Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**2**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **1** | Write a Python program that calculates the absolute as well as relative error present in the following measurements:  **Actual values:** [11.0098, 167.902, 56.0567, 67.9860] **Measured values:** [12.0001, 166.802, 55.0001, 69.0000] |
| **2** | Write a Python program that calculates the square root of following numbers using both the math.sqrt function (which uses floating-point arithmetic) and a custom square root function that uses integer arithmetic. Then, find and compare the results to observe the rounding error.  **(56.90, 100.45, 67.90, 25.67, 56.67)** |
| 3 | Write a Python program which finds the value of PI (π) using Taylor series, and then find the truncating error occurred due to the use of a finite number of terms. |

Submitted On:

\_\_\_\_\_\_\_\_\_\_\_\_

8 October 2024

(Date: DD/MM/YY)

**Task 1**

**Write a Python program that calculates the absolute as well as relative error present in the following measurements:**

**Actual values: [11.0098, 167.902, 56.0567, 67.9860]  
Measured values: [12.0001, 166.802, 55.0001, 69.0000]**

**Solution:**

num\_terms = 1000

true\_pi = 3.141592653589793

def calculate\_pi(terms):

    pi\_over\_4 = 0

    for n in range(terms):

        pi\_over\_4 += ((-1) \*\* n) / (2 \* n + 1)

    return pi\_over\_4 \* 4

def calculate\_truncation\_error(true\_value, estimated\_value):

    return abs(true\_value - estimated\_value)

num\_terms = 1000

true\_pi = 3.141592653589793

estimated\_pi = calculate\_pi(num\_terms)

error = calculate\_truncation\_error(true\_pi, estimated\_pi)

print(f"Estimated value of p: {estimated\_pi}")

print(f"Truncation error: {error}")

act = [11.0098,167.902, 56.0567, 67.9860]

mes = [12.0001,166.802, 55.0001, 69.0000]

for i in range(len(act)):

        error = abs(act[i]- mes[i])

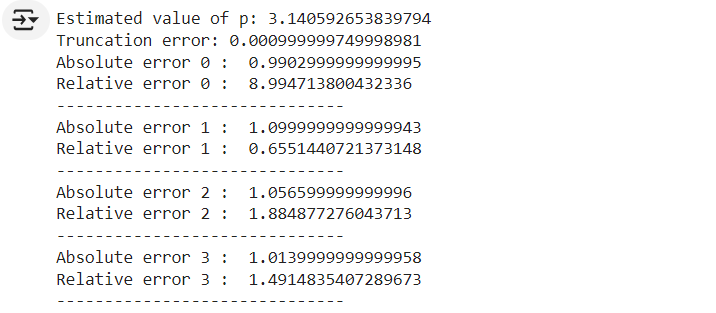
        relative = error / abs(act[i]) \* 100

        print("Absolute error",i,": ",error)

        print("Relative error",i,": ",relative)

        print("------------------------------")

**Output:**

****

**Task 2**

**Write a Python program that calculates the square root of following numbers using both the math.sqrt function (which uses floating-point arithmetic) and a custom square root function that uses integer arithmetic. Then, find and compare the results to observe the rounding error.**

**(56.90, 100.45, 67.90, 25.67, 56.67)**

**Solution:**

import math

def int\_sqrt(n):

    if n < 0:

        raise ValueError("Cannot compute square root of negative number")

    guess = n // 2

    better\_guess = (guess + n // guess) // 2

    while better\_guess < guess:

        guess = better\_guess

        better\_guess = (guess + n // guess) // 2

    return guess

numbers = [56.90, 100.45, 67.90, 25.67, 56.67]

print(f"{'Number':>10} {'math.sqrt':>15} {'Custom sqrt':>15} {'Rounding Error':>15}")

for num in numbers:

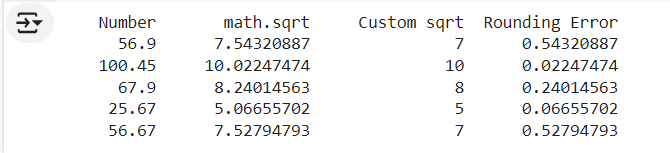
    math\_sqrt\_result = math.sqrt(num)

    custom\_sqrt\_result = int\_sqrt(int(num))

    rounding\_error = math\_sqrt\_result - custom\_sqrt\_result

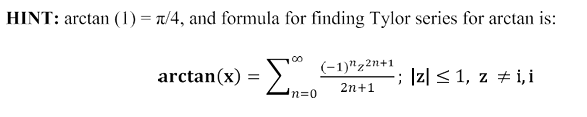
    print(f"{num:>10} {math\_sqrt\_result:>15.8f} {custom\_sqrt\_result:>15} {rounding\_error:>15.8f}")

**Output:**

****

**Task 3**

**Write a Python program which finds the value of PI (π) using Taylor series, and then find the truncating error occurred due to the use of a finite number of terms.**



**Solution:**

import math

# Taylor series for arctan(x)

def arctan\_taylor(x, terms):

    result = 0.0

    for n in range(terms):

        term = ((-1) \*\* n) \* (x \*\* (2 \* n + 1)) / (2 \* n + 1)

        result += term

    return result

# Function to calculate pi using arctan(1) = pi/4

def calculate\_pi(terms):

    arctan\_value = arctan\_taylor(1, terms)

    pi\_value = 4 \* arctan\_value

    return pi\_value

# Truncation error for pi approximation

def truncation\_error(approx\_pi):

    true\_pi = math.pi

    return abs(true\_pi - approx\_pi)

# Number of terms in the Taylor series

terms = 10  # You can change this to increase precision

# Calculating pi and the truncation error

approx\_pi = calculate\_pi(terms)

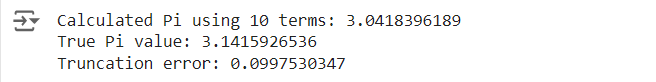
error = truncation\_error(approx\_pi)

print(f"Calculated Pi using {terms} terms: {approx\_pi:.10f}")

print(f"True Pi value: {math.pi:.10f}")

print(f"Truncation error: {error:.10f}")

**Output:**

****