Lambda functions are used when:

- you need a function for a short period of time.
- you want to pass a function as an argument to higher-order functions.

The lambda operator cannot have any statements and it returns a function object that we can assign to any variable.

The lambda's general form :

any number of arguments

but only one expression

The expression is evaluated and returned

lambda *argument1, argument2 , ... argument N* : *expression using arguments*

```
>>> def func(x, y, z):
        return x + y + z
>>> func(2, 3, 4)
9
```

```
>>> f = lambda x, y, z: x + y + z
>>> f(2, 3, 4)
9
```

This example uses a lambda expression to return a function:

```
def makeIncrementor(n):
        return lambda x:x+n
f=makeIncrementor(42)
print(f(1))
>>43
```

- Lambda functions can be used along with built-in functions like **filter(), map() and reduce().**

- **Reduce** is a really useful function for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list. For example, if you wanted to compute the product of a list of integers. So the normal way you might go about doing this task in python is using a basic for loop, but let's try it with reduce:

```
from functools import reduce
product = reduce((lambda x, y: x * y), [1, 2, 3, 4])
# Output: 24
```