Python: Functions and Objects

FOSSEE

1 Function

They allows us to enclose a set of statements and call them again and again instead of repeating the group of statements every-time.

1.1 Function definition

```
In []: def signum(r):
   ...:     if r < 0:
   ...:         return -1
   ...:     elif r > 0:
   ...:         return 1
   ...:     else:
   ...:         return 0
   ...:
   ...:
```

1.2 Usage

```
In []: signum(4)
Out[]: 1
In []: signum(0)
Out[]: 0
In []: signum(-4)
Out[]: -1
In []: signum() # ERROR signum() takes exactly 1 argument(0 given)
```

Note: Arguments passed to a function are passed by-value **only if** they are basic Python data type(int, float). In case one passes immutable types(String, tuples), they cant be modified in the function, but objects like `list` and dictionary can be manipulated.

1.3 Default Arguments

This feature allow the functions to take the arguments optionally. For example:

```
In []: greet = 'hello world'

In []: greet.split()
Out[]: ['hello', 'world']
```

In above case, default argument which `split` function uses is a blank space. One can pass argument also to split the string for a different delimiter.

```
In []: line = 'Rossum, Guido, 54, 46, 55'

In []: line.split(',') #split with ','
Out[]: ['Rossum', ' Guido', ' 54',
                        ' 46', ' 55']
```

Function to work with default argument can be defined as:

```
def welcome(greet, name='world!'):
  print greet, name
```

above function can be used as:

```
In []: welcome("Hello") #using default argument
Hello World!
In []: welcome("Hi", "Guido") #taking name via argument
Hi Guido
```

1.4 Keyword Arguments

This feature provides the facility of passing arguments by specifying the name of the parameter as defined in the function definition. You don't have to remember the order of the parameters in function definition. For example:

```
In []: plot(y, sin(y), 'g', linewidth=2)
In []: plot(y, cos(y), linewidth=1, color='g')
```

Both call to `plot` function will work and parameters are set accordingly. One can define a function such that keyword arguments can be used in following way:

```
def wish(name='World', greetings='Hello'):
  print greetings, name
```

This function can be called as:

```
In [13]: wish() #default arguments will work
Hello World
In [14]: wish(greetings='hey', name='madhu')
hey madhu
In [15]: wish(name='vattam', greetings = 'bye bye')
bye bye vattam
```

2 Self contained python script

Functions like `plot`, `linspace` etc are not inbuilt functions. One have to import them to use them.

```
from scipy import linspace, pi, sin
from pylab import plot, legend, annotate
from pylab import xlim, ylim

x = linspace(-5*pi, 5*pi, 500)
plot(x, x, 'b')
plot(x, -x, 'b')
plot(x, sin(x), 'g', linewidth=2)
plot(x, x*sin(x), 'r', linewidth=3)
legend(['x', '-x', 'sin(x)', 'xsin(x)'])
annotate('origin', xy = (0, 0))
xlim(-5*pi, 5*pi)
ylim(-5*pi, 5*pi)
```

These import statements are necessary to make program self contained. After importing, we can run script via:

```
$ python sine_plot.py
```

We no longer need:

```
$ ipython -pylab
In []: %run -i sine_plot.py
```

3 objects

In Python everything is a object! All variables, lists, tuples, dictionaries and even functions are objects.

```
In []: a = str() # initializing a string object.
In []: b = "Hello World"
In []: b.split() # calling function on object 'b'
Out[]: ['Hello', 'World']
```

"." is a operator used to call functions defined for given object.

4 Links and References

- Some of inbuilt functions available with Python are listed at `http://docs.python.org/library/functions.html`
- Reference manual to describe the standard libraries that are distributed with Python is available at `http://docs.python.org/library/`