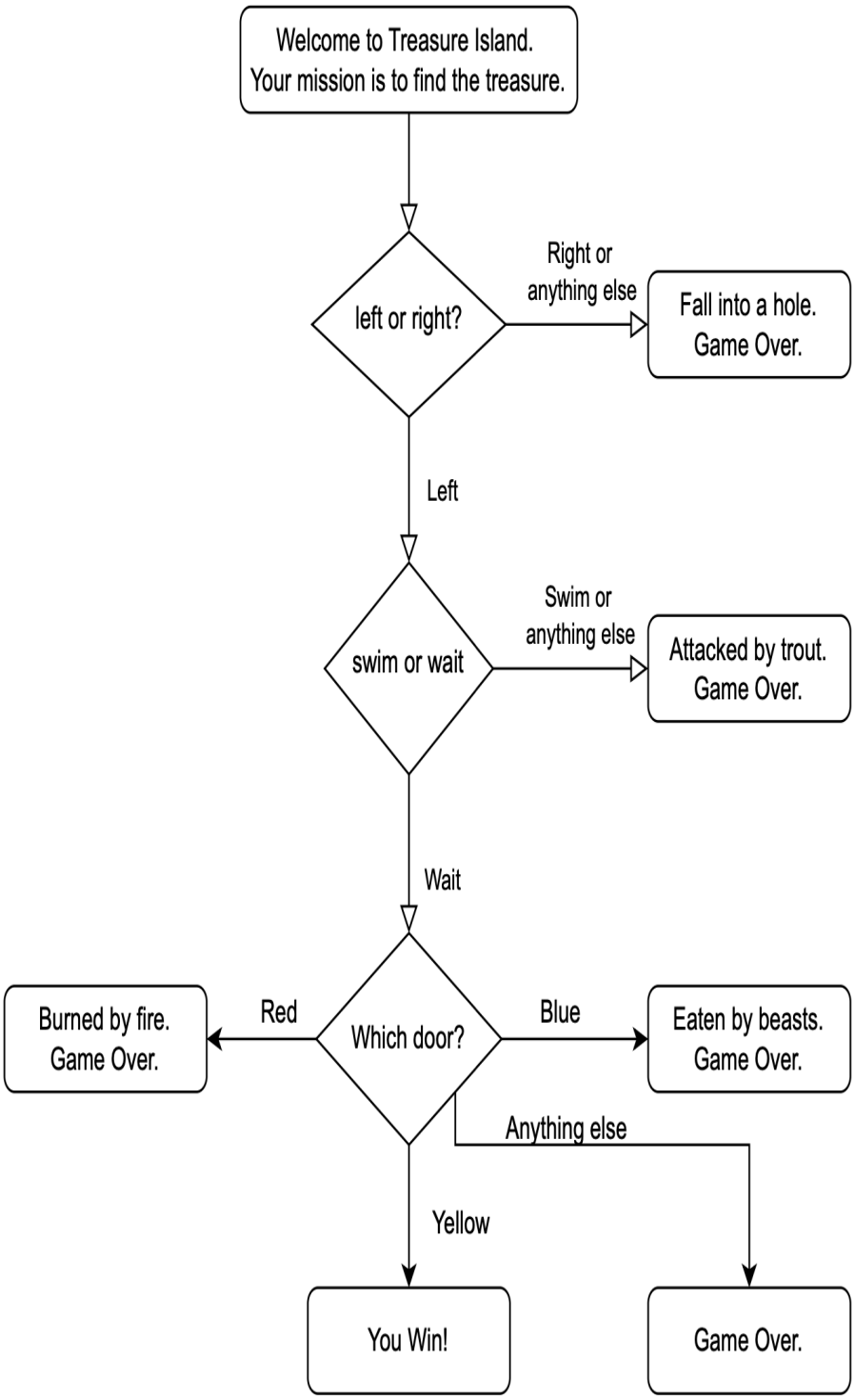
```
base ~/Desktop/PIPlab git:(main)±18 (48.38s)
python3 -m exp15Server

Server is listening on 127.0.0.1:12345
Connected by ('127.0.0.1', 50277)
Received: Hello from the client!
```

```
~/Desktop/PIPlab (0.064s)
python3 -m exp15Client
Server Response: Hello from the server!
```

Experiment 1

Aim: Create a program to simulate the Treasure Island Game Program.

Flow Chart:



Code:

```python
def treasure_island():
    print("Welcome to Treasure Island.")
    print("Your mission is to find the treasure.")

    choice1 = input("left or right? ").strip().lower()
    if choice1 != "left":
        print("Fall into a hole. Game Over.")
        return

    choice2 = input("swim or wait? ").strip().lower()
    if choice2 != "wait":
        print("Attacked by trout. Game Over.")
        return

    choice3 = input("Which door? Red, Blue, or Yellow: ").strip().lower()
    if choice3 == "red":
        print("Burned by fire. Game Over.")
    elif choice3 == "blue":
        print("Eaten by beasts. Game Over.")
    elif choice3 == "yellow":
        print("You Win!")
    else:
        print("Game Over.")

# Start the game
treasure_island()
```

Output:

```
base ~/Desktop/PIPlab git:(main)±8 (22.727s)
python3 -m AddOntreasureIsland
Welcome to Treasure Island.
Your mission is to find the treasure.
left or right? left
swim or wait? wait
Which door? Red, Blue, or Yellow: red
Burned by fire. Game Over.
```

```
base ~/Desktop/PIPlab git:(main)±8 (9.33s)
python3 -m AddOntreasureIsland
Welcome to Treasure Island.
Your mission is to find the treasure.
left or right? left
swim or wait? wait
Which door? Red, Blue, or Yellow: yellow
You Win!
```

```
base ~/Desktop/PIPlab git:(main)±8 (8.883s)
python3 -m AddOntreasureIsland
Welcome to Treasure Island.
Your mission is to find the treasure.
left or right? left
swim or wait? wait
Which door? Red, Blue, or Yellow: blue
Eaten by beasts. Game Over.
```

Experiment 2

Aim: To develop a Python program that simulates the classic game of Rock, Paper, Scissors between a user and the computer, using conditional statements, random number generation, and ASCII art to enhance user interaction and visualize game outcomes.

Code:

```python
import random

# ASCII Art
rock = '''
    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)
'''

paper = '''
    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)
'''

scissors = '''
    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)
'''

```

```python
# Game images list
game_images = [rock, paper, scissors]

# User input
user_choice = int(input("What do you choose? Type 0 for Rock, 1 for Paper or 2 for Scissors:\n"))

if user_choice >= 0 and user_choice <= 2:
    print("You chose:")
    print(game_images[user_choice])
else:
    print("Invalid choice. You lose!")
    exit()

# Computer choice
computer_choice = random.randint(0, 2)
print("Computer chose:")
print(game_images[computer_choice])

# Game logic
if user_choice == computer_choice:
    print("It's a draw!")
elif user_choice == 0 and computer_choice == 2:
    print("You win!")
elif user_choice == 1 and computer_choice == 0:
    print("You win!")
elif user_choice == 2 and computer_choice == 1:
    print("You win!")
else:
    print("You lose.")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±8 (2.407s)
python3 -m AddOnrockPaperScissors
What do you choose? Type 0 for Rock, 1 for Paper or 2 for Scissors:
2
You chose:


    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

Computer chose:


    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

It's a draw!
```

Experiment 3

Aim: To create a Python-based file management system that performs common file operations such as read, write, append, copy, and word count through a user-interactive menu-driven interface.

Code:

```python
import os

def get_filename(prompt, default):
    name = input(f"{prompt} (default: {default}): ").strip()
    return name if name else default

# 1. Writing to a File
def write_to_file():
    filename = get_filename("Enter filename to write to", "sample.txt")
    content = input("Enter content to write to the file: ")
    with open(filename, 'w') as file:
        file.write(content + "\n")
    print(f"Written to {filename}.")

# 2. Reading from a File
def read_file():
    filename = get_filename("Enter filename to read from", "sample.txt")
    if os.path.exists(filename):
        with open(filename, 'r') as file:
            content = file.read()
        print("File contents:\n", content)
    else:
        print("File does not exist.")

# 3. Appending Data to a File
def append_to_file():
    filename = get_filename("Enter filename to append to", "sample.txt")
    content = input("Enter content to append: ")
    with open(filename, 'a') as file:
        file.write(content + "\n")
    print("Data appended.")

```

```python
# 4. Reading a File Line by Line
def read_line_by_line():
    filename = get_filename("Enter filename to read line-by-line", "sample.txt")
    if os.path.exists(filename):
        with open(filename, 'r') as file:
            print("Reading line by line:")
            for line in file:
                print(line.strip())
    else:
        print("File does not exist.")

# 5. Counting Words in a File
def count_words():
    filename = get_filename("Enter filename to count words in", "sample.txt")
    if os.path.exists(filename):
        with open(filename, 'r') as file:
            words = file.read().split()
        print("Word count:", len(words))
    else:
        print("File does not exist.")

# 6. Copying File Content
def copy_content():
    source = get_filename("Enter source filename", "sample.txt")
    dest = get_filename("Enter destination filename", "copy.txt")
    if os.path.exists(source):
        with open(source, 'r') as src, open(dest, 'w') as dst:
            dst.write(src.read())
        print(f"Copied content to {dest}.")
    else:
        print("Source file does not exist.")

# 7. Checking if a File Exists
def check_file_exists():
    filename = get_filename("Enter filename to check", "sample.txt")
    exists = os.path.exists(filename)
    print(f"File exists: {exists}")
    return exist
```

```python
# 8. Writing a List to a File
def write_list_to_file():
    filename = get_filename("Enter filename to write list into", "sample.txt")
    data = input("Enter list items separated by commas: ").split(',')
    with open(filename, 'w') as file:
        for item in data:
            file.write(item.strip() + "\n")
    print("List written to file.")

# Menu to run functions
def menu():
    print("""
1. Write to file
2. Read file
3. Append to file
4. Read line by line
5. Count words
6. Copy content to another file
7. Check file exists
8. Write list to file
9. Exit
    """)
```

```python
    while True:
        choice = input("Enter your choice: ")
        if choice == '1':
            write_to_file()
        elif choice == '2':
            read_file()
        elif choice == '3':
            append_to_file()
        elif choice == '4':
            read_line_by_line()
        elif choice == '5':
            count_words()
        elif choice == '6':
            copy_content()
        elif choice == '7':
            check_file_exists()
        elif choice == '8':
            write_list_to_file()
        elif choice == '9':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    menu()
```

Output:

```
base ~/Desktop/PIPlab git:(main)±8
python3 -m AddOnfileManagement

1. Write to file
2. Read file
3. Append to file
4. Read line by line
5. Count words
6. Copy content to another file
7. Check file exists
8. Write list to file
9. Exit

Enter your choice: 1
Enter filename to write to (default: sample.txt): sample.txt
Enter content to write to the file: this is a test input to sample.txt using fileManagement script
Written to sample.txt.
Enter your choice: 2
Enter filename to read from (default: sample.txt): sample.txt
File contents:
 this is a test input to sample.txt using fileManagement script
```

```
Enter your choice: 3
Enter filename to append to (default: sample.txt): sample.txt
Enter content to append: this is appended content
Data appended.
Enter your choice: 2
Enter filename to read from (default: sample.txt):
File contents:
 this is a test input to sample.txt using fileManagement script
this is appended content

Enter your choice: 5
Enter filename to count words in (default: sample.txt):
Word count: 14
Enter your choice:
```

Experiment 4

Aim: To develop a GUI-based Student Management System using Python and Tkinter that allows users to add, view, update, search, and delete student records stored in a CSV file.

Code:

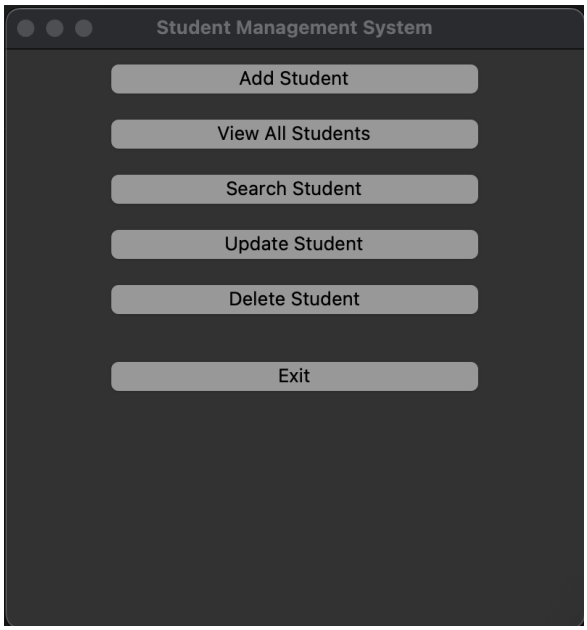
```
import tkinter as tk
from tkinter import messagebox, simpledialog, ttk
import pandas as pd
import os

FILENAME = "students.csv"

```

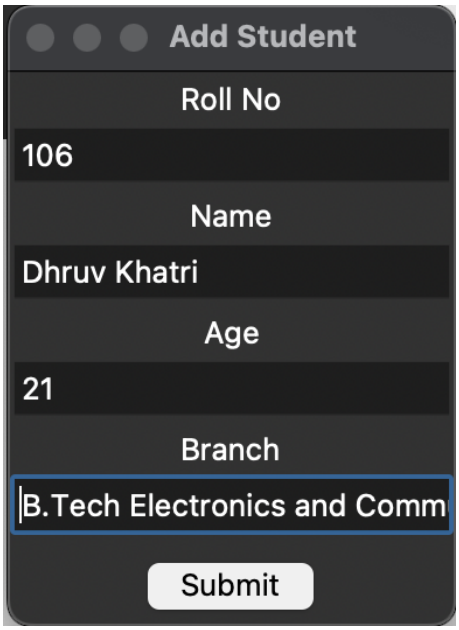
```python
# Ensure the file exists with correct headers
def initialize_file():
    if not os.path.exists(FILENAME):
        df = pd.DataFrame(columns=["Roll No", "Name", "Age", "Branch"])
        df.to_csv(FILENAME, index=False)

def add_student(roll, name, age, branch):
    df = pd.read_csv(FILENAME)
    if str(roll) in df["Roll No"].astype(str).values:
        return False
    df.loc[len(df.index)] = [roll, name, age, branch]
    df.to_csv(FILENAME, index=False)
    return True

def get_all_students():
    return pd.read_csv(FILENAME)

def search_student(roll):
    df = pd.read_csv(FILENAME)
    result = df[df["Roll No"] == roll]
    return result if not result.empty else None

def update_student(roll, name, age, branch):
    df = pd.read_csv(FILENAME)
    if roll not in df["Roll No"].values:
        return False
    df.loc[df["Roll No"] == roll] = [roll, name, age, branch]
    df.to_csv(FILENAME, index=False)
    return True

def delete_student(roll):
    df = pd.read_csv(FILENAME)
    df = df[df["Roll No"] != roll]
    df.to_csv(FILENAME, index=False)

```

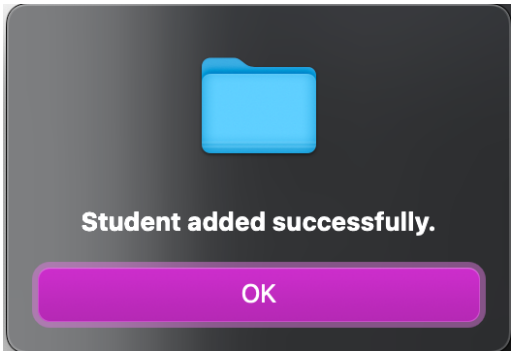
```python
# GUI
class StudentGUI:
    def __init__(self, master):
        self.master = master
        master.title("Student Management System")
        master.geometry("400x400")

        tk.Button(master, text="Add Student", width=25, command=self.add_student_window).pack(pady=5)
        tk.Button(master, text="View All Students", width=25, command=self.view_students).pack(pady=5)
        tk.Button(master, text="Search Student", width=25, command=self.search_student_window).pack(pady=5)
        tk.Button(master, text="Update Student", width=25, command=self.update_student_window).pack(pady=5)
        tk.Button(master, text="Delete Student", width=25, command=self.delete_student_window).pack(pady=5)
        tk.Button(master, text="Exit", width=25, command=master.quit).pack(pady=20)

    def add_student_window(self):
        self.input_window("Add Student", self.add_student_action)

    def update_student_window(self):
        self.input_window("Update Student", self.update_student_action)

    def input_window(self, title, action):
        win = tk.Toplevel(self.master)
        win.title(title)

        labels = ["Roll No", "Name", "Age", "Branch"]
        entries = []

        for label in labels:
            tk.Label(win, text=label).pack()
            entry = tk.Entry(win)
            entry.pack()
            entries.append(entry)

        def on_submit():
            values = [e.get().strip() for e in entries]
            if "" in values:
                messagebox.showerror("Error", "All fields are required.")
                return
            try:
                roll = int(values[0])
                age = int(values[2])
            except:
                messagebox.showerror("Error", "Roll No and Age must be integers.")
                return
            action(roll, values[1], age, values[3])
            win.destroy()

        tk.Button(win, text="Submit", command=on_submit).pack(pady=5)

    def add_student_act
```

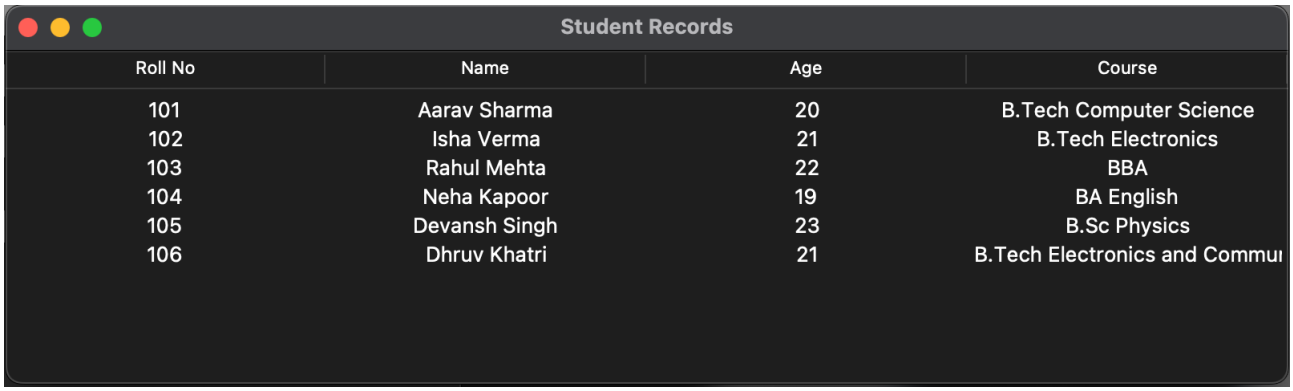
```python
    def add_student_action(self, roll, name, age, branch):
        if add_student(roll, name, age, branch):
            messagebox.showinfo("Success", "Student added successfully.")
        else:
            messagebox.showerror("Error", "Roll No already exists.")

    def update_student_action(self, roll, name, age, branch):
        if update_student(roll, name, age, branch):
            messagebox.showinfo("Success", "Student updated successfully.")
        else:
            messagebox.showerror("Error", "Student not found.")

    def view_students(self):
        data = get_all_students()
        self.display_table(data)

    def search_student_window(self):
        roll = simpledialog.askinteger("Search Student", "Enter Roll No:")
        if roll is not None:
            result = search_student(roll)
            if result is not None:
                self.display_table(result)
            else:
                messagebox.showinfo("Result", "Student not found.")

    def delete_student_window(self):
        roll = simpledialog.askinteger("Delete Student", "Enter Roll No:")
        if roll is not None:
            delete_student(roll)
            messagebox.showinfo("Success", "Student deleted (if existed).")

    def display_table(self, df):
        win = tk.Toplevel(self.master)
        win.title("Student Records")
        tree = ttk.Treeview(win)
        tree["columns"] = list(df.columns)
        tree["show"] = "headings"
        for col in df.columns:
            tree.heading(col, text=col)
            tree.column(col, anchor=tk.CENTER)
        for i in df.values.tolist():
            tree.insert("", "end", values=i)
        tree.pack(expand=True, fill="both")

# Main
if __name__ == "__main__":
    initialize_file()
    root = tk.Tk()
    app = StudentGUI(root)
    root.mainloop()
```

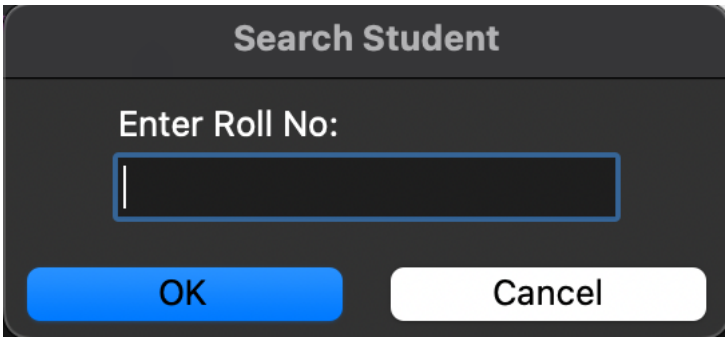
Output:

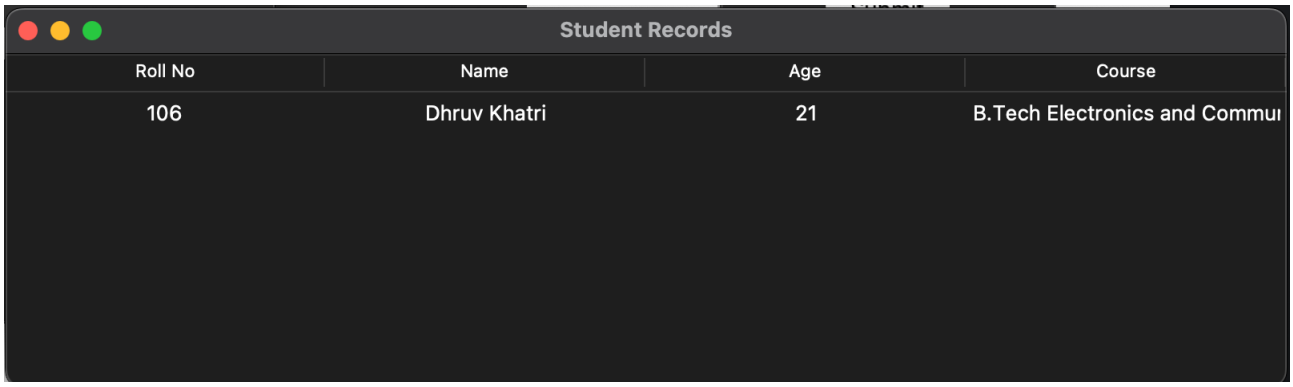

Student Management System
Add Student
View All Students
Search Student
Update Student
Delete Student
Exit

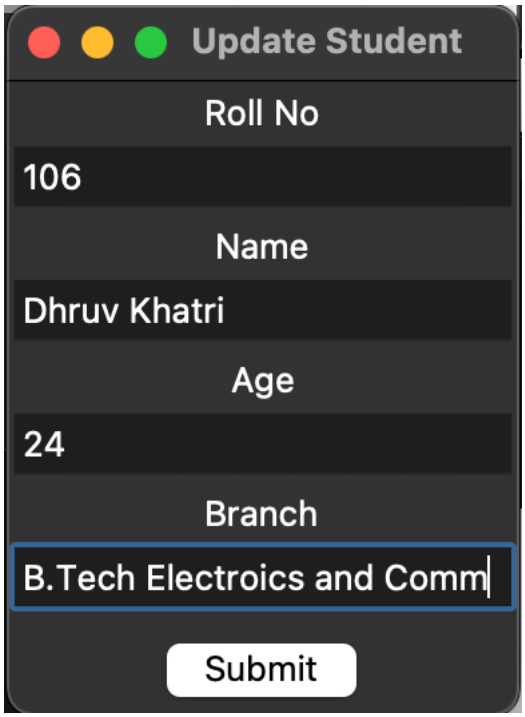

Add Student
Roll No
106
Name
Dhruv Khatri
Age
21
Branch
B.Tech Electronics and Comm
Submit

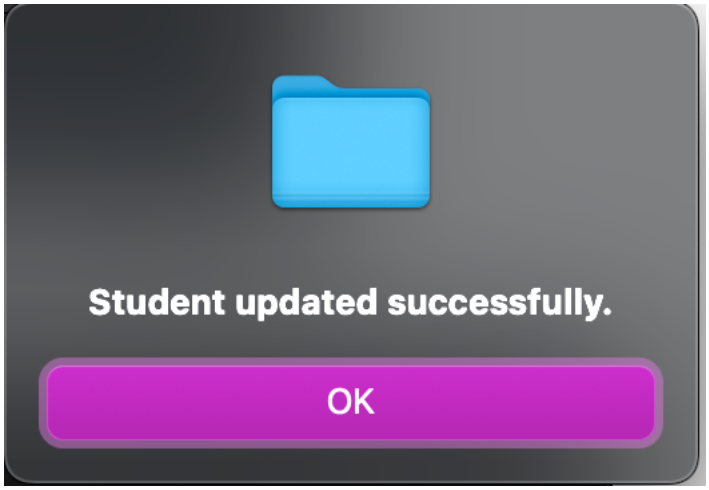

Student added successfully.
OK

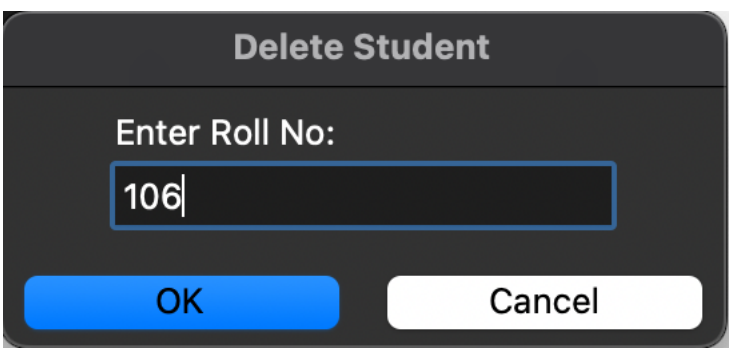

Student Records
Roll No
Name
Age
Course
101
Aarav Sharma
20
B.Tech Computer Science
102
Isha Verma
21
B.Tech Electronics
103
Rahul Mehta
22
BBA
104
Neha Kapoor
19
BA English
105
Devansh Singh
23
B.Sc Physics
106
Dhruv Khatri
21
B.Tech Electronics and Commu

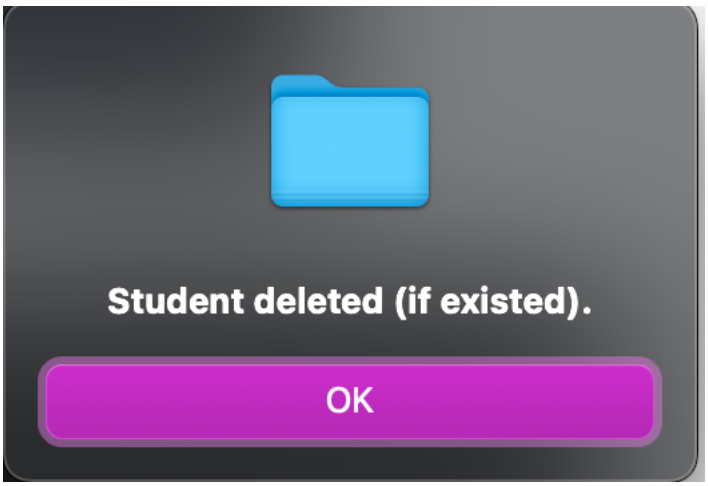

Search Student
Enter Roll No:
OK
Cancel


Student Records
Roll No
Name
Age
Course
106
Dhruv Khatri
21
B.Tech Electronics and Commu


Update Student
Roll No
106
Name
Dhruv Khatri
Age
24
Branch
B.Tech Electroics and Comm
Submit


Student updated successfully.
OK


Delete Student
Enter Roll No:
106
OK
Cancel


Student deleted (if existed).
OK

Add-On Experiment

Programming in Python (IOT-330P)

Lab Record

Name:_____ Enrollment No:_____ Branch:_____

S.No	Experiment Aim	Date	Sign
1	Create a program that prompts the user for their name and age and prints a personalized message.		
2	Create a program that prompts user for their age and tells them if they can vote in the next election.		
3	Create a program that calculates the factorial of a number entered by user using a loop.		
4	Create a program that prompts the user for a list of numbers and then sorts them in ascending order.		
5	Create a program that prompts the user for a string and then prints the string reversed.		
6	Create a program that defines a function to calculate the area of a circle based on the radius of the circle entered by the user.		
7	Create a program that defines a class to represent a car and then create an object of that class with specific attributes.		
8	Create a program that reads data from a file and writes it to another file in a format.		
9	Create a program that uses regular expressions to find all instances of a specific pattern in a text file.		
10	Create a program that prompts users for two numbers and then divides them, handling any exception that may arise.		
11	Create a program that uses a graphical user interface (GUI) to allow users to perform simple calculations.		
12	Create a program that uses a web-scrapping library to extract data from a website and then stores it in a database.		
Add on Experiment			
1	Create a program to simulate the Treasure Island Game Program.		
2	To develop a Python program that simulates the classic game of Rock, Paper, Scissors between a user and the computer, using conditional statements, random number generation, and ASCII art to enhance user interaction and visualize game outcomes.		

3	To create a Python-based file management system that performs common file operations such as read, write, append, copy, and word count through a user-interactive menu-driven interface.		
4	To develop a GUI-based Student Management System using Python and Tkinter that allows users to add, view, update, search, and delete student records stored in a CSV file.		
5	To develop a Python script that generates a Spotify playlist of the Billboard Hot 100 songs for a user-specified historical date by fetching song data from a public JSON API and integrating with the Spotify Web API using OAuth authentication.		

Experiment 5

Aim: To develop a Python script that generates a Spotify playlist of the Billboard Hot 100 songs for a user-specified historical date by fetching song data from a public JSON API and integrating with the Spotify Web API using OAuth authentication.

APIs Used:

- Spotify WebAPI
- github.com/mhollingshead/billboard-hot-100

Code:

```python
import requests
import spotipy
from spotipy.oauth2 import SpotifyOAuth
import os
from dotenv import load_dotenv
load_dotenv()

# --- Spotify credentials ---
CLIENT_ID = os.getenv("SPOTIPY_CLIENT_ID")
CLIENT_SECRET = os.getenv("SPOTIPY_CLIENT_SECRET")
REDIRECT_URI = os.getenv("SPOTIPY_REDIRECT_URI")
SCOPE = "playlist-modify-public"

# --- Spotify Auth ---
spotify = spotipy.Spotify(auth_manager=SpotifyOAuth(
    client_id=CLIENT_ID,
    client_secret=CLIENT_SECRET,
    redirect_uri=REDIRECT_URI,
    scope=SCOPE
))

```

```python
# --- Get date and fetch Billboard JSON ---
date = input("Which year do you want to travel to? Type the date in this format YYYY-MM-DD: ")
url = f"https://raw.githubusercontent.com/mhollingshead/billboard-hot-100/main/date/{date}.json"

response = requests.get(url)
if response.status_code != 200:
    print("Invalid date or chart data not found.")
    exit()

chart = response.json()
songs = [entry['song'] for entry in chart['data']]
artists = [entry['artist'] for entry in chart['data']]

# --- Spotify: Create playlist ---
user_id = spotify.current_user()["id"]
playlist_name = f"Billboard HOT 100 - {date}"
playlist_description = f"Top 100 songs from Billboard on {date}"

# Check if playlist already exists
user_playlists = spotify.current_user_playlists()
existing_names = [playlist["name"] for playlist in user_playlists["items"]]

track_uris = []

```

```python
# --- Search for songs on Spotify ---
for song, artist in zip(songs, artists):
    print(f"Searching: {song} by {artist}")
    result = spotify.search(q=f"track:{song} artist:{artist}", type='track', limit=1)
    items = result["tracks"]["items"]
    if items:
        track_uris.append(items[0]["uri"])
    else:
        print(f"NOT FOUND: {song} by {artist}")

# --- Create or skip playlist ---
if playlist_name in existing_names:
    print("Playlist already exists.")
else:
    playlist = spotify.user_playlist_create(user=user_id, name=playlist_name, description=playlist_description)
    if track_uris:
        spotify.playlist_add_items(playlist["id"], track_uris)
        print(f"Playlist '{playlist_name}' created with {len(track_uris)} tracks.")
    else:
        print("No tracks were added due to missing URIs.")
```

Output:

```
base ~/Desktop/PIPlab git:(main)±2 (36.293s)
python3 -m AddOnSpotifyScapping
Which year do you want to travel to? Type the date in this format YYYY-MM-DD: 2025-04-19
Searching: Luther by Kendrick Lamar & SZA
Searching: Nokia by Drake
Searching: Die With A Smile by Lady Gaga & Bruno Mars
Searching: All The Way by BigXthaPlug Featuring Bailey Zimmerman
```

```
Searching: Khe? by Rauw Alejandro & Romeo Santos
Playlist 'Billboard HOT 100 - 2025-04-19' created with 90 tracks.
```