



# **BACHELOR OF COMPUTER APPLICATIONS SEMESTER 5**

**DCA3104  
PYTHON PROGRAMMING**

# Unit 10

## Methods

### Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	<a href="#">Introduction</a>	-	-	3-4
	1.1 <a href="#">Learning Objectives</a>	-	-	
2	<a href="#">Introduction to Functions</a>	-	<a href="#">1</a>	5-6
3	<a href="#">Types of Functions</a>	-	<a href="#">2</a>	6-10
4	<a href="#">Benefits of Function in Python</a>	-	-	11
5	<a href="#">Creating a Function</a>	<a href="#">1</a>	<a href="#">3, I</a>	11-13
6	<a href="#">Calling of Function</a>	<a href="#">2</a>	<a href="#">4</a>	14-15
7	<a href="#">Return Statement of Function</a>	-	<a href="#">5</a>	15-17
8	<a href="#">Parameters in Functions</a>	-	<a href="#">6</a>	18-19
9	<a href="#">Call by Reference</a>	-	<a href="#">7</a>	20
10	<a href="#">Variable-length Functions</a>	<a href="#">3</a>	<a href="#">8, II</a>	21-22
11	<a href="#">Keyword Arguments</a>	-	-	23-24
12	<a href="#">Summary</a>	-	-	25-26
13	<a href="#">Glossary</a>	-	-	26
14	<a href="#">Case Study</a>	-	-	27
	<a href="#">Terminal Questions</a>			28-32
15	14.1 <a href="#">Answer keys</a>	-	-	
	<a href="#">Suggested Books and e-References</a>	-	-	32

## 1. INTRODUCTION

Python is considered one of the easiest programming languages that can be used for software development, web development, mathematics, data handling, etc. It is one of the widely used programming languages globally on multiple platforms such as Windows, Linux, Mac, Raspberry Pi, etc.

In the previous unit, we learned about tuple, sets, and dictionaries in Python. A tuple is a collection of values assigned to an ordered variable and cannot be changed once formed. Tuple can also contain duplicate values. An example of a tuple is given below.

```
myTuple = ("python", "java", "python", "html")
```

On the other hand, a set is a collection of values of data types that is both unordered and unindexed. It is also unchangeable but does not allow duplicate values like in a tuple, and if present, it will simply ignore it. An example for a set is given below.

```
mySet = {"monitor", "keyboard", "printer", "scanner"}
```

Note that the values will be displayed in the output at random positions every time because a set is unordered. A dictionary stores values in pairs. It means it stores a key and a value for that key together and not one at a time. It is ordered, can be changed and does not allow duplicate values or pairs. An example for the same is given below.

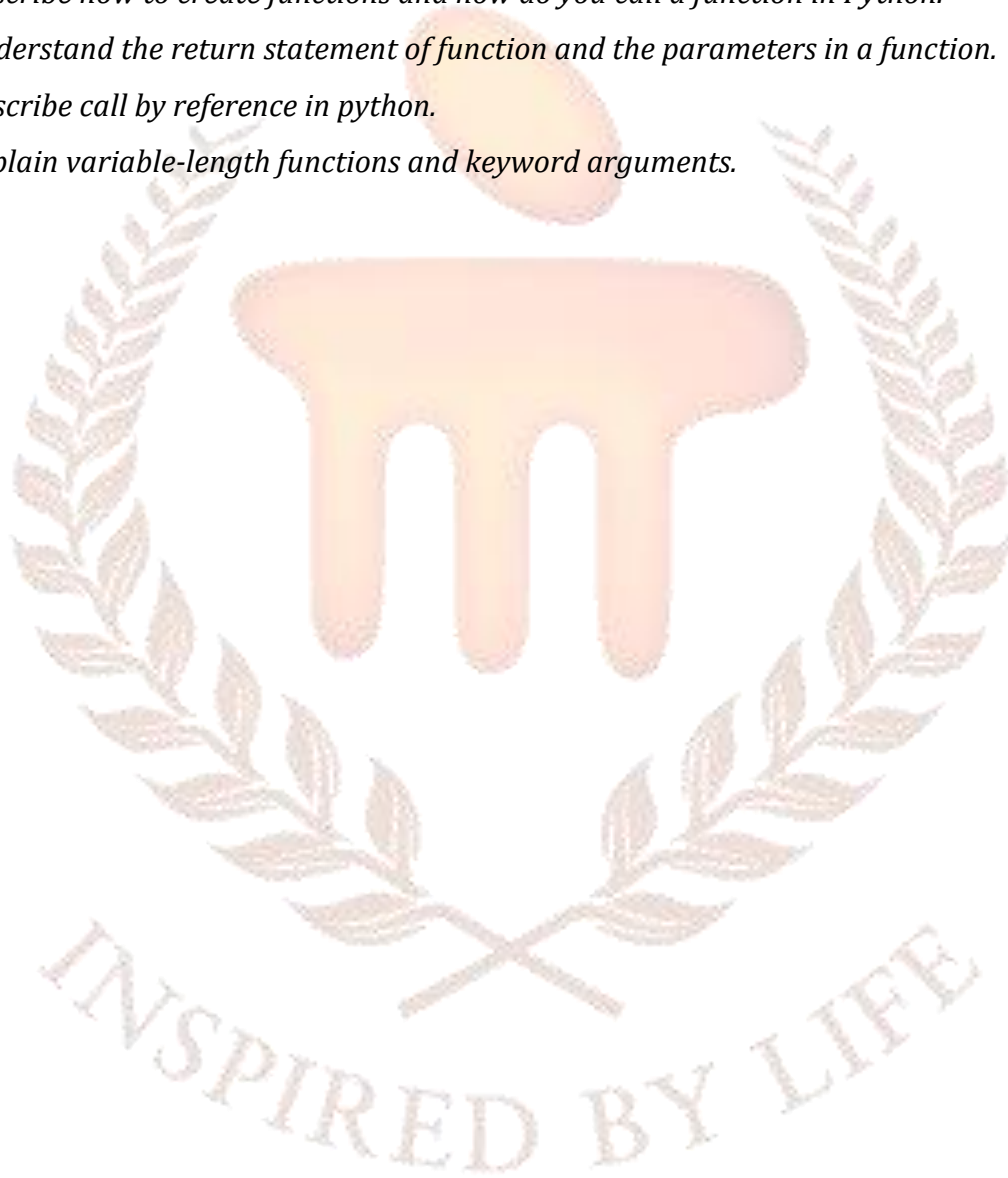
```
myDictionary = {1: 'python', 2: [4, 5, 6]}
```

In this unit, we will learn about functions in Python in detail. We will go through a brief introduction to functions, then learn about the types of functions and benefits of functions in Python. Furthermore, we will understand the process of creating a function and calling a function to use in a code. Moving on, we will understand the return statement of functions and parameters used in a function. Call by reference and variable-length functions will be discussed and the unit will end with keyword arguments.

## 1.1 Learning Objectives

*After studying the chapter, you will be able to:*

- ❖ *Describe the concept of functions and types of functions present.*
- ❖ *Explain the benefits of functions in Python.*
- ❖ *Describe how to create functions and how do you call a function in Python.*
- ❖ *Understand the return statement of function and the parameters in a function.*
- ❖ *Describe call by reference in python.*
- ❖ *Explain variable-length functions and keyword arguments.*



## 2. INTRODUCTION TO FUNCTIONS

Repetition in any type of code for any objective is very problematic, both for the programmer and the reader. Repetition of codes in any program makes the program unclear and rough. Even though the code works and completes the objective for which it was created, it still causes many problems and errors due to its unclear structure.

### STUDY NOTE

Functions must be created at the beginning of the program and before using it. Otherwise, Python will generate an error and display that no such function has been defined.

A structured code not only takes less space but is also very easy to read and interpret and generates fewer errors to deal with. Therefore, it was crucial to introduce a method that eliminates the use of codes multiple times in any program. Functions are created for this purpose. Once you understand something that has to be repeated multiple times in a program, you can simply define a function and call it multiple times in a program whenever you want that activity to be performed.

Efficient use of functions in a code can increase the effectiveness of that program. Whenever we set activities to be executed from time to time in a program, we create a function and assign that set of values to it along with a name. This way, whenever we need those sets of actions to be executed, we have to call that function and let it do the work.

Python programming language already has numerous built-in functions in its huge standard library such as `sort()`, `title()`, `print()`, `range()`, `len()`, and much more that makes programming fun and easy. You can also create your functions using the keyword `def` and assign the set of actions to it that you want to be executed.

The key to solving complex programs is to break them into smaller parts and tackle one part at a time. This method allows the programmer to focus on one task at a time and organize their code by creating small functions to carry out the tasks in an ordered manner. This eliminates the possibility of having a complex and difficult solution to an already complex problem. Functions make coding easier for programmers. This also ensures less error than usual.



**SELF-ASSESSMENT QUESTIONS - 1**

1. Functions that are already present in Python's standard library is called \_\_\_\_\_ functions.
2. Functions makes a code \_\_\_\_\_.
3. Functions make coding difficult and confusing. [True/False]
4. Repetition causes make a code lengthy and confusing. (True/False)
5. Functions can be defined as
  - a) Set of actions
  - b) Combination of commands set by the user
  - c) Both a and b
  - d) None of these

**3. TYPES OF FUNCTIONS**

As we learned in the above section, functions are meant to carry out specific tasks in a given program when called for execution. They might need or do not need input at all. The function can either return a single value or multiple values. All this depends upon the type of function used in the program. There are three types of functions in Python, which are as follows:

- **Built-in functions:** These types of functions are already built into Python. There are numerous functions pre-defined for the benefit of the programmers. Some examples for built-in functions in Python are `print()`, `help()`, `len()`, `max()`, `min()`, `format()`, `input()`, `range()`, `type()`, `map()`, `reduce()`, and `filter()`.
- `'print()'`: This function is used to output data to the standard output device (screen).  
`print("Hello, World!")`  
This will output `'Hello, World!'` to the console.
- `'help()'`: This function is used to display the documentation of modules, functions, classes, keywords, etc.  
`help(print)`  
This will display the documentation of the `'print'` function.

**STUDY NOTE**

Functions are meant to carry out specific tasks. They can return multiple values at once.

- `len()`: This function returns the number of items (length) in an object. The argument can be a sequence (such as a string, bytes, tuple, list, or range) or a collection (such as a dictionary, set, or frozen set).

```
print(len("Hello, World!"))
```

This will output `13`.

- `max()`: This function returns the largest item in an iterable or the largest of two or more arguments.

```
print(max(1, 2, 3, 4, 5))
```

This will output `5`.

- `min()`: This function returns the smallest item in an iterable or the smallest of two or more arguments.

```
print(min(1, 2, 3, 4, 5))
```

This will output `1`.

- `format()`: This function formats specified values and insert them inside the string's placeholder. The placeholder is defined using curly braces: `{}`.

```
print("Hello, {}".format("World"))
```

This will output `Hello, World!`.

- `input()`: This function allows user input.

```
name = input("Enter your name: ")
```

```
print("Hello, {}".format(name))
```

This program will ask for your name and then output `Hello, <Your Name>!`.

- `range()`: This function returns a sequence of numbers, starting from 0 by default, and increments by 1 (also by default), and stops before a specified number.

```
for i in range(5):
```

```
print(i)
```

This will output the numbers `0` through `4`.

- `type()`: This function returns the type of the specified object.

```
print(type(123))
```

This will output `<class 'int'>`.

- `map()`: This function applies a given function to all items in an input list.

```
numbers = [1, 2, 3, 4, 5]
```

```
squared = list(map(lambda x: x ** 2, numbers))
```

```
print(squared)
```

This will output `[1, 4, 9, 16, 25]`, which are the squares of the numbers in the original list.

- `reduce()`: This function is a part of `functools` module. `reduce()` function is for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list.

```
from functools import reduce
```

```
numbers = [1, 2, 3, 4, 5]
```

```
product = reduce((lambda x, y: x * y), numbers)
```

```
print(product)
```

This will output `120`, which is the product of the numbers in the list.

- `filter()`: This function constructs an iterator from elements of an iterable for which a function returns true.

```
numbers = [1, 2, 3, 4, 5, 6]
```

```
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
```



```
print(even_numbers)
```

This will output `[2, 4, 6]`, which are the even numbers from the original list.

- **User-Defined functions:** These are the functions that the programmers create then and there at the start of programming to solve the present problem or perform any activity.
- **Anonymous functions:** Anonymous functions in Python are known as "lambda" functions. They are called "anonymous" because they are functions that are defined without a name. While normal functions are defined using the `def` keyword in Python, anonymous functions are defined using the `lambda` keyword.

Here is the syntax of lambda functions:

```
``python
lambda arguments: expression
``
```

The `lambda` keyword is followed by one or more arguments, separated by commas, then a colon `:`, and then an expression. The expression is what the function returns.

Let's expand on this with some examples:

**\*\*Example 1: A simple lambda function that adds two numbers.\*\***

```
add = lambda x, y: x + y
print(add(5, 3)) # Output: 8
```

In this example, `x` and `y` are arguments to the function, and `x + y` is the expression. The lambda function is assigned to the variable `add`, and it can be called just like a normal function.

**\*\*Example 2: A lambda function that returns the square of a number.\*\***

```
square = lambda x: x ** 2
print(square(5)) # Output: 25
```

In this example, `x` is the argument and `x \*\* 2` (x squared) is the expression.

Lambda functions are used when you need a function for a short period of time, usually for one-time usage. They are often used with functions like ``map()``, ``filter()``, and ``reduce()``, which take a function as an argument.

For instance, you can use a lambda function to square all numbers in a list:

```
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x ** 2, numbers))
print(squared) # Output: [1, 4, 9, 16, 25]
```

In this example, the ``map()`` function applies the lambda function to each element of the list ``numbers``, and returns a new list ``squared`` that contains the squares of the numbers.

### SELF-ASSESSMENT QUESTIONS – 2

6. Functions created by programmers for their use is called \_\_\_\_\_ functions.
7. Anonymous functions are also called \_\_\_\_\_ functions.
8. Name of every function is unique. [True/False]
9. A function can only return single value. [True/False]
10. Which of the following is not a built-in function?
  - a) Print()
  - b) Len()
  - c) Maximum()
  - d) Format()

## 4. BENEFITS OF FUNCTION IN PYTHON

Several benefits of functions in Python is given below:

- Functions eliminate the repetition of code.
- It makes the codes reusable.
- It makes the code structured and manageable.
- Large work can be done in a few lines of code and using a function.
- The generation of errors is reduced with the length of the program.
- Codes are much easier to read if they are structured and simple.
- Clarity if the code is greatly increased.
- Functions make solving complex problems easy when divided and tackled in parts.

### STUDY NOTE

If we have to make changes in the function's output, then we only have to modify the function and the effect will be seen.

## 5. CREATING A FUNCTION

After knowing the concept, types, and benefits of the functions, let's move on to the next section. By creating a function, we mean defining the function. To define a function, the `def` keyword is used. It marks the start of the function header. You have to take care of indentation while defining the function in Python. It will immediately raise an error if the indentation of the function is off even a bit.

### STUDY NOTE

Note that indentation is very important in Python. The body of the functions should be indented properly or else Python will raise indentation error.

Whenever you create or define a function, you have to name it to call it in the program. The name of the function has to be unique because this name will identify the function in the program. The naming of the functions has some rules that you have to follow. It is easy to remember once you get the hang of it.

- The keywords defined in Python cannot be used as a name for any function.
- Spaces are not allowed in naming a function.
- The first character can be the uppercase alphabet, lowercase alphabet, or “\_” character.
- The next characters can be the combination of uppercase, lowercase alphabet, numbers, or “\_” character.

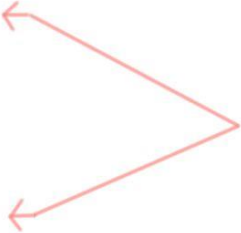
- Remember that special characters such as "\$," "#," "%," "!", "@" are not allowed in the naming of the function.

The next important thing in defining a function is the parameter or the input which is passed or given to the function. This is optional. Then comes the ":" colon that marks the end of the function header.

Even though you define the function, explaining what it does exactly is very much appreciated in a single line. This you can do with the help of documented string (docstring) which is optional too.

Now comes the body where you can write some python statements to allocate some action to the function. Remember that all the statements inside the body of the function must have the same indentation level. Python places heavy emphasis on indentation therefore, you should take extra care on this part.

```
>>> def absolute_value(x):  
...     if x < 0:  
...         return -x  
...     if x > 0:  
...         return x  
...  
>>>
```



Alternative  
conditional

*Source: ProgramBitz.com*

**Fig 1:** Parts of function

A return statement to get a return value from the function is also optional but is very useful.

The syntax of a function is given below.

#defining a function

```
def func_name(par1, par2):  
    #define the action of the function(optional)  
    statement(s)
```

Example:

```
#def func_new1():  
    print('I am learning Python.')
```

### Activity 1

Define a function that can accept the name, marks, and roll numbers of 30 students in a class and display the data of each student in a single line, one by one. Discuss the parameters used in defining the function.

### SELF-ASSESSMENT QUESTIONS - 3

16. The keyword used for defining a function is \_\_\_\_\_.
17. This marks the end of the function header \_\_\_\_\_.
18. Indentation is not necessary in defining a function. [True/False]
19. A Docstring is optional in defining a function. [True/False]
20. Which one of the following is the correct syntax of function?
  - A. 

```
def 1func_name(par1, par2):  
    #define the action of the function(optional)  
    statement(s)
```
  - B. 

```
def func_name(par1, par2):  
    #define the action of the function(optional)  
    statement(s)
```
  - C. 

```
def func_name(par1, par2):  
    #define the action of the function(optional)  
    statement(s)
```
  - D. 

```
def func_name(par1, par2):  
    #define the action of the function  
    statement(s);
```



## 6. CALLING OF FUNCTION

When the time of executing a function that you have created comes, you call it. Calling a function is directly employed to carry out the tasks you have assigned or created in it. When you call any function, you can say that the program's control has been transferred to the function for that time being. It means that the execution of the program is in the hands of the function. After executing the tasks assigned or defined in the function, it will return to the next line after the function call.

To call any function, you must use the unique name used to define the function. Simply type the name of the function and provide values of the parameters, if any. This will prompt the function to take control of the program for that time and execute the commands given in the body of the function.

The syntax for calling a function is given below:

```
#calling a function defined in the previous section  
func_name(val1, val2)
```

The expression present inside the statement gets assessed and then a value is returned when the function is called. This will activate the series of commands associated with the function created with this name.

### STUDY NOTE

The function defined by you will only work when it is defined before the function call or before it is used in the program. So it is very crucial to define the function before you call it.

Example:

```
#defining of function  
def func_new():  
    print('I was invisible and now I am visible.')
```

```
#calling of function  
func_new()
```

Output is as follows:  
I was invisible and now I am visible.

**SELF-ASSESSMENT QUESTIONS – 4**

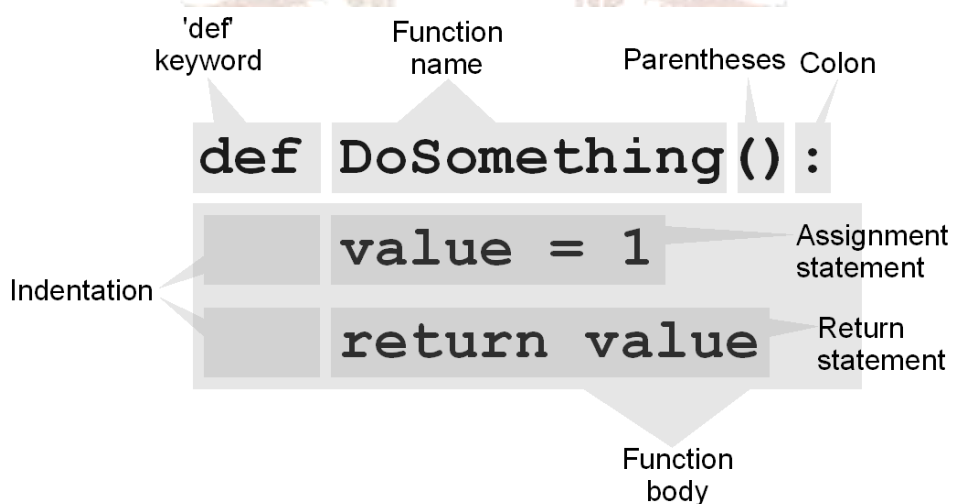
21. To call any function, the \_\_\_\_\_ name of the function should be written that was used to define it.
22. Calling a function means accessing the built-in functions in the Python library. [True/False]
23. Calling a function can be done by using the def keyword. [True/False]
24. Calling a function means accessing the lambda functions. [True/False]
25. Calling of a function does which of the following things?
  - a) Executes the commands defined in the function
  - b) Passes the arguments into the function for evaluation
  - c) Erases the function
  - d) Both a and b

**7. RETURN STATEMENT OF FUNCTION**

Return statement in Python for a function is the way to exit the function when needed and go back to the initial call line. The syntax for return is given below.

```
#writing the return statement  
return [expression]
```

The use of a return statement in defining a function is optional and depends on the tasks required to be done.



*Source: FunctionBits.com*

**Fig 2:** Brief of Return statement

The return statement can be used several times depending on the complexity of the problem.

An example is given below for you to understand the use of the return statement better.

**Example no.1:**

```
#when no return is given
def praise(name):
    """ this function praises the person whose name is
    passed as a parameter"""
    print("Well done" + name + "!")
#calling the function to praise 'Tanmay.'
praise("Tanmay")

#The output of the given code is as follows:
Well done Tanmay!
```

**STUDY NOTE**

The expression present inside the statement gets assessed and then a value is returned when the function is called. In case of no return, the command line goes to the next line after executing the body of the function.

**Example no.2:**

```
#when return statement is present
def mod(number):
    """ this function returns the mod values of any number"""
    if number >= 0:
        return number
    else :
        return -number
print(mod(4))
print(mod(-5))
#calling the mod function
mod(4)
mod(-5)

#The output of the above code is as follows:
4
5
```

**SELF-ASSESSMENT QUESTIONS - 5**

26. The return statement will be initiated with \_\_\_\_\_ keyword.
27. The statement that is evaluated in the return command is called \_\_\_\_\_.
28. The return statement gives only boolean values. [True/False]
29. The execution of the function continues after return statement. [True/False]
30. Which of the following is the correct syntax of return statement?
- a) Return: [expression]
  - b) Return [expression];
  - c) Return [expression]
  - d) Return(expression)



## 8. PARAMETERS IN FUNCTIONS

The parameter is the unbound variable used in defining the function. People often get confused about the difference between the parameter and the argument. If you define a function like `def func(name)`, then the name is the parameter, whereas when you call the function, then the actual values used for the parameter is known as the argument.

### STUDY NOTE

Note that all the commands written after the return statement in any function will never get executed because the execution of the function stops right after it executes the return statement.

Parameters can appear in the function's body, whereas when the function is called, then the argument or the specific value given by the programmer takes its place in processing for the output.

There is no rule to limit the number of parameters to be present in a function. One can have as many parameters in the function as required. The main function of parameter variables is to transfer arguments in the function. There are two types of parameters in functions:

- **Input parameter:** They transfer the absolute values given by the user to the functions.
- **Output/Return parameter:** They return multiple values from the function but are not used very frequently because they confuse.

Several parameters or a parameter list explain the types of values or arguments a function will receive.

Example:

```
#defining a function
def func_new1(obj):
    print("I have a" + obj)
#calling the function
func_new1("computer")

#Output of the above code :
I have a computer
```



**SELF-ASSESSMENT QUESTIONS – 6**

31. \_\_\_\_\_ type of parameter transfer the absolute values given by the user to the functions.
32. \_\_\_\_\_ type of parameter return multiple values from the function but are not used very frequently because they cause confusion.
33. Parameters are the absolute value passed into a function. [True/False]
34. Only one parameter can be introduced at a time. [True/False]
35. What is the main function of the parameter?
  - a) to pass the arguments into the functions.
  - b) to create memory space for the actual value
  - c) to help carry out the tasks assigned to the function
  - d) to give the function a unique name



## 9. CALL BY REFERENCE

The Python programming language uses the 'Call by Object Reference' or 'Call by Assignment'. When mutable arguments are passed into the functions, meaning that the values can be changed in the functions and the change will be reflected in the output, then it is called call by reference. Let us see an example for the same to get a clear idea of the concept.

Example:

```
#demonstration of call by reference
def add(list):
    mylist.append(10)
    print("List inside functions:",list)
mylist=[2,4,6,8]
add(mylist)
print("List outside the function:",mylist)

#The output is as follows:
List inside the function: [2,4,6,8,10]
List outside the function: [2,4,6,8,10]
```

### STUDY NOTE

When you pass arguments that are immutable, meaning that the arguments cannot be changed then it is called call by value. executes the return statement.

You already know that we can add items to a list after it is made. Thus we used to call by reference to add the number. The changed list was reflected in the output.

### SELF-ASSESSMENT QUESTIONS - 7

36. The Python programming language uses the \_\_\_\_\_.
37. When mutable arguments are passed into the function then it is called \_\_\_\_\_.
38. Passing immutable arguments is known as call by reference. [True/False]
39. The change is not reflected in the output in the call by reference method. [True/False]
40. How many arguments can be passed at a time?
  - a) All the arguments
  - b) One
  - c) Two
  - d) None at all

## 10. VARIABLE-LENGTH FUNCTIONS

You may need to put variable amounts of argument in a function that you are defining. A particular operator can be used in declaring a parameter that allows a variable number of arguments. That operator is an asterisk "\*." There are two types of parameters in a function that allows a variable number of arguments. One will accept the numerous variable arguments that will be eventually stored in a tuple. The other parameter will accept the variable number of arguments and the key for every value given as input when the function is called. The other one stores the keys and the values in a dictionary.

Let us talk about the first type here, and we can discuss the second type of parameter in the next section. In the first type, we use to put the asterisk in front of the parameter or use the standard parameter \*args to accept variable amounts of arguments. The syntax for this can be seen below.

### STUDY NOTE

Using the \*args does not mean that you cannot use the regular parameters alongside with it. You can use it and the preceding arguments will be evaluated first.

Example:

```
def func_name( *args ):
```

```
    #body of the function
```

Let's see an example to understand this concept.

```
def biggest_number(*num):
```

```
    #this function returns the biggest number
```

```
    return max(num)
```

```
#calling the function
```

```
print(biggest_number(33,66))
```

```
print(biggest_number(7,99,82,44))
```

```
print(biggest_number(55,93,30,65,72,32))
```

```
#The output is as follows:
```

```
66
```

```
99
```

```
93
```

## Activity 2

Create a list that includes marks obtained by a student in various subjects. In the list, try accessing various elements through indexing and slicing. Use negative indices as well and find if all combinations yield the desired result. Take note where you find that the slicing does not work as expected.

### SELF-ASSESSMENT QUESTIONS – 8

41. The operator used in variable number of arguments is \_\_\_\_\_.
42. \*args have to be used as a parameter to use a variable number of arguments in a function. [True/False]
43. \*args takes three arguments at a time. [True/False]
44. Triple asterisk is used to denote the variable number of arguments [True/False]
45. The variable number of arguments are stored in
  - a) List
  - b) Set
  - c) Dictionary
  - d) Tuple

## 11. KEYWORD ARGUMENTS

The Keyword Arguments or **\*\*kwargs** is the second type of parameter that we discussed in the previous section. By using this parameter, we can accept both the variable amounts of arguments and the key to every argument. Only arguments will not be allowed into the function if this type of parameter is used. A key to each argument should also be given, and else it will not be accepted. Take a look at the example for keywords arguments given below.

Example:

```
#using keyword argument
def func(name, activity):
    """this function tells which person is doing
    what at the moment"""
    print(name + "is" + activity)
#calling the function
func(name = "Arya", activity = "dancing" )
func(name = "Ananya", activity = "singing" )
func(name = "Aditya", activity = "playing" )

#The output is as follows:
Arya is dancing
Ananya is singing
Aditya is playing
```

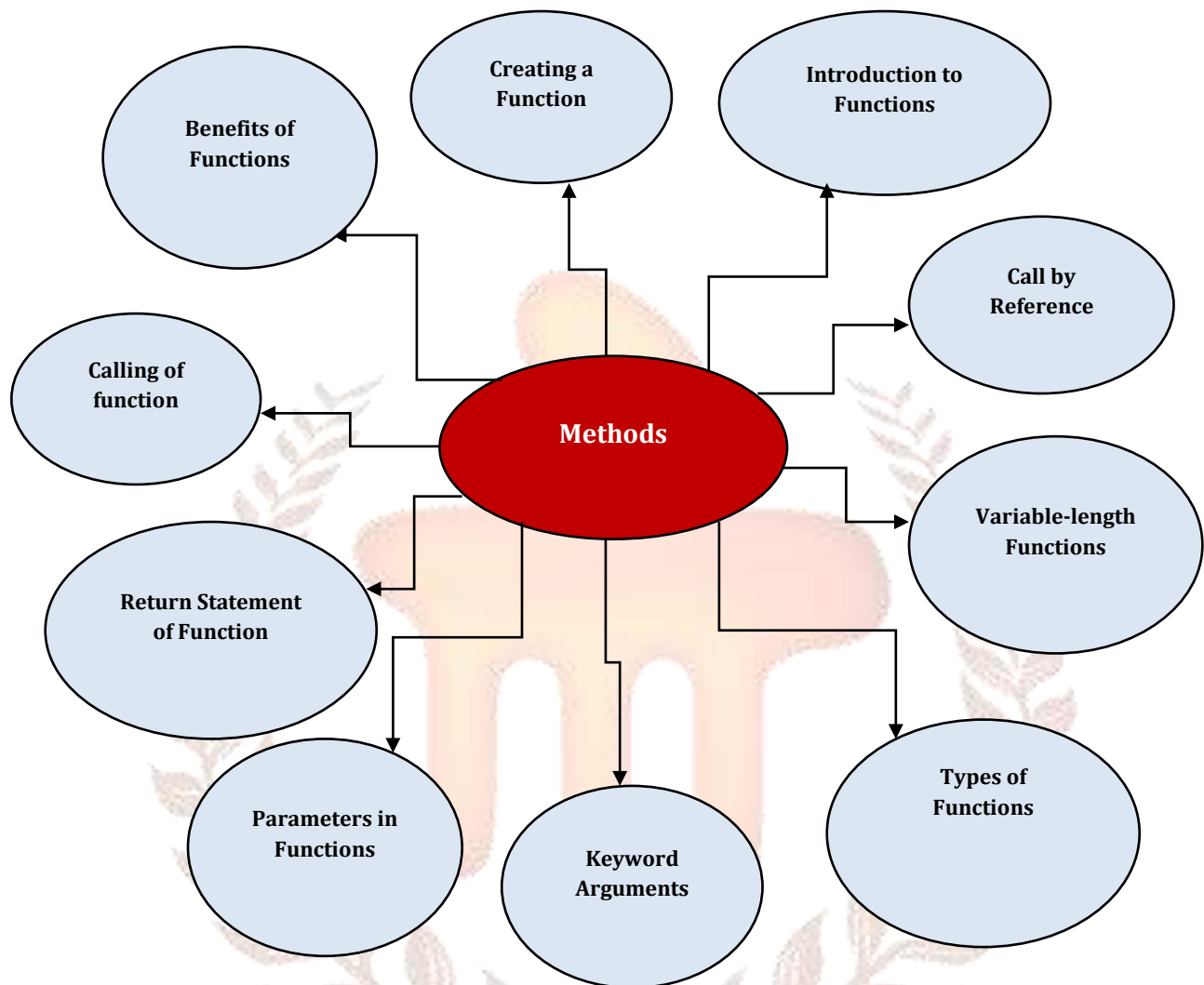
### STUDY NOTE

The position of the arguments does not matter in Python as long as the key is assigned to the relevant parameter. Python will use the relevant value according to the keys to which it is assigned.

### SELF-ASSESSMENT QUESTIONS - 9

46. The Keyword Arguments are denoted as \_\_\_\_\_.
47. The values are accepted in pairs only. [True/False]
48. The position of the arguments does not matter. [True/False]
49. The key for each argument is not necessary. [True/False]
50. The values will be accepted into the function if
  - a) Only arguments are given
  - b) Only keys for each argument are given
  - c) Both the arguments and their keys are given
  - d) None of the above





**Fig 3: Conceptual Map**

## 12. SUMMARY

- Python can be used for software development, web development, mathematics, data handling, and so on.
- A tuple is a collection of values assigned to an ordered variable and cannot be changed once formed.
- A set is a collection of values of data types that is both unordered and unindexed.
- A dictionary stores values in pairs, which means it stores a key and a value for that key together and not one at a time.
- Functions make coding easier for programmers. This also ensures less error than usual.
- Functions must be created at the beginning of the program and before using it; otherwise, Python will generate an error and display that no such function has been defined.
- Built-in functions are the types of functions already built into Python.
- User-Defined functions are the functions that the programmers create then and there at the start of programming to solve the present problem or perform any activity.
- Anonymous functions are not declared outright with the standard keyword `def` and therefore are called lambda functions.
- Functions make solving complex problems easy when divided and tackled in parts.
- Indentation is very important in functions.
- The name of the function has to be unique because this name will identify the function in the program.
- The function defined by you will only work when it is defined before the function call or before it is used in the program.
- Return statement in Python for the function is the way to exit the function when needed and go back to the initial call line.
- When mutable arguments are passed into the functions, meaning that the values can be changed in the functions and the change will be reflected in the output, then it is called call by reference.
- The parameter is the unbound variable used in defining the function.
- The variable-length argument will accept the numerous variable arguments that will be eventually stored in a list.

- The keyword arguments will accept not only the variable number of arguments, but also the key to each argument.

### 13. GLOSSARY

**Argument:** Value passed to the function when it is called.

**Call a Function:** The method to access a function in Python language

**Dictionary:** An array where keys are mapped to their values.

**Docstring:** First expression in a function that explains what does the function do.

**Expression:** Something that can be evaluated to give an output.

**Function:** A set of actions that returns something to the programmer when called.

**Indentation:** Spaces at the start of the command line.

**Keyword Arguments:** A key and value are both passed to the function.

**Lambda function:** Anonymous function in python that does not have a unique name.

**List:** An ordered collection of items that can be changed.

**Parameter:** A variable that specifies an argument.

**Return statement:** The command that returns a value after evaluation.

**Tuple:** An ordered collection of items that cannot be changed.

## 14. CASE STUDY

### Developing an app using Python to purchase groceries online

Gofer App is created to provide the users with a new and wonderful experience of purchasing groceries. the user just has a select a few options and the delivery will be made in no time.

You can approve and track your order while you sit back and relax. cancellation of the order option is also present should the user does not want to place an order any more.

The app has two branches, one in which the delivery person could pick up delivery and the second where the delivery person could purchase the groceries selected by the customer.

The apps for the user and the delivery guy were different. The user has the options to confirm or cancel after the delivery guy makes clear which of the selected groceries are available and which are not. The challenge was to relay the message in real-time and keeping the overall navigation easy for the user to have the best experience.

*Source: Indianic.com*

#### Questions:

1. Discuss the number of screens required and create a wireframe for the same app.
2. What built-in functions are used in the placing of order section?

## 15. TERMINAL QUESTIONS

### SHORT ANSWER QUESTIONS:

- Q1. Write a function such that it can accept a variable length of argument and print all arguments value.
- Q2. Write a function such that it can accept two variables and calculate the addition and subtraction of them.
- Q3. Return the largest item from the given list. List = [5,8,7,6]
- Q4. What are the benefits of keyword arguments?
- Q5. Which keyword is used to define a function?

### LONG ANSWER QUESTIONS:

- Q1. Write a Python function to sum all the numbers in a list. List : [2, 4, 9, 8, 5]
- Q2. Write a Python program to reverse a string. String: "language"
- Q3. Write a Python function to check whether a number is in a given range.
- Q4. Write a Python function to multiply all the numbers in a list.
- Q5. Write a Python function to calculate the factorial of a number (a non-negative integer).

### 15.1 Answers

#### SELF ASSESSMENT QUESTIONS

1. Built-in
2. Clear
3. FALSE
4. TRUE
5. C. both a and b
6. User-defined
7. Lambda
8. TRUE
9. FALSE
10. C. maximum()
11. Repetition
12. Reusable
13. True



- 
14. True
  15. A. Code becomes structured and more manageable.
  16. def
  17. :
  18. False
  19. True
  20. B
  21. #define the action of the function(optional)
  22. statement(s)
  23. Unique
  24. False
  25. False
  26. False
  27. D. Both a and b
  28. Return
  29. Expression
  30. False
  31. False
  32. C. Return [expression]
  33. Input
  34. Output/Return
  35. False
  36. False
  37. A. to pass the arguments into the functions.
  38. 'Call by Object Reference'
  39. Call by reference
  40. False
  41. False
  42. A. all the arguments
  43. \*
  44. True
  45. False

- 46. False
- 47. Tuple
- 48. \*\*kwargs
- 49. True
- 50. True
- 51. False
- 52. C. Both the arguments and their keys are given

### TERMINAL QUESTIONS

### SHORT ANSWER QUESTIONS

**Answer 1:**

```
def func(*args):  
    for x in args:  
        print(x)  
func(3,5,6,7)  
func(22,55,66,77,99)
```

**Answer 2:**

```
def cal(a, b):  
    return a+b, a-b  
res = cal(98, 79)  
print(res)
```

**Answer 3:**

```
List = [5,7,8,6]  
print(max(List))
```

**Answer 4:** No need not to remember the order of the arguments while using this. We can specify the values for the parameters which we want.

**Answer 5:** def keyword is used to define a function.

**LONG ANSWER QUESTIONS:****Answer 1:**

```
def add(*num):  
    total = 0  
    for x in numbers:  
        total += x  
    return total  
print(add((2, 4, 9, 8, 5)))
```

**Answer 2:**

```
def string_rev(str):  
    rstr = ""  
    index = len(str)  
    while index > 0:  
        rstr += str[ index - 1 ]  
        index = index - 1  
    return rstr  
print(string_rev('language'))
```

**Answer 3:**

```
def range(n):  
    if n in range(1,10):  
        print( " %s is in the range"%str(n))  
    else :  
        print("The number is not in the given range.")  
range(5)
```

**Answer 4:**

```
def multiply(num):  
    total = 1  
    for x in num:  
        total *= x  
    return total
```

```
l=[1, 2, 3, 4, 5]
print(multiply(l))
```

**Answer 5:**

```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
n=int(input("Input a number: "))
print(fact(n))
```

## 16. SUGGESTED BOOKS AND REFERENCES

**BOOKS:**

1. Mertz, David (2019). Functional Programming in Python. Dover Publishing.
2. Martin, Brown C., (2018). Python: The Complete Reference. John Wiley & Sons
3. Downey, Allen (2015). Learning with Python. Pearson Publications.
4. Sheetal, Taneja (2017). Python Programming: A modular approach by Pearson (First Edition)

**E-REFERENCES:**

- Python functions, last viewed on March 29, 2021 <  
<https://www.programiz.com/python-programming/function-argument> \ >
- Python functions, last viewed on March 29, 2021 <  
[https://www.w3schools.com/python/python\\_functions.asp](https://www.w3schools.com/python/python_functions.asp) >
- Functions in Python, last viewed on March 29, 2021 <  
<https://www.geeksforgeeks.org/functions-in-python/> >
- Define your own Python Function, last viewed on March 29, 2021 <  
<https://realpython.com/defining-your-own-python-function/> >
- Creating functions, last viewed on March 29, 2021 <  
<https://swcarpentry.github.io/python-novice-inflammation/08-func/index.html> >