Modules

A module allows you to logically organize your Python code. Simply, a module is a file consisting of Python code. A module can define 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 When the interpreter encounters the import statement, it imports the module if it is present

Syntax: Syntax: import module1, module2,......modueN

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]]
Example

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

The from...import * Statement
It is also possible to import all names [functions] from a module by using * (Astrick) symbol − as import *

Syntax: from modname import *

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

Example

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

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.

Example:

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

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 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.

rename( ) To rename a file. It takes two existing_file_name and new_file_name.

Syntax: os.rename(existing_file_name, new_file_name)

Example

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

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.

Syntax: os.remove(file_name)

Example:
```
import os
os.remove('mno.txt')
```

mkdir( ) To create a directory. A directory contains the files. It takes one argument which is the name of the directory.

Syntax: os.mkdir("file_name")

Example:
```
import os
os.mkdir("new")
```

chdir( ) To change the current working directory. It takes one argument which is the name of the directory.

Syntax: os.chdir("file_name")

Example:
```
import os
print os.getcwd( )
os.chdir("python")
print os.getcwd( )
```

getcwd( ) It gives the current working directory.

Syntax: os.getcwd()

Example:
```
import os
print os.getcwd()
```

rmdir( ) To delete a directory. It takes one argument which is the name of the directory.

Syntax: os.rmdir("directory_name”)

Example:
```
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.
In order to remove a non-empty directory we can use the rmtree() method inside the shutil module.

Example
```
import shutil
shutil.rmtree('python’)
```

listdir( ) It diplays all the files and sub directories inside a directory

Syntax: os.listdir( )

Example

```
import os
print(os.listdir('/home/rajani/Desktop/python'))
```

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.

Sys Module

This module provides a number of functions and variables that can be used to manipulate different parts of the Python runtime environment.Like all the other modules, the sys module has to be imported with the import statement, i.e. import sys

sys.path()

This function just displays the PYTHONPATH set in current system. Let’s execute this on the system by making a script

sys.modules()

This function gives the names of the existing python modules current shell has imported.

sys.exit()

This method makes the Python interpretor exits the current flow of execution suddenly.

Sys.version()

This method returns the python version

Example

```
import sys

print (sys.version)

print(sys.modules)

print(sys.path)

print("Hello")

sys.exit()

print("Students")
```

Math Module

Math modules Provides access to mathematical functions like power, logarithmic, trigonometric, hyperbolic, angular conversion, constants etc.

Function	Description
ceil(x)	Returns the smallest integer greater than or equal to x.
copysign(x, y)	Returns x with the sign of y
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
fmod(x, y)	Returns the remainder when x is divided by y
frexp(x)	Returns the mantissa and exponent of x as the pair (m, e)
fsum(iterable)	Returns an accurate floating point sum of values in the iterable
isfinite(x)	Returns True if x is neither an infinity nor a NaN (Not a Number)
isinf(x)	Returns True if x is a positive or negative infinity
isnan(x)	Returns True if x is a NaN

ldexp(x, i)	Returns x * (2**i)
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
expm1(x)	Returns e**x – 1
log(x[, base])	Returns the logarithm of x to the base (defaults to e)
log1p(x)	Returns the natural logarithm of 1+x
log2(x)	Returns the base-2 logarithm of x
log10(x)	Returns the base-10 logarithm of x
pow(x, y)	Returns x raised to the power y
sqrt(x)	Returns the square root of x
acos(x)	Returns the arc cosine of x
asin(x)	Returns the arc sine of x
atan(x)	Returns the arc tangent of x
atan2(y, x)	Returns atan(y / x)
cos(x)	Returns the cosine of x
hypot(x, y)	Returns the Euclidean norm, sqrt(x*x + y*y)
sin(x)	Returns the sine of x
tan(x)	Returns the tangent of x
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians
acosh(x)	Returns the inverse hyperbolic cosine of x
asinh(x)	Returns the inverse hyperbolic sine of x
atanh(x)	Returns the inverse hyperbolic tangent of x
cosh(x)	Returns the hyperbolic cosine of x
sinh(x)	Returns the hyperbolic cosine of x
tanh(x)	Returns the hyperbolic tangent of x
erf(x)	Returns the error function at x
erfc(x)	Returns the complementary error function at x
gamma(x)	Returns the Gamma function at x
lgamma(x)	Returns the natural logarithm of the absolute value of the Gamma function at x
pi	Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)

Example

```
import math
a=-5
b=math.fabs(a)#5
print (math.factorial(b))#120
print (math.fsum([1,5,7]))#13.0
c=12.454646
print (math.trunc(c))#12
print (math.pow(2,2))#4.0
print (math.sqrt(2))#1.41421356237
```

Random Module

Python offers random module that can generate random numbers.

randint()

The function randint() generates the random number between start and end range.

Syntax: random.randint(start, end)

choice(rnlist)

The choice() function return the random value from the list.Here random value would be generated from list rnlist.

Syntax:random.choice(rnlist)

shuffle()

The shuffle function, shuffles the elements in list in place. random.shuffle(list1)
list1 is the list provided by user.

Syntax: shuffle(seq)

randrange()

It returns the random number slected from range (start, stop, step).

Syntax
random.randrange(start, stop[, step])
    start and stop spcify the range of random numbers.
    step Step to be added in a random number.

sample()

The sample() function is used to obtain a sample from the sequence (string, list, tuple) .Sequence can be a string , list or tuple . length specify the length of sample to be obtained .

Syntax: random.sample(sequence, length)

uniform(a, b)

Return a random floating point number between a and b inclusive

Syntax: uniform(a,b)

Exercise

1. write a program to roll the dice untill you give exit(using random function)

2. generate a password with length "passlen" with no duplicate characters in the password

3. Write a program which randomly picks an integer from 1 to 100. Your program should prompt the user for guesses – if the user guesses incorrectly, it should print whether the guess is too high or too low. If the user guesses correctly, the program should print how many guesses the user took to guess the right answer. You can assume that the user will enter valid input.