PYTHON PROGRAMMING
GATE DA/DSA

Agenda:

- → Lambda functions
- → Filter
- → Map

} ✓ Problem Solving

Lambda Function: In Python, Lambda functions are anonymous functions means the functions is without a name.

Lambda is a keyword which used to define an anonymous functions.

Syntax: lambda arguments : expression

```
def fun_name(x):
    return x**2
```

```
lambda x : x**2
```

↑ There is no name to this particular function.

Regular function → None

Lambda function → function object.

Lambda functions → generally people use it if a particular function is either single line or if you don't need to use it for future.

```
def add10(x):
  return x+10

def sum(a):
  return add10(a)
```

add10(10)

20

sum(10)

lambda x : lambda y : x+10

x = 10

f3 = lambda _ : print(x)

x

a = 10

type(f3)

function.

int

a = f3( )

10

a = None

```
def myFunc(n):
  return lambda a: a*n

myLambda1 = myFunc(2)
```

myLambda1 => function

lambda a : a * n

2

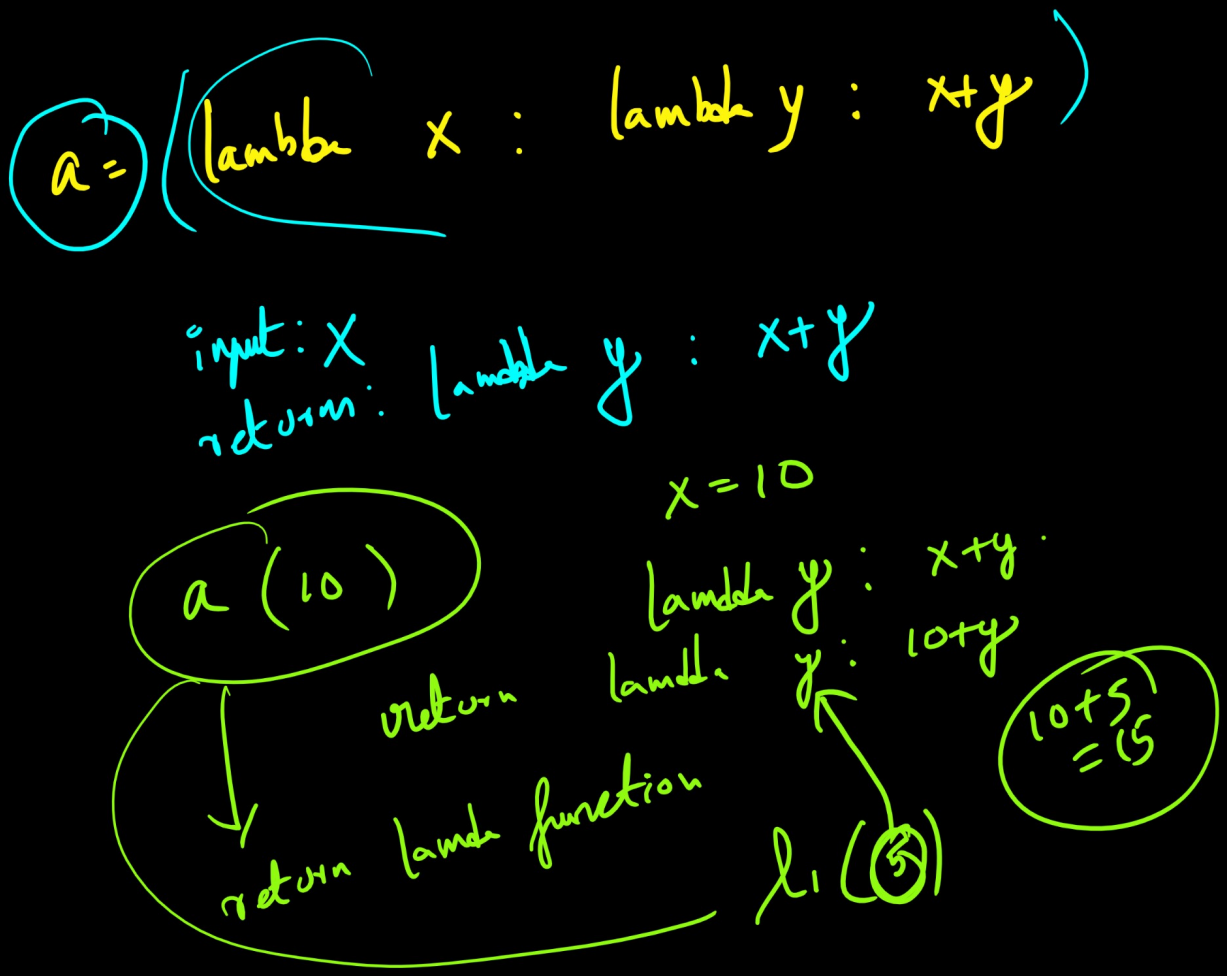
a*

myLambda( )

→ Throw error: myLambda1 (20)

20 : 20*2

40



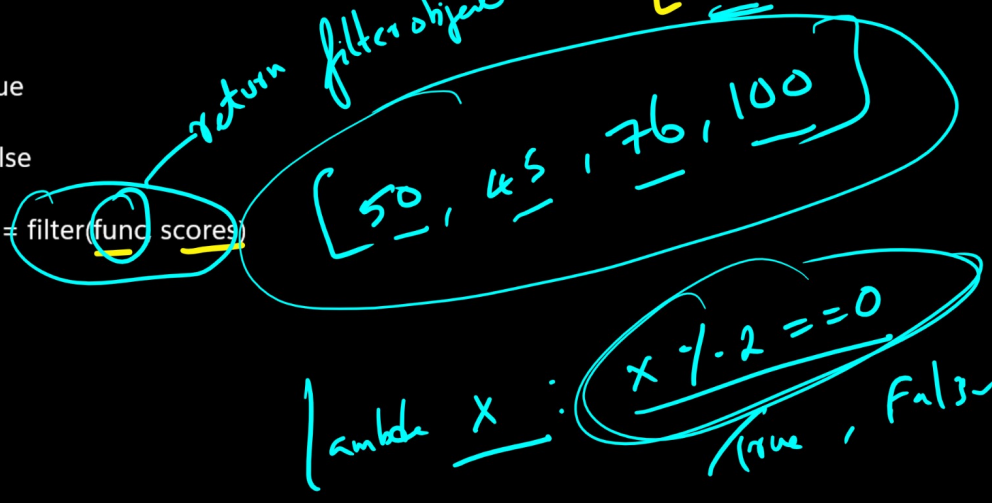
Filter function:

filter (function , iterable)

```
scores = [50, 45, 10, 76, 100, 12, 5]

# create a list which takes the score greater than 40
def func(x):
  if x>40:
    return True
  else:
    return False

filtered_list = filter(func, scores)
```



```
for value in numbers:
  if isPrime(value):
    primeNumbers.append(value)

print(primeNumbers)

primeNumbers = list(filter(isPrime, numbers))
```

def fun(a):
if a > 18:
return false.

fun(2)

None
bool(None) → false

filter(fun, iterable)

bool( ) → false.

```
def myFun(a):
  if a>18:
    return False
  else:
    return a+12
```

a-12
12-12 = 0 bool(0) ⇒ False

1
return a+12
1+12 = 13
bool(13) → true

```
numbers = [1,2,9,12,18,20,25,40,50]

filtered_numbers = list(filter(myFun, numbers))

print(filtered_numbers)
```

[1, 2, 9, 12, 18
13 14]

(func, iterable)

Map(fun, iterable)

Map:

map(function , iterable)

```
def myFun(a):
  if a>18:
    return False

numbers = [1,2,9,12,18,20,25,40,50]

map_numbers = list(map(myFun, numbers))

print(map_numbers)
```

numbers

1

None

```
[None, None, None, None, None, False, False,
False, False]
```

Problem Solving

12:54

Breath

1:

What is the output?

```
x = 50

def func(x):
    print('x is', x)
    x = 2
    print('Changed local x to', x)

func(x)
print('x is now', x)
```

x = 50 — Global variable

x = 50 — local variable.

x is 50

x = 2

Changed local x to 2

inside function.

outside

50

50 2

2. x = 50 — Global variable

```
def func():
    global x
    print('x is', x)
    x = 2
    print('changed x to', x)

func()
print('Value of x is', x)
```

global x → 50

x = 2

func()
print('Value of x is', x) ← 2

3.
```
def statement(msg, times=1):
    print(msg * times)

statement('Hello')
statement('World', 5)
```

a) Hello
World 5

b) Hello
HelloHello HelloHelloHello

c) Hello
World, World, World, World, World.

d) None of the above

e) Error.

Hello
WorldWorld WorldWorldWorld

4. def func(a, b=5, c=10):
print(a, b, c)

default

func(3, 7)
func(25, c=24)
func(c=50, a=100)

a) 7 3 10
25 5 24
5 100 50

b) 3 7 10
5 25 24
50 100 5

c) 2 7 10

25 5 24

100 50 5

d) None of these.

e) Error.

3 7 10

25 5 24

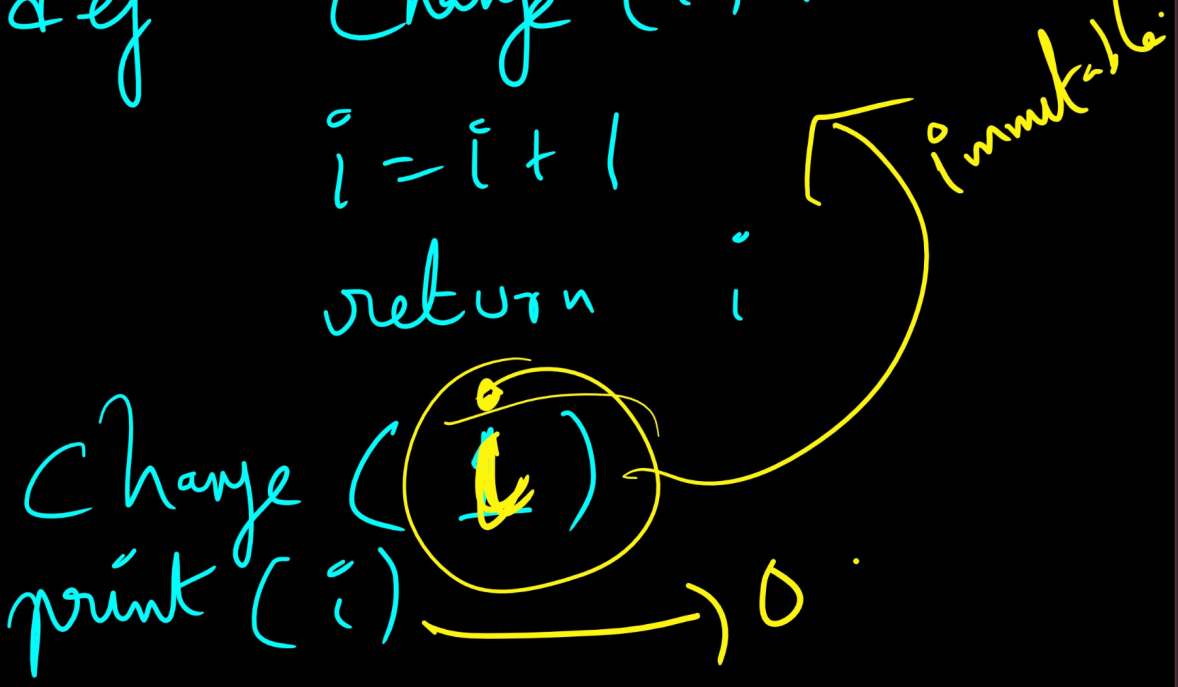
100 5 50

func(c = 50, 100, 5)

↳ Error

5. i = 0

```
def Change(i):
    i = i + 1
    return i
```



Output =1 ?

a) 1 b) Nothing

c) 0 d) Exception.

6.

```
def a(b):
    b = b + [5]
c = [1,2,3,4]
```

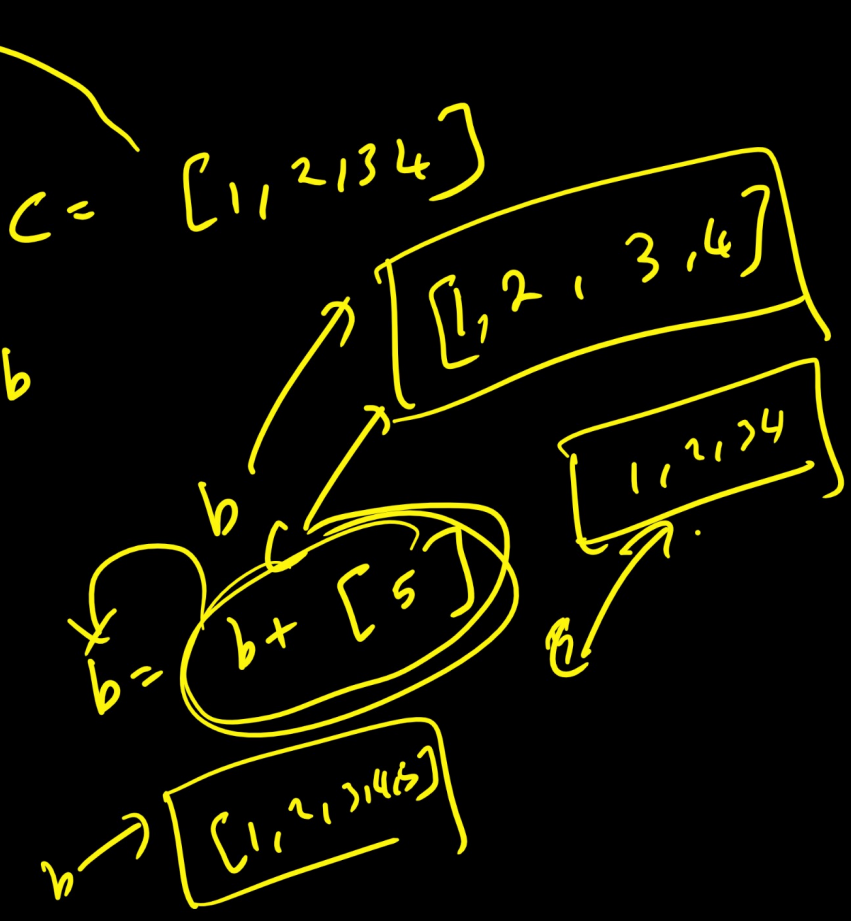
a(c)
print(len(c))

a) 4 b) 5 c) 1 d) Error

e) None of these.

```
def a(b):
  b = b+[5]

c = [1,2,3,4]
a(c)
print(len(c))
```



7. a = 10
b = 20
def Change():
global b
a = 45
b = 56

a) 10 20 c) 10 56
b) 45 56 d) Error.

```
change ()
print (a)
print (b)
```

a = 45

8.
```
def change (i=1 , j=2 ):
    i = i + j
    j = j + 1
    print (i, j)
change (j=1 , i=2 )
```

i = 2 + 1 ⇒ 3

j = 1 + 1 = 2

(3, 2)

9.
```
def change (one, *two):
    print (type (two))
change (1, 2, 3, 4)
```

one = 1

two = (2, 3, 4)

a) Integer
b) Tuple
c) Dict
d) Exception / Error

10.
```
def foo (k):
    k[0] = 1
q = [0]
foo (q)
print (q)
```

a) [0] q → [1] k ↑

b) [0, 1] k ↑

c) [1, 0] k[0] = 1

d) [1]

12.
```
def fun():
    return total + 1
total = 0
print(fun())
```
0 + 1 → return 1 → 1

12.
```
def fun():
    total += 1
    return total
total = 0
print(fun())
```
Error.
```
print(total)
a = total + 1
```

13.
```
def foo(x):
    x = ['def', 'abc']
    return id(x)
q = ['abc', 'def']
print(id(q) == foo(q))
```
x → ['abc', 'def']

1250 → x → ['def', 'abc']

q → ['abc', 'def'] → 1250

False

14.
```
def foo(i, x=[]):
    x.append(i)
    return x
```

```
for i in range(3):
    print(foo(i))
```

a) [0] [1] [2]

b) [0] [0,1] [0,1,2]

c) [1] [2] [3]

d) [1] [1,2], [1,2,3]

e) None of these.

f) Error.

```
def foo(i, x=[]):
    x.append(i)
    return x

for i in range(3):
    print(foo(i))
```

1

foo(0)

[ ]

[0]

[0,1]

[0,1,2]

15. How are variable length arguments specified in the function heading? (*a)

(a) One star followed by a valid identifier.

b) one underscore followed by a valid identifier.

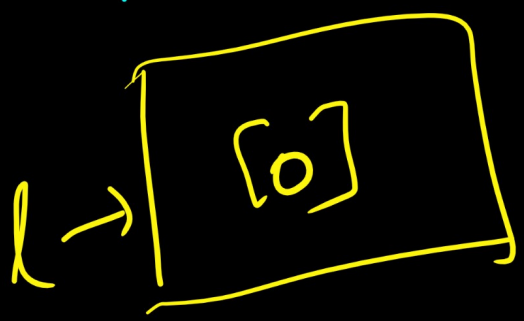
c) two stars followed by valid identifier.

d) two underscored followed by.

16. `l = []`

```
for i in range(3):
    l.append(l.append(i))
print(l)
```

0, 1, 2.



l.append(0)

l.append(l.append(1))

↓ None.

[0, None, 1, None, 2, None]