**MANMEET KAUR**

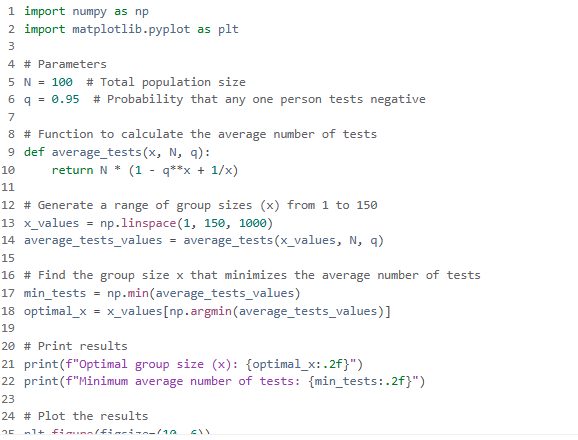
**19945**

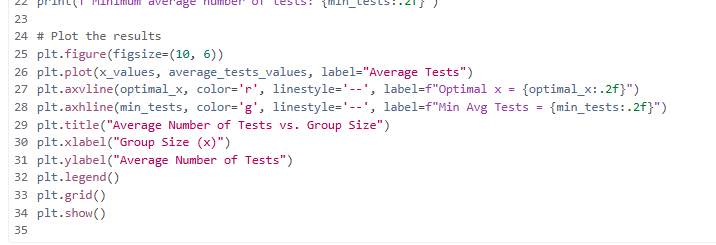
**MATH201 - Calculus-I**

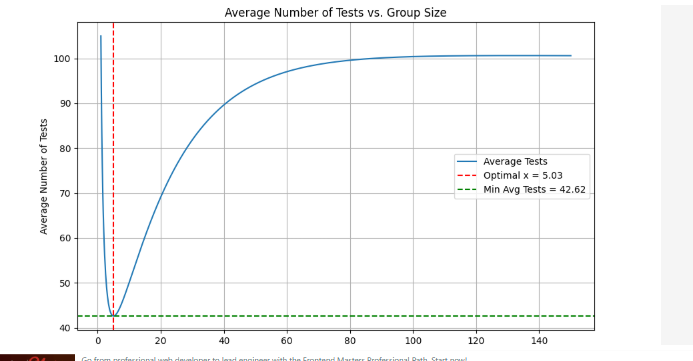
**Homework Assignment #5**

**Ques1**

[**https://www.mycompiler.io/view/FAV2uWrFpPK**](https://www.mycompiler.io/view/FAV2uWrFpPK)

****

****



import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 100 # Total population size

q = 0.95 # Probability that any one person tests negative

# Function to calculate the average number of tests

def average\_tests(x, N, q):

return N \* (1 - q\*\*x + 1/x)

# Generate a range of group sizes (x) from 1 to 150

x\_values = np.linspace(1, 150, 1000)

average\_tests\_values = average\_tests(x\_values, N, q)

# Find the group size x that minimizes the average number of tests

min\_tests = np.min(average\_tests\_values)

optimal\_x = x\_values[np.argmin(average\_tests\_values)]

# Print results

print(f"Optimal group size (x): {optimal\_x:.2f}")

print(f"Minimum average number of tests: {min\_tests:.2f}")

# Plot the results

plt.figure(figsize=(10, 6))

plt.plot(x\_values, average\_tests\_values, label="Average Tests")

plt.axvline(optimal\_x, color='r', linestyle='--', label=f"Optimal x = {optimal\_x:.2f}")

plt.axhline(min\_tests, color='g', linestyle='--', label=f"Min Avg Tests = {min\_tests:.2f}")

plt.title("Average Number of Tests vs. Group Size")

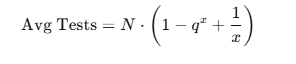
plt.xlabel("Group Size (x)")

plt.ylabel("Average Number of Tests")

plt.legend()

plt.grid()

plt.show()



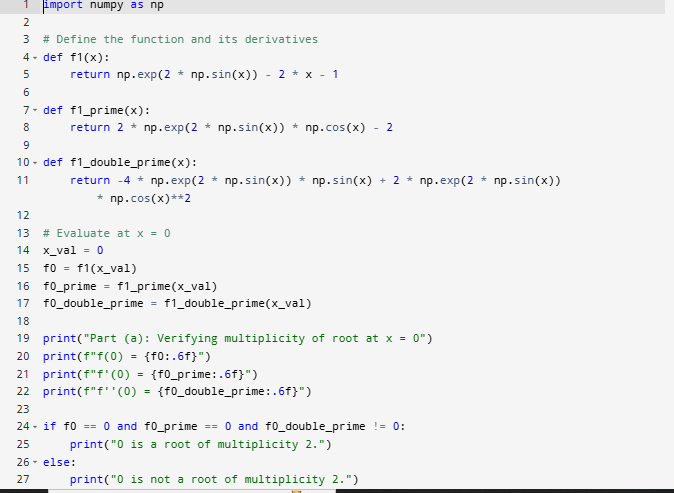
where qis the probability that an individual tests negative (q=0.95) and x*x* is the group size.

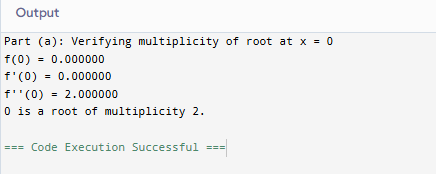
Using Python, we evaluated this function for x ranging from 1 to 150. The results show that the optimal group size is approximately *x*=5.03, which minimizes the average number of tests to 42.62. A plot was generated to visualize this relationship, confirming the result.

Ques2

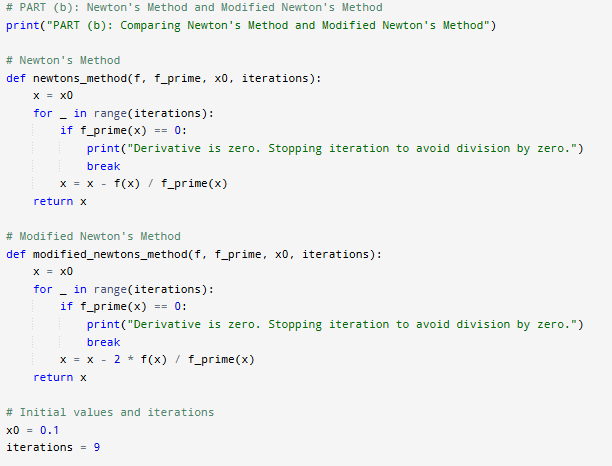
<https://www.programiz.com/online-compiler/85xWw3DOLVUEd>

Part a )

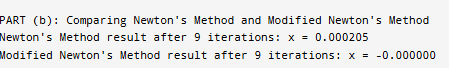




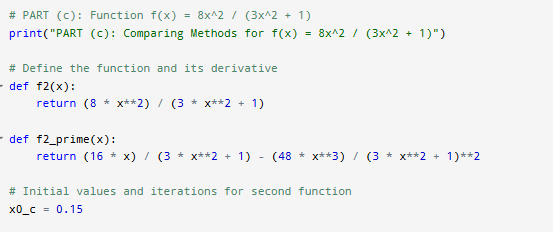
Part b )



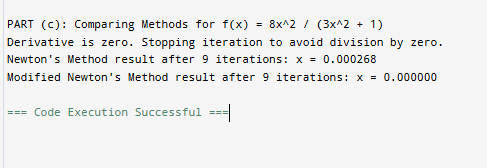
Output:



Part c)



Output:



Part (a):

Verifies whether 0 is a root of multiplicity 2 for the function

f(x) = e^(2sin(x)) - 2x - 1.

Prints f(0), f'(0), f''(0), and confirms multiplicity.

Part (b):

Compares Newton's Method and Modified Newton's Method for the function

f(x) = e^(2sin(x)) - 2x - 1.

Starts with x0 = 0.1, runs 9 iterations, and prints the final results.

Part (c):

Applies Newton's Method and Modified Newton's Method to the function

f(x) = (8x^2) / (3x^2 + 1).

Starts with x0 = 0.15, runs 9 iterations, and prints the final results.