# BCA – 502: Python Programming

Rahul Kumar Singh

**In today's Class we have discussed on Python Modules and Package.**

## The globals() and locals() Functions

The **globals()** and **locals()** functions can be used to return the names in the global and local namespaces depending on the location from where they are called.

If **locals()** is called from within a function, it will return all the names that can be accessed locally from that function.

If **globals()** is called from within a function, it will return all the names that can be accessed globally from that function.

The return type of both these functions is dictionary. Therefore, names can be extracted using the **keys()** function.

## The reload() Function

When the module is imported into a script, the code in the top-level portion of a module is executed only once.

Therefore, if you want to reexecute the top-level code in a module, you can use the **reload()** function. The **reload()** function imports a previously imported module again.

**The syntax of the reload() function is this –**

reload(module_name)

Here, module_name is the name of the module you want to reload and not the string containing the module name.

**For example, to reload hello module, do the following –**

reload(hello)

**Locating Modules:-**

When you import a module, the Python interpreter searches for the module in the following sequences –

- The current directory.
- If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
- If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.

The module search path is stored in the system module sys as the sys.path variable. The sys.path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

**The PYTHONPATH Variable**

The PYTHONPATH is an environment variable, consisting of a list of directories. The syntax of PYTHONPATH is the same as that of the shell variable PATH.

**Here is a typical PYTHONPATH from a Windows system –**

```
set PYTHONPATH = c:\python20\lib;
```

**And here is a typical PYTHONPATH from a UNIX system –**

```
set PYTHONPATH = /usr/local/lib/python
```

**Packages in Python:-**

A python package is a collection of modules. Modules that are related to each other are mainly put in the same package. When a module from an external package is required in a program, that package can be imported and its modules can be put to use.

A package contains all the files you need for a module.

Modules are Python code libraries you can include in your project.

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-

subpackages, and so on.

**For example, let’s take the datetime module, which has a submodule called date. When datetime is imported, it’ll result in an error, as shown below:**

```
import datetime

date.today()
```

**Output:-**

```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    date.today()
NameError: name 'date' is not defined
```

**The correct way to use the date module is shown below:**

```
from datetime import date

print date.today()
```

**Output:-**

```
2022-11-04
```

Consider a file Pots.py available in Phone directory. This file has following line of source code –

```
#!/usr/bin/python
```

```
def Pots():
   print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above −

- Phone/Isdn.py file having function Isdn()
- Phone/G3.py file having function G3()

Now, create one more file __init__.py in Phone directory −

Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

```
#!/usr/bin/python
# Now import your Phone Package.
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
```

When the above code is executed, it produces the following result −

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.