SOEN 6441 - Winter 2023
Advance Programming Practices

"CHEERS"
Deliverable 1

Submitted to : Pankaj Kamthan

Submitted by : TEAM F

Leesha Shah
Vishwassingh Tomar
Jwalit Shah
Shrawan Malyala
Goutham Susarla
Anant Bir Singh

Contents

1 Problem 1

1.1 Introduction

A cold beverage in a glass is among many people's summertime survival strategies. The basic function of a coaster is to soak up condensation that drips along the glass, protecting the tabletop or other surface from damage. Tegestologists are also interested in the coasters. We call it as CHEERS in this project.

1.2 Problem Definition

Let there be two circular coasters of equal area (and negligible height). The purpose of CHEERS is to find how far the two coasters need to be moved on top of each other such that the area of the overlapping region is half the area of any one of the coasters. The length l of the segment X_1X_2 is given by the equation.

$$l = 2R(1-\cos(\alpha/2))$$

and α is given by the equation

$$\alpha-\sin(\alpha) = \pi/2$$

1.3 CRC model

A Class Responsibility Collaborator (CRC) model is a collection of standard index cards that have been divided into three sections, as

- A class represents a collection of similar objects
- a responsibility is something that a class knows or does
- a collaborator is another class that a class interacts with to fulfill its responsibilities

Below are the CRC cards which relate to CHEERS.

1.3.1 CHEERS CRC Model

Length	
• Calculating Length • Precision of the calculation is maintained upto 100 digits	• Sine • Cosine • Pi • Bisection

Pi	
• Calculating Pi. • Pi calculated using Dartboard algorithm • Running multiple times rounds of dartboard algorithm for better accuracy	

Bisection	
• Define the equation to solve for alpha • Calculate initial error between upper and lower bounds of range • Solve for alpha using Bisection method	• Sine • Pi

Factorial	
• Calculates factorial of n. • Checks the given number whether it ranges between 0 to n positive integers	

Sine	
• Calculates Sine of a parameter theta (given in radians) • Uses Taylor series expansion up to 100 values maintaining. • Calculates the answer precision of upto 100 decimal digits.	• Factorial

Cosine	
• Calculates Cosine of a parameter theta (given in radians) • Uses Taylor series expansion up to 100 values maintaining. • Calculates the answer precision of upto 100 decimal digits.	• Factorial

Figure 1: CRC model of CHEERS

1.3.2 Reason for collaboration

Length class: sine, cosine, pi, and bisection are used in the formula to calculate the length of the segment.Sine and Cosine class: Factorial function is required to calculate tailor series. Bisection class: sine and pi are required to solve for alpha. Pi and Factorial class: there are no collaborators

1.4 Object Oriented Principles

The code must be created in a way that allows for future updates to be made without taking a lot of time or money. Hence, we adhere to Object-Oriented design principles during the design and development phases.A highly cohesive and loosely coupled solution should always be our goal. The following are the primary guiding principles:

- Single Responsibility Principle
- Balanced Responsibility Principle
- DRY (Don't repeat yourself)

Each function that was used in Incarnation 1 and Incarnation 2 has a single duty in CHEERS. This would satisfy the SRP by improving cohesion and reducing coupling in each class.

The OOD principle is DRY (don't repeat yourself), which indicates that rather than duplicating code, we call the relevant functions when required.

In both iterations, when defining classes and method allocation, we adhere to the SRP and BRP concepts. For illustration, the main math class is only tasked with computing the solution. So, if manipulating the equation becomes necessary in the future, the developer only needs to edit this class.

2 Problem 2

2.1 Algorithms and Psuedo code

There are five primary functions which are made from scratch, which are used in CHEERS. They are as follows,

2.1.1 Algorithm for Sine

Psuedo code:-

//Design Sine function
$x < -\theta$
$m < -0$
$k < -0$
for $k < -0$ to $k < -10$

$$y = \sum_{k=0}^{10} \frac{(-1^k)(x^1 + 2k)}{(1 + 2k)!}$$

$m < -m + y$
return m

- **Reason for choosing this algorithm :**
  A function can be represented mathematically as an infinite sum of terms using the Taylor series expansion, where each term is a power of the independent variable multiplied by a coefficient. By truncating the infinite sum after a certain number of terms, the Taylor series expansion may be used to approximate a function. The Taylor series expansion is very helpful for sine and cosine functions because it enables us to determine the value of the function at any point using just a small number of expansion terms. In the series expansion by truncating these series after a finite number of terms, we can calculate an approximation of the value of sine or cosine for any given value of x. The more terms we use in the expansion, the more accurate the approximation will be.For sine and cosine functions, the Taylor series expansion is:

  $sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ...$

  $cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + ...$

2.1.2 Algorithm for Cosine

Psuedo code:-

//Design Cosine function
$x < -\theta$
$m < -0$
$k < -0$
for $k < -0$ to $k < -10$

$$y = \sum_{k=0}^{10} \frac{(-1^k)(x^2k)}{(2k)!}$$

$m < -m + y$
return m

- **Reason for choosing this algorithm :**
  Same as for sine (look the desctiption given in sine section)

2.1.3 Algorithm for Pi

Psuedo code:-

//Design Pi function
$score < -0$
$darts < -100000$
for i=0 to darts
  $x_coord < -randomnumberbetween - 1to1$
  $y_coord < -randomnumberbetween - 1to1$
  $if(square(x_coord) + square(y_coord)) <= 1$
    score¡- score + 1
return 4 * (score / darts)

- **Reason for choosing this algorithm :**
  The dartboard algorithm, also known as the Monte Carlo method, is a statistical technique that can be used to estimate the value of pi. The reason this algorithm works is that as you generate more and more random points, the ratio of the number of points that fall within the

circle to the total number of points generated will converge to the true ratio of the areas of the circle and the square. This means that the estimate of pi will become more and more accurate as you generate more points.

2.1.4 Algorithm for Factorial

Psuedo code :-

//Design Factorial function
$ans < -1$
for $i < -1$ to $i < -n$
  $ans < -ans * i$
return ans

- **Reason for choosing this algorithm :**
  Because they show the sum of all positive integers up to a certain number, factorials are frequently employed repeatedly in mathematical computations. In an illustration, 5! (read as "five factorial") equals 5 x 4 x 3 x 2 x 1, or 120.
  We frequently need to compute factorials iteratively when dealing with huge numbers that are difficult to calculate manually. To compute 100! for instance, we would need to multiply 100 x 99 x 98 x.. x 2 x 1, which is a very big number.

2.1.5 Algorithm for Bisection Method

Psuedo code:-

//Design Bisection function
$P < 0$, $N > 0$
Check the sign for two values if one is P and another is N
  $Midpoint < -\frac{P+N}{2}$
Check the sign of $Midpoint$ and N
If the $Midpoint$ ¿ 0
  $N = Midpoint$
Else
  $P = Midpoint$
Return $Midpoint$

- **Reason for choosing this algorithm :**
  The bisection method is a relatively simple algorithm that is easy to implement in code.Also The bisection method is a guaranteed convergent method, meaning that it will always converge to a root of the function within the given interval, as long as the function is continuous and changes sign within the interval. This makes it a reliable method for finding roots.it can be very accurate, particularly when used with a small tolerance value. It is therefore a good choice when high accuracy is required.

3 Problem 3

3.1 Quality of Code

There are serveral quality aspects by which the quality of the code can be improved.

- **Understandablity**
  The use of comments throughout the code makes it intelligible. Furthermore, self-explanatory names are used for the functions, photos, files, etc.
- **Readablity**
  To make the code legible by the user, the entire piece has been appropriately indented.
- **Traceablity**
  Code, documentation, tool sources, and test data have all been kept under version control for CHEERS. This task was completed using GitHub.
- **Complexity**
  Each incarnation contains atmost two files; one one serves as the library and the other as the main file. This method of separating the code makes things simpler. Also, the code was kept in manageable chunks, which further lessens complexity.
- **Testablity**
  Unit testing was done in the debugger after each tiny piece of code, which helps find mistakes and fix them quickly. Also, the developer can quickly fix problems before integrating the entire code.
- **Maintainability**
  In CHEERS, it is simple to change the code, fix errors, manage exceptions, make enhancements, etc. because of the code's high readability, high understandability, and minimal complexity. Maintainability is therefore obtained.
- **Reusability**
  Reusability is the usage of pre-existing resources in the software in some way. For example, sine, cosine, pi, and other functions developed for CHEERS can be utilised for any other problem.

- **Modifiabality**
  As whole code of CHEERS is written under the OOP concepts, we can modify anypart of the code very easily without impacting the other section of it.

- **General**
  We can input any value for the radius in both incarnations dynamically.The code is not specific to any static values.

- **Robust**
  It's handling all the possible exceptions And giving appropriate error messages on wrong input.So when any user tries to break our code We are handling it with try catch and exception Handling

3.2 Style Programming

A set of guidelines used when creating the source code for a computer programme is known as programming style. It is sometimes asserted that adhering to a specific programming style will make it easier for programmers to read and comprehend source code that adheres to the style and prevent errors from being introduced.[4]

- **PEP8**: A Python style manual that covers matters like how to name variables, how to utilise indentation in your code, how to organise your import statements, etc. Other Python developers will find it simpler to read and comprehend our code and to understand how their contributions should be formatted if our follow PEP8. Our code can be examined for PEP8 compliance using the Python library and PEP8 application.[6]

3.3 Experimental results

Below are the outputs(length of line segment X_1X_2) for different values of R which are generated in Scratch method and inbuilt method respectively.

Table 1: Values of length(l)

Radius	incarnation 1	incarnation 2
5	05.95884937227576	5.960245153200647
10	11.91769874455153	11.920490306401295
15	17.87654811682730	17.880735459601944
20	23.83539748910307	23.84098061280259
25	29.79424686137884	29.801225766003235
30	35.75309623365461	35.76147091920389
35	41.71194560593038	41.72171607240453
40	47.67079497820615	47.68196122560518
45	53.62964435048192	53.642206378805824
50	59.58849372275769	9.60245153200647

3.4 Debugger

A debugger or debugging tool is a computer program that is used to test and debug other programs[7]. In CHEERS the code has been debugged in command prompt.

3.4.1 Debugger for CHEERS

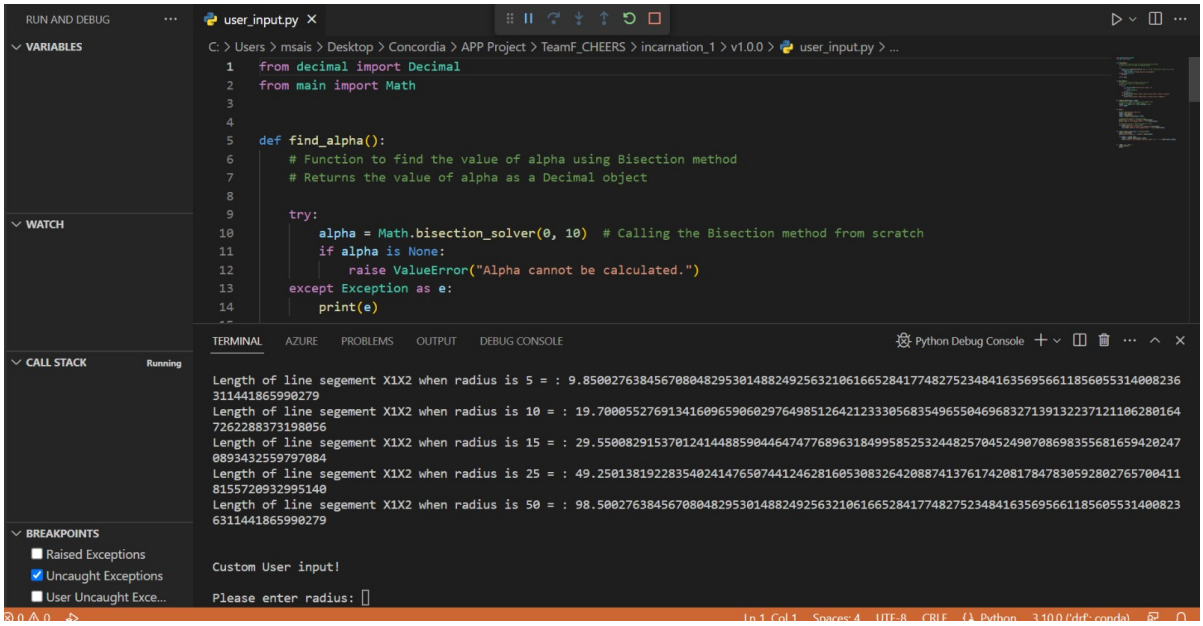

Figure 2: Debugger of CHEERS

3.5 Version Control

A system known as version control keeps track of changes made to a file or set of files over time so that you can later recall particular versions.

The version control for CHEERS can be found below,

- (`https://github.com/ShrawanSai/TeamF_CHEERS`)

3.6 Exception handling

Exception handling is a mechanism that separates code that detects and handles exceptional circumstances from the rest of your program. Note that an exceptional circumstance is not necessarily an error. When a function detects an exceptional situation, you represent this with an object.

Below are the Screenshots for exception handling in CHEERS. This makes it **Robust**.

- **Incarnation 2**

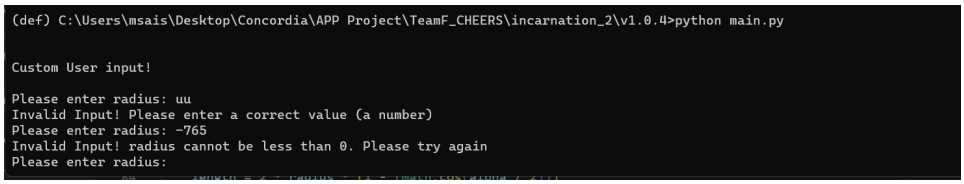

Figure 3: Error message for String as radius as well as negative radius

- **Incarnation 1**

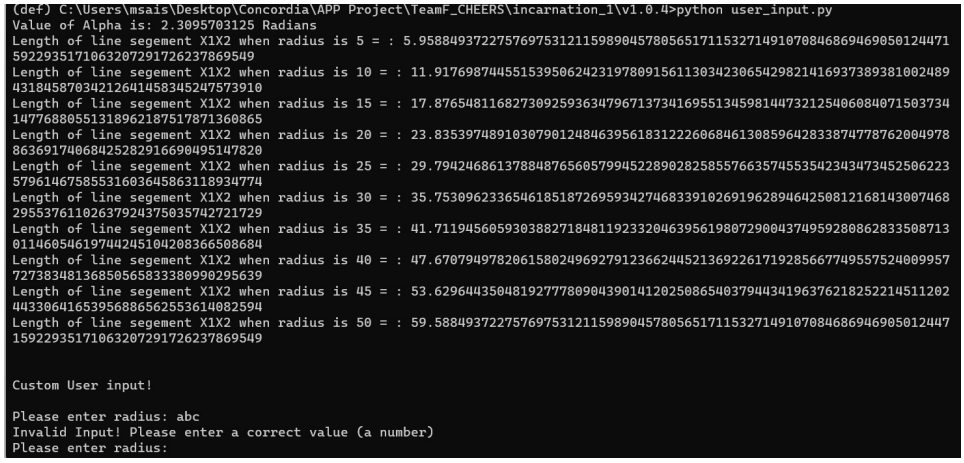

Figure 4: Error message for String as radius

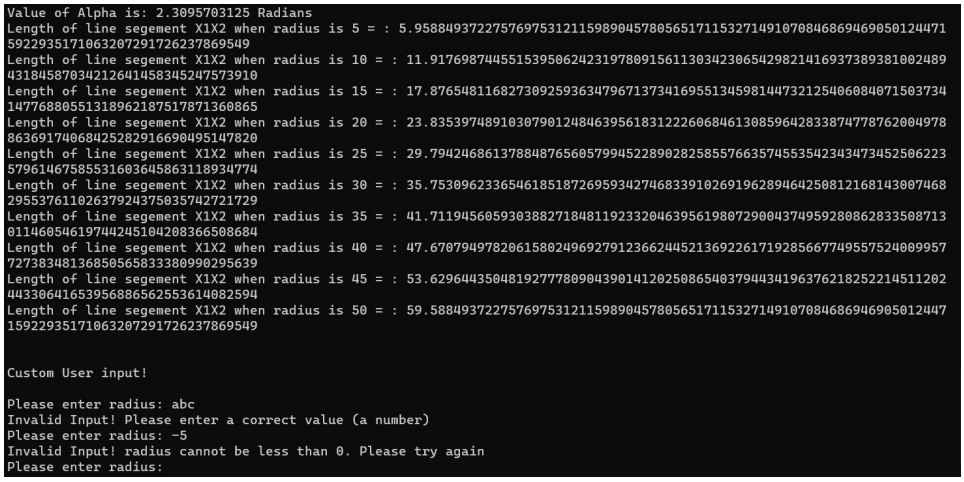

Figure 5: Error message for negative radius

4 Instruction for processing Source code

- For Incarnation 1 :

  1) Navigate to version folder
  2) Open cmd(Comand prompt)/Terminal.
  3) Type python *user_input.py*

- For Incarnation 2 :

  1) Navigate to version folder
  2) Open cmd(Comand prompt)/Terminal.
  3) Type python main.py

- To generate the PyDoc:

  1) Open cmd or command prompt (or terminal)
  2) Type 'python –m pydoc modulename.py',
  Modulename can be main, scratch, inbuilt or main-inbuilt

```
PS C:\Users\gouth\Documents\GitHub\TeamF_CHEERS\incarnation_2\v1.0.4> python -m pydoc main
Help on module main:

NAME
    main

FUNCTIONS
    SubElement(...)

    bisection_solver(lower_bound: float, upper_bound: float, error_tolerance=0.001)
        Solves for a root of a non-linear function, given root boundaries and acceptable error

    compute_length(radius, alpha)
        computes the length

    find_alpha()
        Finds the value of alpha using Bisection method

    get_radius()
        Gets a valid input from user

    main()

    write_to_xml(alpha, length)
        writes the output to XML file

DATA
    VERSION = '1.3.0'

FILE
    c:\users\gouth\documents\github\teamf_cheers\incarnation_2\v1.0.4\main.py
```

Figure 6: pydoc for incarnation2 main.py

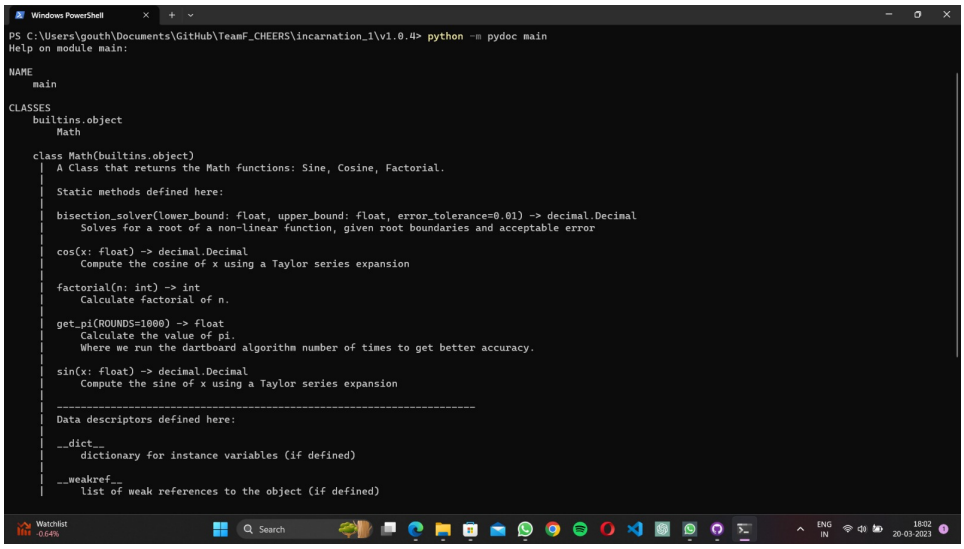

Figure 7: pydoc for incarnation1 main.py

```
PS C:\Users\gouth\Documents\GitHub\TeamF_CHEERS\incarnation_1\v1.0.4> python -m pydoc user_input
Help on module user_input:

NAME
    user_input

FUNCTIONS
    compute_length(radius, alpha)

    find_alpha()

    get_radius()

    main()

    sample_outputs(sample_radii=[5, 10, 15, 20, 25, 30, 35, 40, 45, 50])

FILE
    c:\users\gouth\documents\github\teamf_cheers\incarnation_1\v1.0.4\user_input.py
```

Figure 8: pydoc for incarnation1 user_ input.py

5 Software Tools Used

- PyCharm
- VSCode
- Github
- lucidchart
- Autopep8
- Msais
- Spyder

6 Team Contribution

Team Member	Contributions
Jwalit	• Participated in brainstorming sessions. • Participated in the review meetings. • Implement pi function from scratch • Participated in Coding.
Shrawan	• Participated in brainstorming sessions. • Participated in the review meetings. • Implement sine and cosine function from scratch • Participated in Coding.

Leesha Shah	• Participated in brainstorming sessions. • Participated in the review meetings. • Participated in creating CRC Cards. • Participated in Coding. • Participated in Documentation in Latex.
Vishwassingh Tomar	• Participated in brainstorming sessions. • Participated in the review meetings. • Participated in creating CRC Cards. • Participated in Coding. • Participated in Documentation in Latex.
Goutham	• Participated in brainstorming sessions. • Participated in the review meetings. • Participated in Coding. • Implement Factorial function
Anant Bir Singh	• Participated in brainstorming sessions. • Participated in the review meetings. • Participated in Coding. • Implement Bisection Method

7 References

[1] https://en.wikipedia.org/wiki/Beverage_coaster
[2] http://surl.li/fqjbs
[3] http://www.softpanorama.org/SE/programming_style.shtml
[4] http://www.softpanorama.org/SE/programming_style.shtml
[5] https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
[6] https://alistairwalsh.github.io/python-novice-gapminder/20-style/
[7] https://en.wikipedia.org/wiki/Debugger
[8] http://www.informit.com/articles/article.aspx?p=2223710
[10] https://users.encs.concordia.ca/ kamthan/courses/soen-6441/project_description.pdf
[11] https://thebojan.ninja/2015/04/08/high-cohesion-loose-coupling/ [12] https://en.wikipedia.org/wiki/Exception_handling