Lab 07 - Taylor Series

In this lab, you will create a Python function that calculates the *Taylor Series* of two mathematical functions. A Taylor Series is an infinite series of mathematical terms that when summed together approximate a mathematical function. A Taylor Series can be used to approximate e^x, *sine*, and *cosine*.

An example of a Taylor Series is below:

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \ ...$$

Pre-Lab

Before starting this lab, read through this entire document. Review to programming concepts:

- For Loops
- While Loops

Practice using *for loops* and *while loops* in numerical method calculations.

Lab

Create a new Python script called **taylor_series.py** which will contain a user-defined function called `taylor()`. The `taylor()` function will have the function name, inputs, outputs and doc string information as shown.

```
def taylor(func, x, terms):
"""
Performs the Taylor series expansion for either the sine of x or for Euler's number raised to x.
The taylor() function returns the taylor series approximation of sin(x) or e^(x) with an optional n

 Input:
     - func: A string describing which function to to calculate the Taylor
             series expansion of.  ('sin' for sine(x), 'exp' for e^x)
     - x: Input argument that the function will operate on.  Must be a float or an integer between
     - terms: number of terms in the Taylor Series. Must be an int greater than zero up to and incl
              Defaults to 10. Has to be an integer between 1 and 50.

 Output:
     - series_sum: equals the sum of all of the terms in the Taylor Series up to the number of term
                   Will be a float.

"""
```

Above the function definition, include a block comment (using #) with the lab title, your name, course, quarter, and date.

Underneath the comment block, create the `taylor()` function that fulfills the description below.

Your function should verify that all three of its input variables have values as specified in the description, otherwise, use an `if: raise Exception()` structure to tell the user which variable is invalid and give the constraints.

For example, if a user tries to call the function with `x` set equal to -60, the following exception is triggered.

```
if x > 50:
    raise Exception('x should not exceed 50')
```

The Taylor Series expansion for our two functions e^x and $\sin(x)$ are below:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ...$$

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - ...$$

Note that as n increases, each successive term gets smaller in absolute value. Knowing this, we can assert that if, for instance, the 10^{th} term in a Taylor series is 0.005, then 10 iterations of the function will bring the sum of terms 1 through 10 to be within +/- 0.005 of the actual answer. In other words, 10 iterations yields an answer that has an error with absolute value of 0.005 or less.

Your function should use a **while loop** or **for loop** to compute and sum each of these terms. After the last term in the Taylor Series is calculated and added, use an **if break** statement to exit the **loop**. Otherwise, the **loop** should keep running until it has been run n times.

Note that your `taylor()` function will not compute the exact value of $\sin(x)$ or e^x, the Taylor Series to a finite number of terms is an estimate compared to the actual value of $\sin(x)$ or e^x.

Below are a couple of tips to help you complete the lab:

- Start working with your .py-file as a regular script (not a function) with hard coded input values. This way you can easily run and re-run it for testing.
- Get either the e^x or the $\sin(x)$ capability working before adding the other capability.
- Once both of these capabilities are working, complete the input validation for all of the input variables.
- Test your code often!
- Wait until you have verified your function works, along with its input validation, before changing it into a function.
- Now call your function `taylor()` from within the **taylor_series.py** file . It is easier to change values and re-run this way, versus importing the function and calling `>>>taylor('sin',4,10)` from the Python REPL.
- Verify your results. Compare the results of you Taylor Series function to Python's **math** module functions `math.exp()` and `math.sin()`.

Deliverables

After your taylor() function is complete in a .py file, run the following calculations with your taylor() function in a Jupyter notebook called `lab7.ipynb`. Make sure to import your `taylor()` function from `taylor_series.py` at the top of your notebook. Within your `lab7.ipynb` notebook, run the two calculations below using your `taylor()` function.

sin(30*) (find the sine of 30 degrees)

e^2 (find e raised to the 2nd power)

You should run enough terms so that there is less than floating point error between your `taylor()` function and `math.sin()` or `math.exp()`

Upload your `taylor_series.py` file and your `lab7.ipynb` in D2L.

By P. Kazarinoff, Portland Community College, 2021