25-09-2025, Thursday.

⇒ Create a fn to print prime no. from the given range using with input & without return method.

#without fn

```
s1=2          # Range
s2=20
for i in range(s1, s2+1, 1):
    for j in range(2, i, 1):
        if(i%j==0):
            break
    else:
        print(i)
```

o/p 2 3 5 7 11 13 17 19

i=2 i=3 i=4 i=5
j= j=2 j=2,3 j=2
i=2
j=2,3,.....19

#With fn

return only execute once and return 1 value.

```
def prime(r1, r2)
    for i in range(r1, r2+1, 1):
        for j in range(2, i, 1):
            if(i%j==0):
                break
        else:
            return i
prime(2, 10)
```

o/p 2

Using List

```
def prime(n1, n2):
    l1 = []
    for i in range(n1, n2+1, 1):
        for j in range(2, i, 1):
            if (i%j==0):
                break
        else:
            l1.append(i)
    return l1
prime(2,10)
```

o/p [2, 3, 5, 7]

append is used to add new element at the end of list

⇒ Create fn with input & with return to find largest among 3 nos

```
def lar(a,b,c):
    if (a>b and a>c):
        return a
    elif (b>a and b>c):
        return b
    else:
        return c
lar(9,5,3)
```

o/p 9

OR `return f"{a} is large"`

`return f"{b} is large"`

Function as Parameter: We can assign them to variables

- We can pass them as parameters to other fns.
- We can return them from fns

Ex

```
def square(x):
    return x*x
def cube(x):
    return x*x*x
def apply_fun(fun_name, num):
    return fun_name(num)
apply_fun(square, 5)   # fn calling
```

o/p 25

```
apply_fun(square, 5)
return square(5)
```

⇒ Recursion fn : It is a fn that calls itself until a base condition is satisfied.

```
def fact(n):
    if(n==1):
        return 1
    else:
        return n* fact (n-1)
```

fact (5)

o/p 120

fact(5):
5==1 False

return 5 * fact (5-1)
5 * fact (4)

fact(4):
4==1 False

return 5 * 4 * fact (3)

fact (3):
3==1 False

return 5*4*3 *fact(2)

fat(2):
2==1 False

return 5*4* 3*2 *fact(1)

fact(1):
1==1 True
return 1

o/p 5* 4* 3 * 2 =120

⇒ Nested fn:

```
def outer-fun (p1, p2... pn):
    def inner-fun(p1, p2... pn):
        return value
    return value
```

⇒ Lambda function: It is a small (bcz it is having only 1 line fn), anonymous function in python.

- It is defined using a keyword lambda instead of def.
- It can take any no. of arguments but must contain only one expression.
- Expression is automatically returned. (No need to use return).

Syntax: lambda arguments : expression

Ex
```
s = lambda num : num * num
s(5)
```
o/p = 25

No need to call fn name, only variable calling with input/argument.

⇒ Addition of 2 no.s

```
add = lambda a,b : a+b
add(5,6)
```
o/p 11

⇒ Nested lambda: when lambda returns another lambda.

```
add = lambda x: (lambda y: x+y)
add (5)(3)          o/p 8
```

#Outer lambda

add(5) x=5 and returns (lambda y: 5+y)

#Inner lambda

y: 5+y, take y=3 return 5+3=8

⇒ multiply = lambda a: (lambda b = a * b)

multiply (4)(6) o/p 24

Outer lambda

multiply(4) a=4 return (lambda b = 4 * b)

Inner lambda

b=6, b= 4*6, 24 = 4 * 6 = 24