**RSE Assessment Task 2**

**Results Calculated with program:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **N** | Value of π | Error (%) |
| **Trapezoidal** | **4** | **3.1311764705882354** | **0.33073400643584533** |
| **Simpson’s** | **4** | **3.14156862745098** | **-6.135317706782376e-05** |
| **Trapezoidal** | **12** | **3.1404352468468506** | **0.03601556997498159** |

**I originally wrote the program in Python but when looking through the job posting I saw C++ listed as desirable so I also wrote the program in C++. They are the same program just written in both languages and give the same result. Only the Python code is commented.**

**Python Source Code:**

**import** **math**  
  
truePi = **3.1415667**  
  
**def** **f**(x):  
 'Defines the function to pass x through'  
 **return** **4** /(**1** + x\*\***2**)  
  
def **percentageError**(trueValue, approxValue):  
 'Takes both values and calculates the percentage error of the approximate value'  
 percError = ((trueValue-approxValue)/ trueValue) \* **100**  
 **return** percError  
  
def **deltax** (a, b, n):  
 'Calculates delta x using the input values'  
 **return** (b - a)/n

def **trapezoidalRule**(a, b, n):  
 '''  
 Takes the upper and lower limits of the integration as a and b and n as the  
 number of subintervals to be used in the trapezoidal rule  
 '''  
 subinterval = deltax(a, b, n) # sets subinterval as delta x because it will be used for the increment of x and in the formula  
 sigmaFx = **0** # initialises the sum total of the sigma part of the formula  
 **for** i **in** range(**1**, n): # performs the sigmaf(xi) part of the formula   
 sigmaFx += f(i \* subinterval)  
 result = subinterval\*(sigmaFx + ((f(n \* subinterval) + f(**0** \* subinterval))/**2**)) # calculates the trapezoidal rule approximate value  
 **return** result

**def** **simpsonsRule**(a, b, n):  
 '''  
 Takes the upper and lower limits of the integration as a and b and n as the  
 number of subintervals to be used in the Simpson's rule  
 '''  
 subinterval = deltax(a, b, n) # sets subinterval as delta x because it will be used for the increment of x and in the formula  
 sigmaN1 = **0** # initialises the sum total of the n-1 sigma  
 sigmaN2 = **0** # and the n-1 sigma  
 **for** i **in** range(**1**,n): # performs the sigmaf(xi) for both sigmas of the value, stops at n-1 for sigmaN1  
 **if** i % **2** == **1**: # if i is odd i.e. 1,3,5   
 sigmaN1 += **4**\*f(i \* subinterval) # multiplies by 4 like 4f(x) in the formula  
 **elif** i % **2** == **0** **and** i < n - **1**: # if i is even i.e. 2,4,6 and stops at n-2  
 sigmaN2 += **2**\*f(i \* subinterval) # multiplies by 2 like 2f(x)   
 result = (subinterval / **3**) \* (f(**0** \* subinterval) + sigmaN1 + sigmaN2 + f(n \* subinterval))  
 **return** result

**def** **main**():  
 '''  
 Gets the users inputs to be used in the formula  
 Also creates a simple menus so all the required calculations can be do in one execution of the program  
 '''  
 n = int(input("**\n**Enter the amount of subintervals: ")) # gets the 3 inputs to be used in the approximation formulas  
 a = int(input("Enter the lower limit: "))  
 b = int(input("Enter the upper limit: "))  
 # creates the menu to run multiple caluclations  
 choice = input("Enter 1 for Trapezoidal Rule, 2 for Simpsons Rule or 3 Change Inputs: ")  
 **if** choice == "1": # calls the trapezoidal rule and the percentage error and outputs the results  
 approxValue = trapezoidalRule(a, b, n)  
 **print**("**\n**True Value of pi: ", truePi, "**\n**Approximate Value: ", approxValue, "**\n**Percentage Error :", percentageError(truePi, approxValue), "**\n**")  
 choice = input("Enter 4 to Calculate more or anything else to exit: ")  
 **if** choice == "4":  
 main()  
 **else**:  
 **return**  
 **elif** choice == "2": # calls the simpson's rule and the percentage error and outputs the results  
 approxValue = simpsonsRule(a, b, n)  
 **print**("**\n**True Value of pi: ", truePi, "**\n**Approximate Value: ", approxValue, "**\n**Percentage Error :", percentageError(truePi, approxValue), "**\n**")  
 choice = input("Enter 4 to Calculate more or anything else to exit: ")  
 **if** choice == "4":  
 main()  
 **else**:  
 **return**  
 **elif** choice == "3":  
 main()  
 **else**:  
 **print**("Please enter a valid input")  
 main()  
 **return**  
  
main()

**C++ Code**

#include <iostream>  
#include <iomanip>  
#include <cmath>  
#include <string>  
  
using **namespace** std;  
  
double **f**(**double** x)  
{  
 **return** **4.0** / (**1.0** + x \* x);  
}  
  
double **deltax**(**double** a, **double** b, **double** n)  
{  
 **return** (b - a) / n;  
}  
  
double **trapezoidalRule**(**double** a, **double** b, **int** n)  
{  
 **double** subinterval = deltax(a, b, n);  
 **double** sigmafx = **0.0**;  
 **for** (**int** i = **1**; i < n; i++)  
 {  
 sigmafx += f(i \* subinterval);  
 }  
 **return** subinterval \* (sigmafx + ((f(n \* subinterval) + f(**0** \* subinterval)) / **2**));  
}

**double** **simpsonsRule**(**double** a, **double** b, **int** n)  
{  
 **double** subinterval = deltax(a, b, n);  
 **double** sigmaN1 = **0.0**;  
 **double** sigmaN2 = **0.0**;  
 **for** (**int** i = **1**; i < n; i++)  
 {  
 **if** (i % **2** == **1**)  
 {  
 sigmaN1 += **4** \* f(i \* subinterval);  
 }  
 **else** **if** (i % **2** == **0** || i < n - **1**)  
 {  
 sigmaN2 += **2** \* f(i \* subinterval);  
 }  
 }  
 **return** (subinterval / **3**) \* (f(**0** \* subinterval) + sigmaN1 + sigmaN2 + f(n \* subinterval));  
}  
  
double **percentageError**(**double** trueValue, **double** approxValue)  
{  
 **return** ((trueValue - approxValue) / trueValue) \* **100**;  
}

**int** **main**()  
{  
 **double** truePi = **3.1415667**;  
 **int** n;  
 **double** a;  
 **double** b;  
 **double** approxValue;  
 cout << "Enter the amount of subintervals: ";  
 cin >> n;  
 cout << "Enter the lower limit: ";  
 cin >> a;  
 cout << "Enter the upper limit: ";  
 cin >> b;  
 std::string choice;  
 cout << "Enter 1 for Trapezoidal Rule, 2 for Simpsons Rule or 3 Change Inputs : ";  
 cin >> choice;  
**switch** (std::stoi(choice))  
 {  
 **case** **1**:  
 approxValue = trapezoidalRule(a, b, n);  
 std::cout << "True value of Pi: " << truePi << "**\n**Approximation using Trapezoidal rule: " << std::setprecision(**20**) << approxValue << "**\n**Percentage Error: " << percentageError(truePi, approxValue);  
 cout << "**\n**Enter 4 to calculate more or 5 to close: ";  
 cin >> choice;  
 **if** (choice == "4")  
 {  
 main();  
 }  
 **else** **if** (choice == "5")  
 {  
 **return** **0**;  
 **break**;  
 }

**case** **2**:  
 approxValue = simpsonsRule(a, b, n);  
 std::cout << "True value of Pi: " << truePi << "**\n**Approximation using Simpson's rule: " << std::setprecision(**20**) << approxValue << "**\n**Percentage Error: " << percentageError(truePi, approxValue);  
 cout << "**\n**Enter 4 to calculate more or 5 to close: ";  
 cin >> choice;  
 **if** (choice == "4")  
 {  
 main();  
 }  
 **else** **if** (choice == "5")  
 {  
 **return** **0**;  
 **break**;  
 }  
 **case** **3**:  
 main();  
 **break**;  
 **default:**  
 cout << "Please enter a valid input**\n**";  
 main();  
 }  
  
 **return** **0**;  
}