Lambda function which does not have any name associated with it.
To define a lambda, we define a keyword lambda
We don’t have any return statement in lambda functions
We use lambda function when functionality is very simple and they are not very used often **Syntax**: lambda arguments : expression

```
# without lambda function
def d1():
   return print("Hello World")
d1()

Output
Hello World

# with lambda function
# lambda arguments : expression
r = lambda d1: print("Hello World")
print(type(r))
r(d1)

Output
<class 'function'>
Hello World
```

```
# without lambda function
def d1():
    return "Hello World"
d = d1()
print(d)

Output
Hello World

# with lambda function
r = lambda d1: "Hello World"
print(r(d1))

Output
Hello World
```

```
# without lambda expression
def d1(a):
    print(a)
d1(5)

Output
5

# with lambda expression
r = lambda a : a
print(r(5))

Output
5
```

```
#Without lambda expression addition
def d1(a, b):
   return a+b
r = d1(5,10)
print(r)
Output
15

#With lambda expression addition
#lambda argument: expression
r = lambda a, b : a+b
print(r(5,10))
Output
15
```

```
#Saving lambda function into a variable
square = lambda a, b : a+b
print(type(square)) # <class 'function'>
result = square(10,5)
print(type(result)) # <class 'int'>
print(result) # 15
print(type(result)) # <class 'int'>
```

```
# passing multiple expressions
r = lambda a, b, c, d: (a+b, a-b, c*d, c//d)
a, b, c, d = r(5,10,50,10)
print(a, b, c, d)

Output
15 -5 500 5
```

```
# passing default value
r = lambda a, b=5: a+b
print(r(10))

Output
15
```

Note:
Here default value is b = 5
If we are not passing any value for b, it will pass that value

```
# passing default value
r1 = lambda a=5, b=5: a+b
print(r1(10))

Output
15
```

Note:
a=5 is default value, r1(10) will override the a=5 default value
The expression a+b is equivalent to 10+5 = 15

```
# Nested Lambda Function
l = lambda a=10: (lambda b : a+b)
print(l())
 # <function <lambda>.<locals>.<lambda> at 0x00000036FF7711F0>
result = l()
print(result(5))
 # 15

# Nested Function
def outerFunction(a=10):
    def innerFunction(b):
        return a+b
    return innerFunction

o = outerFunction()
print(o)
# <function outerFunction.<locals>.innerFunction at 0x00000004DF361310>
print(o(5))
# 15
```