ME1 Computing- Session 6: Recursive functions and Consolidation

Learning outcomes:

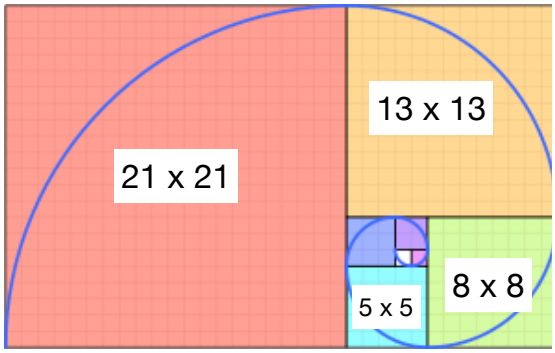
- Understanding the principles of recursion
- Being able to implement recursive functions
- Consolidating use of functions in general

Before you start

In your H drive create a folder *H:\ME1MCP\Session6* and work within it.

Task 1: Recursive functions: Fibonacci sequence

The Fibonacci sequence is found in nature, to represent physical events such as spiral galaxies, hurricanes and the evolution of prolific species (like rabbits). An example is the spiral of a seashell, following this pattern:



The Fibonacci sequence is generated, starting from the seeds 1 and 1, by adding up the two previous numbers of the sequence, i.e.

$$F(n) = F(n-1) + F(n-2)$$

	1	1	**2**	**3**	**5**	**8**	**13**	**21**	…
n^{th}	1^{st}	2^{rd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	

Write a **recursive** function, *Fibonacci(n),* to compute the *n*-th Fibonacci number.

Generate the first *N* numbers of the sequence.

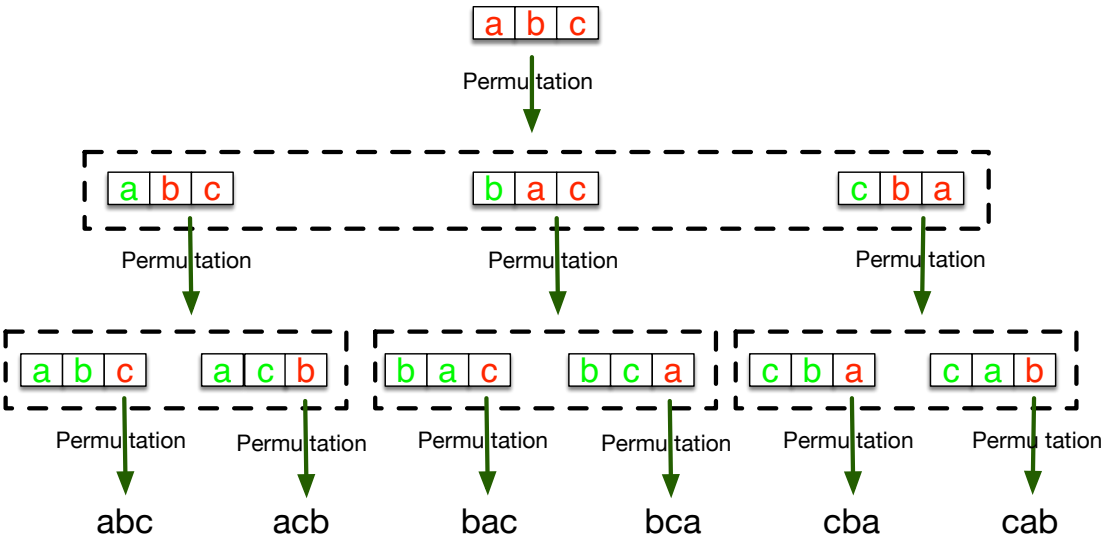
Answer Question 1

Task 2: Recursive functions: Reverse a string

Write a **recursive** function, *Reverse(string),* to reverse a given string, i.e., from '*stressed*' to '*desserts*'.

Task 3: Recursive functions: Permutation of letters (How to crack a password)

Given a list of N values, write a recursive function, *Permute*, to determine all the possible permutations of these values. (When dealing with lists, remember about the aliasing, which is typical for lists in Python).



Answer Question 2

Task 4: Adding mathematical fractions

1. *Recursive functions: Greatest Common Divisor*
   Write a **recursive** function, *GCD(a,b),* that evaluates the Greatest Common Divisor (i.e., the largest common factor) between two numbers, *a* and *b*.
   The GCD can be determined with the following process:

   a) Divide one number by the other.

   b) If the remainder is zero, then the divisor is the GCD.
   Otherwise, divide the divisor of step a) by the remainder and repeat step a).

2. *Least Common Multiple*
   Write a function, *LCM(a,b),* that evaluates the Least Common Multiple (i.e., the largest common factor) between two numbers, *a* and *b*.
   You can make use of the GCD, by using the definition:

$$LCM(a,b) = \frac{|ab|}{GCD(a,b)}$$

3. Write a function, *AddFractions(N,D),* that receives two lists of integer values, N and D, representing the numerators and denominators of some fractions, respectively, and returns two values, N_t and D_t, defined as:

$$\frac{N_t}{D_t} = \frac{N[0]}{D[0]} + \frac{N[1]}{D[1]} + \cdots + \frac{N[-1]}{D[-1]}$$

Ensure that the final fraction, $\frac{N_t}{D_t}$, is at its minimal form.

Answer Question 3

Task 5: Experimental analysis

The performance of a boiling system has been monitored with daily measurements of its water temperature.

Measurements were performed throughout ten different days. Each day, measurements were taken at every hour, for a total of 24 data per day.

The data measured are stored sequentially, one per line, in the file *Temperatures.txt*.

Compute the average temperature, T_{av}, of the water for each day, and plot the result in a graph with T_{av} vs *day*.

Compute also the maximum and minimum temperatures overall the testing period.

Answer Question 4

Task 6: Fractals

Fractals are complex patterns that repeat themselves at different scales. Given an initial point in space, (x_0, y_0), every other point being generated, (x_i, y_i), depends on the previous one (x_{i-1}, y_{i-1}).

Write a script to plot *N* points, with coordinates (x_i, y_i), generated from the following iterative sequence (also known as Julia set):

$$x_i = y_{i-1} - k * \sqrt{|b * x_{i-1} - c|}$$
$$y_i = a - x_{i-1}$$

$$k = \begin{cases} -1 & \text{if } x_{i-1} < 0 \\ 0 & \text{if } x_{i-1} = 0 \\ +1 & \text{if } x_{i-1} > 0 \end{cases}$$

Plot all the points in blue.

Run the script with:

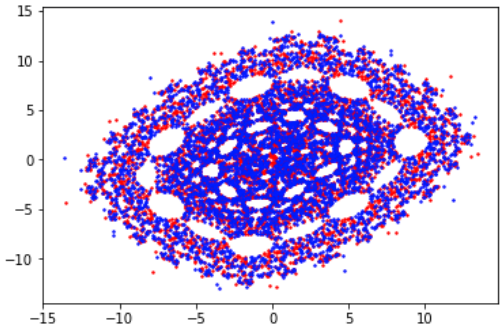
Setup 1: *a = 0.4, b = 1, c = 0,* $(x_0, y_0) = (0,1)$*, N = 20000*

Setup 2 / 3: *a = 0.8, b = 1.3, c = 0.9,* $(x_0, y_0) = (0,2)$*, N = 10000 / 50000*

Enjoy exploring more plots, with different values of *a, b, c* and (x_0, y_0).

If you want to spice up the graph, plot the odd and even points with different colours.

Enjoy running the script with different parameters. You will notice that for minor changes of *a, b, c* and (x_0, y_0) you get completely different patterns: this is the typical behaviour of a *chaotic*.



Answer Question 5

Task 7: Bonus question: Merge sort (Divide et Conquer)

Write a **recursive** function, *MergeSort(List),* that receives a list of values and returns the same list sorted in ascending values, by using the Merge Sort method algorithm.

The merge sort involves the following steps:

1. Divide the unsorted array into two halves, multiple times, until each half contains a single element. *(DIVIDE)*
2. Take adjacent pairs of two single-element array and merge them to form an array of two elements, in ascending order. *(CONQUER)*
3. Repeat the process till a single sorted array is obtained.