

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:

```
● ● ●
1 # Program to prompt the user for their name and age, then display a personalized message
2
3 # Taking user input for name and age
4 name = input("Enter your name: ") # Accepts a string input for the name
5 age = input("Enter your age: ")    # Accepts a string input for the age
6
7 # Type conversion: Convert age from string to integer
8 try:
9     age = int(age)
10 except ValueError:
11     print("Invalid age! Please enter a numeric value.")
12     exit()
13
14 # Processing and output
15 # Displaying a personalized message using string formatting
16 print(f"Hello, {name}! You are {age} years old.")
17
18 # Simple operation: Calculate the age in 5 years
19 future_age = age + 5
20 print(f"In 5 years, you will be {future_age} years old.")
21
22 # Thanking the user
23 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:

```
● ● ●
1 # Program to check if a user is eligible to vote
2
3 # Taking user input for age
4 age = input("Enter your age: ")
5
6 # Type conversion and validation
7 try:
8     age = int(age)
9 except ValueError:
10    print("Invalid input! Please enter a numeric value.")
11    exit()
12
13 # Checking voting eligibility
14 if age >= 18:
15     print("You are eligible to vote in the next election.")
16 else:
17     print("You are not eligible to vote yet.")
18
19 # Thanking the user
20 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!
```

Programming in Python Lab (IOT-330P)

```
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:

```
● ● ●

1 # Program to calculate the factorial of a number using a loop
2
3 # Taking user input
4 num = input("Enter a number to calculate its factorial: ")
5
6 # Type conversion and validation
7 try:
8     num = int(num)
9     if num < 0:
10         print("Factorial is not defined for negative numbers.")
11         exit()
12 except ValueError:
13     print("Invalid input! Please enter a numeric value.")
14     exit()
15
16 # Initializing factorial to 1
17 factorial = 1
18
19 # Calculating factorial using a loop
20 for i in range(1, num + 1):
21     factorial *= i
22
23 # Displaying the result
24 print(f"The factorial of {num} is {factorial}.")
25
26 # Thanking the user
27 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:

```
● ● ●
1 # Program to sort a list using a simple sorting algorithm and an in-built function
2
3 # Taking user input for the list size
4 n = int(input("Enter the number of elements required: "))
5 # Creating an empty list
6 l2 = []
7 # Taking list elements from the user
8 print("Enter the elements:")
9 for i in range(n):
10     element = int(input(f"Element {i + 1}: "))
11     l2.append(element)
12 # Displaying the original list
13 print("\nOriginal List:", l2)
14 # Sorting using a simple sorting algorithm (Bubble Sort)
15 for i in range(len(l2)):
16     for j in range(i + 1, len(l2)):
17         if l2[i] > l2[j]:
18             # Swapping elements
19             l2[i], l2[j] = l2[j], l2[i]
20 # Displaying the sorted list
21 print("Sorted List (using custom algorithm):", l2)
22 # --- Sorting using an in-built function ---
23 # Taking input again for comparison
24 l1 = []
25 print("\nUsing in-built sort function:")
26 for i in range(n):
27     element = int(input(f"Element {i + 1}: "))
28     l1.append(element)
29 # Displaying the original list
30 print("\nOriginal List:", l1)
31 # Sorting using Python's built-in `sort()` function
32 l1.sort()
33 # Displaying the sorted list
34 print("Sorted List (using in-built sort):", l1)
35 # Thanking the user
36 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:

```
 1 # Program to reverse a string using a loop and a library function
 2
 3 # Function to reverse a string using a loop
 4 def reverseString(string):
 5     reverseStr = ""
 6     for i in range(len(string) - 1, -1, -1):
 7         reverseStr += string[i]
 8     return reverseStr
 9
10 # Function to reverse a string using a library function
11 def reverseString2(string):
12     return "".join(reversed(string))
13
14 # Taking user input
15 string = input("Enter the string: ")
16
17 # Calling both functions
18 reverseStr = reverseString(string)
19 reverseStr2 = reverseString2(string)
20
21 # Displaying results
22 print("\nOriginal String:", string)
23 print("Reversed String (using loop):", reverseStr)
24 print("Reversed String (using library function):", reverseStr2)
25
26 # Thanking the user
27 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:

```
● ● ●
1 # Importing the math library
2 import math as m
3
4 # Function to calculate the area of a circle
5 def areaOfCircle(radius):
6     area = m.pi * radius**2
7     return area
8
9 # Taking user input
10 try:
11     radius = float(input("Enter the radius of the circle: "))
12     if radius < 0:
13         print("Radius cannot be negative. Please enter a valid value.")
14     else:
15         # Calling the function and displaying the result
16         print(f"The area of the circle with radius {radius} is: {areaOfCircle(radius):.2f}")
17 except ValueError:
18     print("Invalid input! Please enter a numeric value.")
19
20 # Thanking the user
21 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:

```
● ● ●
1 # Defining the Car class
2 class Car:
3     # Class variable (common to all instances)
4     wheels = 4
5
6     # Constructor to initialize object attributes
7     def __init__(self, make, model, color, year):
8         self.make = make
9         self.model = model
10        self.color = color
11        self.year = year
12
13    # Method to display car details
14    def displayCar(self):
15        print(f"Car Make: {self.make}, Model: {self.model}, Wheels: {Car.wheels}, Color: {self.color}, Year: {self.year}")
16
17 # Creating objects of the Car class
18 car1 = Car("Toyota", "Corolla", "Black", "2022")
19 car2 = Car("Honda", "Civic", "Silver", "2023")
20
21 # Displaying car details
22 print("\nCar Details:")
23 car1.displayCar()
24 car2.displayCar()
25
26 # Thanking the user
27 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:

```
● ● ●
1 import os
2
3 # Displaying the list of files before copying
4 print("List of files before copying:")
5 print(os.listdir())
6 print("\n")
7 # Opening file1.txt in read mode
8 with open('file1.txt', 'r') as file1:
9     # Opening file2.txt in write mode
10    with open('file2.txt', 'w') as file2:
11        # Copying content line by line
12        for line in file1:
13            file2.write(line)
14 # Reading and displaying the contents of file2.txt
15 with open('file2.txt', 'r') as file2:
16     print("Contents of file2.txt:\n")
17     print(file2.read())
18 # Displaying the list of files after copying
19 print("\nList of files after copying:")
20 print(os.listdir())
21 print("\n")
22 # Thanking the user
23 print("Thank you for using the file handling program!")
```

Programming in Python Lab (IOT-330P)

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__', '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']

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:

```
 1 import re
 2
 3 # Function to find a pattern in a file using regex
 4 def find_pattern_in_file(file_name, pattern):
 5     try:
 6         # Open the file and read its content
 7         with open(file_name, 'r') as file:
 8             content = file.read()
 9
10         # Find all occurrences of the pattern
11         matches = re.findall(pattern, content)
12         return matches
13     except FileNotFoundError:
14         print(f"Error: The file '{file_name}' was not found.")
15         return []
16     except Exception as e:
17         print(f"An error occurred: {e}")
18         return []
19 # Taking user input for file name and regex pattern
20 file_name = input("Enter the filename (including extension): ")
21 pattern = input("Enter the regex pattern to search for: ")
22 # Calling the function and storing the result
23 matches = find_pattern_in_file(file_name, pattern)
24 # Displaying the results
25 if matches:
26     print(f"\nFound {len(matches)} matches:")
27     for match in matches:
28         print(match)
29 else:
30     print("\nNo matches found.")
31 # Thanking the user
32 print("\nThank you for using the regex pattern matching program!")
```

Sample.txt File



```
1 The rain in Spain stays mainly in the plain.
```

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:

```
1 # Function to divide two numbers with exception handling
2 def divide_numbers():
3     try:
4         # Taking user input for two numbers
5         num1 = float(input("Enter the first number: "))
6         num2 = float(input("Enter the second number: "))
7         # Performing division
8         result = num1 / num2
9         # Displaying the result
10        print(f"\nThe result of {num1} divided by {num2} is: {result:.2f}")
11    # Handling invalid numeric input
12    except ValueError:
13        print("\nError: Please enter valid numeric values.")
14    # Handling division by zero
15    except ZeroDivisionError:
16        print("\nError: Division by zero is not allowed.")
17    # Handling other unexpected errors
18    except Exception as e:
19        print(f"\nAn unexpected error occurred: {e}")
20 # Calling the function
21 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:

```

● ● ●
1 import tkinter as tk
2
3 # Function to handle button clicks
4 def on_click(button_text):
5     current_text = entry_var.get()
6     if button_text == "=":
7         try:
8             # Evaluate the expression and update the entry
9             result = str(eval(current_text))
10            entry_var.set(result)
11        except Exception:
12            entry_var.set("Error")
13    elif button_text == "C":
14        # Clear the entry field
15        entry_var.set("")
16    else:
17        # Append the clicked button text to the entry field
18        entry_var.set(current_text + button_text)
19 # Create the main window
20 root = tk.Tk()
21 root.title("Simple Calculator")
22 root.configure(bg="#f0f0f0")
23 # Entry field for displaying input and results
24 entry_var = tk.StringVar()
25 entry = tk.Entry(root, textvariable=entry_var, font=("Arial", 24), justify='right', bd=8, relief=tk.GROOVE)
26 entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

```

```

● ● ●
1 # Define button labels
2 buttons = [
3     ('7', '8', '9', '/'),
4     ('4', '5', '6', '*'),
5     ('1', '2', '3', '-'),
6     ('C', '0', '=', '+')
7 ]
8 # Create buttons and add them to the grid
9 for r, row in enumerate(buttons):
10    for c, text in enumerate(row):
11        button = tk.Button(root, text=text, font=("Arial", 20), width=5, height=2,
12                            bg="#e0e0e0", fg="#333",
13                            command=lambda t=text: on_click(t))
14        button.grid(row=r + 1, column=c, padx=5, pady=5)
15 # Run the Tkinter event loop
16 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-scraping library to extract data from a website and then stores it in a database.

Software Used: Visual Studio Code.

Code:

```
● ● ●
1 import requests
2 from bs4 import BeautifulSoup
3
4 # URL to scrape
5 url = "http://quotes.toscrape.com"
6
7 # Send GET request to the URL
8 response = requests.get(url)
9
10 # Check if the request was successful (Status code 200)
11 if response.status_code == 200:
12     # Parse the HTML content using BeautifulSoup
13     soup = BeautifulSoup(response.text, "html.parser")
14
15     # Extract quotes using the specific CSS class
16     quotes = [quote.text.strip() for quote in soup.find_all("span", class_="text")]
17
18     # Save the extracted quotes to a text file
19     with open("scraped_data.txt", "w", encoding="utf-8") as file:
20         for quote in quotes:
21             file.write(quote + "\n")
22
23     print("✓ Data scraped and stored successfully in 'scraped_data.txt'!")
24 else:
25     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'!
```

Programming in Python Lab (IOT-330P)

```
● ● ●  
1 "scraped_data.txt"  
2  
3 "The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."  
4 "It is our choices, Harry, that show what we truly are, far more than our abilities."  
5 "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."  
6 "The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."  
7 "Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring."  
8 "Try not to become a man of success. Rather become a man of value."  
9 "It is better to be hated for what you are than to be loved for what you are not."  
10 "I have not failed. I've just found 10,000 ways that won't work."  
11 "A woman is like a tea bag; you never know how strong it is until it's in hot water."  
12 "A day without sunshine is like, you know, night."  
13
```

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:

```

● ● ●
1 # Import necessary libraries
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 # File path to the dataset
7 file_path = "Toyota.csv"
8
9 # Load the dataset into a pandas DataFrame
10 df = pd.read_csv(file_path)
11
12 # Convert relevant columns to numeric, handling errors gracefully
13 df['KM'] = pd.to_numeric(df['KM'], errors='coerce')    # Convert KM column to numeric
14 df['HP'] = pd.to_numeric(df['HP'], errors='coerce')    # Convert HP column to numeric
15 df['Doors'] = pd.to_numeric(df['Doors'], errors='coerce') # Convert Doors column to numeric
16
17 # Set Seaborn style for better plot aesthetics
18 sns.set_style("whitegrid")

```

```

● ● ●
1 # =====
2 # 1. Distribution of Car Prices
3 # =====
4 plt.figure(figsize=(8, 5))
5 sns.histplot(df['Price'], bins=30, kde=True, color='blue')
6 plt.title('Distribution of Car Prices')
7 plt.xlabel('Price')
8 plt.ylabel('Count')
9 plt.show()
10
11 # =====
12 # 2. Price vs. Age of Cars
13 # =====
14 plt.figure(figsize=(8, 5))
15 sns.scatterplot(x=df['Age'], y=df['Price'], alpha=0.6)
16 plt.title('Price vs. Age of Cars')
17 plt.xlabel('Age (months)')
18 plt.ylabel('Price')
19 plt.show()
20

```

```
● ● ●

1 # =====
2 # 3. Price Distribution by Fuel Type
3 # =====
4 plt.figure(figsize=(8, 5))
5 sns.boxplot(x=df['FuelType'], y=df['Price'])
6 plt.title('Price Distribution by Fuel Type')
7 plt.xlabel('Fuel Type')
8 plt.ylabel('Price')
9 plt.show()
10
11 # =====
12 # 4. Count of Cars by Fuel Type
13 # =====
14 plt.figure(figsize=(8, 5))
15 sns.countplot(x=df['FuelType'], palette='pastel')
16 plt.title('Count of Cars by Fuel Type')
17 plt.xlabel('Fuel Type')
18 plt.ylabel('Count')
19 plt.show()
```

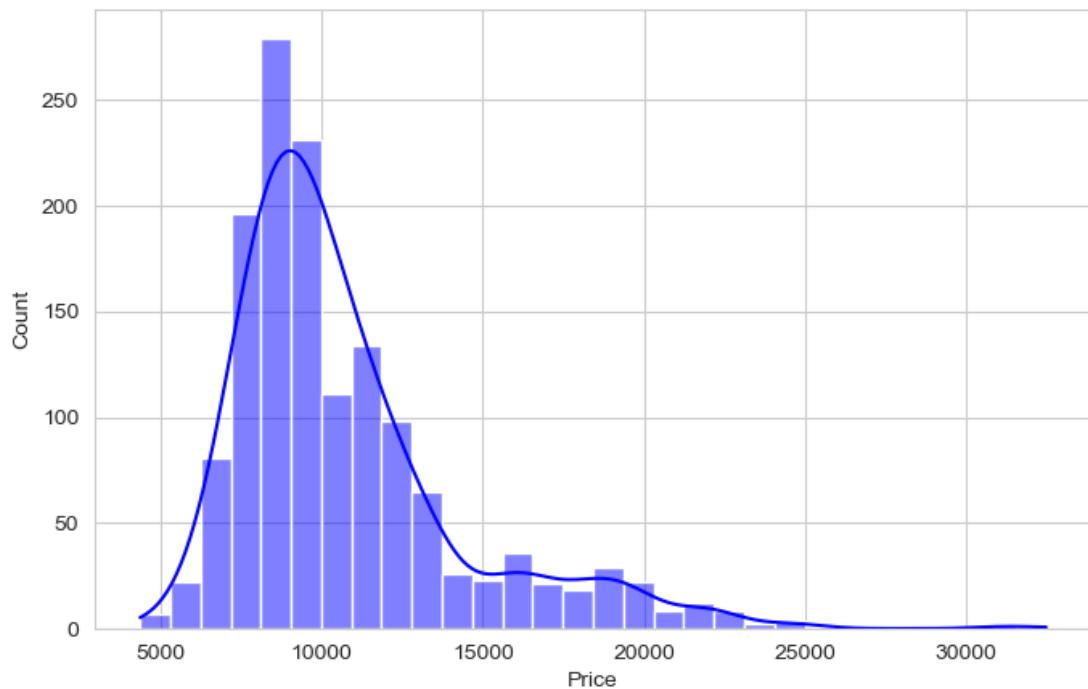
Output:

```
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:

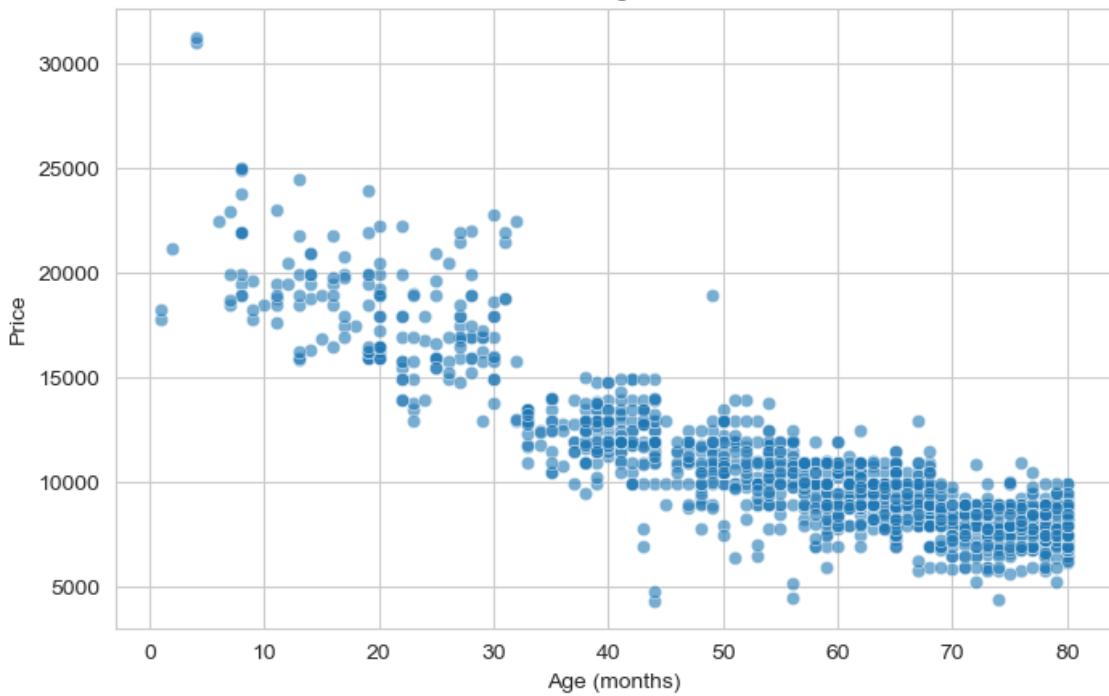
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

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

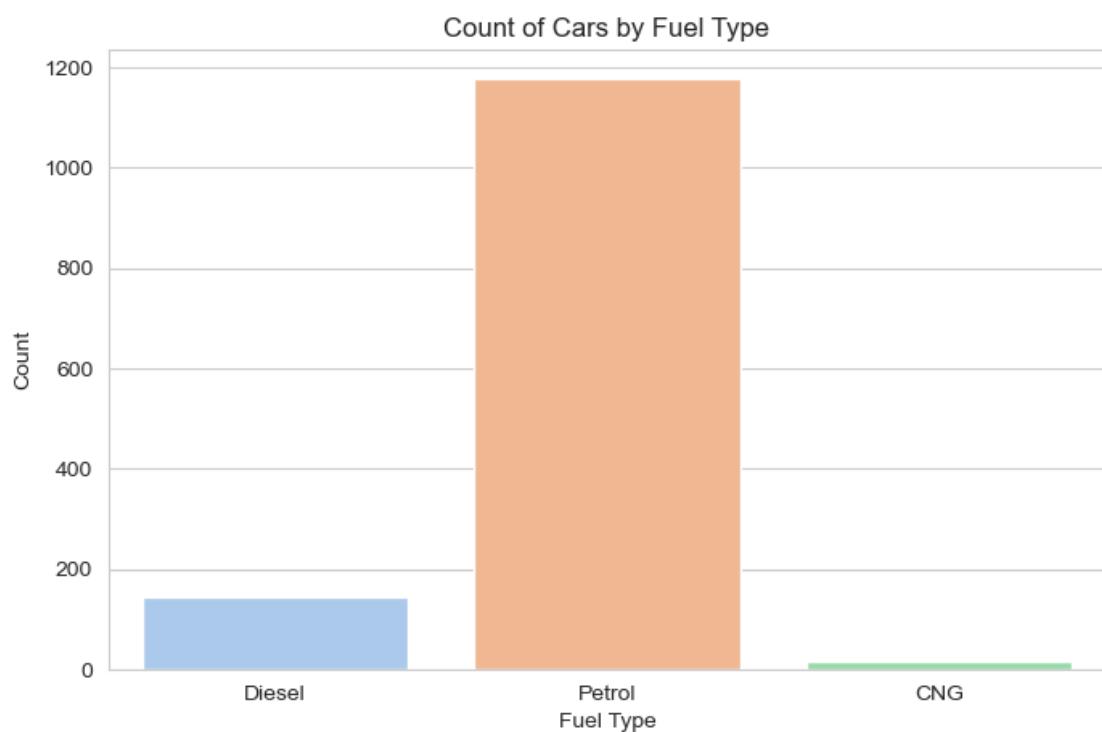
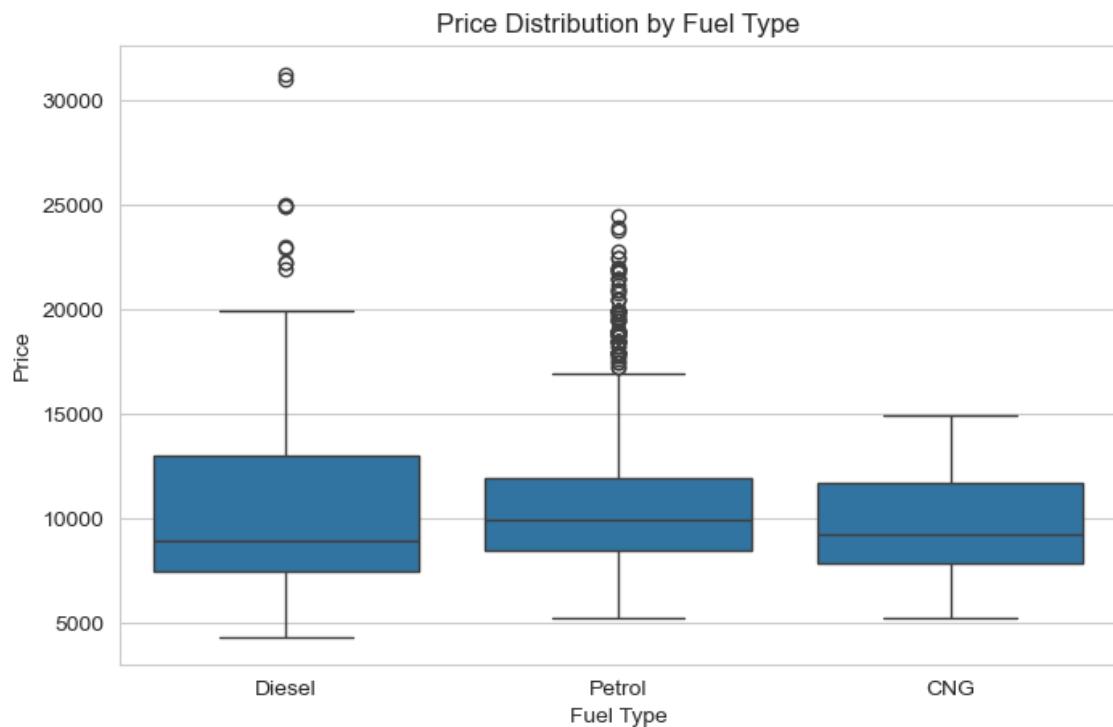
Distribution of Car Prices



Price vs. Age of Cars



Programming in Python Lab (IOT-330P)



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:

```
● ● ●
1 # Import necessary libraries
2 from sklearn import datasets
3 from sklearn.model_selection import train_test_split
4 from sklearn.svm import SVC
5 from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
6 import matplotlib.pyplot as plt
7 # =====
8 # 1. Load the Digits Dataset
9 # =====
10 digits = datasets.load_digits()
11 # =====
12 # 2. Split the Data into Training and Test Sets
13 # =====
14 X_train, X_test, y_train, y_test = train_test_split(
15     digits.data, digits.target, test_size=0.2, random_state=42
16 )
```

```
● ● ●
1 # =====
2 # 3. Create and Train the SVM Classifier
3 # =====
4 clf = SVC(kernel='linear')    # Using a linear kernel
5 clf.fit(X_train, y_train)    # Train the model
6 # =====
7 # 4. Make Predictions
8 # =====
9 y_pred = clf.predict(X_test)
10 # =====
11 # 5. Evaluate the Model
12 # =====
13 # Accuracy Score
14 print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")
15 # Classification Report
16 print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```

1 # =====
2 # 6. Display the Confusion Matrix
3 # =====
4 disp = ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, cmap=plt.cm.Blues)
5 plt.title("Confusion Matrix")
6 plt.show()
7 # =====
8 # 7. Display Sample Predictions
9 # =====
10 fig, axes = plt.subplots(1, 5, figsize=(10, 3))
11 for i, ax in enumerate(axes):
12     ax.set_axis_off()
13     ax.imshow(X_test[i].reshape(8, 8), cmap=plt.cm.gray_r)
14     ax.set_title(f"Pred: {y_pred[i]}")
15 plt.suptitle("Sample Test Predictions")
16 plt.show()

```

Output:

```

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

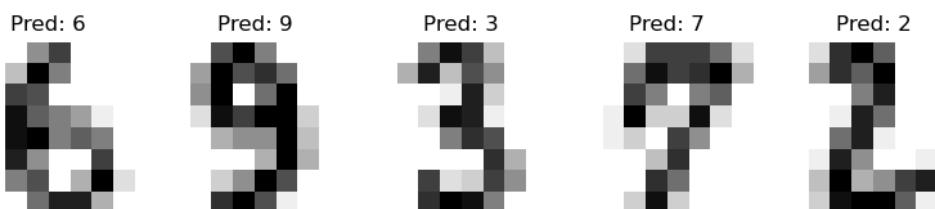
Classification Report:
precision    recall    f1-score   support
          0       1.00      1.00      1.00      33
          1       0.97      1.00      0.98      28
          2       1.00      1.00      1.00      33
          3       0.97      0.94      0.96      34
          4       0.98      0.98      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

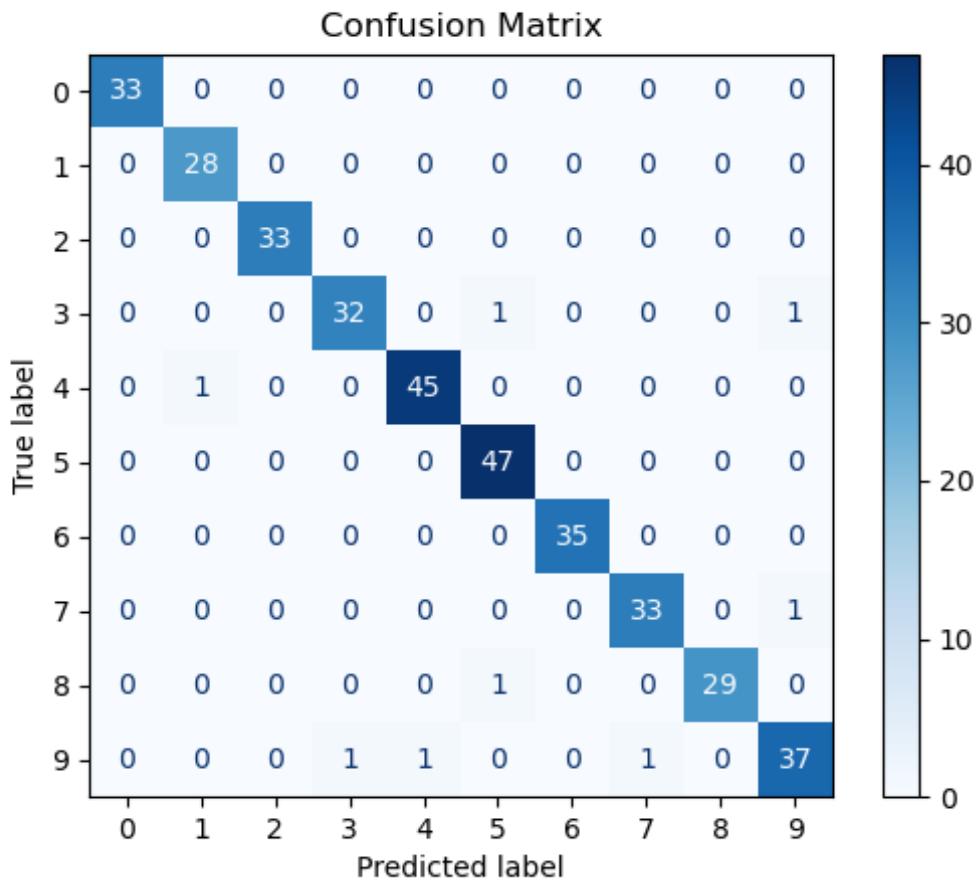
accuracy                           0.98      360
macro avg       0.98      0.98      0.98      360
weighted avg    0.98      0.98      0.98      360

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'

```

Sample Test Predictions





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:

```
● ● ●
1 import socket
2
3 # Create a socket object
4 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
6 # Bind the socket to a specific address and port
7 HOST = '127.0.0.1' # Localhost
8 PORT = 12345
9 server_socket.bind((HOST, PORT))
10
11 # Start listening for connections
12 server_socket.listen()
13
14 print(f"Server is listening on {HOST}:{PORT}")
15
16 # Accept a connection
17 client_socket, addr = server_socket.accept()
18 print(f"Connected by {addr}")
19
20 # Receive data from the client
21 data = client_socket.recv(1024).decode('utf-8')
22 print(f"Received: {data}")
23
24 # Send a response
25 response = "Hello from the server!"
26 client_socket.send(response.encode('utf-8'))
27
28 # Close the connection
29 client_socket.close()
30 server_socket.close()
```

Client Side:

```
● ● ●
1 import socket
2
3 # Create a socket object
4 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
6 # Connect to the server
7 HOST = '127.0.0.1'
8 PORT = 12345
9 client_socket.connect((HOST, PORT))
10
11 # Send a message
12 message = "Hello from the client!"
13 client_socket.send(message.encode('utf-8'))
14
15 # Receive a response
16 response = client_socket.recv(1024).decode('utf-8')
17 print(f"Server Response: {response}")
18
19 # Close the connection
20 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!
```

Programming in Python Lab (IOT-330P)