Recursion

- A function calls itself is known as "Recursion"
- Base case(Terminating condition)
- Recursive case(Calling itself)

Factorial of a given number using recursion

```
In [ ]:
```

```
n=5
res=fact(5)--->120

5*fact(5-1)---->5*24

4*fact(4-1)---->4*6

3*fact(3-1)---->3*2

2*fact(2-1)---->2
```

In [1]:

```
def fact(n):
    if n==0 or n==1:
        return 1
    else:
        return n*fact(n-1)
n=int(input("Enter any value"))
res=fact(n)
print("Factorial of a given number is:",res)
```

```
Enter any value5
Factorial of a given number is: 120
```

Fibonacci Series of Nth number-Recursive

• 1 1 2 3 5 8 13 21 34 55-----> first 10 fibonacci numbers

```
In [8]:
```

```
def fib(n):
    if n==1:
        return 1
    elif n==2:
        return 1
    elif n>2:
        return fib(n-1)+fib(n-2)
for n in range(1,6):
    print(n, "st:",fib(n),end='')

1 st: 12 st: 13 st: 24 st: 35 st: 5

In []:

fib(5)
    fib(4)+fib(3)
    fib(3)+1+2
    fib(2)+fib(1)+
```

Momization

- · Explicit Approach
- Implicit Approach

In [5]:

```
# Explicit Approach
fcache={}
def fib(n):
    # If we have cached the value, then it return it
    if n in fcache:
        return fcache[n]
   # Computing the Nth term
    if n==1:
        val=1
    elif n==2:
        val=1
    elif n>2:
        val=fib(n-1)+fib(n-2)
    # Cache the value and return it
    fcache[n]=val
    return val
for n in range(1,1001):
    print(n,":",fib(n),end=' ')
```

In [7]:

```
# Implicit Approach
from functools import lru_cache
@lru_cache(maxsize=1000) # default 1024
def fib(n):
    if n==1:
        return 1
    elif n==2:
        return 1
    elif n>2:
        return fib(n-1)+fib(n-2)
for n in range(1,1001):
        print(n,":",fib(n),end=' ')
```

In []: