(6.1) Recursion

Introduction

```
A function that calls itself is known as Recursive Function.
Eg:
factorial(3)=3*factorial(2)
            =3*2*factorial(1)
            =3*2*1*factorial(0)
            =3*2*1*1
            =6
factorial(n)= n*factorial(n-1)
```

The main advantages of recursive functions are

```
1. We can reduce length of the code and improves readability
2. We can solve complex problems very easily.
```

Tail Recursion

In [11]:

```python
def func(x):
    if x:
        func(x-1)
        print(x)

func(5)
```

```
1
2
3
4
5
```

Head Recursion

In [15]:

```python
def func(x):
    if x:
        print(x)
        func(x-1)

func(5)
```

```
5
4
3
2
1
```

Recursion depth

In [6]:

```python
import sys
print(sys.getrecursionlimit())
```

```
3000
```

In [3]:

```
c = 0
def hello():
    global c
    c = c + 1
    hello()
```

In [4]:

```
hello()
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10808/2674044599.py in <module>
----> 1 hello()

C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10808/850710868.py in hello()
      3     global c
      4     c = c + 1
----> 5     hello()

... last 1 frames repeated, from the frame below ...

C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10808/850710868.py in hello()
      3     global c
      4     c = c + 1
----> 5     hello()

RecursionError: maximum recursion depth exceeded
```

We can change limis ?

In [9]:

```
sys.setrecursionlimit(1000)
print(sys.getrecursionlimit())
#if we restart the kernal it will automatic become default limit
```

```
1000
```

In [10]:

```
sys.setrecursionlimit(3000)
print(sys.getrecursionlimit())
```

```
3000
```

Problems On Recursion

Number Table

In [14]:

```
def func(num, c=1):
    if c != 11:
        print(num*c)
        func(num,c+1)

func(5)
```

```
5
10
15
20
25
30
35
```

```
40
45
50
```

Factorial

In [18]:

```python
def fact(num):
    if num == 0:
        return 1
    return num*fact(num-1)

fact(5)
```

Out[18]:

```
120
```

Fibonacci

```
fib(8) -> fib(7) + fib(6)
fib(7) -> fib(6) + fib(5)

fib(1) -> 0
fib(2) -> 1

fib(3) -> fib(2) + fib(1)
fib(4) -> fib(3) + fib(2)
```

In [19]:

```python
def fib(n):
    if n == 1:
        return 0
    if n == 2:
        return 1
    return fib(n-1) + fib(n-2)

fib(8)
```

Out[19]:

```
13
```

Prime Number

In [ ]: