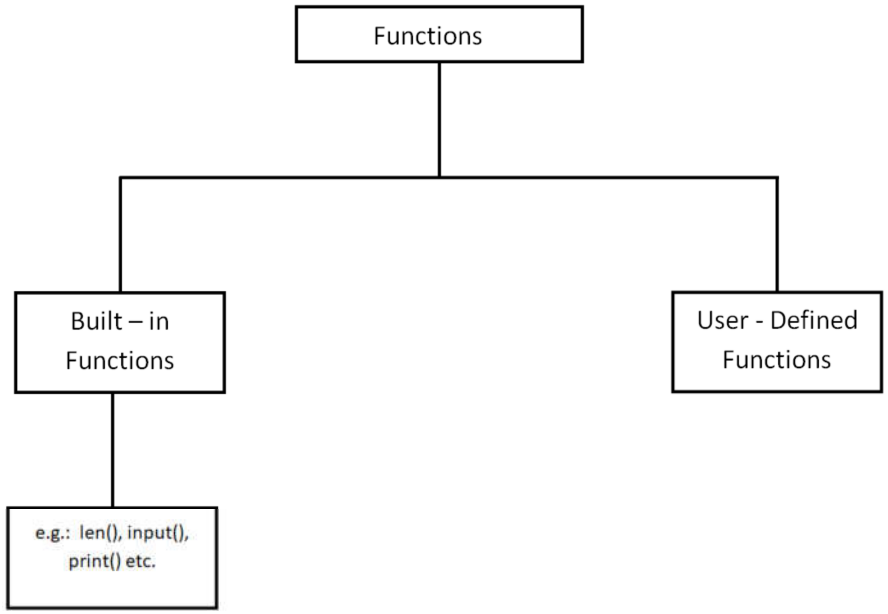
Functions



User-Defined Function

Syntax

```
def function_name(parameters):
   """documantion string"""
   statement(s)
   return result(s)
```

- ☐ parameters(optional): can be one or more.
- ☐ return(optional): statement ends the execution of the function call and returns the results (one or more).
- ☐ If no return statement in function's body, the function ends, when the control flow reaches the end of the function body and the value "None" will be returned.

User-Defined Function

- ☐ A function is a block of statements that perform a specific task.
- ☐ Functions allows us to break our code into manageable, bite-sized chunks.
- ☐ Functions are used to utilize the code in more than one place in a program.
- ☐ Use of functions increases program readability.
- ☐ Use of functions reduces the chances of error.
- ☐ Program modification becomes easier by using function.

Example

```
#print using function
def display():
   """Display function"""
   print("Welcome to IC152 Class")

display()
print("Function documentation:",display.__doc__)
```

Output:

```
Welcome to IC152 class
Function documentation: Display Function
```

Example

```
# function to calculate area and perimeter of circle
def circle(r):
    area = 3.14 * r * r
    perimeter = 2 * 3.14 * r
    return area, perimeter

radius = 5
print("Function documentation:",circle.__doc__)
area,perimeter = circle(radius)
print("Area of circle is {0:.2f} and perimeter is
    {1:.2f}".format(area,perimeter))
result = circle(radius)
print(result)
```

```
Function documentation: None
Area of circle is 78.50 and perimeter is 31.40
(78.5, 31.400000000000002)
```

Default Parameter Value

Any number of arguments in a function can have a default value. But once we have a default argument, all the arguments to its right also must have default values.

```
def fun(a , b = 10 ):
    print("a :",a)
    print("b :",b)

a = 5
b = 20
fun(a)
```

```
a: 5
b: 10
```

```
def fun(a = 10, b):
    print("a :",a)
    print("b :",b)

a = 5
b = 20
fun(a,b)
```

```
SyntaxError: non-default
    argument follows default
    argument
```

Default Parameter Value

```
# function to calculate area and perimeter of circle
def circle(r = 3):
    area = 3.14 * r * r
    perimeter = 2 * 3.14 * r
    return area, perimeter

radius = 5
print("Function documentation:",circle.__doc__)
area,perimeter = circle()
print("Area of circle is {0:.2f} and perimeter is
    {1:.2f}".format(area,perimeter))
result = circle(radius)
print(result)
```

```
Function documentation: None
Area of circle is 28.26 and perimeter is 18.84
(78.5, 31.400000000000002)
```

Keyword Arguments

In keyword arguments we passes arguments as key = value. Order of arguments does not matter in keyword argument method. .

```
def fun(a = 5, b = 10):
    print("a :",a)
    print("b :",b)

fun(b = 20)
```

```
a: 5
b: 20
```

```
def fun(a, b):
    print("a :",a)
    print("b :",b)

fun(b = 20, a = 5)
```

```
a: 5
b: 20
```

Lifetime and Scope of Variables

```
def func():
    a = 20
    b = 10
    print("Value of a inside function is",a)

a = 30
func()
print("Value of a outside function is",a)
print("Value of b ouside function is",b)
```

Output

```
Value of a inside function is 20
Value of a outside function is 30
NameError: name 'b' is not defined
```

Python Anonymous/Lambda Function

```
>>>cube = lambda a: a*a*a
>>>print(cube(5))
125
```

Use:

```
>>>old_list = [1, 2, 3, 4 , 5]
>>>new_list = list(map(lambda a: a*a, old_list))
>>>print(new_list)
[1, 4, 9, 16, 25]
```

Python Anonymous/Lambda Function

- lambda keyword is used to create anonymous functions.
- Lambda Function can have any number of arguments but only one expression, which is evaluated and returned.
- lambda functions are used when we require a nameless function for a short period of time.

Syntax

```
lambda arguments: expression
```