PYTHON PROGRAMMING LANGUAGE

Unit-5

Modules

Module is a python program file which contains a python code including python functions, class, statements and variables. Modules in Python provide us the flexibility to organize the code in a logical way. Modules provide reusability of code. Similar type of attributes can be placed in one module.

A file containing Python code, for e.g., example.py, is called a module and its module name would be example.

Example 39:

create a module and named it as example.py

```
# Python Module example
def add(a, b):
        """This program adds two
        numbers and return the
        result""" result = a + b
        return result
```

we have defined a function add() inside a module named example. The function takes in two numbers and returns their sum.

Loading the module in our python code

We need to load the module in our python code to use its functionality. Python provides two types of statements as defined below.

1. The import statement
2. The from-import statement

1) The import Statement:

The import statement is used to import all the functionality of one module into another.

Syntax:
import module_name

Example 40:
Accessing other module using import statement

```
import example
print (example.add(4,5.5))
```

Python has a ton of standard modules available.

We can import multiple modules separated by comma with a single import

statement. Syntax:
import module1,module2,..... module n

Import all names

import all names (definitions) from a module using import * statement

```
# import all names from the standard module
math from math import *
print("The value of pi is", pi)
```

Output:
The value of pi is 3.14

Renaming a module

Python provides us the flexibility to import some module with a specific name so that we can use this

2) The from-import statement:

from? import statement is used to import only the specific attributes of a module instead of importing the whole module.

Syntax:
from <module-name> import <name 1>, <name 2>..,<name n>

Example 41:

Consider the following module named as calculation which contains three functions as summation, multiplication, and divide.

calculation.py:

```
def summation(a,b):
        return a+b
def multiplication(a,b):
        return a*b
def divide(a,b):
        return a/b
```

Main.py:

```
from calculation import summation
#it will import only the summation() from
calculation.py a = int(input("Enter the first
number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b))
```

Output:

```
Enter the first number10
Enter the second
number20 Sum =  30
```

name to use that module in our python source file.

Syntax:

```
import <module-name> as <specific-name>
```

Example:

```
import calculation as
cal; a = int(input("Enter
a?"));
b = int(input("Enter b?"));
print("Sum =
",cal.summation(a,b))
```

Output:

```
Enter a?10
Enter b?20
Sum = 30
```

The dir() built-in function

dir() function is used to list out names that are defined inside a module.

Example:

```
import math
print
(dir(math))
```

Output:

['__doc_', '__loader__', '__name__', '_package__', '_spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', ' modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

Built in Modules in Python:

There are many built in modules in Python. Some of them are as follows: math, random , threading , collections , os , mailbox , string , time , tkinter etc..

Each module has a number of built in functions which can be used to perform various functions.

Math Module

Using math module , you can use different built in mathematical functions.

Function Name	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...)
E	mathematical constant e (2.71828...)

Useful Example of math module:

```
import math

a=4.6

print math.floor(a)

b=9

print math.sqrt(b)

print math.exp(3.0)

print math.log(2.0)

print math.pow(2.0,3.0)

print math.sin(0)

print math.cos(0)

print math.tan(45)
```

Output:

```
4.0

3.0

20.0855369232

0.69314718056

8.0

0.0

1.0

1.61977519054
```

random Module :

The random module is used to generate the random numbers. Random numbers are used for games, simulations, testing, security, and privacy applications. Some of functions from random module are.

Name of the function	Description	Example
random ( )	It returns a random float x, such that 0 ≤ x<1	>>>random.random ( ) 0.281954791393 >>>random.random ( ) 0.30909046520

randint (a, b)	It returns a int x between a & b such that a ≤ x ≤ b	>>> random.randint (1,10)
		5
		>>> random.randint (2,20) -
		1

uniform (a,b)	unif It returns a floating point number x, such that a <= x < b	>>>random.uniform (5, 10) 5.52615217015
randrange ([start,] stop [,step]	It returns a random item from the given range	from the given range >>>random.randrange(100 ,1000,3) 150

Composition

Composition is an art of combining simple function(s) to build more complicated ones, i.e., result of one function is used as the input to another.

Example Suppose we have two functions fn1 & fn2, such

that a= fn2 (x)

b= fn1 (a)

then call to the two functions can be

combined as b= fn1 (fn2 (x))

Similarly, we can have statement composed of more than two functions. In that result of one function is passed as argument to next and result of the last one is the final result.

Example

math.exp (math.log (a+1))

Example

degrees=270

math.sin

((degrees/360.0)*2*math.pi) 136

Composition is used to package the code into modules, which may be used in many different unrelated places and situations. Also it is easy to maintain the code.

Recursive Function

A function called by itself is called as recursion

Advantages of recursion

1. Recursive functions make the code look clean and elegant.
2. A complex task can be broken down into simpler sub-problems using recursion.
3. Sequence generation is easier with recursion than using some nested iteration.

Disadvantages of recursion

1. Sometimes the logic behind recursion is hard to follow through.
2. Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
3. Recursive functions are hard to debug.

Example

```
# An example of a recursive

function to # find the factorial of a

number


def calc_factorial(x):

"""This is a recursive function

to find the factorial of an integer"""


if x == 1:

        return 1

else:

        return (x * calc_factorial(x-1))


num = 4

print("The factorial of", num, "is", calc_factorial(num))
```

Exercise

Python program to display the Fibonacci sequence up to n-th term using recursive functions

```
def recur_fibo(n):
"""Recursive function to
print Fibonacci
sequence""" if n <= 1:
        return n
else:
        return(recur_fibo(n-1) + recur_fibo(n-
2)) # Change this value for a different result
nterms = 10
# uncomment to take input from the
user #nterms = int(input("How many
terms? ")) # check if the number of
terms is valid
if nterms <= 0:
        print("Please enter a positive integer")
else:
        print("Fibonacci
        sequence:") for i in
        range(nterms):
                print(recur_fibo(i))
```

Python program to find the sum of natural numbers up to n using recursive function

```
def recur_sum(n):
"""Function to return the sum
of natural numbers using
recursion""" if n <= 1:
        return n
else:
        return n + recur_sum(n-1)
# change this value for a different
result num = 16
# uncomment to take input from the user
#num = int(input("Enter a number: "))
if num < 0:
        print("Enter a positive number")
else:
        print("The sum is",recur_sum(num))
```

Assignment-V

1. Define module and write any few functions from math and random module.
2. Define function and explain its advantages. Explain what are the functions available in python with an example are.
3. Differentiate between local and global variables.
4. What is recursion? What are the advantages and disadvantages of recursion? Write a program to calculates exp(x,y) using recursion functions.