User Defined Functions

Introduction

- In Python, a user-defined function's declaration begins with the keyword def and followed by the function name.
- The function may take arguments(s) as input within the opening and closing parentheses, just after the function name followed by a colon.
- After defining the function name and arguments(s) a block of program statement(s) start at the next line and these statement(s) must be indented.

```
print( help('modules'))
```

BUILT-IN FUNCTIONS

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Syntax

Function with arguments:-

```
def function_name(argument1, argument2, ...) :
        statement_1
        statement_2
        ....
```

Function without arguments:-

```
def function_name() :
        statement_1
        statement_2
        ....
```

Call a function

- Calling a function in Python is similar to other programming languages, using the function name, parenthesis (opening and closing) and parameter(s).
- Syntax:

```
function_name(arg1, arg2)
```

For Example:-

```
def avg_number(x, y):
    print("Average of ",x," and ",y, " is ",(x+y)/2)
avg_number(3, 4)
```

Return statement

- In Python the return statement (the word return followed by an expression.) is used to return a value from a function, return statement without an expression argument returns none.
- Syntax:-

```
def function_name(argument1, argument2, ...) :
    statement_1
    statement_2
    ....
    return expression
function_name(arg1, arg2)
```

Return Statement

For Example: -

```
def nsquare(x, y):
    return (x*x + 2*x*y + y*y)
print("The square of the sum of 2 and 3 is : ",
nsquare(2, 3))
```

Lambda/Anonymous Forms/Function

- In Python, small anonymous (unnamed) functions can be created with lambda keyword.
- Lambda forms can be used as an argument to other function where function objects are required but syntactically they are restricted to a single expression.
- Syntax: -

```
lambda arguments: expression
```

Lambda/Anonymous Forms/Function

- A function like this:

```
def average(x, y):
    return (x + y)/2
print(average(4, 3))
```

may also be defined using lambda

```
print((lambda x, y: (x + y)/2)(4, 3))
```

Use of Lambda Function in python

- We use lambda functions when we require a nameless function for a short period of time.
- In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments).
- Lambda functions are used along with built-in functions like filter(), map() etc.

filter() function

- The filter() function in Python takes in a function and a list as arguments.
- The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True.

```
# Program to filter out only the even items from a list

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)
```

map () function

- The map() function in Python takes in a function and a list.
- The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.

```
# Program to double each item in a list using
map()
my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(map(lambda x: x * 2 , my_list))
print(new_list)
```