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** |
| **2** | **Write a Python program the find the propagated error in the area and perimeter found for the following measurements** |
| **3** | **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.** |
| 4 | Write a python program which find the value of PI (π) using Taylor series, and then find the truncating error occurred due to the use of finite number of terms. |

Submitted On:

**Date: \_\_\_\_\_\_\_\_\_\_\_**

**Task No. 01:**

**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:**

def absolute\_error(actual,measured):

return abs(actual-measured)

def realtive\_error(actual,measured):

return abs((absolute\_error(actual,measured)/actual)\*100)

actual\_values = [11.0098, 167.902, 56.0567, 67.9860]

measured\_values = [12.0001, 166.802, 55.0001, 69.0000]

absolute\_errors=[]

realtive\_errors=[]

for i in range(len(actual\_values)):

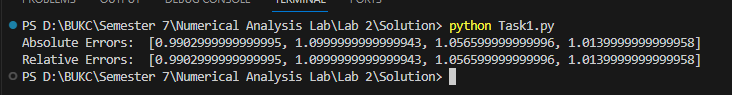
absolute\_errors.append(absolute\_error(actual\_values[i],measured\_values[i]))

realtive\_errors.append(absolute\_error(actual\_values[i],measured\_values[i]))

print('Absolute Errors: ',absolute\_errors)

print('Relative Errors: ',realtive\_errors)

**Output:**



**Task No. 02:**

**Write a Python program the find the propagated error in the area and perimeter found for the following measurements:**

**SQUARE (length: 45.09 cm; uncertainty: 0.01 cm)**

**CIRCLE (radius: 34.90 cm; uncertainty: 0.05 cm)**

**TRIANGLE (side1: 70.9 m; uncertainty: 0.23 m, side2: 89.07cm; uncertainty: 0.07 m,**

**base: 76.07cm; uncertainty: 0.04 m, height: 100.07cm; uncertainty: 0.05 m)**

**- TRAPEZIUM (side1: 670.9 m; uncertainty: 0.53 m (parallel one), side2: 849.07cm; uncertainty: 0.27 m (parallel one) side3: 376.07cm; uncertainty: 0.74 m, side4: 716.07cm; uncertainty: 0.14 m, height: 231.07cm; uncertainty: 0.25 m)**

**Solution:**

import math

def square\_area\_perimeter(length, length\_error):

area = length\*\*2

perimeter = 4 \* length

area\_error = 2 \* length \* length\_error

perimeter\_error = 4 \* length\_error

return area, perimeter, area\_error, perimeter\_error

def circle\_area\_perimeter(radius, radius\_error):

area = math.pi \* radius\*\*2

perimeter = 2 \* math.pi \* radius

area\_error = 2 \* math.pi \* radius \* radius\_error

perimeter\_error = 2 \* math.pi \* radius\_error

return area, perimeter, area\_error, perimeter\_error

def triangle\_area\_perimeter(side1, side2, base, height, side1\_error, side2\_error, base\_error, height\_error):

area = 0.5 \* base \* height

perimeter = side1 + side2 + base

area\_error = 0.5 \* height \* (base\_error + height\_error) + 0.5 \* base \* height\_error

perimeter\_error = side1\_error + side2\_error + base\_error

return area, perimeter, area\_error, perimeter\_error

def trapezium\_area\_perimeter(side1, side2, side3, side4, height, side1\_error, side2\_error, side3\_error, side4\_error, height\_error):

area = 0.5 \* (side1 + side3) \* height

perimeter = side1 + side2 + side3 + side4

area\_error = 0.5 \* height \* (side1\_error + side3\_error) + 0.5 \* (side1 + side3) \* height\_error

perimeter\_error = side1\_error + side2\_error + side3\_error + side4\_error

return area, perimeter, area\_error, perimeter\_error

length\_sq = 45.09

length\_error\_sq = 0.01

radius\_circ = 34.90

radius\_error\_circ = 0.05

side1\_tri = 70.9

side2\_tri = 89.07

base\_tri = 76.07

height\_tri = 100.07

side1\_error\_tri = 0.23

side2\_error\_tri = 0.07

base\_error\_tri = 0.04

height\_error\_tri = 0.05

side1\_trap = 670.9

side2\_trap = 849.07

side3\_trap = 376.07

side4\_trap = 716.07

height\_trap = 231.07

side1\_error\_trap = 0.53

side2\_error\_trap = 0.27

side3\_error\_trap = 0.74

side4\_error\_trap = 0.14

height\_error\_trap = 0.25

area\_sq, perimeter\_sq, area\_error\_sq, perimeter\_error\_sq = square\_area\_perimeter(length\_sq, length\_error\_sq)

area\_circ, perimeter\_circ, area\_error\_circ, perimeter\_error\_circ = circle\_area\_perimeter(radius\_circ, radius\_error\_circ)

area\_tri, perimeter\_tri, area\_error\_tri, perimeter\_error\_tri = triangle\_area\_perimeter(side1\_tri, side2\_tri, base\_tri, height\_tri, side1\_error\_tri, side2\_error\_tri, base\_error\_tri, height\_error\_tri)

area\_trap, perimeter\_trap, area\_error\_trap, perimeter\_error\_trap = trapezium\_area\_perimeter(side1\_trap, side2\_trap, side3\_trap, side4\_trap, height\_trap, side1\_error\_trap, side2\_error\_trap, side3\_error\_trap, side4\_error\_trap, height\_error\_trap)

print('Square:')

print('Area:', area\_sq, 'cm^2 (±', area\_error\_sq, 'cm^2)')

print('Perimeter:', perimeter\_sq, 'cm (±', perimeter\_error\_sq, 'cm)')

print('Circle:')

print('Area:', area\_circ, 'cm^2 (±', area\_error\_circ, 'cm^2)')

print('Perimeter:', perimeter\_circ, 'cm (±', perimeter\_error\_circ, 'cm)')

print('Triangle:')

print('Area:', area\_tri, 'cm^2 (±', area\_error\_tri, 'cm^2)')

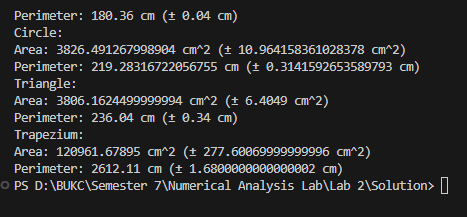
print('Perimeter:', perimeter\_tri, 'cm (±', perimeter\_error\_tri, 'cm)')

print('Trapezium:')

print('Area:', area\_trap, 'cm^2 (±', area\_error\_trap, 'cm^2)')

print('Perimeter:', perimeter\_trap, 'cm (±', perimeter\_error\_trap, 'cm)')

**Output:**



**Task No. 03:**

**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 difference(list1):

actual\_sq=[]

measured\_sq=[]

diffList=[]

for i in range(len(list1)):

actual\_sq.append(math.sqrt(list1[i]))

measured\_sq.append(list1[i]\*\*(1/2))

diff=actual\_sq[i]-measured\_sq[i]

diffList.append(diff)

return actual\_sq,measured\_sq,diffList

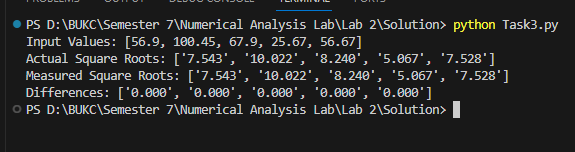
list1=[56.90, 100.45, 67.90, 25.67, 56.67]

actual\_square\_root, measured\_square\_root, differences = difference(list1)

print("Input Values:", list1)

print("Actual Square Roots:", actual\_square\_root)print("Measured Square Roots:", measured\_square\_root)print("Differences:", differences)

**Output:**



**Task No. 04:**

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

**HINT: arctan (1) = π/4, and formula for finding Tylor series for arctan is:**

**𝐚𝐫𝐜𝐭𝐚𝐧(𝐱)= Σ (−1) 𝑛𝑧2𝑛+12𝑛+1; ∞𝑛=0 |z|≤1, z ≠i,i**

**Solution:**

import math

def arcTan(x,N):

res=0.0

for i in range(N):

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

res+=n

return res

print('Actual PI: ',math.pi)

terms=100

approx\_pi=4\*arcTan(1,terms)

print('Approximate Value of PI: ',approx\_pi)

truncationError=abs(approx\_pi-math.pi)

print('Truncation Error: ',truncationError)

**Output:**

