Recursive Functions and Search Algorithms

Objectives
Using Python to solve complex problems

- Implement simple programs using Python
- Implement recursive algorithms
- Implement search algorithms

Requirements

1. Implement algorithms for the following problems:
   a. Recursive functions for factorial and Fibonacci
   b. A recursive version of the function f(n) = 3 * n
   c. A recursive function that returns the sum of the first n integers
   d. A function which implements the Pascal's triangle:

```
            1
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5   10   10   5   1
```

   e. Write a function that returns the min /max of a list.
   f. Write a function, `recursive_min`, that returns the smallest value in a nested number list.

```
recursive_min [2, 9, [1, 13], 8, 6] == 1
recursive_min [2, [[13, -7], 90], [1, 100], 8, 6] == -7
```

   g. Write a function `count` that returns the number of occurrences of `target` in a nested list:

```
count(2, []) == 0
count(2, [2, 9, [2, 1, 13, 2], 8, [2, 6]]) == 4
count(7, [[9, [7, 1, 13, 2], 8], [7, 6]]) == 2
```

2. Implement search algorithms and establish their complexity:
   a. Sequential search (for unordered and ordered lists)
   b. Binary search (for ordered lists)