**Module 2 – Critical Thinking Project**

**Salary Double**

Overview:

To build a python program that uses loop and recursion to build a salary double program.

Principles:

* Create the program using Loop.
* Create the program using Recursion.
* Penny is 1/100 i.e. 1th part of 100 units e.g $1=100Cents

Discussions:

Loop

def sal\_loop(days):

    salary = 0.01  # 1 penny

    for day in range(1, days + 1):

        print(f"Day {day}: ${salary:.2f}")

        salary \*= 2  # sal double

sal\_loop(30)

Recursion

def sal\_recur(day, salary=0.01):

    if day > 0:

        print(f"Day {day}: ${salary:.2f}")

        sal\_recur(day - 1, salary \* 2)

sal\_recur(1000)

What problems relating to number storage are you likely to encounter if you implement your solutions on an actual machine?

* Call stack overflow – with recursion as and when the numbers of calls will accumulate the call stack will overflow resulting in the recursion error

e.g if I call the second program 1000 times it will through an error

A screen shot of a computer

Description automatically generated

* Precision Limitations- Most computers use finite precision arithmetic for floating-point numbers, commonly represented in IEEE 754 format. This means there's a limit to the precision of real numbers that can be stored. When repeatedly doubling a value, as in the salary example, the numbers can become extremely large, leading to a loss of precision.
* Memory Usage - Representing extremely large numbers may require more memory than the standard data types provide. This can lead to increased memory usage and potential resource constraints.

Here is a small comparison of the 2 approaches -

**Loop-Based Approach:**

Space Complexity: Loop approach will be memory efficient as it requires constant memory on every iteration.

Efficiency: Loop will be more efficient as it doesn’t need repetitive functions call.

Iteration Control: Loops offer precise control over the iteration process, allowing for easy management of variables, loop conditions, and increment/decrement operations.

**Recursive Approach:**

Readability and Elegance: Recursive solution is more readable as it imitates the salary double problem in a more relatable way.

Function Call Overhead: Recursion involves function calls, which add overhead to the program's execution. This can lead to a higher memory consumption due to the call stack.

Stack Overflow Risk: If not properly optimized or if the recursion depth is too large, there is a risk of a stack overflow, leading to a program crash. This is a concern for problems with deep recursion as you can see in my program for 100 days above.

Ref - [Recursive Practice Problems with Solutions - GeeksforGeeks](https://www.geeksforgeeks.org/recursion-practice-problems-solutions/)

[Python For Loops (w3schools.com)](https://www.w3schools.com/python/python_for_loops.asp)