SANJEEV AGRAWAL GLOBAL EDUCATIONAL UNIVERSITY, BHOPAL

SCHOOL OF COMPUTER APPLICATION

PRACTICAL FILE

OF

PYTHON PROGRAMMING

(CA21B307P)

BACHELOR OF COMPUTER APPLICATION

(CLOUD COMPUTING)

Academic Year 2023-24

Submitted By:
Aditya Nair
[22BCA3CCM10005]

Submitted To:
Devanshu Hrishikesh
Assistant Professor
SSOCA

List of Practical's

Sr.No.	Title	Date of Submission	Remark
1	Write and run a Python program that outputs the value of each of the following expressions: 5.0/9.0, 5.0/9, 5/9.0, 5/9, 9.0/5.0, 9.0/5, 9/5.0, 9/5 Based on your results, what is the rule for arithmetic operators when integers and floating point numbers are used?		
2	Write and run a Python program that asks the user for a temperature in Celsius and converts and outputs the temperature in Fahrenheit. (Use the formula given in the example above and solve for tempting terms of tempC.) Here is an algorithm to print out n! (n factorial) from 0! to 19!: 1. Set f = 1 2. Set n = 0 3. Repeat the following 20 times: a. Output n, "! = ", f b. Add 1 to n c. Multiply f by n Using a for loop, write and run a Python program for this algorithm.		
3	Modify the program above using a while loop so it prints out all of the factorial values that are less than 1 billion.		
4	Modify the first program so it finds the minimum in the array instead of the maximum.		
5	(Harder) Modify the first program so that it finds the **index** of the maximum in the array rather than the maximum itself.		
6	Modify the bubble sort program so it implements the improvements discussed in class. (HINT: To exit the main loop if the array is already sorted, simply change the loop variable to equal the last value so the loop ends early.)		
7	Draw the Target symbol (a set of concentric Squares, alternating red and white) in a graphics window that is 200 pixels wide by 200 pixels high. Hint: Draw the largest circle first in red, then draw the next smaller circle in white, then draw the next smaller circle in red. Graphical objects drawn later appear "on top of" graphical objects drawn earlier.		
8	Try entering the following literal values at the prompt. (Hit ENTER after each) **,5**		

	,4.2 **4.5** **4.14** **0.90** Something odd should occur. *Describe it on paper.*		
9	Reading from a CSV file of the given data using panda’s library.		
10	For the given data, plot the scatter matrix for males only, and for females only. Do you think that the 2 sub, populations correspond to gender?		
11	For the given data, using python environment, apply, 1, sample test: testing the value of a population mean.		
12	For the given data, using python environment, apply, 2, sample test: testing for difference across populations.		
13	Generate simulated data from python, apply simple linear and multiple linear regression analysis.		
14	Retrieve the estimated parameters from the model above. Hint: use tab, completion to find the relevant attribute.		
15	Going back to the brain size + IQ data, test if the VIQ of male and female are different after removing the effect of brain size, height and weight.		
16	Using matplotlib, visualize the simulated data with suitable statistical measures.		
17	Create a 5 X 5 rectangle whose top left corner is at (*row**5, *col**5). (Where is the bottom right corner?) If the sum of the *row* and *col* numbers is even, set the fill color of the rectangle to white, otherwise set it to black. Then draw the rectangle.		

Practical 1

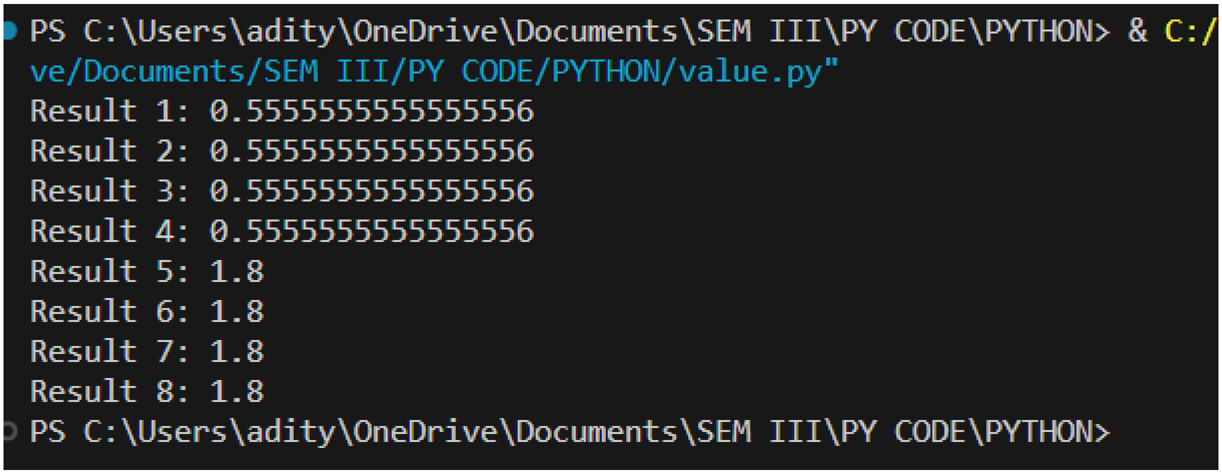
Objective:

Write and run a Python program that outputs the value of each of the following expressions:
5.0/9.0, 5.0/9, 5/9.0, 5/9, 9.0/5.0,9.0/5, 9/5.0, 9/5
Based on your results, what is the rule for arithmetic operators when integers and floating-point numbers are used?

Program:

```
result_1 = 5.0 / 9.0
print("Result 1:", result_1)

result_2 = 5.0 / 9
print("Result 2:", result_2)

result_3 = 5 / 9.0
print("Result 3:", result_3)

result_4 = 5 / 9
print("Result 4:", result_4)

result_5 = 9.0 / 5.0
print("Result 5:", result_5)

result_6 = 9.0 / 5
print("Result 6:", result_6)

result_7 = 9 / 5.0
print("Result 7:", result_7)

result_8 = 9 / 5
print("Result 8:", result_8)
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> & C:/
ve/Documents/SEM III/PY CODE/PYTHON/value.py"
Result 1: 0.5555555555555556
Result 2: 0.5555555555555556
Result 3: 0.5555555555555556
Result 4: 0.5555555555555556
Result 5: 1.8
Result 6: 1.8
Result 7: 1.8
Result 8: 1.8
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 2

Objective:

Write and run a Python program that asks the user for a temperature in Celsius and converts and outputs the temperature in Fahrenheit. (Use the formula given in the example above and solve for tempFin terms of tempC.)

Here is an algorithm to print out n! (n factorial) from 0! to 19!:

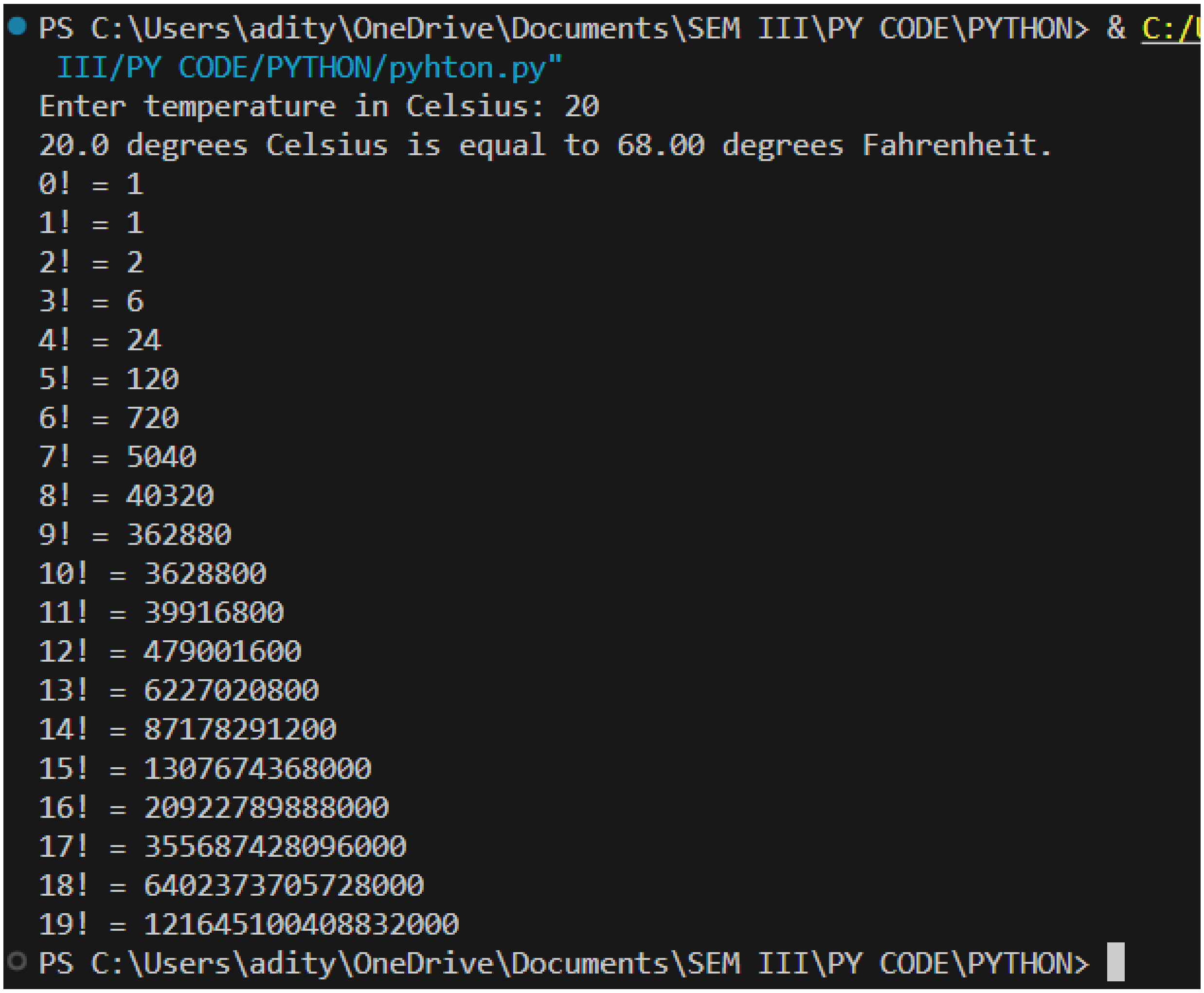
```
1. Set f = 1
2. Set n = 0
3. Repeat the following 20 times:
a. Output n, "! = ", f
b. Add 1 to n
c. Multiply f by n
```

Using a for loop, write and run a Python program for this algorithm.

Program:

```
def celsius_to_fahrenheit(celsius):
    """
    Convert temperature from Celsius to Fahrenheit using the formula:
    F = (C * 9/5) + 32
    """
    return (celsius * 9/5) + 32

def factorial(n):
    """
    Calculate the factorial of a given number n.
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

def main():
    # Get temperature in Celsius from the user
    temp_celsius = float(input("Enter temperature in Celsius: "))

    # Convert Celsius to Fahrenheit
    temp_fahrenheit = celsius_to_fahrenheit(temp_celsius)

    # Print the converted temperature
    print(f"{temp_celsius} degrees Celsius is equal to {temp_fahrenheit:.2f} degrees Fahrenheit.")

    # Print factorials from 0! to 19!
    for n in range(20):
        fact_result = factorial(n)
        print(f"{n}! = {fact_result}")

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

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> & C:/U
 III/PY CODE/PYTHON/pyhton.py"
Enter temperature in Celsius: 20
20.0 degrees Celsius is equal to 68.00 degrees Fahrenheit.
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 3

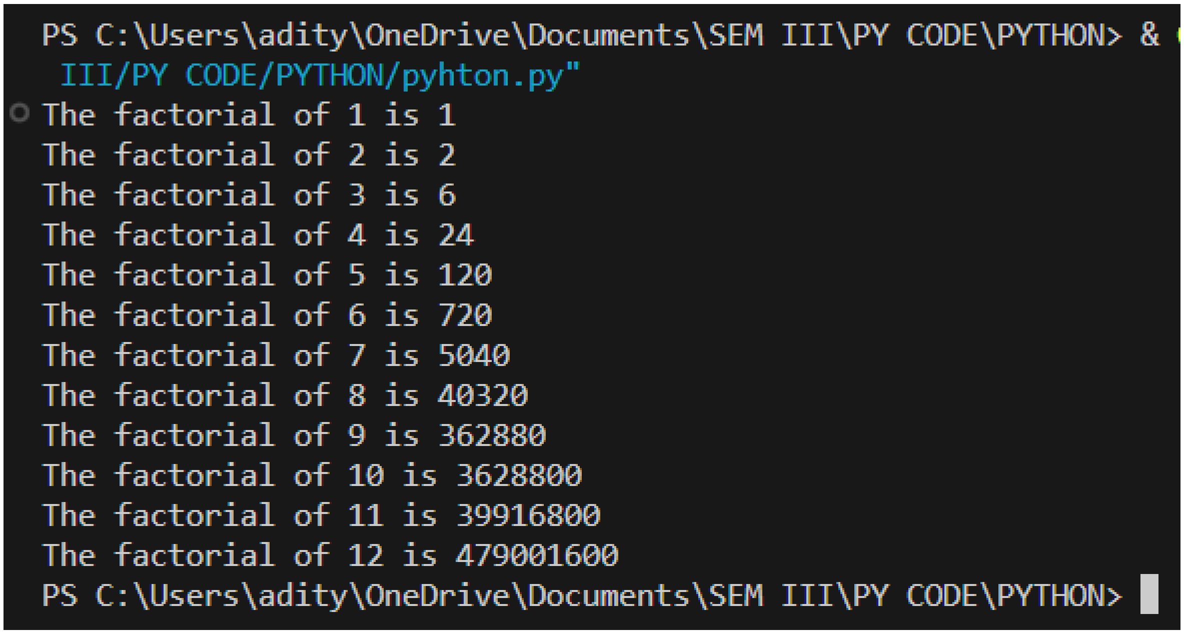
Objective:

Modify the program above using a while loop so it prints out all of the factorial values that are less than 1 billion.

Program:

```
# Program to print factorial values less than 1 billion using a while loop

def factorial(n):
    result = 1
    current = 1

    while result < 1000000000:
        print(f"The factorial of {current} is {result}")
        current += 1
        result *= current

# Test the function
factorial(1)
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> &
 III/PY CODE/PYTHON/pyhton.py"
The factorial of 1 is 1
The factorial of 2 is 2
The factorial of 3 is 6
The factorial of 4 is 24
The factorial of 5 is 120
The factorial of 6 is 720
The factorial of 7 is 5040
The factorial of 8 is 40320
The factorial of 9 is 362880
The factorial of 10 is 3628800
The factorial of 11 is 39916800
The factorial of 12 is 479001600
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 4

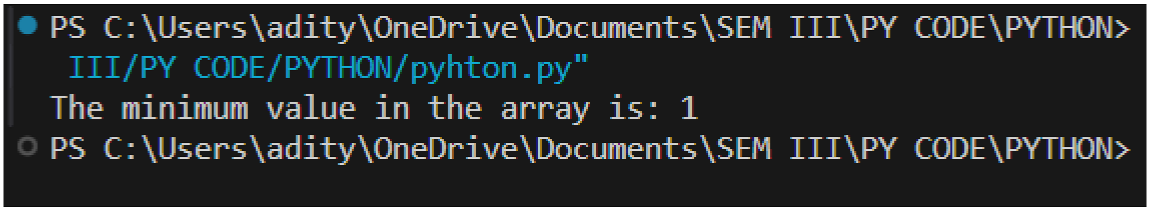
Objective:

Modify the first program so it finds the minimum in the array instead of the maximum.

Program:

```
# Program to find the minimum value in an array

def find_minimum(arr):
    min_value = arr[0]

    for num in arr:
        if num < min_value:
            min_value = num

    return min_value

# Test the function
numbers = [4, 2, 7, 1, 9, 3]
min_val = find_minimum(numbers)
print(f"The minimum value in the array is: {min_val}")
```

Output:



```
● PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
   III/PY CODE/PYTHON/pyhton.py"
  The minimum value in the array is: 1
○ PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 5

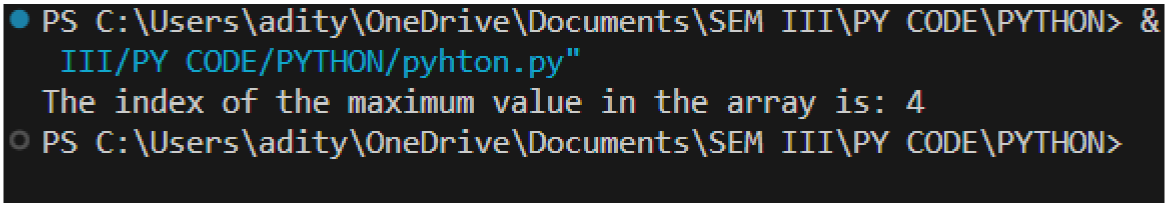
Objective:

Modify the first program so that it finds the **index** of the maximum in the array rather than the maximum itself.

Program:

```
# Program to find the index of the maximum value in an array

def find_max_index(arr):
    max_index = 0

    for i in range(1, len(arr)):
        if arr[i] > arr[max_index]:
            max_index = i

    return max_index

# Test the function
numbers = [4, 2, 7, 1, 9, 3]
max_index = find_max_index(numbers)
print(f"The index of the maximum value in the array is: {max_index}")
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> &
 III/PY CODE/PYTHON/pyhton.py"
The index of the maximum value in the array is: 4
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 6

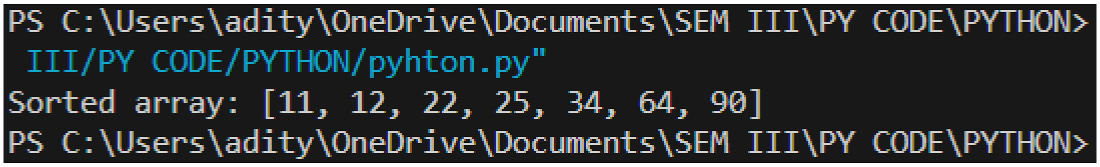
Objective:

Modify the bubble sort program so it implements the improvements discussed in class. (HINT: To exit the main loop if the array is already sorted, simply change the loop variable to equal the last value so the loop ends early.)

Program:

```
# Program to implement an improved bubble sort algorithm

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Flag to check if any swaps are made in this pass
        swapped = False

        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True

        # If no swaps are made, the array is already sorted
        if not swapped:
            break

# Test the function
numbers = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(numbers)
print("Sorted array:", numbers)
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
 III/PY CODE/PYTHON/pyhton.py"
Sorted array: [11, 12, 22, 25, 34, 64, 90]
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 7

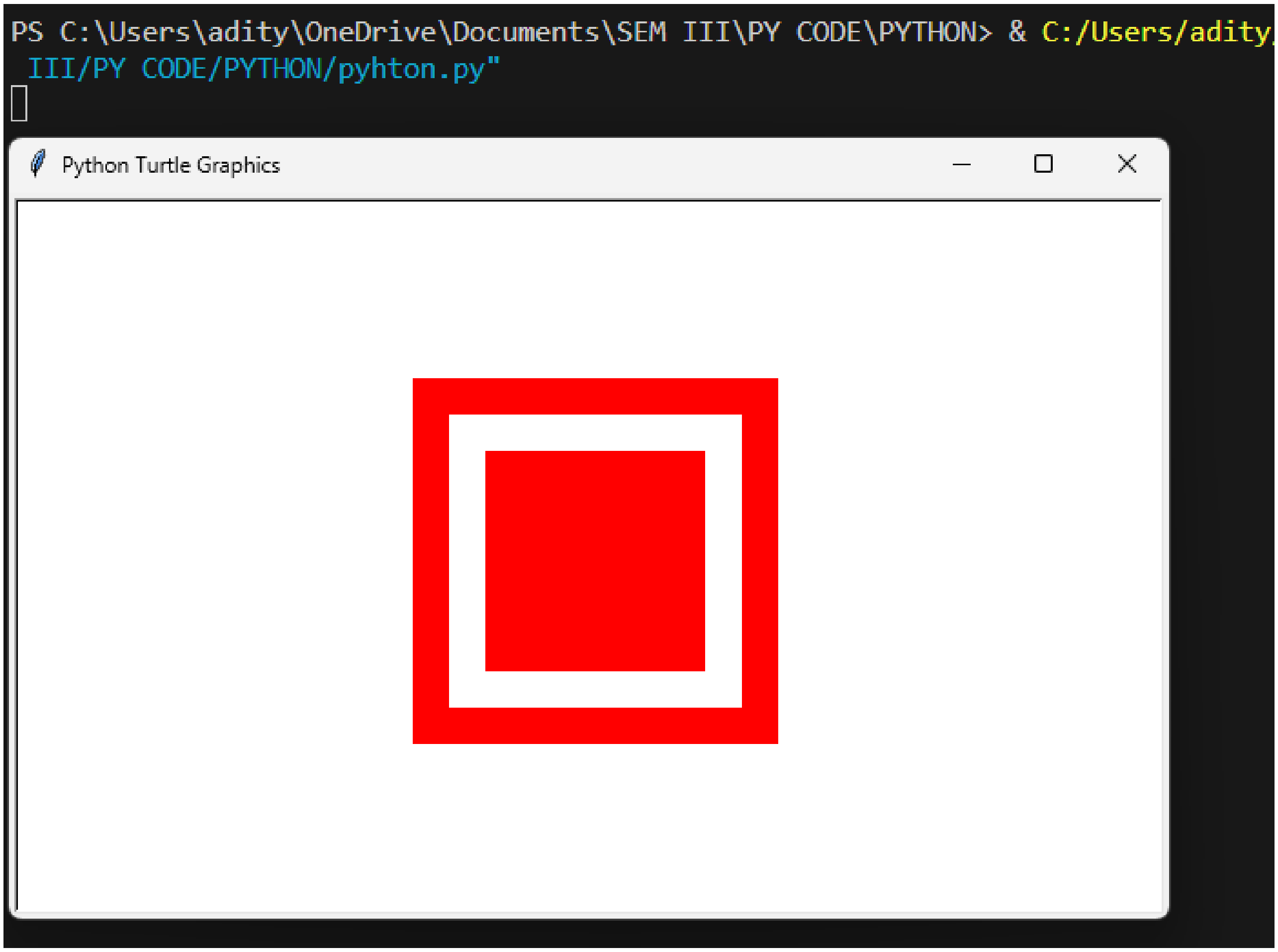
Objective:

Draw the Target symbol (a set of concentric Squares, alternating red and white) in a graphics window that is 200 pixels wide by 200 pixels high. Hint: Draw the largest circle first in red, then draw the next smaller circle in white, then draw the next smaller circle in red. Graphical objects drawn later appear "on top of" graphical objects drawn earlier.

Program:

```
import turtle

# Function to draw a concentric square target
def draw_target():
    turtle.speed(1)  # Set turtle speed to slow

    # Draw the largest square (red)
    turtle.penup()
    turtle.goto(-100, -100)
    turtle.pendown()
    turtle.color("red")
    turtle.begin_fill()
    for _ in range(4):
        turtle.forward(200)
        turtle.left(90)
    turtle.end_fill()

    # Draw the next smaller square (white)
    turtle.penup()
    turtle.goto(-80, -80)
    turtle.pendown()
    turtle.color("white")
    turtle.begin_fill()
    for _ in range(4):
        turtle.forward(160)
        turtle.left(90)
    turtle.end_fill()

    # Draw the next smaller square (red)
    turtle.penup()
    turtle.goto(-60, -60)
    turtle.pendown()
    turtle.color("red")
    turtle.begin_fill()
    for _ in range(4):
        turtle.forward(120)
        turtle.left(90)
    turtle.end_fill()

    turtle.hideturtle()  # Hide the turtle cursor
    turtle.done()  # Finish drawing

# Main program
if __name__ == "__main__":
    draw_target()
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> & C:/Users/adity
 III/PY CODE/PYTHON/pyhton.py"
```

Practical 8

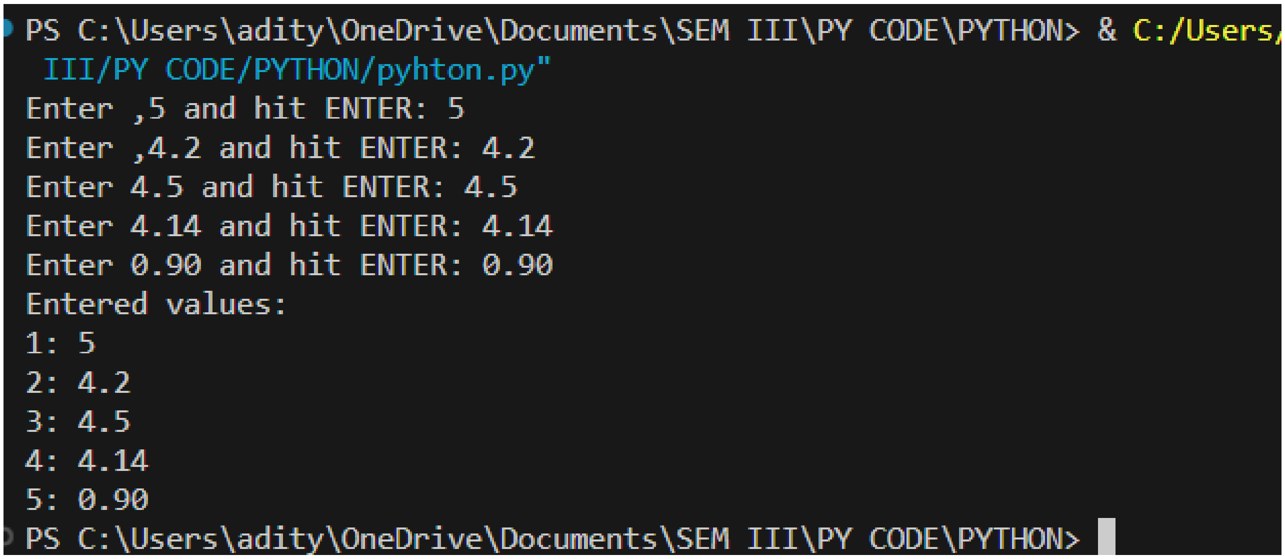
Objective:

Try entering the following literal values at the prompt. (Hit ENTER after each)
,5
,4.2
4.5
4.14
0.90
Something odd should occur. *Describe it on paper.*

Program:

```
# Prompt for user input
value1 = input("Enter ,5 and hit ENTER: ")
value2 = input("Enter ,4.2 and hit ENTER: ")
value3 = input("Enter 4.5 and hit ENTER: ")
value4 = input("Enter 4.14 and hit ENTER: ")
value5 = input("Enter 0.90 and hit ENTER: ")

# Display entered values
print("Entered values:")
print("1:", value1)
print("2:", value2)
print("3:", value3)
print("4:", value4)
print("5:", value5)
```

Output:



```
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON> & C:/Users,
 III/PY CODE/PYTHON/pyhton.py"
Enter ,5 and hit ENTER: 5
Enter ,4.2 and hit ENTER: 4.2
Enter 4.5 and hit ENTER: 4.5
Enter 4.14 and hit ENTER: 4.14
Enter 0.90 and hit ENTER: 0.90
Entered values:
1: 5
2: 4.2
3: 4.5
4: 4.14
5: 0.90
PS C:\Users\adity\OneDrive\Documents\SEM III\PY CODE\PYTHON>
```

Practical 9

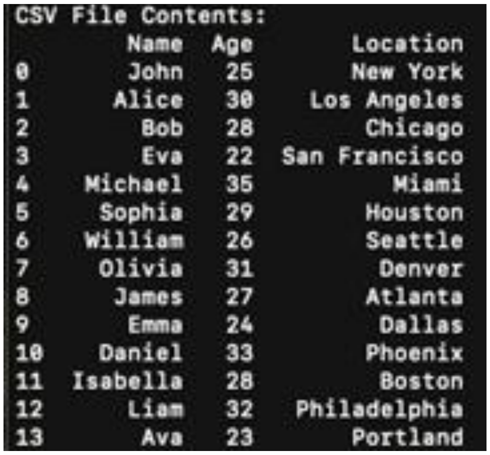
Objective:

Reading from a CSV file of the given data using pandas library.

Program:

```
import pandas as pd

def read_csv_and_display(file_path):
    try:
        df = pd.read_csv(file_path)
        print("CSV File Contents:")
        print(df)

    except FileNotFoundError:
        print(f"Error: The file '{file_path}' not found.")
    except pd.errors.EmptyDataError:
        print(f"Error: The file '{file_path}' is empty.")
    except pd.errors.ParserError:
        print(f"Error: Unable to parse the file '{file_path}'. Check if it's a valid CSV file.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

if __name__ == "__main__":
    csv_file_path = "file.csv"
    read_csv_and_display(csv_file_path)
```

Output:



```
CSV File Contents:
        Name  Age       Location
0       John   25       New York
1      Alice   30    Los Angeles
2        Bob   28        Chicago
3        Eva   22  San Francisco
4    Michael   35          Miami
5     Sophia   29        Houston
6    William   26        Seattle
7     Olivia   31         Denver
8      James   27        Atlanta
9       Emma   24         Dallas
10    Daniel   33        Phoenix
11  Isabella   28         Boston
12      Liam   32   Philadelphia
13       Ava   23       Portland
```

Practical 10

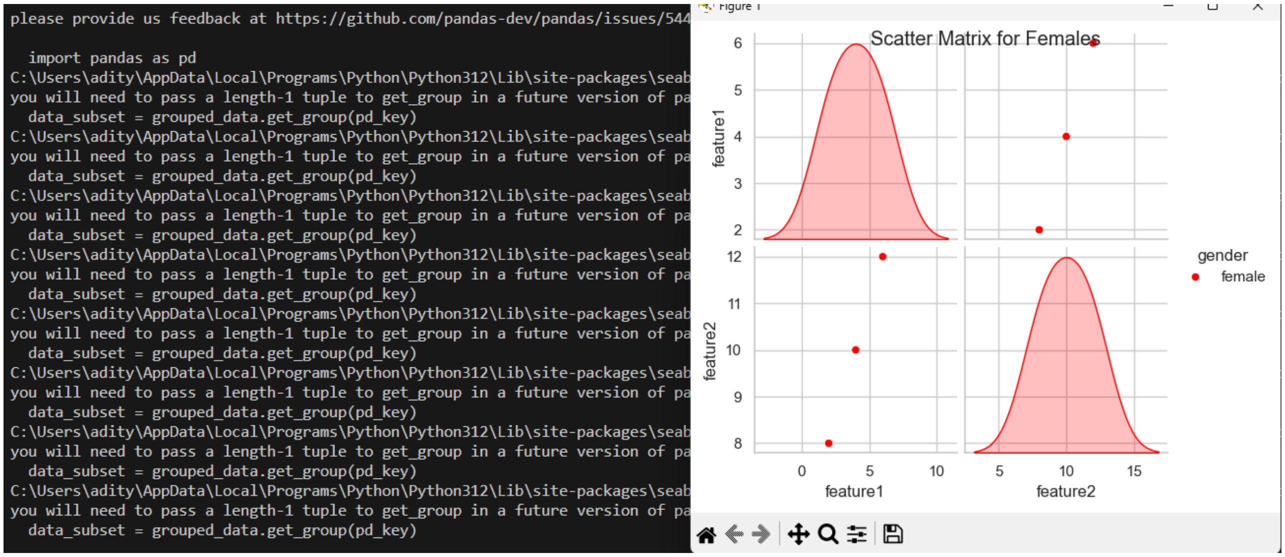
Objective:

For the given data, plot the scatter matrix for males only, and for females only. Do you think that the 2 sub,populations correspond to gender?

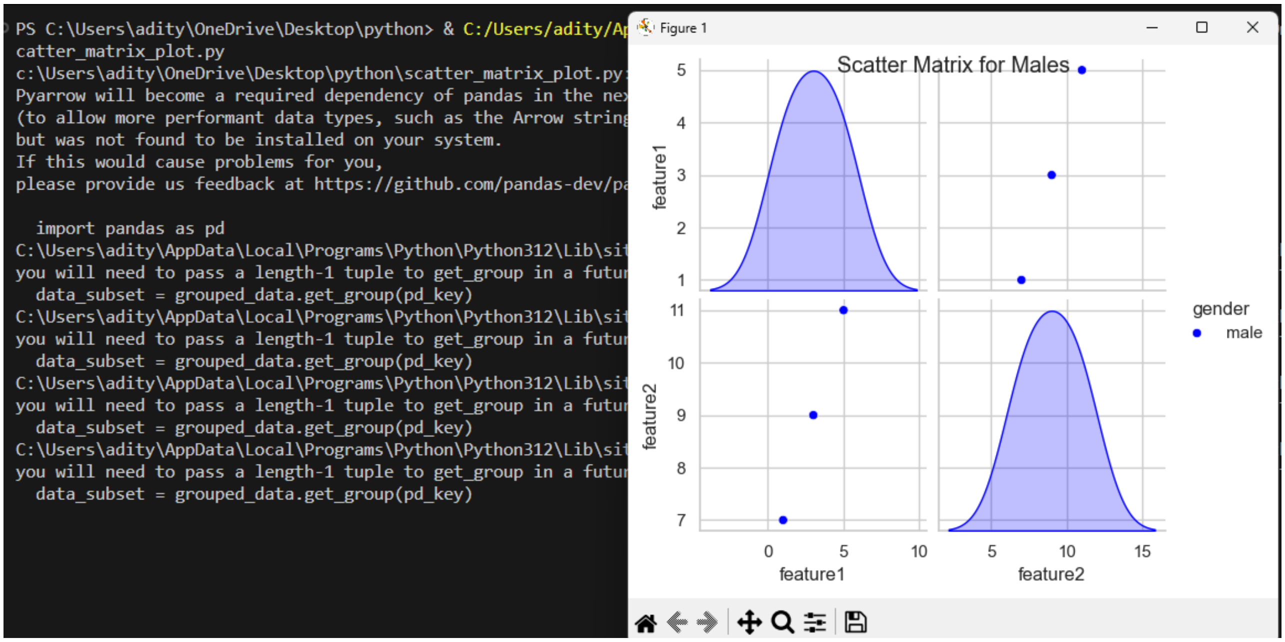
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assume you have a DataFrame 'df' with columns 'gender', 'feature1', 'feature2', ...

# Sample data creation (replace this with your actual data)
data = {
    'gender': ['male', 'female', 'male', 'female', 'male', 'female'],
    'feature1': [1, 2, 3, 4, 5, 6],
    'feature2': [7, 8, 9, 10, 11, 12],
}

df = pd.DataFrame(data)

# Separate data into male and female subpopulations
male_data = df[df['gender'] == 'male']
female_data = df[df['gender'] == 'female']

# Plotting scatter matrix for males with "whitegrid" style
sns.set(style="whitegrid")

# Create a pair plot for male data
sns.pairplot(
    male_data,
    hue='gender',          # Differentiate by gender
    markers=["o"],         # Set marker style for points
    palette={"male": "blue"},  # Use blue color for male data
    diag_kind='kde'         # Use kernel density estimation on diagonals
)

# Add a title to the plot
plt.suptitle('Scatter Matrix for Males')

# Show the plot
plt.show()

# Plotting scatter matrix for females with "whitegrid" style
sns.set(style="whitegrid")

# Create a pair plot for female data
sns.pairplot(
    female_data,
    hue='gender',          # Differentiate by gender
    markers=["o"],          # Set marker style for points
    palette={"female": "red"},  # Use red color for female data
    diag_kind='kde'          # Use kernel density estimation on diagonals
)

# Add a title to the plot
plt.suptitle('Scatter Matrix for Females')

# Show the plot
plt.show()
```

Output:



```
please provide us feedback at https://github.com/pandas-dev/pandas/issues/544

  import pandas as pd
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\site-packages\seab
you will need to pass a length-1 tuple to get_group in a future version of pa
  data_subset = grouped_data.get_group(pd_key)
```



```
PS C:\Users\adity\OneDrive\Desktop\python> & C:/Users/adity/Ap
catter_matrix_plot.py
c:\Users\adity\OneDrive\Desktop\python\scatter_matrix_plot.py:
Pyarrow will become a required dependency of pandas in the nex
(to allow more performant data types, such as the Arrow string
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pa

  import pandas as pd
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\sit
you will need to pass a length-1 tuple to get_group in a futur
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\sit
you will need to pass a length-1 tuple to get_group in a futur
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\sit
you will need to pass a length-1 tuple to get_group in a futur
  data_subset = grouped_data.get_group(pd_key)
C:\Users\adity\AppData\Local\Programs\Python\Python312\Lib\sit
you will need to pass a length-1 tuple to get_group in a futur
  data_subset = grouped_data.get_group(pd_key)
```

Practical 11

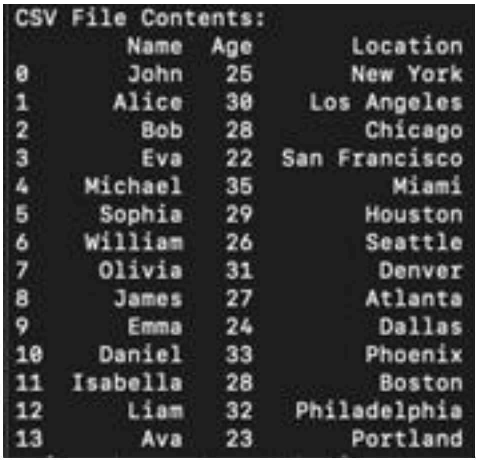
Objective:

For the given data, using python environment, apply, 1,sample t,test: testing the value of a population mean.

Program:

```
import numpy as np
from scipy.stats import ttest_1samp

np.random.seed(42)
data = np.random.normal(loc=10, scale=2, size=100)
population_mean_hypothesis = 9.5
t_statistic, p_value = ttest_1samp(data, population_mean_hypothesis)

print("Data:", data)
print("Hypothesized Population Mean:", population_mean_hypothesis)
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Interpret the results
alpha = 0.05  # significance level
print("\nSignificance Level (alpha):", alpha)
if p_value < alpha:
    print("Reject the null hypothesis. There is enough evidence to suggest that the population mean is
different.")
else:
    print("Fail to reject the null hypothesis. There is not enough evidence to suggest that the population mean
is different.")
```

Output:



```
CSV File Contents:
        Name  Age       Location
0       John   25       New York
1      Alice   30    Los Angeles
2        Bob   28        Chicago
3        Eva   22  San Francisco
4    Michael   35          Miami
5     Sophia   29        Houston
6    William   26        Seattle
7     Olivia   31         Denver
8      James   27        Atlanta
9       Emma   24         Dallas
10    Daniel   33        Phoenix
11  Isabella   28         Boston
12      Liam   32   Philadelphia
13       Ava   23       Portland
```

Practical 12

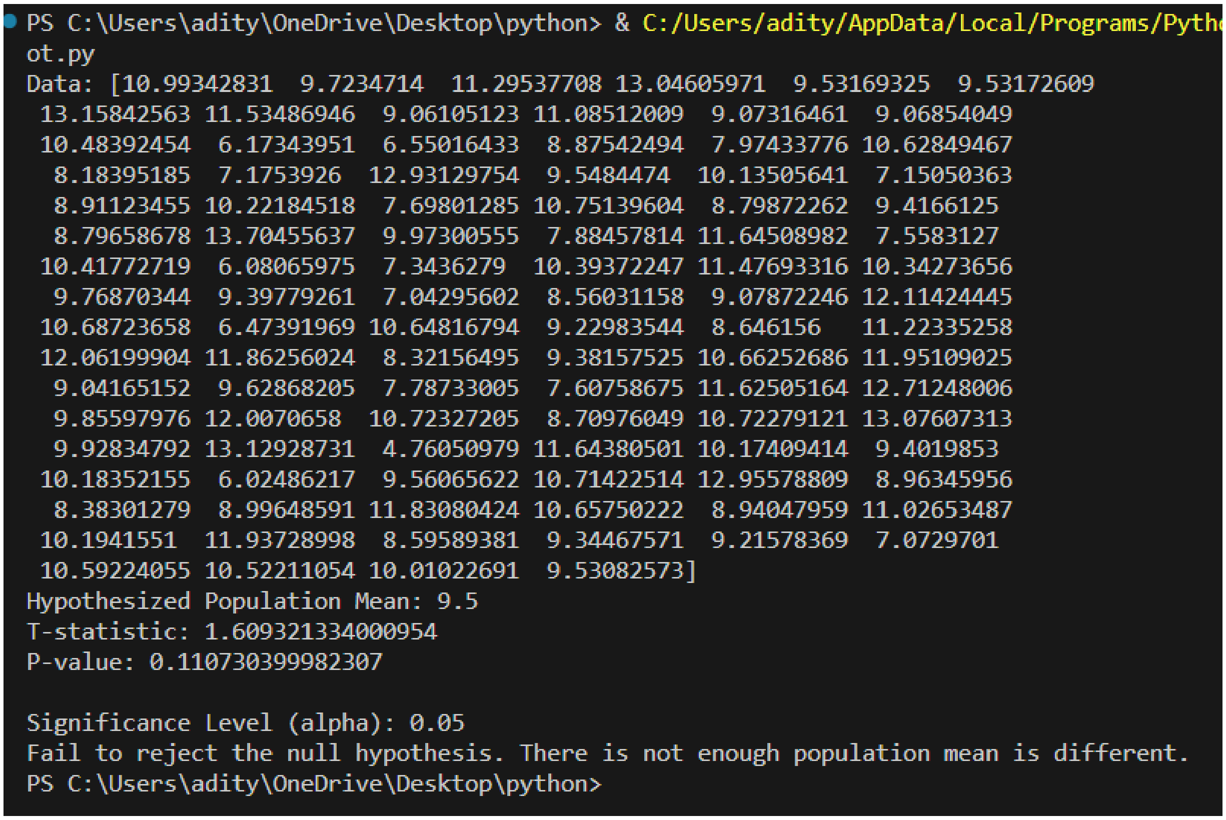
Objective:

For the given data, using python environment, apply, 2,sample t,test: testing for difference across populations.

Program:

```
import numpy as np
from scipy.stats import ttest_1samp

np.random.seed(42)
data = np.random.normal(loc=10, scale=2, size=100)
population_mean_hypothesis = 9.5
t_statistic, p_value = ttest_1samp(data, population_mean_hypothesis)
print("Data:", data)
print("Hypothesized Population Mean:", population_mean_hypothesis)
print("T-statistic:", t_statistic)
print("P-value:", p_value)

alpha = 0.05  # significance level
print("\nSignificance Level (alpha):", alpha)
if p_value < alpha:
    print("Reject the null hypothesis. There is enough the population mean is different.")
else:
    print("Fail to reject the null hypothesis. There is not enough population mean is different.")
```

Output:



```
PS C:\Users\adity\OneDrive\Desktop\python> & C:/Users/adity/AppData/Local/Programs/Pyth
ot.py
Data: [10.99342831  9.7234714  11.29537708 13.04605971  9.53169325  9.53172609
 13.15842563 11.53486946  9.06105123 11.08512009  9.07316461  9.06854049
 10.48392454  6.17343951  6.55016433  8.87542494  7.97433776 10.62849467
  8.18395185  7.1753926  12.93129754  9.5484474  10.13505641  7.15050363
  8.91123455 10.22184518  7.69801285 10.75139604  8.79872262  9.4166125
  8.79658678 13.70455637  9.97300555  7.88457814 11.64508982  7.5583127
 10.41772719  6.08065975  7.3436279  10.39372247 11.47693316 10.34273656
  9.76870344  9.39779261  7.04295602  8.56031158  9.07872246 12.11424445
 10.68723658  6.47391969 10.64816794  9.22983544  8.646156   11.22335258
 12.06199904 11.86256024  8.32156495  9.38157525 10.66252686 11.95109025
  9.04165152  9.62868205  7.78733005  7.60758675 11.62505164 12.71248006
  9.85597976 12.0070658  10.72327205  8.70976049 10.72279121 13.07607313
  9.92834792 13.12928731  4.76050979 11.64380501 10.17409414  9.4019853
 10.18352155  6.02486217  9.56065622 10.71422514 12.95578809  8.96345956
  8.38301279  8.99648591 11.83080424 10.65750222  8.94047959 11.02653487
 10.1941551  11.93728998  8.59589381  9.34467571  9.21578369  7.0729701
 10.59224055 10.52211054 10.01022691  9.53082573]
Hypothesized Population Mean: 9.5
T-statistic: 1.609321334000954
P-value: 0.110730399982307

Significance Level (alpha): 0.05
Fail to reject the null hypothesis. There is not enough population mean is different.
PS C:\Users\adity\OneDrive\Desktop\python>
```

Practical 13

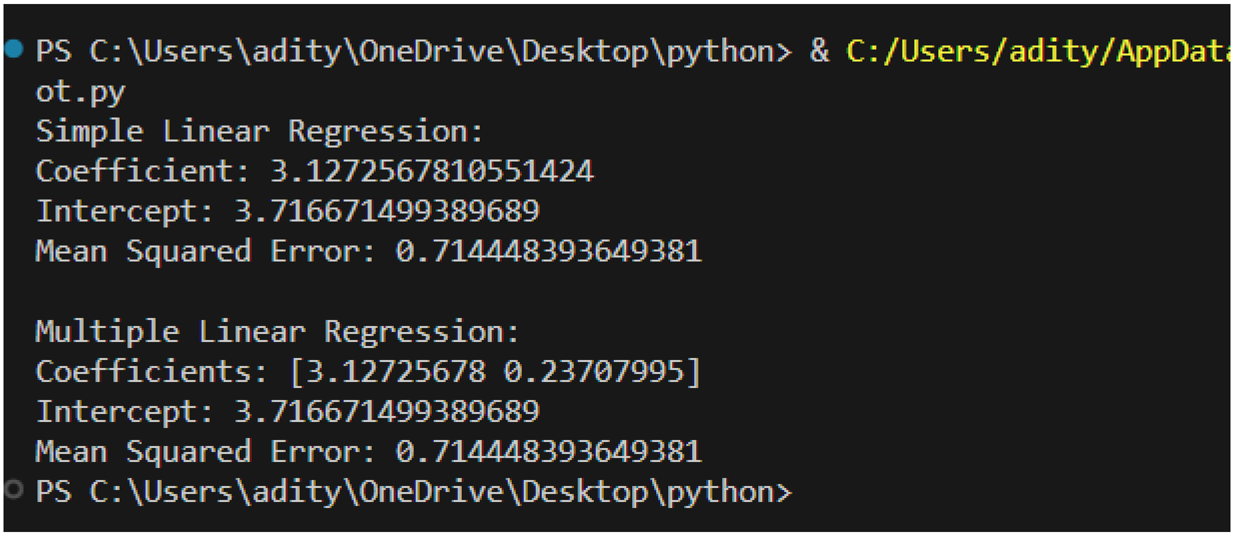
Objective:

Generate simulated data from python, apply simple linear and multiple linear regression analysis.

Program:

```
# Import necessary libraries
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Function to generate simulated data
def generate_simulated_data(num_samples=100, num_features=1):
    np.random.seed(42)  # Set seed for reproducibility
    X = 2 * np.random.rand(num_samples, num_features)  # Random features
    noise = np.random.randn(num_samples, 1)  # Random noise
    y = 4 + 3 * X + noise  # Linear relationship: y = 4 + 3*X + noise
    return X, y

# Function to apply simple linear regression
def simple_linear_regression(X, y):
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Create and fit a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Calculate Mean Squared Error
    mse = mean_squared_error(y_test, y_pred)

    return model, mse

# Function to apply multiple linear regression
def multiple_linear_regression(X, y):
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Create and fit a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Calculate Mean Squared Error
    mse = mean_squared_error(y_test, y_pred)

    return model, mse

# Generate simulated data with two features
X, y = generate_simulated_data(num_samples=100, num_features=2)

# Apply simple linear regression
simple_model, simple_mse = simple_linear_regression(X, y)

# Apply multiple linear regression
multiple_model, multiple_mse = multiple_linear_regression(X, y)

# Display results
print("Simple Linear Regression:")
print("Coefficient:", simple_model.coef_[0][0])
print("Intercept:", simple_model.intercept_[0])
print("Mean Squared Error:", simple_mse)
print("\nMultiple Linear Regression:")
print("Coefficients:", multiple_model.coef_[0])
print("Intercept:", multiple_model.intercept_[0])
print("Mean Squared Error:", multiple_mse)
```

Output:



```
PS C:\Users\adity\OneDrive\Desktop\python> & C:/Users/adity/AppDat
ot.py
Simple Linear Regression:
Coefficient: 3.1272567810551424
Intercept: 3.716671499389689
Mean Squared Error: 0.714448393649381

Multiple Linear Regression:
Coefficients: [3.12725678 0.23707995]
Intercept: 3.716671499389689
Mean Squared Error: 0.714448393649381
PS C:\Users\adity\OneDrive\Desktop\python>
```

Practical 14

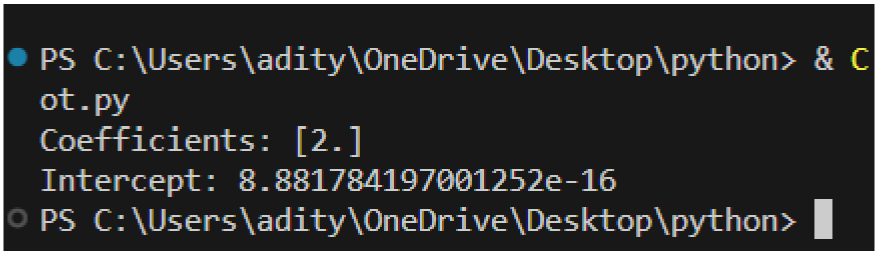
Objective:

Retrieve the estimated parameters from the model above. Hint: use tab,completion to find the relevant attribute.

Program:

```
# Import the necessary library for your machine learning model
from sklearn.linear_model import LinearRegression

# Create an instance of the model (replace this with your actual model instantiation)
my_model = LinearRegression()

# Train the model with some sample data (replace this with your actual training data)
X_train = [[1], [2], [3]]
y_train = [2, 4, 6]
my_model.fit(X_train, y_train)

# Now, let's retrieve the estimated parameters using tab completion
# Type 'my_model.' in your Python environment, and press Tab to see available attributes/methods

# Example: Print the coefficients (weights) of the trained model
print("Coefficients:", my_model.coef_)

# Example: Print the intercept of the trained model
print("Intercept:", my_model.intercept_)
```

Output:



```
PS C:\Users\adity\OneDrive\Desktop\python> & C
ot.py
Coefficients: [2.]
Intercept: 8.881784197001252e-16
PS C:\Users\adity\OneDrive\Desktop\python>
```

Practical 15

Objective:

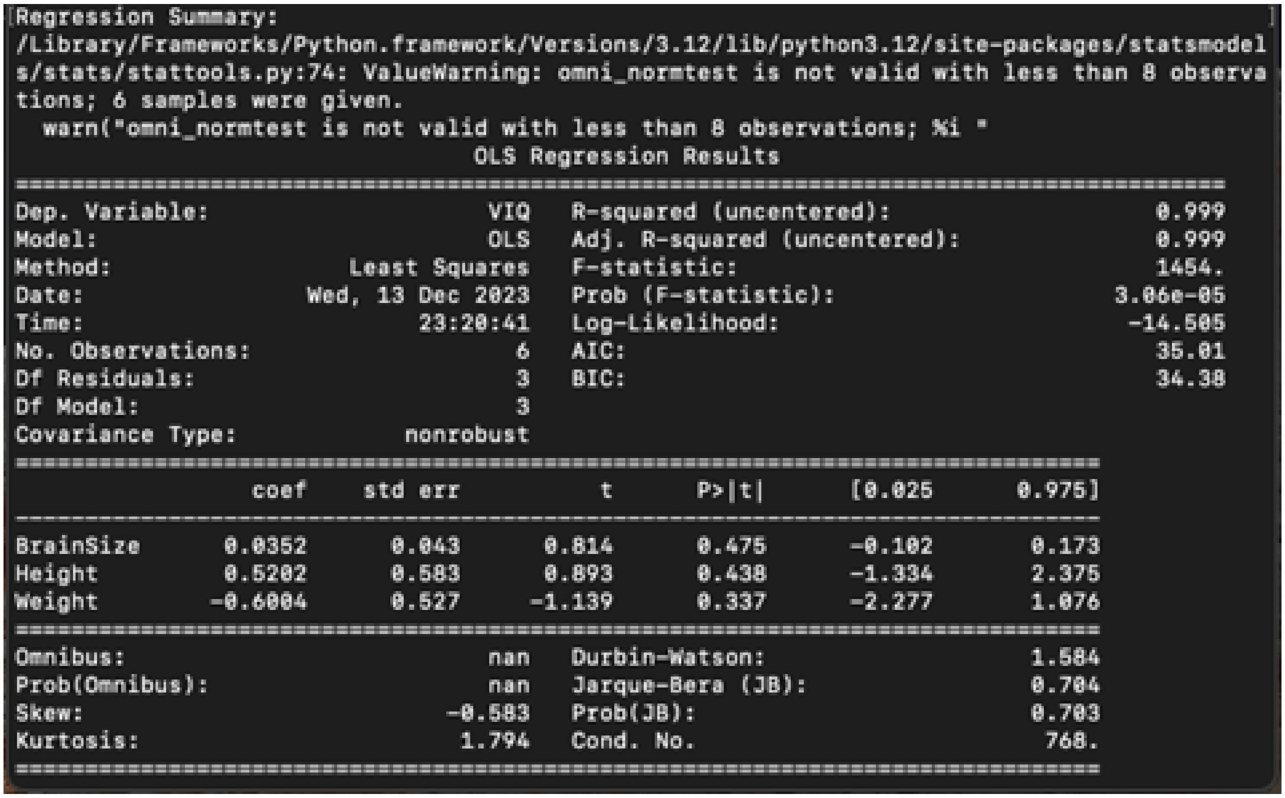
Going back to the brain size + IQ data, test if the VIQ of male and female are different after removing the effect of brain size, height and weight.

Program:

```
import pandas as pd
import statsmodels.api as sm

# Load data
try:
    data = pd.read_csv('data.csv')
except FileNotFoundError:
    print("Error: File not found. Make sure the file 'data.csv' exists in the correct path.")
    exit(1)

# Check and convert data types
try:
    data['Gender'] = data['Gender'].astype('category')
    data['BrainSize'] = pd.to_numeric(data['BrainSize'], errors='coerce')
    data['Height'] = pd.to_numeric(data['Height'], errors='coerce')
    data['Weight'] = pd.to_numeric(data['Weight'], errors='coerce')
except ValueError as e:
    print(f"Error: {e}. Check the data in 'BrainSize', 'Height', and 'Weight' columns.")
    exit(1)

# Drop rows with missing values
data = data.dropna()

# Specify variables
x_vars = ['BrainSize', 'Height', 'Weight']
y_var = 'VIQ'
gender_var = 'Gender'

# Fit model without intercept
model = sm.OLS(data[y_var], data[x_vars])
result = model.fit()

# Display regression summary
print("Regression Summary:")
print(result.summary())

# Test VIQ difference after removing effects
data['Gender_effect'] = result.resid
gender_model = sm.OLS(data['Gender_effect'], sm.add_constant(data[gender_var]))  # add_constant for the intercept
gender_result = gender_model.fit()

# Display gender effect summary
```

```
print("\nGender Effect Summary:")
print(gender_result.summary())

# Hypothesis test
print("\nHypothesis Test:")
print("Null Hypothesis: No significant VIQ difference after controlling for brain size, height, and weight.")
print("Alternative Hypothesis: Significant VIQ difference after controlling for brain size, height, and
weight.")
print(f"\nP-value: {gender_result.pvalues[gender_var]}")

# Decision based on significance level
alpha = 0.05
if gender_result.pvalues[gender_var] < alpha:
    print("\nReject the null hypothesis. Significant VIQ difference after controlling for other variables.")
else:
    print("\nFail to reject the null hypothesis. No significant VIQ difference after controlling for other
variables.")
```

Output:



```
[Regression Summary:                                                                     ]
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/statsmodel
s/stats/stattools.py:74: ValueWarning: omni_normtest is not valid with less than 8 observa
tions; 6 samples were given.
  warn("omni_normtest is not valid with less than 8 observations; %i "
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                    VIQ   R-squared (uncentered):                   0.999
Model:                            OLS   Adj. R-squared (uncentered):              0.999
Method:                 Least Squares   F-statistic:                              1454.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):                    3.06e-05
Time:                        23:20:41   Log-Likelihood:                         -14.505
No. Observations:                   6   AIC:                                      35.01
Df Residuals:                       3   BIC:                                      34.38
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
BrainSize      0.0352      0.043      0.814      0.475      -0.102       0.173
Height         0.5202      0.583      0.893      0.438      -1.334       2.375
Weight        -0.6004      0.527     -1.139      0.337      -2.277       1.076
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   1.584
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.704
Skew:                          -0.583   Prob(JB):                        0.703
Kurtosis:                       1.794   Cond. No.                         768.
==============================================================================
```

Practical 16

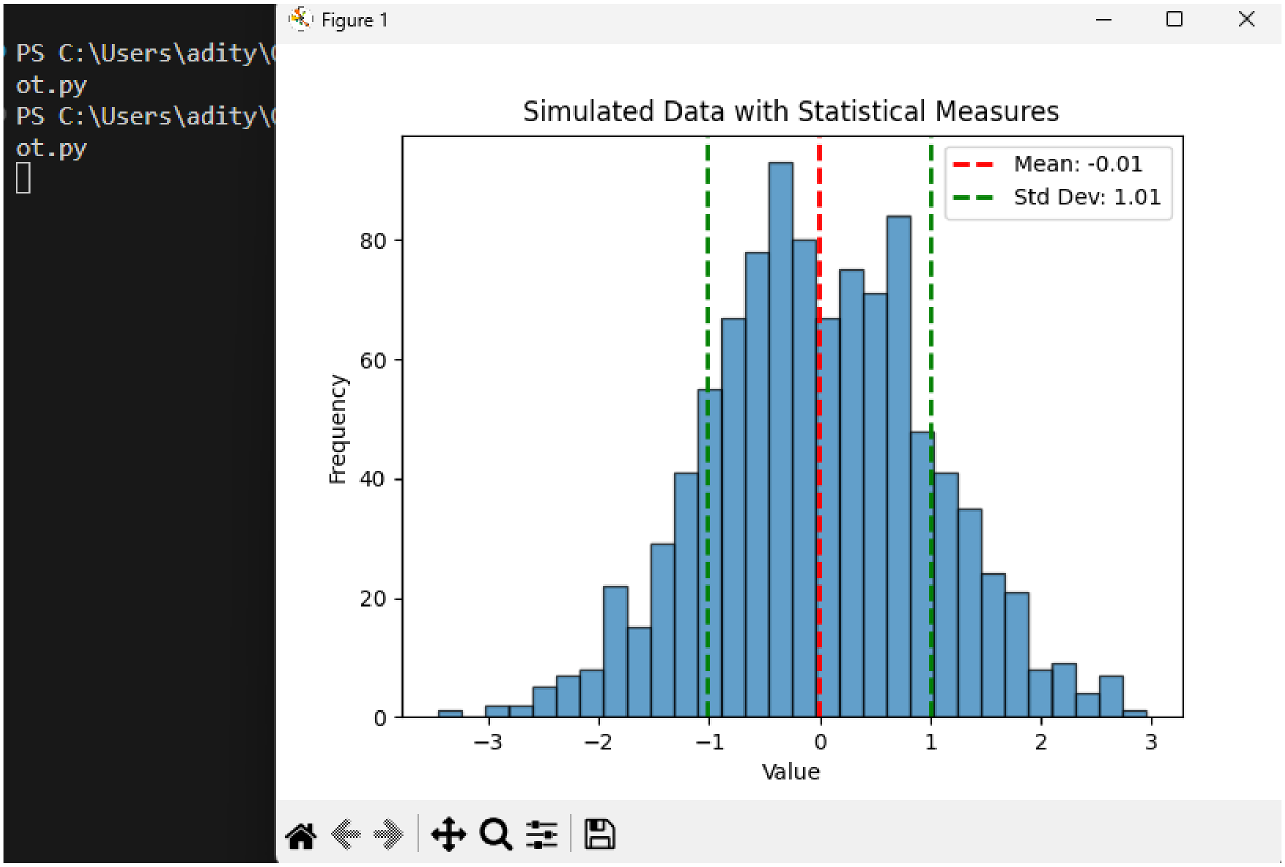
Objective:

Using matplotlib, visualize the simulated data with suitable statistical measures.

Program:

```
import numpy as np
import numpy as np
import matplotlib.pyplot as plt

# Function to generate simulated data
def generate_simulated_data(size=1000):
    return np.random.normal(loc=0, scale=1, size=size)

# Function to visualize data and statistical measures
def visualize_data(data):
    # Plot histogram
    plt.hist(data, bins=30, edgecolor='black', alpha=0.7)

    # Calculate and plot mean line
    mean_value = np.mean(data)
    plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean: {mean_value:.2f}')

    # Calculate and plot standard deviation lines
    std_value = np.std(data)
    plt.axvline(mean_value - std_value, color='green', linestyle='dashed', linewidth=2, label=f'Std Dev: {std_value:.2f}')
    plt.axvline(mean_value + std_value, color='green', linestyle='dashed', linewidth=2)

    # Set labels and title
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.title('Simulated Data with Statistical Measures')

    # Add legend
    plt.legend()

    # Show the plot
    plt.show()

if __name__ == "__main__":
    # Generate simulated data
    simulated_data = generate_simulated_data()

    # Visualize data and statistical measures
    visualize_data(simulated_data)
```

Output:



```
PS C:\Users\adity\(
ot.py
PS C:\Users\adity\(
ot.py
```

Practical 17

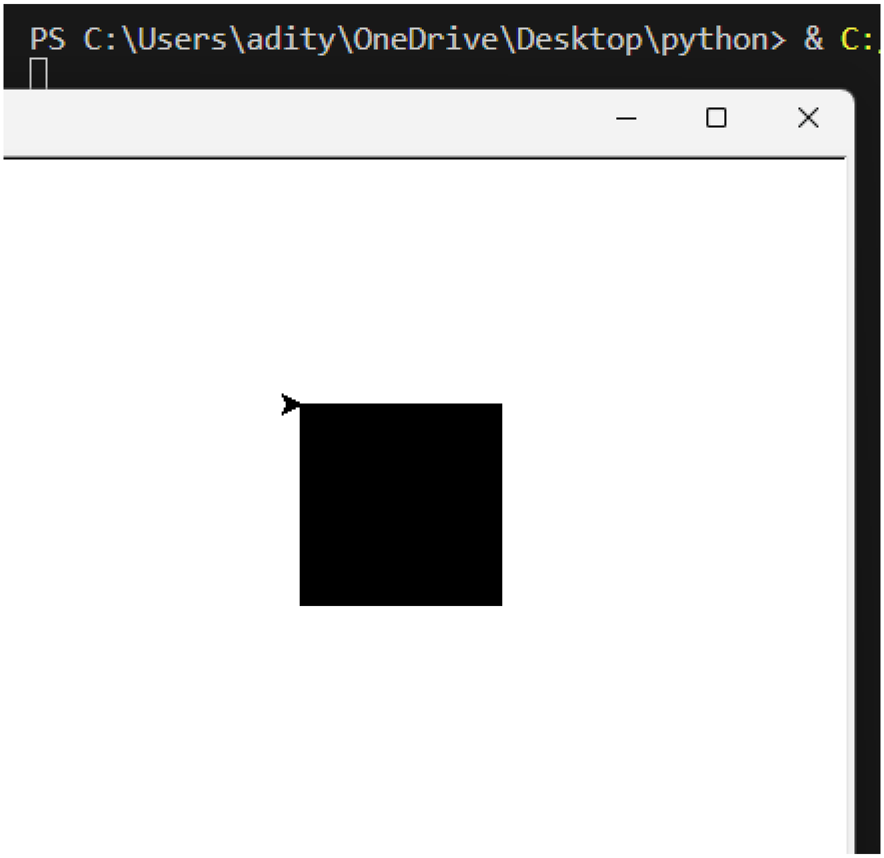
Objective:

Create a 5 X 5 rectangle whose top left corner is at (*row**5, *col**5). (Where is the bottom right corner?) If the sum of the *row* and *col* numbers is even, set the fill color of the rectangle to white, otherwise set it to black. Then draw the rectangle.

Program:

```
import turtle

def draw_rectangle(row, col):
    # Set up the turtle screen
    screen = turtle.Screen()
    screen.bgcolor("white")

    # Create a turtle object
    pen = turtle.Turtle()
    pen.speed(2)  # Set the drawing speed

    # Calculate the coordinates of the top left and bottom right corners
    x_top_left = col * 5 * 20  # Assuming each unit is 20 pixels
    y_top_left = row * 5 * 20
    x_bottom_right = x_top_left + 5 * 20
    y_bottom_right = y_top_left + 5 * 20

    # Set the fill color based on the sum of row and col
    if (row + col) % 2 == 0:
        pen.fillcolor("white")
    else:
        pen.fillcolor("black")

    # Move to the top left corner
    pen.penup()
    pen.goto(x_top_left, y_top_left)
    pen.pendown()

    # Draw the rectangle
    pen.begin_fill()
    for _ in range(4):
        pen.forward(5 * 20)  # Assuming each unit is 20 pixels
        pen.right(90)
    pen.end_fill()

    # Close the turtle graphics window on click
    screen.exitonclick()

# Example usage
row_number = 1
col_number = 2
draw_rectangle(row_number, col_number)
```

Output:



```
PS C:\Users\adity\OneDrive\Desktop\python> & C:
```