PYTHON PROGRAMMING
GATE DA/DSA

Agenda:

→ Scope of Variables
- Local Variable
- Global Variable
- Non local Keyword

→ Nested Functions in Functions

→ Higher Order Functions

→ Anonymous Function (Lambda Functions)

Functions
- Syntax
- access
- arguments
  - pos
  - Key
  - combination of both
  - Default argument
  - *args
  - **kwargs

Function(l):
    operation
    l.append(3)

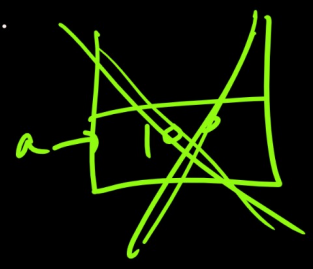
l = [1,2]
function(l)
print(l)
[1,2,3]

Local Variables: — Scope of this is only inside that specific function.

* Local Variables are those which are initialized inside a function and belong only to that particular function.
* It cannot be accessed anywhere outside the function.

```
def func_a():
    a = 100
    print(a)

func_a()
print(a)
```



a is not defined

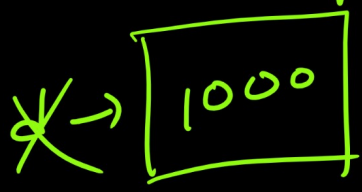
Global Variables:

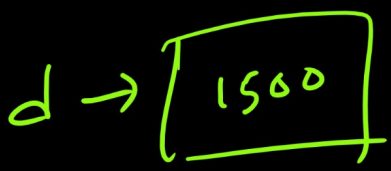
- Global Variables are those which are defined outside any function and can be accessible throughout the program i.e both inside / outside of every function.

```
def func_d():
    d += 500
```

d is global variable. Can be accessed throughout the program.

```
d = 1000
d += 500
print(d) → 1500
func_d()
print(d)
```

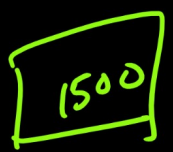


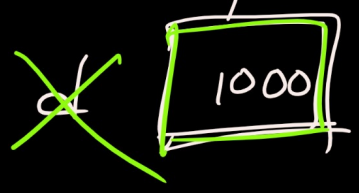



```
def func_new():
    global d
    d = 1000
    print(d)
    d += 500
    print(d) → 1500
d = 1000
func_new()
print(d) → 1500
```

1000 ←

local variable



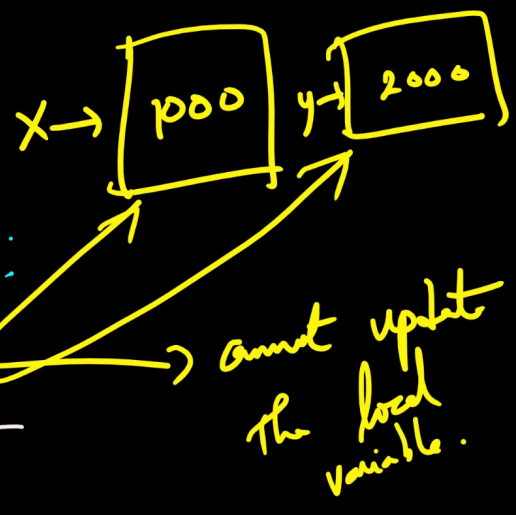


Nested Functions :

* A nested function is a function defined inside another function. The outer function can call the inner function and the inner function can access the variables and parameters of the outer function.

```
def func_outer():
    x = 1000
    y = 1000
    def func_inner():
        print(x,y)
    func_inner()

func_outer()
```



Python → nonlocal variables :

→ Nonlocal variables are used in nested functions whose local scope is not defined. This means that the variable can be neither in the local scope nor the global scope.

→ We use 'nonlocal' keyword to create nonlocal variables.

```
def outer( ):
    v1 = 100
    def inner( ):
        v1 += 1000
        print(v1)
    print(v1)
```

global v1
v1 = 100

local variable

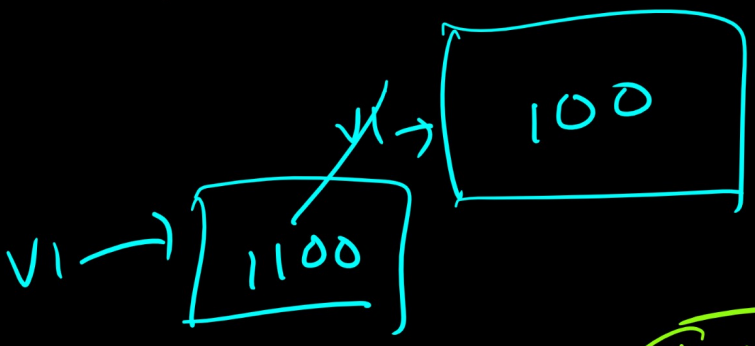
inner ← 1100

outer ← 100

```
outer()
print(v1)
```

main code. Throw error.

Case 1: inner fun → local variable v1

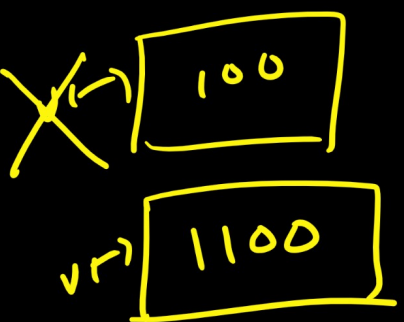


Case 2:

inner : 1100 ← global var.
outer : 100 ← local var.

v1 : 1100 ← main code

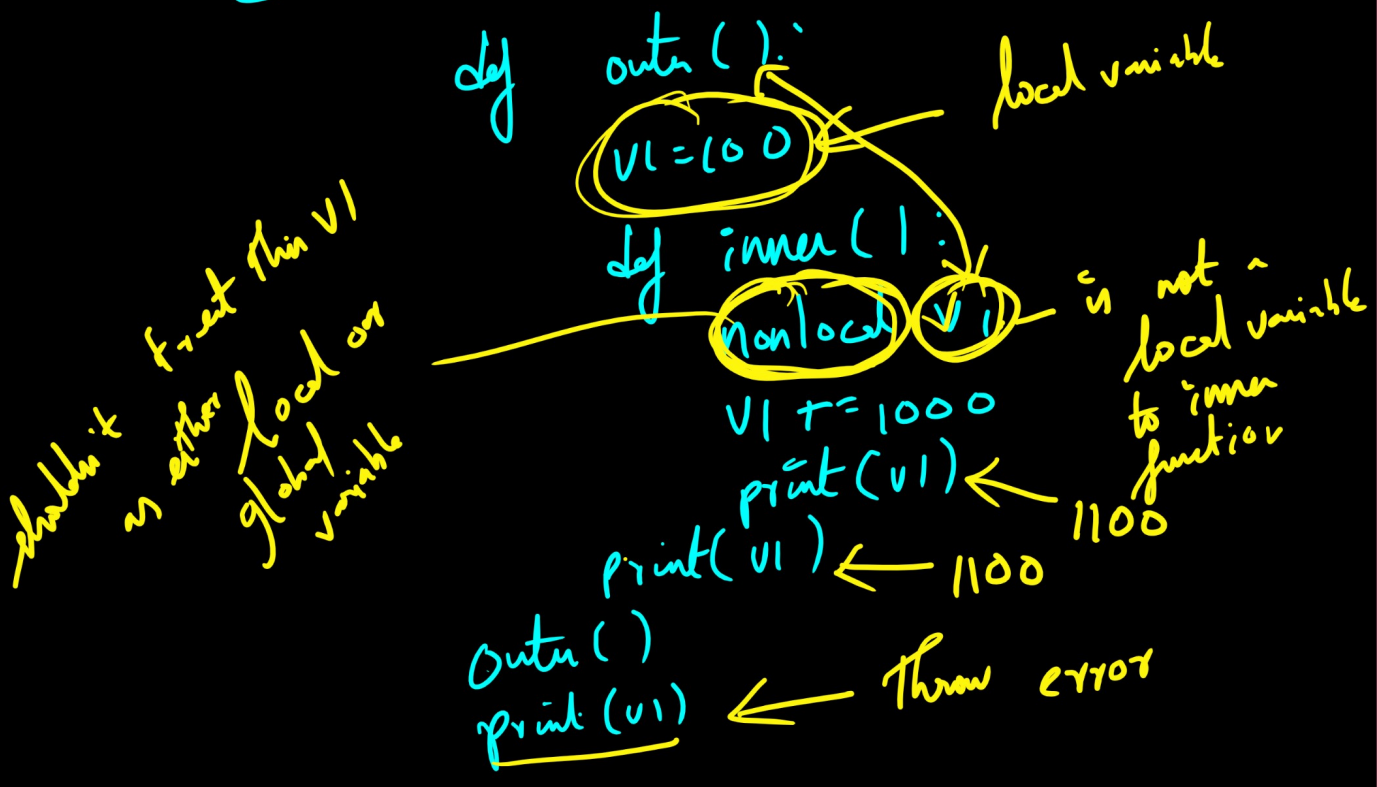
```
def outer ( ):
    global v1
    v1 = 100
    def inner( ):
        global v1
        v1 = 100
```

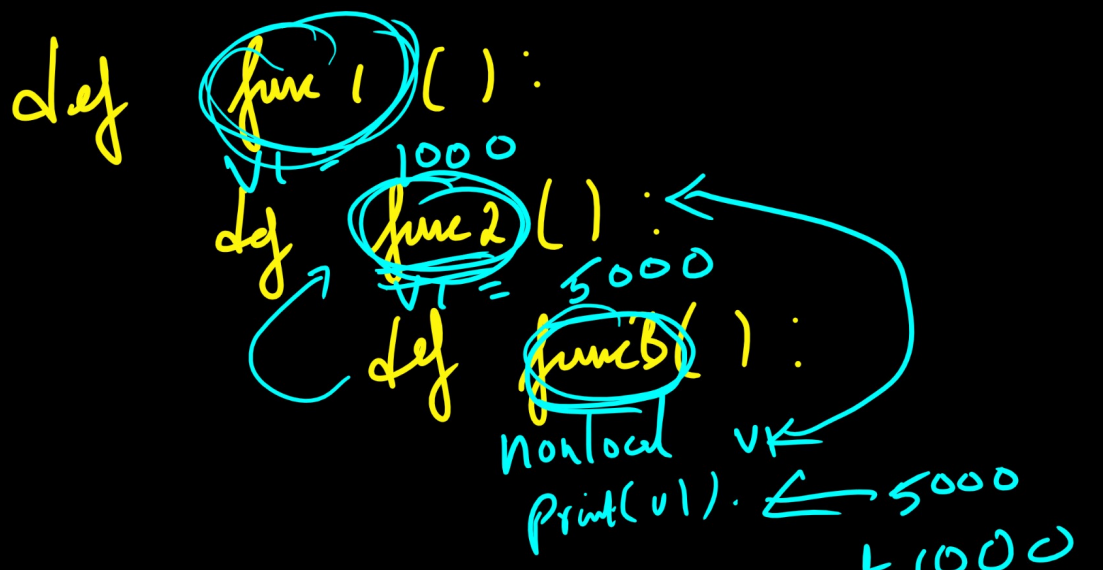


v1 += 1000
print(v1) ← 1100
Print(v1) ← 1100

outer()
print(v1) ← 1100

Case 4:





```
def Calculate (a, b, Operation):
    return Operation(b, a)

def sub (a, b)
    return a-b

Calculate (10, 15, sub)
```

sub(10, 15)

Operation → sub

function

function inside function

Recursion