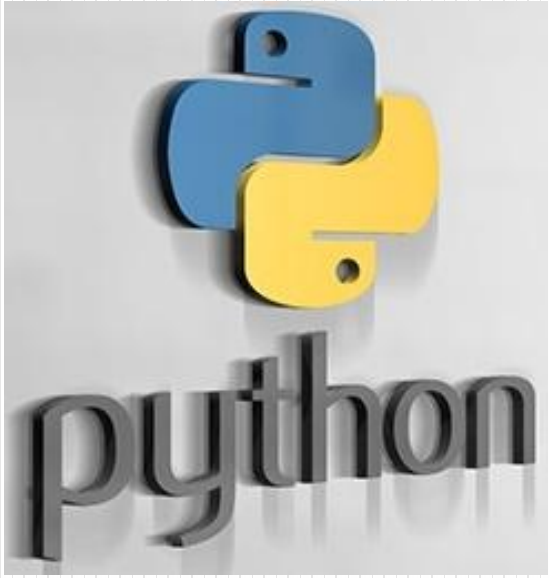
Scripting Language

UNIT-IV



K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Modules

python
Programming Language

Modules

- Modules are used to catagorize the code in python into smaller parts
- Grouping related code into modules makes us easier to understand and use
- Module is a file contains functions, classes and variables
- A module must be imported into your file by using "import"
- It is a file and may have runnable code
- **import Statement**
  - Any python source file can be used as module by executing an import statement in any other python source code
  - **Syntax: import module1, module2,……modueN**
  - When the interpreter encounters the import statement, it imports the module if it is present
  - Module is loaded only once regardless of the number of times it is imported

- **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.
- **Writing modules**
- Writing Python modules is very simple.
- To create a module of your own, simply create a hello .py file with the module name, and then import it using the Python file name (without the .py extension) using the import command.

- **Example**

```
#save this program as hello.py
   def print_func( par ):
         print( "Hello : ", par)
         return
#save this program as import.py
# Import module hello
   import hello
# Now you can call defined function that module as follows
hello.print_func("Ravikanth")
```

- **Output**

```
         Hello : Ravikanth
```

NOTE: *A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.*

- Ex: create a file by name called **addition.py**

```
def add(a,b):
    c=a+b
    print(c)
    return
```

This file can be used in another python file by using import statement

now create a file by name called **test.py**

```
import addition as a        # a is a code for import file called addition
a.add(10,20)
a.add(100,200)
```

Output:

Multiple Import Statements

Filename: message.py

```
def msg_method():
    print("Today looks something different in my life")
    return
```

Filename: display.py

```
def display_method():
    print("You know U R working with Me")
    return
```

Filename: multiimport.py

```
import message,display
message.msg_method()
display.display_method()
```

from import statement

- Python's from statement lets you import specific attributes from a module into the current namespace. The from...import has the following
- syntax –from modname import name1[, name2[, ... nameN]]
- The another way we can import the modules as from import
- **Example**

```
from math import sqrt
#import math
print ("math.sqrt(100) : ", sqrt(100))
print ("math.sqrt(7) : ",sqrt(7))
```

Output:

The *from...import * Statement*

- It is also possible to import all names [functions] from a module by using * (Astrick) symbol – as **import ***

from modname import *

  - modname indicates the Module Name
  - * indicates all the functions of that module

- Ex:

```
from math import *
print ("math.sqrt(7) : ", sqrt(7))
print ("math.sqrt(math.pi) : ", sqrt(pi))
print (pi)
```

Output:

The dir( ) Function

- The dir() built-in function returns a sorted list of strings containing the names defined by a module.
- The list contains the names of all the modules, variables and functions that are defined in a module.
- Ex:

```
#Import built-in module math
        import math
        content = dir(math)
        print( content)
#Import built-in module turtle
        import turtle
        content = dir(turtle)
        print( content)
```

OS Module

- The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.
- Python os module can be used to perform tasks such as finding the name of present working directory, changing current working directory, checking if certain files or directories exist at a location, creating new directories, deleting existing files or directories, walking through a directory and performing operations on every file in the directory that satisfies some user-defined criteria, and a lot more.

Method	Description
rename( )	To rename a file. It takes two arguments, existing_file_name and new_file_name.
remove( )	To delete a file. It takes one argument. Pass the name of the file which is to be deleted as the argument of method.
mkdir( )	To create a directory. A directory contains the files. It takes one argument which is the name of the directory.
chdir( )	To change the current working directory. It takes one argument which is the name of the directory.
getcwd( )	It gives the current working directory.
rmdir( )	To delete a directory. It takes one argument which is the name of the directory.
listdir( )	It diplays all the files and sub directories inside a directory

Syntax with example:

Syntax: os.rename(existing_file_name, new_file_name)

- Ex: **import** os

  os.rename('mno.txt','pqr.txt')

Syntax: os.remove(file_name)

- Ex: **import** os

  os.remove('mno.txt')

Syntax: os.mkdir("file_name")

- Ex: **import** os

  os.mkdir("new")

Syntax: os.chdir("file_name")

- Ex: **import** os

  os.chdir("new")

Syntax: os.getcwd()

•Ex: **import** os

print os.getcwd()

Syntax: os.rmdir("directory_name”)

•Ex: **import** os

os.rmdir("new")

NOTE: In order to delete a directory, it should be empty. In case directory is not empty first delete the files

Syntax: os.listdir( )

import os

print(os.listdir('D:\\Materail\\SL 2017'))

Note: This method takes in a path and returns a list of sub directories and files in that path. If no path is specified, it returns from the current working directory.

```
import os
os.system("notepad")  # It opens a Notepad
```

- **Note :** That rmdir() method can only remove empty directories.
- In order to remove a non-empty directory we can use the rmtree() method inside the shutil module.
- >>>import shutil

```
shutil.rmtree('test')
os.listdir()
```


K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT-Basara, T.S, IND.

Sys module

- The sys module provides information about constants, functions and methods of the Python interpreter. dir(system) gives a summary of the available constants, functions and methods. Like all the other modules, the sys module has to be imported with the import statement, i.e. import sys
- Like all other modules,the sys module has to be imported with import stmt.
- INFORMATION ON THE PYTHON INTERPRETER
- COMMAND-LINE ARGUMENTS
- CHANGING THE OUTPUT BEHAVIOUR OF THE INTERACTIVE PYTHON SHELL
- STANDARD DATA STREAMS
- REDIRECTIONS
- Help(“sys”)

- Ex:

```
import sys
print (sys.version)
print(sys.version_info)
print(sys.paltform)

import sys
print(sys.path)
print(sys.modules)
print(sys.maxunicode)
print(sys.executable)

import sys
 for i in (sys.stdin, sys.stdout, sys.stderr):
      print(i)
```

Math Module

- Provides access to mathematical functions like power, logarithmic, trigonometric, hyperbolic, angular conversion, constants etc

Function	Description
abs()	Absolute value of x: the positive distance between x & zero
ceil()	Ceiling of x:Smallest integer not less than x
cmp(x,y)	-1 if x<y,0 if x==y, or 1 if x>y
exp(x)	Exponential of x:eX
floor(x)	Floor of x : The largest integer not greater than x
max(x1,x2….)	Largest of its arguments: the value closest to positive infinity
min(x1,x2….)	Smallest of its arguments: the value closest to negative infiy
pow(x,y)	Value of x**y
round(x[,n])	x rounded to n digits from the decimal point
sqrt(x)	Square root of x for x>0

String Module

•Includes built-in methods to manipulate strings. Consider the string , str=Infosys

Function	Description
str.count(“s”)	Returns count of occurrence of character “s” in string str
str.startswith(“s”)	Returns true if string str starts with character “s”
str.endswith(“s”)	Returns true if string str ends with character “s”
str.find(“s”)	Returns index position of character “s” in string str if found else -1
str.replace(“s”,”S”)	Replaces all occurrences of character “s” with character “S” in string str
str.isdigit()	Checks if all the characters in string str are digits and returns true or false accordingly
str.upper()	Converts all the characters in string str to uppercase
str.lower()	Coverts all the characaters in string str to lowercase

List Module

Function	Description
cmp(list1,list2)	Compares elements of both lists
len(list)	Gives total length of list
max(list)	Returns items from the list with maximum value
min(list)	Converts a tuple to list
list(seq)	Converts a tuple to list
list.append(obj)	Appends object obj to list
list.count(obj)	Returns count of how many times obj occurs in list
list.insert(index,obj)	Inserts object obj into list of offset index
obj=list.pop()	Removes the items at position -1 from list and assigns it to obj
list.remove(obj)	Removes object obj from list
list.reverse()	Returns the order of items in list
sorted(list)	Sorts items in list

Date and Time Module

- Supplies classes for manipulating dates and times in both simple and complex ways
- The time module provides a number of functions that deal with dates and the time within a day
- import time module : Ex: print(time.localtime())

Function	Description
time.clock ( )	Returns current time in seconds, given as a floating point number
time.gmtime()	Returns current UTC date and time(not affected by timezone)
time.localtime()	Returns the number of hours difference between your timezone and the UTC timezone (London)
time.time()	Returns the number of seconds
time.sleep(secs)	Suspends execution of the current thread for the given number of seconds
time.daylight()	Returns 0 if you are not currently in Daylight Saving time

Ex:

```
import time
print(time.localtime())
```

Output:

```
import time
print(time.ctime())
```

Output:

```
import time
now = time.localtime(time.time())
print (time.asctime(now) )
print (time.strftime("%y/%m/%d %H:%M", now) )
print (time.strftime("%a %b %d", now) )
print (time.strftime("%c", now) )
```

Output:

- **Tick**: Time intervals are floating-point numbers in units of seconds.
- import time

  ticks=time.time()

  print(" number of ticks since 09:00AM Sep 14,2017:",ticks)
- import time

  localtime = tim e.localtime(time.time())

  print ("Local current time :", localtime)
- import time

  localtime = time.asctime(time.localtime(time.time()))

  print ("Local current time :", localtime)

- **What is TimeTuple?**
- Many of Python's time functions handle time as a tuple of 9 numbers, as shown below −

Index	Field	Values
0	4-digit year	2017
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 61 60*or61areleap* − *seconds*
6	Day of Week	0 to 6 0*isMonday*
7	Day of year	1 to 366 *Julianday*
8	Daylight savings	-1, 0, 1, -1 means library determines DST

Index	Attributes	Values
0	tm_year	2008
1	tm_mon	1 to 12
2	tm_mday	1 to 31
3	tm_hour	0 to 23
4	tm_min	0 to 59
5	tm_sec	0 to 61 60*or61areleap − seconds*
6	tm_wday	0 to 6 0*isMonday*
7	tm_yday	1 to 366 *Julianday*
8	tm_isdst	-1, 0, 1, -1 means library determines DST

Calendar Module

- The calendar module supplies calendar-related functions, including functions to print a text calendar for a given month or year.
- By default, calendar takes Monday as the first day of the week and Sunday as the last one. To change this, call calendar.setfirstweekday function.
- **calendar.firstweekday**
- **calendar.isleap**
- **calendar.month**
- **calendar.monthrange**
- import calendar

  cal = calendar.month(2017, 1)

  print ("Here is the calendar:")

  print (cal)

random Module

- The random module contains a number of random number generators. Ex: Dias Game, OTP Generator
- It is most commonly used in Cryptography
- **Example 1:**

```
# import the random module
import random
print(random.randint(0,9))
```

Output:

- **Example 2:**

```
import random
# random choice from a list
for i in range(5):
    print (random.choice([1, 2, 3, 5, 9]))
```

Output:

Turtle Module

- Python supports both
  - Character User Interface [Core]
  - Graphical User Interface [Advance]
- Turtle has 70+ methods as of python3.6 version
- Turtle graphics is a popular way for introducing programming to kids.
- Roughly it has the following features : Better animation of the turtle movements, Different turtle shapes, Fine control over turtle movement and screen updates, ontrolling background color background image, window and canvas size, Appearance of the TurtleScreen and the Turtles at startup and many more behind the screen

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT-Basara, T.S, IND.

Ex:1

```
import turtle as tt
tt.fd(100)
tt.pencolor("red")
tt.bgcolor("blue")
tt.done()
```

Output:

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT-Basara, T.S, IND.

```
import turtle as tt
for i in range(5):
    tt.forward(50)
    tt.left(30)
    tt.pensize(200)
    tt.pencolor("orange")
    tt.bgcolor("yellow")
    tt.color("green")
```

Turtle Methods

```
import turtle as tt
tt.forward(100)
tt.right(100)
tt.left(100)
tt.write("E2 ROCK STARTS",font("Arial",100,"BOLD"))
tt.up()
tt.down()
tt.bye()
tt.done()
tt.dot()
tt.degree()
tt.position()
```

```
tt.delay(10)
tt.speed(10)
tt.position()
tt.windowwidth()
tt.windowheight()
tt.screensize()
tt.circle()
tt.tutle() # it supports 3 shapes turtle,angle,arrow
```

- Many more….

Packages

- 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
- Packages are namespaces which contains multiple packages and modules
- Collection of modules in directory
- **Must have _init_.py file** # May contain subpackages
- _init_.py can be empty or it contains valid python code
- _init_.py indicates that the directory it contains is a package and it can be imported the same way as a module
  - **Ex: foo.abc , here Module abc belongs to package named foo**
- Users of the package can import individual modules from the package

  Ex: import sound.effects.echo
- This loads the submodule sound.effects.echo. It must be referenced with its full name

IIIT BASARA

Infosys | Campus Connect

For Details Contact Me @ :

9247448766

ravikanth27787@gmail.com

python

Programming Language