**Logarithm Function Notes**

The logFunction module is designed for evaluating logarithm of given base number and argument number. LogFunction class have 3 attributes: base, argument, result. Base represent the base number. Argument represent argument number and result is the final calculated result. \_\_init\_\_ is the constructor of this class which used to initialize those three attributes. The set functions of base number and argument will set the numbers by given input respectively and convert the number to float if input entered is fraction. The fraction will also be check if there is an exception of division by zero. The set\_values method is used to take user input as base number and argument as well as validate the number entered. Three exceptions are used to in this method to handle the exceptions to keep the base number and argument stay in their domain:

1. if base number is equal to 1.
2. If base number is less than 1.
3. If the argument entered is less than 0.

For all three cases, there will be an error message pop up for each of them and ask the user to re-enter the number. The ln\_helper method is responsible to calculate the natural logarithm of given number and the cal\_log method is for calculate the final answer by taking the division of the result of two natural logarithm. Those two methods are created based on the Taylor Series and the properties of logarithm. For Taylor Series, the domain of x needs to be |x-1|1 and x 0. However, the other situations, like x > 1 need to be considered as well. According to the series section of natural logarithm on Wikipedia, I separated the domain of x to three sections: x 1, x < 0.5 and 0.5 x 1. Then we can get the answer of logb(x) through ln(x)/ln(b). The detail of the algorithm implemented in pseudocode. The get\_result method is used to get the final result of this function.

**pseudocode**

function ln\_helper(x):

If x is greater or equal to 1:

return ln\_helper(x/2) – ln\_helper(1/2)

else if x is less than 0.5:

return ln\_helper(2\*x) + ln\_helper(1/2)

n 🡨 10000

sum 🡨 0

for i in the range between i and n:

sum 🡨 sum + (-1) ^ (i+1) \* ((x-1) ^ i) / i

return sum

function cal\_log():

if argument is greater than 0 and base number is greater than 0 but not equal to 1:

result 🡨 ln\_helper(argument) / ln\_helper(base number)

return result