COL100 Assignment 3

Due date: Wednesday, 26^{th} January 2021 23:59

Instructions: All programs are to be written in Python. Functions must be documented with proper comments explaining

- the purpose of the function
- the types and meanings of the input arguments
- any conditions on the input arguments (input specifications)
- the type and meaning of the return values
- conditions on the outputs (output specifications)

For example:

```
def mean(a, b):
    # The mean of two numbers.
    #
    # INPUT parameters
    # a: float -- the first number
    # b: float -- the second number
    #
    # Returns: float -- the mean of a and b
    # OUTPUT (a+b)/2

    return (a + b)/2
```

Within the program you MUST document the following:

- Loop invariants
- Highlight (via a comment) the decreasing measure in each iteration which guarantees termination
- Time complexity of the function. Use appropriate notation in your arguments/explanations. You will get 0 for not writing appropriate explanations. You will be asked to explain the same during the demo.

Write all your code in a single Python file named `<entryNumber>_assignment_3.py`.

Other specific instructions: Use the Python math library to get the value of π and functions like $\sin(x)$, $\cos(x)$, e^x (for Question 2). You cannot use in-built functions factorial or exponentiation. Define your own functions for these purposes and document them properly.

Part 1

Question 1

In this assignment we will work with iteration (loops) and write their loop invariants. We already learnt in class how to calculate the value of $\sin(x)$ using the Taylor series expansion. We will apply the same ideas to calculate some more useful functions.

Implement the following functions using Taylor's Series expansions to find the value of the function for a given x.

1. $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

   Write a function `expn(x, n)` where `n` is the number of terms up to which the series is to be expanded and `x` is the point at which the function is to be evaluated.

   Example: As $e^0 = 1$ and let us suppose the number of terms is 2, the function should behave as

```
>>> expn(0, 2)
1
```

2. $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots$

   Example: As $cos(\frac{\pi}{2}) = 0$ and let the number of terms be 3. You should get

   ```
   >>> cosine(pi/2, 3)
   0.0199689
   ```

3. $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$ for $|x| < 1$

   Example:

   ```
   >>> inverse(0.5, 3)
   1.75
   ```

4. $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} \ldots$

   Example:

   ```
   >>> natural_log(0.5, 2)
   0.375
   ```

5. $\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \ldots$

   Example:

   ```
   >>> tan_inv(1, 3)
   0.86666667
   ```

Instructions:

1. The functions are to be implemented iteratively (using loops).
2. Loop invariants are to be provided for each function using comments.

3. Provide the time complexity of your functions.
4. Your answer should be correct upto 6 decimal digits

Question 2

Bisection method: The Bisection method is one of the simplest and most reliable of iterative methods for finding the solutions of nonlinear equations. This method, also known as binary chopping or half-interval method, relies on the fact that if $f(x)$ is real and continuous in the interval $a < x < b$, and $f(a)$ and $f(b)$ are of opposite signs, that is, $f(a) \cdot f(b) < 0$, then there is at least one real root between a and b (there may be more than one root in the interval). Let $x_1 = a$ and $x_2 = b$. Let us also define another point x_0, which is the midpoint of a and b, that is,

$$x_0 = \frac{x_1 + x_2}{2}$$

Now, the following three conditions arise:

1. If $f(x_0) = 0$, we have a root at x_0.
2. If $f(x_0) \cdot f(x_1) < 0$, then there is a root between x_0 and x_1.
3. If $f(x_0) \cdot f(x_2) < 0$ then there is a root between x_0 and x_2.

It follows that by testing the sign of the function at midpoint, we can deduce that which part of the interval contains the root.

Note: While some books suggest finding if two numbers are of opposite sign by multiplying them and checking if the product is negative, this idea (while correct mathematically) is not a good idea computationally (where the approximation may not precisely reflect the true mathematical value).

Write a program to find a root of the following functions using Bisection Method. Take the last 2 digits of your entry number and find its remainder when divided by 4. Depending on the result, you get one of the following functions:

$0 \rightarrow f(x) = \cos(x) + e^{-x}$

$1 \rightarrow f(x) = e^{-x} - \sin(x)$

2 $\rightarrow f(x) = \sin(x) + \frac{1}{1-x}$

3 $\rightarrow f(x) = \frac{1}{1-x} - \cos(x)$

For example, if your entry number is 2021CS12345, then your function is $f(x) = e^{-x} - \sin(x)$

Define a function named `bisect(a,b,eps)` to find the root of your assigned function via iterative bisection. The functions are evaluated in the interval $[a, b]$, and should give a result with tolerance `eps`. The output of your function should be a 3-tuple (`Found, Value, IterList`).

- `Found` (type `bool`): denotes if there is a root of the function in the given interval
- `Value` (type `float`): the root of the function in the given interval, if it exists
- `IterList` (type `float list`): denotes the series of approximate results as the iteration converges towards the root (or when you exit the loop).

Constraints:

- $f(a) \cdot f(b) < 0$
- `Found == True` implies $|f(\texttt{Value})| \leq$ `eps`

Example: Consider the case when the remainder is 0, then $f(x) = \cos(x) + e^{-x}$. Then the function should behave as follows:

```
>>> bisect(4, 5, 1e-2)
(True, 4.703125, [4.5, 4.75, 4.625, 4.6875, 4.71875, 4.703125])
```

Analysis: Write down the time complexity of the bisection method using comments in the code. Assume that in your algorithm, the width of the bisected interval is always greater than or equal to a threshold tolerance: `tol`. Use this parameter in your analysis. Do NOT use this parameter in your code.

Note: Copying from the internet or from each other will lead to a straight 0 in the entire assignment and lead to a grade down or an F grade.

Part 2

Question 3

For the Taylor Series given in Part 1 Question 1, write functions that return the coefficient of x^n. Your function names should be similar to Part 1 question 1. For eg: expn_coeff, inverse_coeff etc

1. **Example:** if we need to find coefficient of x^3 in e^x. Our function would be as follows:

```
>>> expn_coeff(3)
0.16666666666
```

2. **Remember:** There might be cases when the coefficient is zero, then you must return zero. For example: if we need to find the coefficient of x^3 in the expansion of cos(x), this should be the behaviour of the function:

```
>>> cosine_coeff(3)
0.0
```

3. Functions to be implemented are

```
expn_coeff(n):
cosine_coeff(n):
inverse_coeff(n):
natural_log_coeff(n):
tan_inv_coeff(n):
```

Question 4

For all the Taylor series expansions in Part 1 Question 1, write functions that return the sum of series up to to a specific power of x, i.e., $\sum_{i=0}^{n} c_i x^i$.

You need to use the functions defined in Question 3 for this part.

1. **Example:** If we want to calculate e^3 up to the term x^3 in its expansion we would get

```
>>> expn_t(3,3)
13.0
```

2. **Watch Closely:** This question is different from that in Part 1. For example: in case of expansion of $\cos(\frac{\pi}{2})$ up to x^5, we get 1 - $\frac{(\frac{\pi}{2})^2}{2!}$ + $\frac{(\frac{\pi}{2})^4}{4!}$

```
>>> cosine_t(pi/2,5)
0.199689
```

3. Functions to be implemented are

```
expn_t(x,n):
cosine_t(x,n):
inverse_t(x,n):
natural_log_t(x,n):
tan_inv_t(x,n):
```

Question 5

Suppose we have two functions $f(x)$ and $g(x)$ represented as Taylor series, $f(x) = a(0) + a(1)x + a(2)x^2 + ...$ and $g(x) = b(0) + b(1)x + b(2)x^2 + ...$

Take the last 2 digits of your entry number and find its remainder when divided by 4. Depending on the result, you get one of the following function combinations:

- $0 \rightarrow f(x) = \cos(x)$ and $g(x) = \ln(1 + x)$
- $1 \rightarrow f(x) = e^{-x}$ and $g(x) = \frac{1}{1-x}$
- $2 \rightarrow f(x) = \frac{1}{1-x}$ and $g(x) = e^{-x}$
- $3 \rightarrow f(x) = \ln(1 + x)$ and $g(x) = \cos(x)$

Write the following functions:

(a) `add_coeff(n)` returns the coefficient of x^n in the Taylor series of $f(x)+g(x)$.

(b) `mul_coeff(n)` returns the coefficient of x^n in the Taylor series of $f(x) \cdot g(x)$. *First try this out by hand, then write the code after having figured out the pattern.*

(c) `diff_coeff(n)` returns the coefficient of x^n in the Taylor series of $\frac{d \cdot f(x)}{dx}$.

You need to use the functions defined in Question 3 for this part.

Question 6

Define a function `limit_diff(x,eps)` to evaluate the first order differential of the following functions using first principles.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Take the last 2 digits of your entry number and find its remainder when divided by 4. Depending on the result, you get one of the following functions: Evaluate the corresponding one of the following functions:

$0 \rightarrow f(x) = \frac{\sin(x)}{\cos(x)}$

$1 \rightarrow f(x) = e^{-x} \cdot \sin(x)$

$2 \rightarrow f(x) = \frac{\cos(x)}{1-x}$

$3 \rightarrow f(x) = \ln(1+x) \cdot \cos(x)$

Constraints:

- Use functions definition from Question 1.
- Consider $h = \epsilon$.
- Fix the value of $n = 20$.

Related Links

Moodle course: Here

Queries / FAQs: Here

Submission instructions: Here