***Lambda***

“***Everything in python is an object***” – as we discussed in our very first notes. Just keep this in mind.

Python supports the creation of anonymous functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda".

In other words, python let you to create a function without function name but which can be assigned a variable and called.

Let’s see the below code which is in normal function definition,

def f (x):

return x\*\*2

print f(8)

This is a function to return square if given number, so output will be,

64

Let’s see a same function using lambda,

g = lambda x: x\*\*2

print g(8)

and output will be,

64

Bothe the example does the same job but lambda is not having a name or return statement. Lambda function will always have an expression which will be returned automatically

Lambda function gets created at runtime and where as normal function created during compilation.

Let’s see another little more complex scenario. Here we are going to have the lambda inside a normal function (similar to closure or generator we discussed in last document)

def make\_incrementor (n):

return lambda x: x + n

f = make\_incrementor(2)

g = make\_incrementor(2)

print f(3)

print g(2)

In the above example, **make\_incrementor(2)** sets the n value (we can call as base value) and whereas **f(3)** assigns x value (we can call it as increment value) and output will be like,

5

4

Main purpose of the lambda is writing a small recallable/ recursive functionalities and to reduce the line of code.

One last example:

In this one we have a sentence and split them into words and returns each words length.

sentence = 'need atleast two hands to make sound'

words = sentence.split()

print words

lengths = map(lambda word: len(word), words)

print lengths

and output will be like,

['need', 'atleast', 'two', 'hands', 'to', 'make', 'sound']

[4, 7, 3, 5, 2, 4, 5]

Note: I hope you went through map and zip function as an exercise at the end of previous document.