SOEN 6441

ADVANCED PROGRAMMING PRACTICES

DELIVERABLE 1

TEAM E

Pouria Pirian (40207625)
Sachin Prakash (40218678)
Nishant Saini (40195801)
Chandra Sagar Reddy Sangu (40234194)
Omer Sayem (40226505)
Kajal Sehrawat (40230025)

Instructor:

Prof. PANKAJ KAMTHAN

Department of Computer Science and Software Engineering
Gina Cody School of Engineering and Computer Science
Concordia University, Montreal

March 20, 2023

Contents

Overview

1.1 Introduction

Coasters prevent condensation from dripping along the glass, which can damage the surface of tables. The project referred to as **CHEERS** deals with two circular coasters overlapping each other. The objective of **CHEERS** is to compute the diameter of the overlapping region such that the real estate is half that of any of the coasters.

1.2 Background

In this project, we determine the roots of an equation to calculate the angle made at the vertex of a coaster using the **Secant approximation method**. This method approximates the value of a function using a secant line passing through two points on a graph.

$$x_{n+1} = x_n - \frac{f(x_n) \cdot (x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

The Sin/Cos value is approximated using the **McLaurin Series** which is the sum of derivatives of a function.

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

1.3 Scope

The length **l** of the overlapping coaster region can be computed by the equation

$$l = 2R\left(1 - \cos\frac{\alpha}{2}\right)$$

R $\rightarrow$ radius of the coaster
$\alpha \rightarrow$ angle created with the vertex at the centre point of the left coaster

The angle, α can be calculated using

$$\alpha - \sin(\alpha) = \frac{\pi}{2}$$

- The roots of the equation need to be determined to compute the value of α. We use the Secant approximation method to arrive at a solution.
- The McLaurin series has been used to approximate the Sine and Cos values and, subsequently, calculate l.

1.4 Objectives

- To compute the value of Π.
- To calculate the exponent of a number using iteration.
- To figure out the factorial of a number using iteration.
- To approximate Sin and Cos using McLaurin Series.
- To find the roots of an equation using Secant Approximation.
- To generate XML files valid with respect to a DTD.

1.5 Assumptions

- The initial guess values for approximating the roots of an equation will be pre-determined.
- The default precision for approximating Sine and Cos functions has been set to 4.
- A maximum of 100 attempts will be made to find the roots of the equation, after which the function will be considered non-convergent.

Problem 1

2.1 Solution approach

The product CHEERS was built using RDD principles. Hence the roles and responsibilities were split among multiple classes. Each class was designed using the Single Responsibility Principle (SRP). The list of classes are as follows.

- Root Approximation: Calculates the roots of a given equation using the Secant method.
- Math Library: Calculates the power, factorial and Pi values.
- Trigonometry: Trigonometric approximations including Sine and Cos are computed using the McLaurin series.
- Output Generator: Generates textual output
- Controller: It calculates the angle at the vertex, and length of the overlapping segment and generates the output using the aforementioned classes.

When the user runs the program to find out the overlapping length L, the Controller class fetches the radius of circle from the user. Next, the Root Approximation, Math Library and Trigonometry classes are initialized using their respective wrapper classes. A wrapper class provides a way to modify or extend the behaviour of an existing class without having to modify the class itself.

To calculate the overlapping length l, the angle created with the vertex at the centre point of the left coaster will have to be computed. To achieve this, the roots of the equation $\alpha - \sin(\alpha) = \frac{\pi}{2}$ should be determined. The roots of this equation are calculated by the secant approximation method in the Root Approximation class. Secant approximation calls custom Sine function, which in turn uses the factorial and exponent methods in the Math Library to calculate the Sine value of a given number. Here, the Sine of a given number is calculated using the McLaurin series. The calculated Sin value is then returned Secant Approximation method. Finally, the calculated value for alpha flows back to the Controller class.

Next, the distance is calculated using the equation $l = 2R\left(1 - \cos\frac{\alpha}{2}\right)$. Similar to Sine, the Cos value is calculated using the McLaurin series. Finally, the overlapping length is displayed to the user in a textual format.

2.2 Object Oriented Design

2.2.1 Sequence Diagram

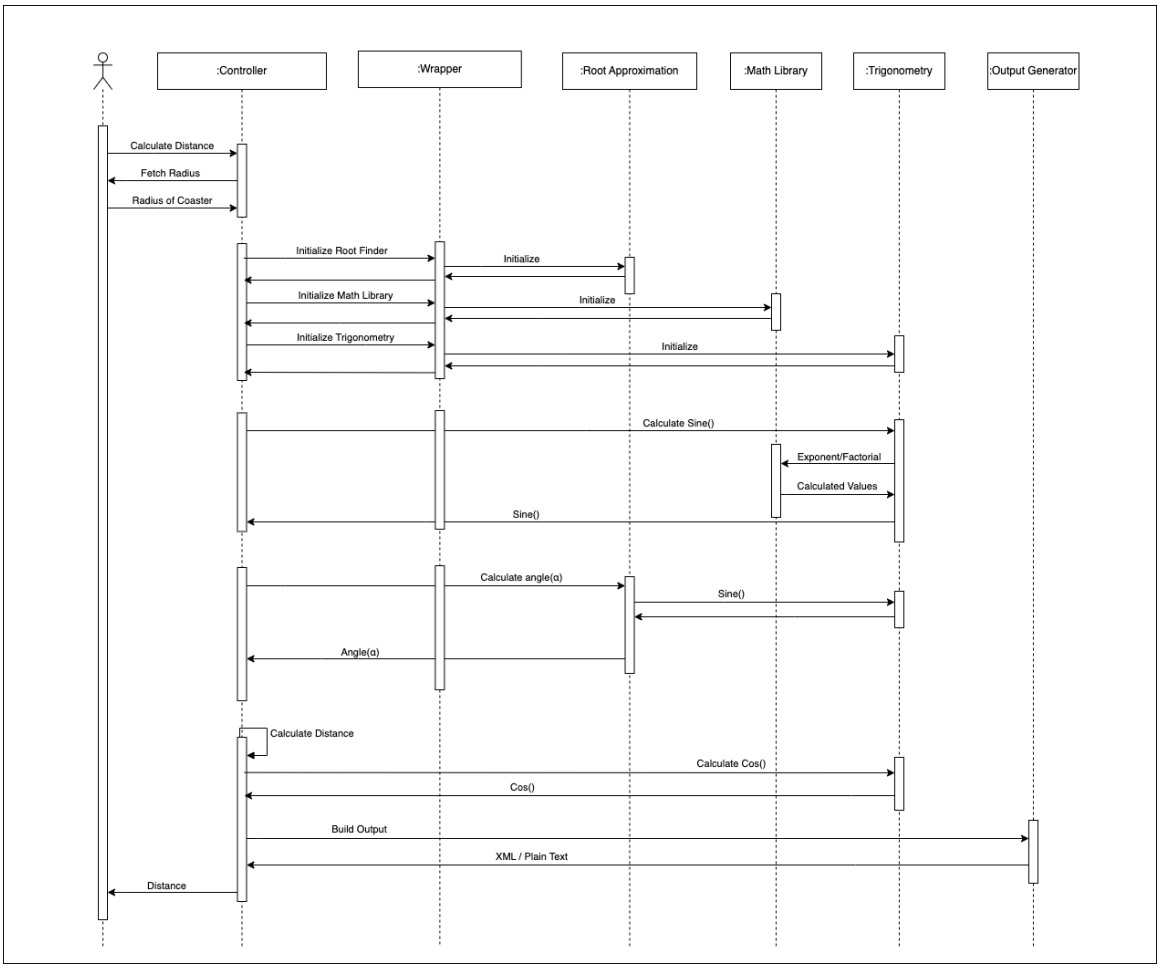

Figure 2.1: Flow of events for calculating the length

2.2.2 CRC Cards

Trigonometry

The trigonometry class is responsible for calculating the Sine and Cos of a given number. Its role is to calculate the Sin and Cos of a number using the McLaurin series to a pre-defined precision level. It collaborates with the Math Library class for accessing the exponent and factorial functions and with the Root Approximation class that need Sin and Cos for calculating the roots of the function.

Trigonometry	
Inject classes at runtime	Controller
Calculate Sine using McLaurin series	Math Library
Calculate Cos using McLaurin series	Error Handler

Figure 2.2: CRC Card - Trigonometry Class

Math Library

The responsibility of Math Lib class is to provide the function definition of Mathematical operations such as factorial and power of a given number. Its role is to calculate the value of PI using Leibniz formula, factorial and power of a given number using recursion. It collaborates with trigonometry, Root approximation and error handling class.

Math Library	
Inject classes at runtime	
Find exponent of a number	Controller
Find factorial of a number	Error Handler
Calculate Pi	
Handle runtime errors	
Return calculated output	

Figure 2.3: CRC Card - Math Library Class

Root Approximation

The root approximation class is the core of the entire project. It contains two functions that calculate the roots of the given equation. It collaborates with Trigonometry and Math Lib classes as they contain necessary functions used for finding roots of an equation. As we always try to reduce the coupling with multiple classes and increase the cohesion within the class, we chose this class as to perform only one task of calculating the roots of equation.

Root Approximation	
Inject classes at runtime	Controller
Find roots using Secant Method	Trigonometry
Handle runtime errors	Math Library
	Error Handler

Figure 2.4: CRC Card - Root Approximation Class

Validator

The role of the input validator class is to ensure that the data received from the user is in the appropriate format (positive integer). If the user enters invalid data, it prompts the user to provide valid data.

Validator	
Validate user input	Controller

Figure 2.5: CRC Card - Validator Class

Error Handler

The role of the error handler class is to handle errors that are generated. It collaborates with the controller class. The main responsibility of this class is to handle the errors generated and return an appropriate error message that is comprehensible to the user. The rationale for choosing this class is to define and handle all possible errors that are generated in the process of calculating length l.

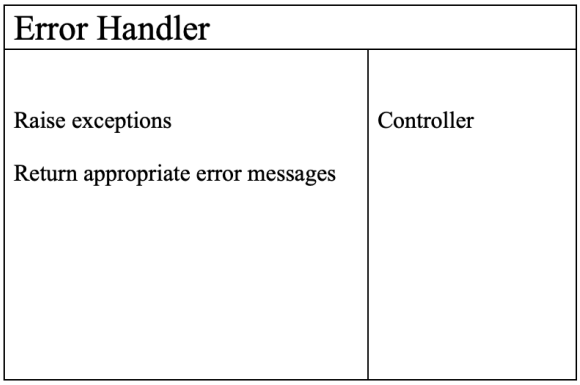

Figure 2.6: CRC Card - Error Handler Class

Output Generator

The main role of the Output Generator class is to generate XML files. It stores the radius of the circle and the overlapping length l. The rationale behind choosing this class is to make sure that the processing and output are separated so that any changes to the output format do not affect the processing capabilities. This reduces the coupling between the classes.

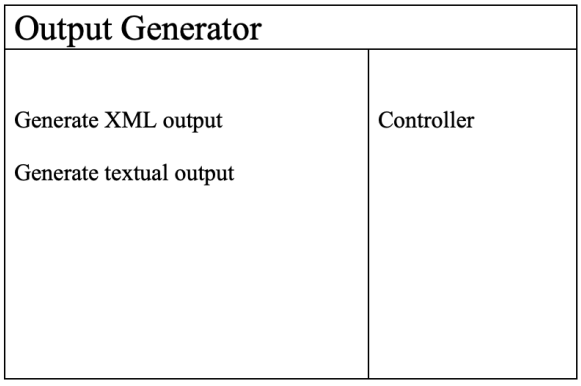

Figure 2.7: CRC Card - OutputGenerator Class

Controller

The main role of this class is to calculate α, the angle with a vertex at the centre of the left circle. It also calculates the length of the overlapping segment. To do this, it collaborates with other classes, which include Math Library, Trigonometry and Root Approximation. Each of these classes contains various mathematical functions such as Sine, Cos and Secant Approximation that are required for calculating the angle α and length l.

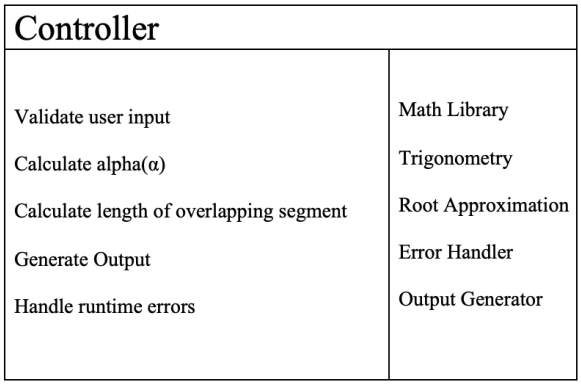

Figure 2.8: CRC Card - Controller Class

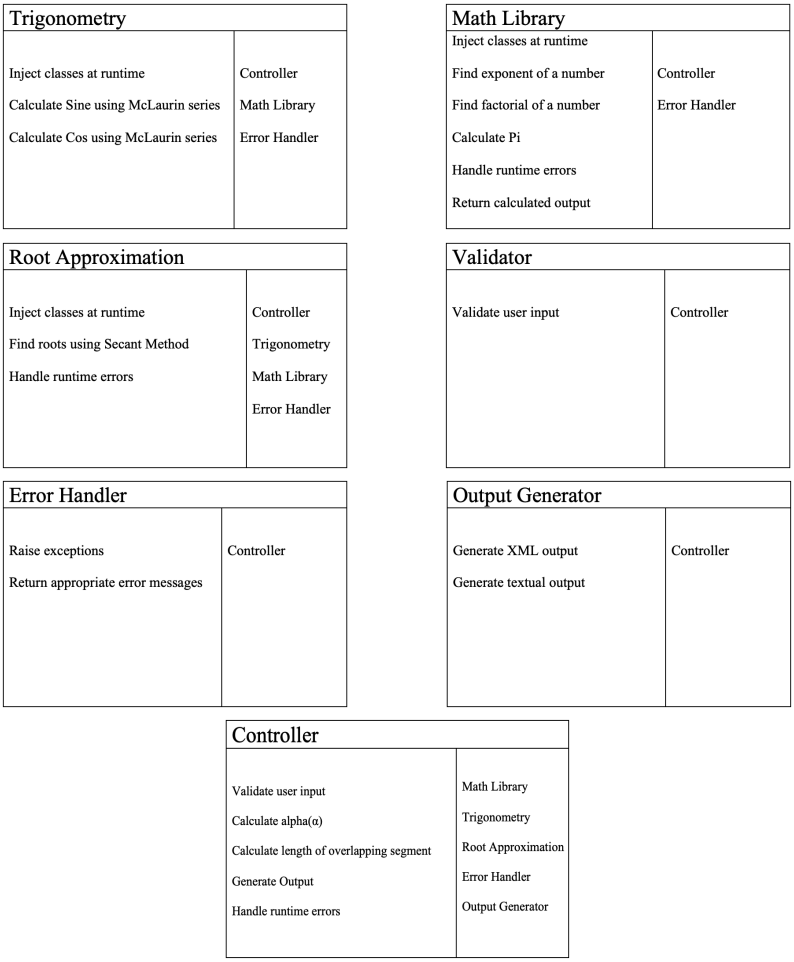

Figure 2.9: Collection of CRC Cards for CHEERS

Problem 2

3.1 Pseudocode

3.1.1 Sine() & Cos()

We arrive at a value for sine/cos of a number using the Mclaurin series expansion.

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Algorithm 1 McLaurin Series

Parameters:

rad(float): The input angle value in radians for which we want to calculate the sine.

precision(int): Number of terms in the series expansion used to approximate the sine value.

Returns:

float: The calculated sine value for the input angle in radians.

```
1: function CALCULATE_SIN(rad, precision)
2:     sin_val ← 0
3:     for i ← 0 to precision do                ▷ Iterate upto the desired precision
4:         term ← (−1)^i ∗ rad^(2i+1)/(2i + 1)!  ▷ Value of the current term in the expansion
5:         sin_val ← sin_val + term
6:     end for
7:     return sin_val
8: end function
```

```
1: function CALCULATE_COS(rad, precision)
2:     cos_val ← 1
3:     for i ← 1 to precision do                ▷ Iterate upto the desired precision
4:         term ← (−1)^i ∗ rad^(2i)/(2i)!        ▷ Value of the current term in the expansion
5:         cos_val ← cos_val + term
6:     end for
7:     return cos_val
8: end function
```

There are various methods of calculating sine/cos of a number using various infinite series like Taylor series, McLauren series, power series and Euler's formula. While each of the methods have advantages and disadvantages of their own, McLaurin Series has some distinct advantages over other methods. They are as follows :

- **Simplicity** : MacLaurin Series simple to use, as it only uses relatively simple mathematical functions. Due to the same reason it is easy to implement a computer program of it.
- **Convergence** : The McLaurin series converges for all values of x, which means that it can be used to calculate the sine function accurately for any angle.
- **Accuracy** : The accuracy of Maclaurin series increases with every new term that is added. So, with every iteration it rapidly converges to true value of sin.
- **Versatility**: The McLaurin series can be easily extended to calculate other trigonometric functions, such as cosine and tangent, by simply changing the coefficients of the terms in the series.

3.1.2 Pi(π)

The value of pi is calculated using Leibniz formula[1].

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Algorithm 2 Pi Approximation

Returns:
float: An approximate value of Pi.

```
pi_estimate ← 0
sign ← 1
denominator ← 1
for i in range(1000) do
    pi_estimate ← pi_estimate + sign/denominator
    sign ← sign * −1
    denominator ← denominator + 2        ▷ Compute next term in series and update variables
end for
pi_estimate ← pi_estimate * 4                    ▷ Multiply by 4 to get estimated value of pi
return pi_estimate
```

Leibniz formula is a special case of $arctan(x)$ series, that is as follows.

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

When x is equal to 1, we get $\arctan(1) = \frac{1}{4}\pi$. The advantages of Leibniz formula is as follows:

- **Simple implementation** : It is a simple formula that can be easily programmed, as it requires only basic mathematical operations like power and division.
- **Accuracy** : The Leibniz formula gives an accurate approximation of π, with each additional term of the series increasing the accuracy of the approximation. The formula can be used to calculate π to any desired level of precision, depending on how many terms of the series are used.

3.1.3 Secant Approximation

The roots of an equation can be derived using the Secant Approximation method.

$$x_{n+1} = x_n - \frac{f(x_n) \cdot (x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

Algorithm 3 Secant Approximation

Parameters:
func (callable): The function to find the root of.
x0 (float): The first initial guess.
x1 (float): The second initial guess.
eps (float): The desired level of accuracy.
max_iter (int): The maximum number of iterations to perform.
Returns:
float: Roots of the function.

```
function SECANT_METHOD(func, x_0, x_1, eps, max_iter)
    for step in range(max_iter) do
        x_2 ← x_0 − (x_1 − x_0) · func(x_0) / (func(x_1) − func(x_0))    ▷ Calculate new guess for root
        if |func(x2)| < eps then
            return x2
        else
            x0, x1 ← x1, x2                                              ▷ Update previous guesses
        end if
    end for
    raise ValueError("Maximum number of iterations reached.")
end function
```

```
function func(x)
    square ← x^2
    sine ← sin(x)
    result ← square + sine − 2
    return result
end function
```

The secant method is similar to the Newton-Raphson method, but it uses an approximation of the derivative instead of the actual derivative. Secant method has various advantages over other methods like bisection method and Newton-Raphson method. They are as follows:

- **No derivative required** : One of the main advantage of secant method is that we do not need to calculate the derivative of the function whose roots we are trying to calculate. So, writing a program is lot easier that that of Newton-Raphson method, which requires derivative of the function.
- **Convergence rate** : The secant method has a convergence rate that is faster than the bisection method but slower than the Newton-Raphson method. However, it is less likely to diverge than the Newton-Raphson method.
- **Applicability to nonlinear functions**: The secant method is applicable to nonlinear functions, whereas the bisection method is applicable only to continuous functions that change sign over an interval.

- **Multiple roots** : Unlike bisection method, secant method can detect multiple roots in a given interval.

3.1.4 Factorial(!)

This is the algorithm used to obtain the factorial of a number.

Algorithm 4 Factorial

Parameters:
num(int): This argument represents the number to calculate the factorial of.
Returns:
int: A numerical value that represents the factorial of the input number.

```
1: function FACTORIAL(num)
2:     if num < 0 then
3:         raise Exception("Factorial can't be calculated for negative numbers")
4:     end if
5:     result ← 1
6:     for i ← 1 to num do
7:         result ← result × i
8:     end for
9:     return result
10: end function
```

3.1.5 Power(e^x)

The power of a number can be obtained using the following algorithm.

Algorithm 5 Power

Parameters:
base(int): This parameter represents the base of the exponentiation.
exponent(int): An integer value that specifies how many times the number should be multiplied by itself.
Returns:
int: A numerical value that represents the result of the exponentiation.

```
procedure POWER(base, exponent)
    if exponent == 0 then
        return 1
    else
        return base * Power(base, exponent − 1)
    end if
end procedure
```

This algorithm provides a clear and concise way of expressing a mathematical operation, such as power. The recursive algorithm for power is simple and intuitive. It can also be used to calculate power with non-integer exponents.

Problem 3

4.1 Incarnation 1

The logic for computing Cos, Sin, Pi and other functions was written from the ground up in this incarnation. Code snippets are below.

4.1.1 Snippets

```python
class MathLib_WBI(ML):
    def __init__(self):
        self.PI = None

    def pow(self, number: float, power: int) -> float:
        if power == 0:
            return 1
        elif power % 2 == 0:
            temp = pow(number, power // 2)
            return temp * temp
        else:
            return number * pow(number, power - 1)

    def factorial(self, num: int) -> int:
        if num < 0:
            raise ValueError("Factorial can't be calculated for negative numbers")
        result = 1
        for i in range(1, num + 1):
            result *= i
        return result

    def get_pi(self):
        if self.PI is not None:
            return self.PI
        return calculate_pi()
```

Figure 4.1: Logic to compute Exponent, Factorial and Pi

```
class RootApproximation_WBI(RootApproximation):
    def __init__(self):
        self.num_terms = 100
        super().__init__()

    def get_roots(self, func, e) -> list:
        x0, x1 = 0, 99
        x2 = 0
        step = 1
        while True:
            if func(x0) == func(x1):
                break

            x2 = x0 - (x1 - x0) * func(x0) / (func(x1) - func(x0))
            print('Iteration-%d, x2 = %0.6f and f(x2) = %0.6f' % (step, x2, func(x2)))

            x0 = x1
            x1 = x2
            step += 1

            if step > self.num_terms:
                raise Exception("Not convergent")
                break

            if func(x2) - func(x1) > e:
                break

        return [x2]
```

Figure 4.2: Secant Method of root approximation

```
class TrignometryWBI(Trig):
    def __init__(self, precision=4):
        self.precision = precision
        self.m_lib = MathLib_WBI()

    def sin(self, rad):
        total_iterations = self.precision
        result = 0
        for i in range(1, total_iterations):
            result += (self.m_lib.pow(-1, i) * self.m_lib.pow(rad, 2 * i + 1)) / self.m_lib.factorial(2 * i + 1)

        return result

    def cos(self, rad: float) -> float:
        total_iterations = self.precision
        result = 0
        for i in range(0, total_iterations):
            result += (self.m_lib.pow(-1, i) * self.m_lib.pow(rad, 2 * i)) / self.m_lib.factorial(2 * i)
        return result
```

Figure 4.3: Logic to compute trigonometry functions

4.1.2 Output

```
SachinscBookAir:APP_Project sachinprakash$ /usr/local/bin/python3 /Users/sachinprakash/vscode-workspace/app-project/APP_Project/controller.py
Enter the radius: 4
Iteration-1, x2 = -0.000000 and f(x2) = -2.000000
Iteration-2, x2 = -0.000000 and f(x2) = -2.000000
Iteration-3, x2 = 2.000000 and f(x2) = 2.907937
Iteration-4, x2 = 0.815006 and f(x2) = -0.608035
Iteration-5, x2 = 1.019933 and f(x2) = -0.107666
Iteration-6, x2 = 1.064028 and f(x2) = 0.006469
Iteration-7, x2 = 1.061529 and f(x2) = -0.000059
Iteration-8, x2 = 1.061552 and f(x2) = -0.000000
Iteration-9, x2 = 1.061552 and f(x2) = 0.000000
Iteration-10, x2 = 1.061552 and f(x2) = 0.000000
Iteration-11, x2 = 1.061552 and f(x2) = 0.000000
Alpha with WBI is  2.4389047302915734
The length with WBI is :5.2476745813337065
Alpha with BI is  2.438908527759335
The length with BI is :5.246734435549717
SachinscBookAir:APP_Project sachinprakash$
```

Figure 4.4: Output for Incarnation 1

4.2 Incarnation 2

The python Math module and SciPy library were used to compute the various values. Code snippets are below.

4.2.1 Snippets

```python
class MathLib_BI(ML):
    def __init__(self) -> None:
        super().__init__(4)

    def pow(self, number: float, power: int) -> float:
        return math.pow(number, power)

    def factorial(self, number: int) -> int:
        return math.factorial(number)

    def get_pi(self) -> float:
        return math.pi
```

Figure 4.5: Math module functions

```python
class RootApproximation_BI(RootApproximation):
    def __init__(self) -> None:
        super().__init__()
        self.num_terms = 100

    def get_roots(self, func, e) -> list:
        root = fsolve(func, 0)
        return root
```

Figure 4.6: Root approximation using SciPy

```python
class TrignometryBI(Trig):
    def __init__(self, precision=4):
        super().__init__()
        self.precision = precision
        self.m_lib = MathLib_BI()

    def sin(self, rad: float) -> float:
        return math.sin(rad)

    def cos(self, rad: float) -> float:
        return math.cos(rad)
```

Figure 4.7: Math module Trigonometry functions

4.2.2 Output

```
dtd = '''<!DOCTYPE root [
    <!ELEMENT root (element*)>
    <!ELEMENT element (radius, output)>
    <!ELEMENT radius (#PCDATA)>
    <!ELEMENT output (#PCDATA)>
]>'''
```

Figure 4.8: DTD of xml file

```
<root>
  <element>
    <radius>5</radius>
    <output>6.559593226667133</output>
  </element>
  <element>
    <radius>10</radius>
    <output>13.119186453334265</output>
  </element>
  <element>
    <radius>15</radius>
    <output>19.678779680001398</output>
  </element>
</root>
```

Figure 4.9: XML file generated from user data

4.3 Quality Attributes

The following attributes apply to both incarnations.

4.3.1 Modifiability

Modifiability refers to the efforts that will be needed to modify the software product according to the specific requirements of customers[2]. The source code was built using the RDD paradigm. This inherently makes it amenable to updates in the future.

4.3.2 Readability

Adequate comments were provided to make the source code more meaningful to the reader. The variable and method names adhere to the guidelines provided under PEP 8.

4.3.3 Reusability

The source code was made reusable by following the Single Responsibility Principle (SRP) for every class and method. Abstract types have been used to hide the implementation details and improve the composability.

4.3.4 Understandability

The understandability of the source code was improved through

- The use of meaningful variable names
- Inclusion of method and class descriptions
- Minimizing the complexity within each method in a class

4.3.5 Testability

Due to the object-oriented nature of the project design, each class and method in the source code can be unit tested. Each class has its own set of tests which can swiftly determine if the actual value matches the expected.

4.3.6 Robustness

All possible exceptions have been handled throughout the program and the potential points of failure have been minimized. In case of an exceptional event, a meaningful error message will be returned to the end user.

4.3.7 Generality

The program has been made more general by avoiding the use of OS-specific path(s). E.g. When writing to an output file, a relative path name was used.

4.3.8 Usability

The usability of the program has been improved by

- Using meaningful prompt messages throughout the lifecycle
- Minimizing the potential points of failure
- Returning meaningful error messages in case of an exceptional event

4.4 Version Control

The project was executed on GitHub with each of the team members working on the agreed-upon tasks. The final build of the project was achieved by merging the efforts from all team members.

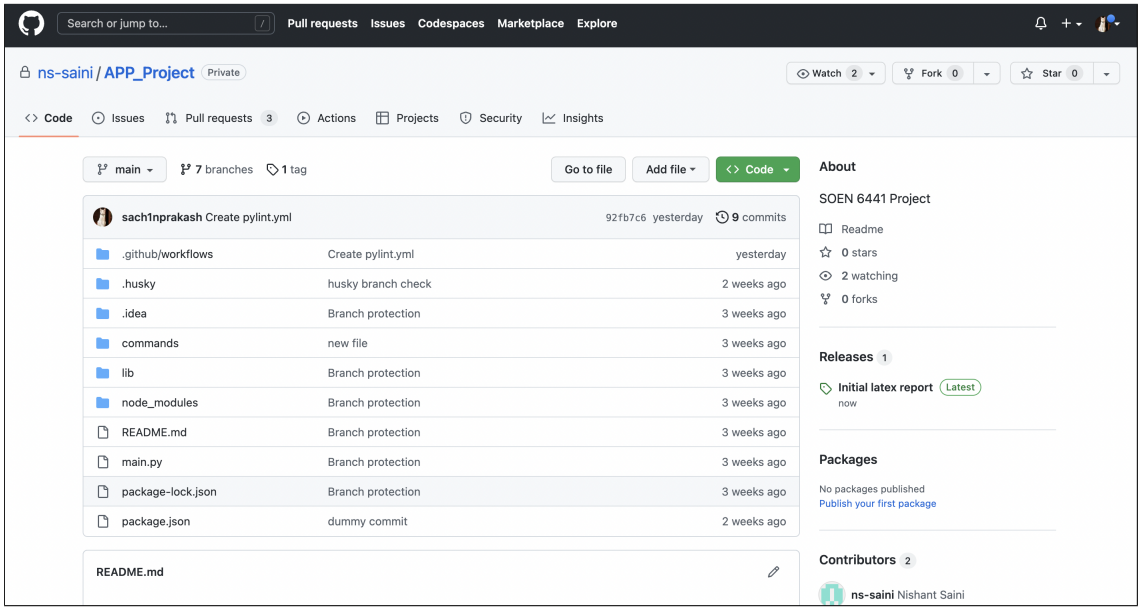

Figure 4.10: `https://github.com/ns-saini/APP_Project`

4.5 Debugger

The python debugger built into vs code was used during the development of the source code.

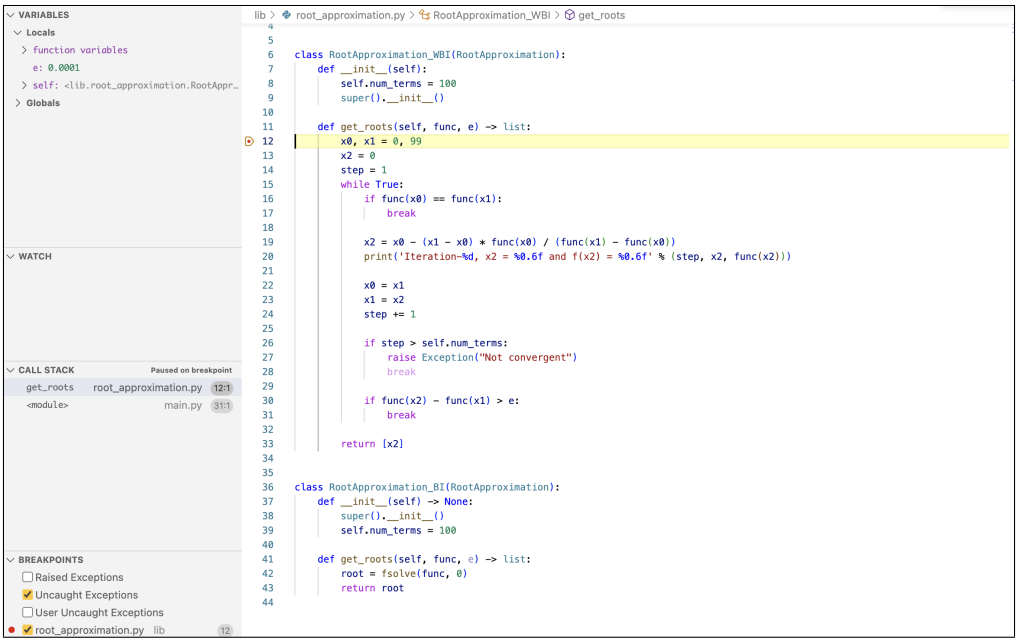

Figure 4.11: Debugger in action

4.6 Programming Style

Python Enhancement Proposals or PEPs are the documents that establish standards in the Python community. Most PEPs describe new features for Python or Python's standard library, but a few of them are more nebulous. PEP 8 is one of those[3]. The guidelines provided under PEP 8 were followed during the development of the source code. The PyLint linter was used to enforce this.

4.7 PyDoc

The PyDoc documentation system was used to document the source code

Figure 4.12: Documentation using Pydoc

Bibliography

[1] Giovanni Ferraro. *Geometrical quantities and series in Leibniz*, pages 25–44. Springer New York, New York, NY, 2008.

[2] Daniel Galin. *Software Quality Factors (Attributes)*, pages 23–44. 2018.

[3] Daniel Arbuckle. *PEP 8 — guidelines for Python code.* 2017.