5.2 - Week 5 - Applied - Practical

Exercise 1 - O(1) Complexity

Write a python function that asks for an input from the user and tells if that number is `higher` or `lower` than a target number provided. User the scaffold provided.

Make sure the program has a complexity of $O(1)$

Exercise 2 - O(n) Complexity

Extend your previous program that accepts `n` inputs from a user until the user is able to guess the right answer.
Use the scaffold provided below:

Exercise 3 - O(log(n)) Complexity

Write a python function where, given a sorted list of numbers, it asks for an integer input and then returns `true` if the inputted integer exists in the list and `false` otherwise.

Use the scaffold given to you.

Exercise 4 - Complex Complexity

We will now write a program to calculate the digital root of a number.

The **digital root** of a decimal integer is obtained by adding up its digits and then repeating this process to the result, and so on until you get a single digit. The single-digit is the digital root of the decimal integer you started with.

As an example: to find the digital root of 979853562951413, we calculate: sum of digits = 9 + 7 + 9 + 8 + 5 + 3 + 5 + 6 + 2 + 9 + 5 + 1 + 4 + 1 + 3 = 77, then sum of digits = 7 + 7 = 14, then sum of digits = 1 + 4 = 5. Now we have just one digit, 5, so that's the digital root of the number we started with.

Write a program that calculates the digital root of an integer.

Make sure the complexity of this program does not exceed $O(n)$

Extra Exercises - For Complex Complexity Warriors

We heard y'all like challenges. Complexity has just begun. Proceed if you dare

Exercise 5 - Varying Complexity

We now have to write 3 python functions to understand how we can do the same thing, but in 3 different complexities.

The question will remain the same for all 3 functions: Write a program that outputs the n^{th} Fibonacci number.

The only difference is:

1. `fib_constant(n: int)` has a constant time complexity
2. `fib_linear(n: int)` has a linear time complexity
3. `fib_exponential(n: int)` has an exponential time complexity

Use the file provided in the scaffold to write out the solutions for all 3. Ensure that you get the correct answer.

Hint: You can use Binet's Formula to find the nth Fibonacci number:

$$F_n = \frac{\left(\frac{\left(1+\sqrt{5}\right)}{2}\right)^n - \left(\frac{\left(1-\sqrt{5}\right)}{2}\right)^n}{\sqrt{5}}$$

where F_n is the nth fibonacci number