```
60

50
```

The Anonymous Functions

These functions are called anonymous because they are not declared in the standard manner by using the *def* keyword. You can use the *lambda* keyword to create small anonymous functions.

- Lambda forms can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.
- An anonymous function cannot be a direct call to print because lambda requires an expression.
- Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.
- Although it appears that lambda's are a one-line version of a function, they are not equivalent to inline statements in C or C++, whose purpose is by passing function stack allocation during invocation for performance reasons.

Syntax

The syntax of *lambda* functions contains only a single statement, which is as follows:

```
lambda [arg1 [,arg2,.....argn]]:expression
```

Following is the example to show how *lambda* form of function works:

```
#!/usr/bin/python


# Function definition is here

sum = lambda arg1, arg2: arg1 + arg2;


# Now you can call sum as a function

print "Value of total : ", sum( 10, 20 )
```