# Bahria University,

## Karachi Campus



LAB EXPERIMENT NO.

**\_02\_**

LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| 01 | Write a Python program that calculates the absolute as well as relative error present in the following measurements:   * Actual values: [111.0098, 167.902, 56.0567, 67.9860] * Measured values: [112.0001, 166.802, 55.0001, 69.0000] |
| 02 | Write a Python program that calculates the square root of the 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.  Numbers: (56.90, 100.45, 67.90, 25.67, 56.67) |
| 03 | Write a Python program to find the value of π using the Taylor series, and then find the truncating error due to the use of a finite number of terms.  Hint: arctan(1) = π/4, and the formula for finding the Taylor series for arctan is: arctan⁡(x)=∑n=0∞(−1)nx2n+12n+1\arctan(x) = \sum\_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}arctan(x) |

Submitted On:

Date: 1/10/2024

**Task No 01:** Write a Python program that calculates the absolute as well as relative error present in the following measurements:

* Actual values: [111.0098, 167.902, 56.0567, 67.9860]
* Measured values: [112.0001, 166.802, 55.0001, 69.0000]

**Solution:**

actual\_values = [111.0098, 167.902, 56.0567, 67.9860]

measured\_values = [112.0001, 166.802, 55.0001, 69.0000]

def calculate\_errors(actual, measured):

abs\_errors = []

rel\_errors = []

for a, m in zip(actual, measured):

abs\_error = abs(a - m)

rel\_error = abs\_error / abs(a)

abs\_errors.append(abs\_error)

rel\_errors.append(rel\_error)

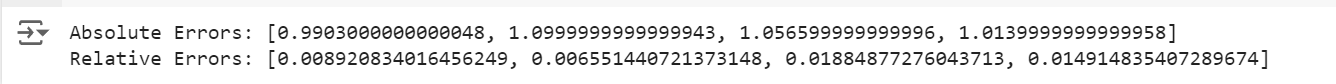
return abs\_errors, rel\_errors

abs\_errors, rel\_errors = calculate\_errors(actual\_values, measured\_values)

print("Absolute Errors:", abs\_errors)

print("Relative Errors:", rel\_errors)

**Output:**

****

**Task No 02:** Write a Python program that calculates the square root of the 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.

Numbers: (56.90, 100.45, 67.90, 25.67, 56.67)

**Solution:**

import math

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

def custom\_sqrt(n):

x = n

while True:

root = 0.5 \* (x + (n / x))

if abs(root - x) < 1e-10:

break

x = root

return root

for num in numbers:

sqrt\_math = math.sqrt(num)

sqrt\_custom = custom\_sqrt(num)

rounding\_error = sqrt\_math - sqrt\_custom

print(f"Number: {num}")

print(f"math.sqrt: {sqrt\_math}, custom\_sqrt: {sqrt\_custom}")

print(f"Rounding Error: {rounding\_error}\n")

**Output:**

**A computer code with numbers and symbols

Description automatically generatedTask No 03:** Write a Python program to find the value of π using the Taylor series, and then find the truncating error due to the use of a finite number of terms.

Hint: arctan(1) = π/4, and the formula for finding the Taylor series for arctan is: arctan

**Solution:**

def arctan\_taylor\_series(x, terms):

    result = 0

    for n in range(terms):

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

        result += term

    return result

def calculate\_pi(terms):

    pi\_estimate = 4 \* arctan\_taylor\_series(1, terms)

    return pi\_estimate

pi\_estimate = calculate\_pi(terms)

pi\_actual = math.pi

truncation\_error = abs(pi\_actual - pi\_estimate)

print(f"Estimated value of pi: {pi\_estimate}")

print(f"Actual value of pi: {pi\_actual}")

print(f"Truncation Error: {truncation\_error}")

**Output:**

**A black text with numbers

Description automatically generated**