Recursion

Recursion is when a function is calling itself (multiple time) with the smaller input. Thus, resulting in reducing the size of problem (i.e., Converging the problem towards base case/ trivial case).

Math behind the Recursion

**PMI (Principle of Mathematical Induction)**

For a function f(x)

Step 1: To prove for lower value like f(0) or f(1) is true | ***Base Case/ Trivial Case***

Step 2: Assume f(k) is true | ***Induction hypothesis***

Step 3: Prove f(k+1) is true | ***Induction Step***

**Extended form of PMI**

Step 1: Assume f(0), f(1), f(2), … f(k) is true

Step 2: Using above f(0)…f(k) prove f(k+1) is true

*def* Printer(*n*):

    if n == 0:

        return

    Printer(n - 1) # printing while popping out from stack

    print(n) # printing after all the function is returned

Printer(10) #1, 2, 3, 4, … 10

*def* Printer(*n*):

    if n == 0:

        return

    print(n) # printing before then calling the next function

    Printer(n - 1) # printing before inserting into the stack

Printer(10) #10, 9, 8, … 1

Fibonacci series requires 2 base cases in a recursion.(Extended PMI)

RecursionError: maximum recursion depth exceeded in comparison

The reason we get it is because every recursion call uses some system memory and python has certain memory limit to used by recursive call if it gets exceeded, it gives us a recursion error.

import sys

sys.setrecursionlimit(3000) #to increase the recursion limit in python

First & Last Index using Recursion

#First index using multiple copy of array

arr = [20, 12, 3, 33, 20, 2]

key = 2

*def* FirstIndex(*a*, *k*):

    if len(a) == 0:

        return -1

    if a[0] == k:

        return 0

    OP = FirstIndex(a[1:], k)

    if OP == -1:

        return -1

    else:

        return OP + 1

print(FirstIndex(arr, key)) #5

#First index using index as a third parameter

arr = [20, 12, 3, 33, 20, 2]

key = 2

index = 0

*def* FirstIndex(*a*, *k*, *i*):

    if i >= len(a):

        return -1

    if a[i] == k:

        return i

    return FirstIndex(a, k, i+1)

print(FirstIndex(arr, key, index))

#Last Index using index as a third parameter

arr = [20, 2, 3, 33, 20, 62]

key = 2

index = len(arr) - 1

*def* LastIndex(*a*, *k*, *i*):

    if i < 0:

        return -1

    if a[i] == k:

        return i

    return LastIndex(a, k, i-1)

print(LastIndex(arr, key, index)) # 1

#Last index using multiple copy of array

arr = [20, 12, 3, 33, 20, 2]

key = 2

*def* LastIndex(*a*, *k*):

    if len(a) == 0:

        return -1

    output = LastIndex(a[1:], k)

    if output == -1:

        if a[0] == k:

            return 0

        else:

            return -1

    else:

        return output +1

print(LastIndex(arr, key))