CSE 231 **Fall 2022**

Computer Project #4

This assignment focuses on the design, implementation and testing of a Python program to calculate and display the values of selected mathematical functions (see below).

It is worth 40 points (4% of course grade) and must be completed no later than **11:59 PM on Monday, October 10. After the due date, your score will be deducted by 2pt for every 2 hours late.**

Assignment Overview

In this assignment, you will practice with functions.

Assignment Background

Many common mathematical functions can be expressed as the sum of a series of terms. In some cases (such as n!, the factorial of n), the function can be expressed as a finite series of terms.

In other cases, the expression contains an infinite number of terms. For example, the tangent function can be expressed as the infinite series:

$$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n(1-4^n)}{(2n)!}x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots \qquad \text{for } |x| < \frac{\pi}{2}$$

(GIF courtesy of Wikipedia)

Obviously, we cannot generate an infinitely long sum. We will handle that by defining a very small epsilon:

```
EPSILON = 0.0000001
```

so if the absolute value of the next term to be added to the sum is *smaller than* `EPSILON`, we will *not* add in the term and stop the calculation. We provide (and require) the `EPSILON` constant so all students calculate the same result. Note that the value we chose is arbitrary. Also, note the italicized "smaller than" and "not" which will be important in your code.

Assignment Specifications

You will develop a Python program which allows the user to select from a menu of options and which performs the requested calculations. You must use at least the four functions specified below. You are encouraged to use more functions of your own design. Global variables are not allowed. That is, any variable names used within any function must be parameters or be created in the function. You are not allowed to use advanced data structures such as list, dictionaries, sets, classes, etc. However, you are allowed to read ahead and use try-except.

`factorial(N)` → `int`:

a. This function takes one argument `N` as a string and returns `N!` (the factorial of `N`), if `N` is not a non-negative `int`, return `None` (hint: the string method `isdigit()` may be useful). Your factorial calculation must be based on the following formula (you are allowed to calculate the values in reverse order, but you are not allowed to simply call `math.factorial(N)` or similar):

$$n! = n \times (n-1) \times \cdots \times 1$$

Note: by definition `0! = 1`

b. Parameters: `str`
c. Returns : `int` or `None`
d. The function displays nothing.

`e()` → `float`:

a. This function takes no arguments and returns an approximate value of e. Your calculation must be based on the following formula, i.e. you cannot use `math.e` or similar (hint: the previous factorial function will be useful or you may use `math.factorial()`):

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$

Note the discussion above on the use of the `EPSILON` that we provide to handle when to stop adding on new terms to the sum. The number should be **rounded** to 10 decimal digits after the decimal point (we ask you to round to achieve more consistent results which allows for more consistent testing). Do not round the terms, always round at the end.

b. Parameters: no parameters
c. Returns : `float`
d. The function displays nothing.

`pi()` → `float`:

a. This function accepts no input and returns the approximate of Pi (π). Your calculation must be based on the following formula, i.e. you cannot use `math.pi` or similar. The number is calculated as:

$$\pi = 4 * \left(\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \right)$$

Note the discussion above on the use of the `EPSILON` that we provide to handle when to stop adding on new terms to the sum. Note that the `EPSILON` is for the terms in the summation (before you multiply by 4). The number should be **rounded** to 10 decimal digits after the decimal point (we ask you to round to achieve more consistent results which allows for more consistent testing). Do not round the terms, always round at the end.

b. Parameters: no parameters
c. Returns : `float`
d. The function displays nothing.

`sinh(x)→ float:`

a. This function accepts as input a numeric value X (measured in radians) as a string and returns the approximate value of the hyperbolic sine of X.). If `x` cannot be converted to a `float`, return `None` (hint: see item #9 of Notes below). The number is calculated as:

$$\sinh x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots \quad \text{for all } x$$

Note the discussion above on the use of the `EPSILON` that we provide to handle when to stop adding on new terms to the sum. The number should be **rounded** to 10 decimal digits after the decimal point (we ask you to round to achieve more consistent results which allows for more consistent testing). Do not round the terms, always round at the end.

b. Parameters: `str`
c. Returns : `float` or `None`
d. The function displays nothing.

`main():`

a. This function is used to interact with the user. It takes no input and returns nothing. Call the functions from here. The program should prompt the user to choose between 6 options (capitalization does not matter) until the user enters `'X' or 'x'`:

```
'F': Factorial of N.
'E': Approximate value of e.
'P': Approximate value of Pi.
'S': Approximate value of the sinh of X.
'M': Display the menu of options.
'X': Exit.
```

b. The program will repeatedly prompt the user to enter a letter (upper or lower case) which corresponds to one of the supported options. For example, the user will enter the letter F (or the letter f) to select Option F (display the value of the factorial of N).
c. The program will display the menu of options once when execution begins, whenever the user selects Option `M`, and whenever the user selects an invalid menu option.
d. If the user enters option `F`, the program will display the message `"Factorial"` then ask the user to enter a numeric non-negative value N. If N is a non-negative integer, the program will calculate and display:
   - the calculated value of the factorial of N (using your function),
   - the math module value, i.e., `math.factorial(N)`
   - the absolute value of the difference between these two values.

   Otherwise, the program will display an appropriate message (see `strings.txt`).
e. If the user enters option `E`, the program will display the message `"e"` then calculate and display the approximate value of `e`. It will then display the corresponding value from the Python math module, i.e. `math.e`, as well as the absolute value of the difference between the calculated value (which was rounded in the function) and the rounded math module

value . The program should display 10 decimal digits after the decimal point for both values and the difference. That is, find the difference between the rounded values, but do not round the difference. Use the appropriate string formatting.

f. If the user enters option `P`, the program will display the message `"pi"` and calculate and display the approximate value of `Pi`. It will then display the corresponding value from the Python math module, i.e. `math.pi`, as well as the absolute value of the difference between the calculated value (which was rounded in the function) and the rounded math module value . The program should display 10 decimal digits after the decimal point for both values and the difference. That is, find the difference between the rounded values, but do not round the difference. Use the appropriate string formatting.
g. If the user enters option `S`, the program will print the message `"sinh"` then ask the user to enter a numeric value X (measured in radians). If X is a numeric value (integer or a float), the program will calculate and display the approximate value of the `sinh` of X. It will then display the corresponding value from the Python math module,i.e. `math.sinh` as well as the absolute value of the difference between the calculated value (which was rounded in the function) and the rounded math module value . The program should display 10 decimal digits after the decimal point for both values and the difference. That is, find the difference between the rounded values, but do not round the difference. Use the appropriate string formatting. If the value entered for X cannot be converted to a float (see note #9), the program will display an appropriate message (see `strings.txt`).
h. If the user enters option `M`, the program will print the menu.
i. If the user enters an invalid option, the program will display an appropriate message (see `strings.txt`) followed by the invalid option in uppercase. Then, it will display the menu.
j. If the user quits, the program should display a goodbye message.

Assignment Notes and Hints

1. The coding standard for CSE 231 is posted on the course website:

   http://www.cse.msu.edu/~cse231/General/coding.standard.html

Items 1-8 of the Coding Standard will be enforced for this project.

2. The program will produce reasonable and readable output, with appropriate labels for all values displayed.

3. You may assume that the user enters a string representing a numeric value when prompted for a number. That is, a user will not enter letters where digits are expected. However, you should not assume that the value is valid in that particular context. For example, the user might enter 45.7 for Option F; that numeric value is not a positive integer.

4. Be sure to prompt the user for the inputs in the correct order. Also, your program cannot prompt the user for any supplementary inputs.

5. Several items from the `math` module might be useful for this project (remember to `import math`):

```
math.pi
math.e
math.fabs()
math.factorial()
math.sinh()
```

Note that the difference between `abs()` and `math.fabs()` is that the latter always returns a `float`.

6. We provide a `proj04.py` program for you to start with.

7. You are not allowed to use advanced data structures such as list, dictionaries, classes, etc. However, you are allowed to read ahead and use try-except.

8. If you "hard code" answers, you will receive a grade of zero for the whole project. An example of hard coding is to simply print the approximate value of e rather than calculating it and then printing the calculated average.

9. There are multiple ways to check whether a string contains a float. One is to use the `float_check` function you developed in Lab 4. That function doesn't consider a float in the format such as `4e2`, but that is fine for this project—we will assume that the user will not use that format for input. However, the best way to check for a float is to use the `try-except` from Section 6.6.2 of the text. You `try` to convert a value to a `float` and if a `ValueError` exception is raised you handle it, e.g. `return None`. Feel free to read ahead and use the `try-except` statement in this project.

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

- Cycle through the following steps to incrementally develop your program:
  - Edit your program to add new capabilities.
  - Run the program and fix any errors.
  - Use the **Coding Rooms** system to submit the current version of your program.

- Be sure to use the **Coding Rooms** system to submit the final version of your program.

- Be sure to log out when you leave the room, if you're working in a public lab.

The last version of your solution is the program which will be graded by your TA.

*You should use the **Coding Rooms** system to back up your partial solutions, especially if you are working close to the project deadline. That is the easiest way to ensure that you won't lose significant portions of your work if your machine fails or there are other last-minute problems.*

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj04.py` – the source code for your Python program

Be sure to use the specified file.

Grading Rubric

```
Computer Project #04                                Scoring Summary
General Requirements:
  ( 5 pts) Coding Standard 1-8
     (descriptive comments, function headers, mnemonic identifiers,
     format, etc...)

Implementation:
 ( 6 pts)  factorial() function

 ( 6 pts)  e() function

 ( 6 pts)  pi() function

 ( 6 pts)  sinh() function

 ( 2 pts)  Test 1

 ( 6 pts)  Test 2

 ( 3 pts)  Test 3


Note: hard coding an answer earns zero points for the whole project
-10 points for not using main()
```

Test 1:

```
Options below:
    ‘F’: Factorial of N.
    ‘E’: Approximate value of e.
    ‘P’: Approximate value of Pi.
    ‘S’: Approximate value of the sinh of X.
    ‘M’: Display the menu of options.
    ‘X’: Exit.

Choose an option: X

Thank you for playing.
```

Test 2:

```
Options below:
    ‘F’: Factorial of N.
    ‘E’: Approximate value of e.
    ‘P’: Approximate value of Pi.
    ‘S’: Approximate value of the sinh of X.
    ‘M’: Display the menu of options.
    ‘X’: Exit.

Choose an option: F

Factorial

Input non-negative integer N: 3

Calculated: 6
Math: 6
Diff: 0


Choose an option: f

Factorial

Input non-negative integer N: 6

Calculated: 720
Math: 720
Diff: 0


Choose an option: E

e
Calculated: 2.7182818011
Math: 2.7182818285
Diff: 0.0000000274


Choose an option: p
```

```
pi
Calculated: 3.1415924536
Math: 3.1415926536
Diff: 0.0000002000


Choose an option: S

sinh

X in radians: 1

Calculated: 1.1752011684
Math: 1.1752011936
Diff: 0.0000000252


Choose an option: s

sinh

X in radians: 2.1

Calculated: 4.0218566892
Math: 4.0218567422
Diff: 0.0000000530


Choose an option: m

Options below:
    ‘F’: Factorial of N.
    ‘E’: Approximate value of e.
    ‘P’: Approximate value of Pi.
    ‘S’: Approximate value of the sinh of X.
    ‘M’: Display the menu of options.
    ‘X’: Exit.


Choose an option: a

Invalid option: A

Options below:
    ‘F’: Factorial of N.
    ‘E’: Approximate value of e.
    ‘P’: Approximate value of Pi.
    ‘S’: Approximate value of the sinh of X.
    ‘M’: Display the menu of options.
    ‘X’: Exit.


Choose an option: x

Thank you for playing.
```

Test 3:

```
Options below:
    ‘F’: Factorial of N.
    ‘E’: Approximate value of e.
    ‘P’: Approximate value of Pi.
    ‘S’: Approximate value of the sinh of X.
    ‘M’: Display the menu of options.
    ‘X’: Exit.


Choose an option: F

Factorial

Input non-negative integer N: -2

Invalid N.


Choose an option: f

Factorial

Input non-negative integer N: a

Invalid N.


Choose an option: s

sinh

X in radians: 2a

Invalid X.


Choose an option: x

Thank you for playing.
```