Parameters	Variables that are listed within the parentheses of a function header
Function Body	The block of statements / indented statements beneath function header that defines the action performed by the function The function body may or may not return any value. A function returns a value through a return statement. e.g. above given *sum()* is returning a value stored in variable *res* but function *greet()* is not returning a value A function not returning any value can still have a *return* statement without any expression or value. Examples below will make it clearer
Indentation	The blank space in the beginning of a statement (convention is four spaces) within a block. All statements within same block have same indentation

Let us now have a look at some more function definitions

```
# Sample Code 1
def sumOf3Multiples1( n ) :
    s = n * 1 + n * 2 + n * 3
    return s

# Sample Code 2
def sumOf3Multiples2( n ) :
    s = n * 1 + n * 2 + n * 3
    print(s)
```

*Both these functions are doing the same thing BUT first one is **returning** the computed value using return statement and second function is **printing** the computed value using print( ) statement*

Consider some more function definitions:

```
# Sample Code 3
def areaOfSquare ( a ) :
    return a * a

# Sample Code 4
def areaOfRectangle ( a, b) :
    return a * b

# Sample Code 5
def perimeterCircle( r ) :
    return (2 * 3.1459 * r)

# Sample Code 6
def perimeterRectangle( l, b) :
    return 2 * ( l + b )

# Sample Code 7
def Quote( ) :
    print("\t Quote of the Day")
    print("Act Without Expectation!!")
    print("\t -Lao Tzu")
```

For all these function definitions, try identifying their parts. (Not as an exercise, just d casually, while reading them.)

A function definition defines a user-defined object function. The function definition does execute the function body; this gets executed only when the function is called or invoked. In following lines, we are discussing how to invoke functions, but before that it would be usef know the basic structure of a Python program.

Structure of a Python Program

In a Python program, generally all function definitions are given at the top followed by statements which are not part of any functions. These statements are not indented at all. These are often called from the top-level statements (the ones with no indentation). The Python interpreter starts the execution of a program/script from the top-level statements. The top level statements are part of the main program. Internally Python gives a special name to top-level statements as *__main__* .

NOTE

By default, Python names the segment with top-level statements (main program) as *__main__*.

The structure of a Python program is generally like the one shown below :

```
def function1( ) :
    :
def function2( ) :
    :
def function3( ) :
    :
    :
#top-level statements here
statement1
statement2
    :
```

Python names the segment with top-level statements (no identation) as __main__. Python begins execution of a program from the top-level statements i.e., from __main__.

Python stores this name in a built-in variable called **__name__** (i.e., you need not declare this variable ; you can directly use it). You can see it yourself. In the ***__main__*** segment of your program if you give a statement like :

```
print( __name__)
```

Python will show you this name. For example, run the following code and see it yourself.

```
def greet( ) :
    print("Hi there!")

print("At the top-most level right now")
print("Inside" , __name__)
```

The top-level statements, i.e., the __main__ segment of this Python program. Python will start execution of this program from the segment.

Upon executing above program, Python will display :

```
At the top-most level right now
Inside __main__
```

Notice word ' __main__ ' in the output by Python interpreter. This is the result of statement : print(.... __name__)