Recursion, means a defined function can call itself.

- Benfit : without using loop we can execute our task

Recursion Concept/Aim:

- starts by solving smaller portions of your problem until the original, larger problem is solved
- Its about simulating a loop

ex: real life : searching a pic from a group of folders

- we started by creating a method to find the pic inside 1 folder, and by applying the same logic on all the folders recursively. We solve the larger problem

Disadvantages of recursion :

- following the logic behind recursive fuctn might be hard sometimes
- recursive calls are expensive (inefficient) as they take up a lot of memory and time
- They are too hard to debug

Advantages of recursion:

- code is elegant and clean in recursive fuctn
- a larger task can be broken down into smaller sub problems using recursive fuctn
- generating sequence is easier in case of recursion rather then using some iterative fuctn

In [ ]:

In [ ]:

Creating a mul fuctn using loop

In [5]:

```python
def mul(a,b):
    res = 0
    for i in range(b):
        res+=a
    return res
```

In [6]:

```python
mul(5,6)
```

Out[6]:

```
30
```

In [ ]:

- > using recursion

In [4]:

```
def mulr(a,b):
    if b==1:
        return a
    else:
        return a+mulr(a,b-1)

print(mulr(5,6))
```

```
30
```

In [ ]:

Factorial of a no.

In [8]:

```
def fact(no):
    res = 1
    for i in range(1,no+1):
        res*=i
    return res
```

In [10]:

```
print(fact(5))

# 5x4x3x2x1
```

```
120
```

In [ ]:

- using recursion

```
5! = 5x4x3x2x1

5! = 5x(4!)
5! = 5x(4*3!)
5! = 5x4x(3*2!)
5! = 5x4x3x(2x1!)
5! = 5x4x3x2x(1)
when n = 1 stop
```

In [22]:

```
def factr(no):
    if no==1:
        return 1
    else:
        # print('no : ',no)
        return no*factr(no-1)
```

In [23]:

```
print(factr(5))
```

```
no :  5
no :  4
no :  3
no :  2
120
```

```
In [ ]:
```

Palindrome

```
In [4]: def palin(txt):
            if txt==txt[::-1]:
                print('Its palindrome')
            else:
                print('Not a palindrome no')
```

```
In [5]: palin('madam')

        Its palindrome
```

```
In [6]: palin('maab')

        Not a palindrome no
```

- > using recursion

```
In [7]: def palinr(txt):
            if len(txt)==1:
                print('Plaindrome no')
            else:
                if txt[0]==txt[-1]:
                    palinr(txt[1:-1])
                else:
                    print('Not a palindrome no')
```

```
In [8]: palinr('madam')

        Plaindrome no
```

```
In [9]: palinr('moob')

        Not a palindrome no
```

```
In [10]: palinr('mom')

         Plaindrome no
```

```
In [ ]:
```

Generating fibonacci series

- rabbit problem

In [66]:
```
def fibon(n):
    a,b=0,1
    for i in range(n):
        print(a, end=' ')
        tmp=a
        a = b
        b=tmp+b
```

In [67]:
```
fibon(5)
```

```
0 1 1 2 3
```

- Using recursion

In [68]:
```
def fibonr(n):
    if n <= 1:
        return n
    else:
        return(fibonr(n-1) + fibonr(n-2))
```

In [65]:
```
fibonr(10)
```

Out[65]:
```
55
```

In [ ]:

In [37]:
```
# for the soln of rabbit que

def fibonrn(n):
    if n==0 or n == 1:
        return 1
    else:
        return(fibonrn(n-1) + fibonrn(n-2))
```

In [38]:
```
fibonrn(5)
```

Out[38]:
```
8
```

In [ ]:

Q. Generating the fibbonaci series upto given no

In [89]:
```
def fibo1(n):
    a,b = 1,1
    for i in range(n):
        print(a, end=' ')
        tmp = a
        a = b
        b = tmp+b
```

In [90]:

```
fibo1(11)
```

```
1 1 2 3 5 8 13 21 34 55 89
```

Q. Find a fibonnaci number present at the given posn

In [81]:

```
def fibo2(n):
    a,b = 1,1
    for i in range(n):
        if (i+1)==n:
            print(f'f({n}) is {a}')
        tmp = a
        a = b
        b = tmp+b
```

In [82]:

```
fibo2(10)
```

```
f(10) is 55
```

In [ ]:

Finding the time consumed by each of the fibo fuctn

- By using recursion

In [92]:

```
import time

def fibo1(n):
    if n==1 or n==0:
        return n
    else:
        return fibo1(n-1)+fibo1(n-2)


start = time.time()
print(fibo1(10))
print('time taken : ',time.time()-start)
```

```
55
time taken :  0.00021743774414062
```

In [ ]:

- By using iteration

In [84]:

```
import time

def fibo2(n):
    a,b = 1,1
    for i in range(n):
        if (i+1)==n:
            print(f'f({n}) is {a}')
        tmp = a
        a = b
        b = tmp+b

start = time.time()
fibo2(10)
print('time taken : ',time.time()-start)
```

```
f(10) is 55
time taken :  0.0007922649383544922
```

In [ ]:

The main problem is recursion is it takes longer time

- to solve this issue we can use Dynamic pogramming
- Memoization

We will use a dict to store the fibo of no. so that the repeated fibo. of no. will not be calculated again and the data will be fetched directly from the dict,

- It;s kinda trade off of storage space for less execution time

In [ ]: