9 Functions

In Python a function is a block of code that is executed when it is called. A function can take parameters and return a value. A function is defined using the def keyword. The syntax of a function is as follows:

```python
def function_name(parameters):
    """docstring"""
    statement(s)
    return [expression]
```

9.1 Why use functions?

Functions are used to break down a large program into small manageable and organized chunks. This makes program easy to understand, maintain and debug. Furthermore, it avoids repetition and makes the code reusable. Imagine you have to write a program that calculates the area of a rectangle. You can write a function that calculates the area of a rectangle and use it whenever you want to calculate the area of a rectangle. This saves you from writing the same code again and again. You can also use the same function to calculate the area of a square.

Example

Let us write a function that calculates the area of a rectangle. The function takes two parameters, length and width, and returns the area of the rectangle.

```python
def area(length, width):
    """This function calculates the area of a rectangle"""
    return length * width

print(area(5, 6)) # 30
print(area(2, 3)) # 6
```

The function definition starts with the def keyword. It is followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.

9.2 Parameters

Information can be passed to functions as parameters. Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

```python
def greet(name, msg):
    """This function greets to
    the person with the provided message."""
    print("Hello", name + ', ' + msg)

greet("Monica", "Good morning!") # Hello Monica, Good morning!
```

9.3 Default arguments

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument. The default value is specified after the parameter name in the function definition. The default value is evaluated at the point of function definition in the defining scope.

```python
def greet(name, msg="Good morning!"):
    """This function greets to
    the person with the provided message."""
    print("Hello", name + ', ' + msg)

greet("Monica", "Good morning!") # Hello Monica, Good morning!
greet("John") # Hello John, Good morning!
```

9.4 Return values

A function can return a value. The return statement is used to exit a function and go back to the place from where it was called. The statement consists of the return keyword followed by an expression that is evaluated and returned. If the return statement is without any expression, then the special value None is returned.

```python
def greet(name, msg):
    """This function greets to
    the person with the provided message."""
    print("Hello", name + ', ' + msg)
    return

greet("Monica", "Good morning!") # Hello Monica, Good morning!
```

9.5 Scope of variables

The scope of a variable is the portion of the program where the variable is recognized. There are two basic scopes of variables in Python.

- Global variables
- Local variables

9.5.1 Global variables

A variable that is defined outside of a function or in global scope is a global variable. Global variables can be accessed inside or outside of the function.

```python
x = "global"

def foo():
    print("x inside:", x) # x inside: global

foo()
print("x outside:", x) # x outside: global
```

9.5.2 Local variables

A variable that is defined inside a function's body or in the local scope is a local variable. Local variables can only be accessed inside the function.

```python
x = "global" # global variable

def foo():
    x = "local" # local variable
    print("x inside:", x) # x inside: local

foo()
print("x outside:", x) # x outside: global
```

9.6 Recursion

A function can call other functions. It is even possible for the function to call itself. These type of construct are termed as recursive functions. In a recursive function, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

```python
def factorial(x):
    if x == 1: # base case
        return 1
    else: # recursive case
        return x * factorial(x-1)

print(factorial(5)) # 120
```