As you can make out that the above function's name is **cube( )** and it takes one argument. Now its function call statement(s) would be similar to the ones shown below :

(*i*) Passing literal as argument in function call

```
cube(4)          # it would pass value as 4 to argument x
```

(*ii*) Passing variable as argument in function call

```
num = 10
cube(num)        # it would pass value as variable num to argument x
```

(*iii*) taking input and passing the input as argument in function call

```
mynum = int (input ( "Enter a number :" ))
cube(mynum)      # it would pass value as variable mynum to argument x
```

(*iv*) using function call inside another statement

```
print(cube(3))   # cube(3) will first get the computed result
                 # which will be then printed
```

(*v*) using function call inside expression

```
double OfCube = 2 * cube(6)
                # function call's result will be multiplied with 2
```

NOTE

The syntax of the function call is very similar to that of the declaration, except that the key word ***def*** and colon (:) are missing.

3.2.2 Python Function Types

Python comes preloaded with many *function-definitions* that you can use as per your needs. You can even create new functions. Broadly, Python functions can belong to one of the following *three* categories :

1. Built-in functions — These are pre-defined functions and are always available for use. You hav used some of them – *len( ), type( ), int( ), input( )* etc.

2. Functions defined in modules — These functions are pre-defined in particular modules and can only used when the corresponding module is *imported*. *For example*, if you w to use pre-defined functions inside a module, say **sin( )**, you need to *import* the module *math* (that contains definition of **sin( )**) in your prog

3. User defined functions — These are defined by the programmer. As programmers you can your own functions.

In this chapter, you will learn to write your own Python functions ar them in your programs.

PriP STRUCTURE OF FUNCTIONS — Progress In Python

This PriP session is aimed at making anatomy of Python functions clear to you. You'll be required to practice about structure of Functions.

Please check the practical component-book – Progress in Computer Science with Python

3.3 DEFINING FUNCTIONS IN PYTHON

As you know that we write programs to do certain things. Functions can be thought of as key-doers within a program. A function once defined can be invoked as many times as needed by using its name, without having to rewrite its code.

In the following lines, we are about to give the general form *i.e.*, syntax of writing function code in Python. Before we do that, just remember these things.

In a syntax language :

- item(s) inside angle brackets < > has to be provided by the programmer.
- item(s) inside square brackets [ ] is optional, *i.e.*, can be omitted.
- items/words/punctuators outside < > and [ ] have to be written as specified.

A function in Python is defined as per following general format :

```
def <function name> ( [parameters] ) :
    [ " " "<function's docstring>" " " ]
    <statement>
    [<statement>]
    ⋮
```

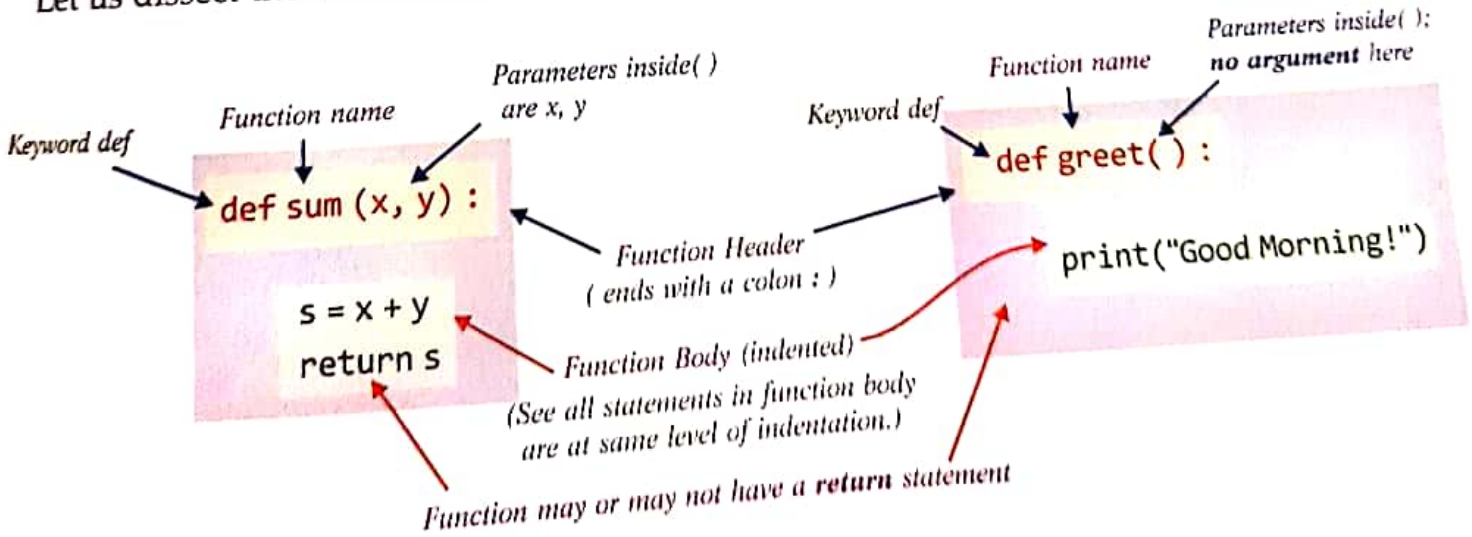
For example, consider some function definitions given below:

```
def sum (x, y) :
    s = x + y
    return s
```

Or

```
def greet( ):
    print("Good Morning!")
```

Though you know about various elements in a function-definition, still let us talk about it again. Let us dissect these functions' definitions to know about various components.



Let us define these terms formally :

Function Header The first line of function definition that begins with keyword *def* and ends with a col (:), specifies the *name of the function* and its *parameters*