Python Lambda

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

Syntax

```
lambda arguments : expression
```

The expression is executed and the result is returned:

A lambda function that adds 10 to the number passed in as an argument, and print the result:

```
x = lambda a : a + 10
print(x(5))
```

```
Output: 15
```

Lambda functions can take any number of arguments:

A lambda function that multiplies argument a with argument b and print the result:

```
x = lambda a, b : a * b
print(x(5, 6))
```

```
Output: 30
```

A lambda function that sums argument a, b, and c and print the result:

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

Output: 13

Why Use Lambda Functions?

The power of lambda is better shown when you use them as an anonymous function inside another function.

Say you have a function definition that takes one argument, and that argument will be multiplied with an unknown number:

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

Use that function definition to make a function that always doubles the number you send in:

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

mydoubler = myfunc(2)

print(mydoubler(11))
```

Output: 22

Or, use the same function definition to make a function that always *triples* the number you send in:

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

mytripler = myfunc(3)

print(mytripler(11))
```

Output: 33

Or, use the same function definition to make both functions, in the same program:

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

mydoubler = myfunc(2)
mytripler = myfunc(3)

print(mydoubler(11))
print(mytripler(11))
```

```
Output:
22
33
```

Note: Use lambda functions when an anonymous function is required for a short period of time.