# Recursive Functions and Search Algorithms

## Objectives

*Using Python to solve complex problems*

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

## Requirements

i. Implement **recursive** algorithms for the following problems:

1. Calculating the factorial of a number
2. Calculating the $n^{th}$ element of the Fibonacci sequence
3. A recursive version of the function $f(n) = 3 * n$
4. Calculating the sum of the first $n$ integers
5. A function which implements the Pascal's triangle (returns the $n^{th}$ row):

```
      1
    1   1
  1   2   1
1   3   3   1
1   4   6   4   1
```

6. Calculating the min/max of a list.
7. Write a function, `recursive_min`, that returns the smallest value in a nested list.

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

8. Write a function `count` that returns the number of occurrences of a 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
```

ii. Implement search algorithms and establish their complexity:

1. Sequential search (for unordered and ordered lists)
2. Binary search (for ordered lists)