6 MODULES

A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module. Modules in Python provides us the flexibility to organize the code in a logical way.

To use the functionality of one module into another, we must have to import the specific module.

Example

In this example, we will create a module named as file.py which contains a function func that contains a code to print some message on the console.

Let's create the module named as file.py.

```
#displayMsg prints a message to the name being passed.
def displayMsg(name)
  print("Hi "+name);
```

Here, we need to include this module into our main module to call the method displayMsg() defined in the module named file.

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

The import statement

The import statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file.

We can import multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

The syntax to use the import statement is given below.

import module1,module2,..... module n

Hence, if we need to call the function displayMsg() defined in the file file.py, we have to import that file as a module into our module as shown in the example below.

Example:

```
import file;
name = input("Enter the name?")
file.displayMsg(name)
```

abc.py

Output:

Enter the name?Kavi
Hi Kavi

The from-import statement

Instead of importing the whole module into the namespace, python provides the flexibility to import only the specific attributes of a module. This can be done by using from? import statement. The syntax to use the from-import statement is given below.

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

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

calculation.py:

```
#place the code in the 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)) #we do not need to specify the module name while access
ummation()
```

Output:

Enter the first number10
Enter the second number20
Sum = 30

The from...import statement is always better to use if we know the attributes to be imported from the module in advance. It doesn't let our code to be heavier. We can also import all the attributes from a module by using *.

Consider the following syntax.

from <module> import *

Renaming a module

Python provides us the flexibility to import some module with a specific name so that we can use this name to use that module in our python source file.

The syntax to rename a module is given below.

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

Example

```
#the module calculation of previous example is imported in this example as cal.
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

Standard Modules – sys

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. You will learn some of the important features of this module here.

sys.exit

This causes the script to exit back to either the Python console or the command prompt. This is generally used to safely exit from the program in case of generation of an exception.

sys.maxsize

Modules

Returns the largest integer a variable can take.

```
>>> import sys
>>>sys.maxsize
9223372036854775807
```

sys.path

This is an environment variable that is a search path for all Python modules.

```
>>>sys.path
['', 'C:\\python36\\Lib\\idlelib', 'C:\\python36\\python36.zip', 'C:\\python36\\DLLs',
'C:\\python36\\lib', 'C:\\python36',
'C:\\Users\\acer\\AppData\\Roaming\\Python\\Python36\\site-packages', 'C:\\python36\\lib\\site-
packages']
```

sys.version

This attribute displays a string containing the version number of the current Python nterpreter.

```
>>>sys.version
'3.7.0 (v3.7.0:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)]'
```

Standard Modules – math

Some of the most popular mathematical functions are defined in the math module. These nclude trigonometric functions, representation functions, logarithmic functions, angle conversion unctions, etc. In addition, two mathematical constants are also defined in this module.

Pie (π) is a well-known mathematical constant, which is defined as the ratio of the circumference to the diameter of a circle and its value is 3.141592653589793.

```
>>> import math
>>>math.pi
3.141592653589793
```

Another well-known mathematical constant defined in the math module is e. It is called Euler's number and it is a base of the natural logarithm. Its value is 2.718281828459045.

```
>>>math.e
2.718281828459045
```

The math module contains functions for calculating various trigonometric ratios for a given angle. The functions (sin, cos, tan, etc.) need the angle in radians as an argument. We, on the other hand, are used to express the angle in degrees. The math module presents two angle conversion functions: degrees() and radians(), to convert the angle from degrees to radians and vice versa. For example, the following statements convert the angle of 30 degrees to radians and back (Note: π radians is equivalent to 180 degrees).

```
>>>math.radians(30)
0.5235987755982988
>>>math.degrees(math.pi/6)
29.999999999999996
```

The following statements show sin, cos and tan ratios for the angle of 30 degrees (0.5235987755982988 radians):

```
>>math.sin(0.5235987755982988)
0.49999999999999994
>>>math.cos(0.5235987755982988)
0.8660254037844387
>>>math.tan(0.5235987755982988)
0.5773502691896257
```

You may recall that sin(30)=0.5, cos(30)=32 (which is 0.8660254037844387) and tan(30)= 13 (which is 0.5773502691896257).

math.log()

The math.log() method returns the natural logarithm of a given number. The natural logarithm is calculated to the base e.

```
>>>math.log(10)
2.302585092994046
```

math.log10()

The math.log10() method returns the base-10 logarithm of the given number. It is called the standard logarithm.

```
>>>math.log10(10)
1.0
```

math.exp()

The math.exp() method returns a float number after raising e (math.e) to given number. In other words, exp(x) gives e**x.

```
>>>math.log10(10)
1.0
```

This can be verified by the exponent operator.

```
>>>math.e**10
22026.465794806703
```

math.pow()

The math.pow() method receives two float arguments, raises the first to the second and returns the result. In other words, pow(4,4) is equivalent to 4**4.

```
>>>math.pow(2,4)
16.0
>>>2**4
16
```

math.sqrt()

The math.sqrt() method returns the square root of a given number.

```
>>>math.sqrt(100)
10.0
>>>math.sqrt(3)
1.7320508075688772
```

The following two functions are called representation functions. The ceil() function approximates the given number to the smallest integer, greater than or equal to the given floating

point number. The floor() function returns the largest integer less than or equal to the given number.

```
>>>math.ceil(4.5867)
5
>>>math.floor(4.5687)
4
```

Standard Modules - time

In the real world applications, there are the scenarios where we need to work with the date and time. There are the examples in python where we have to schedule the script to run at some particular timings.

In python, the date is not a data type, but we can work with the date objects by importing the module named with datetime, time, and calendar.

Tick

In python, the time instants are counted since 12 AM, 1st January 1970. The function time() of the module time returns the total number of ticks spent since 12 AM, 1st January 1970. A tick can be seen as the smallest unit to measure the time.

Consider the following example.

Example

```
import time;
#prints the number of ticks spent since 12 AM, 1st January 1970
print(time.time())
```

Output:

1545124460.9151757

The localtime() functions of the time module are used to get the current time tuple. Consider the following example.

Example

```
import time;
#returns a time tuple
print(time.localtime(time.time()))
```

odules

utput:

time.struct_time(tm_year=2018, tm_mon=12, tm_mday=18, tm_hour=15, tm_min=1, tm_sec=32, tm_wday=1, tm_yday=352, tm_isdst=0)

ime tuple

The time is treated as the tuple of 9 numbers. Let's look at the members of the time tuple.

Index	Attribute	Values
0	Year	4 digit (for example 2018)
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 60
6	Day of weak	0 to 6
7	Day of year	1 to 366
8	Daylight savings	-1, 0, 1, or -1

The datetime Module

The datetime module enables us to create the custom date objects, perform various operations on dates like the comparison, etc.

To work with dates as date objects, we have to import datetime module into the python source code.

Consider the following example to get the datetime object representation for the current time.

Example

```
import datetime;
#returns the current datetime object
print(datetime.datetime.now())
```

Output:

2018-12-18 16:16:45.462778

The calendar module

Python provides a calendar object that contains various methods to work with the calendars. Consider the following example to print the Calendar of the last month of 2018.

Example

```
import calendar;
cal = calendar.month(2018,12)
#printing the calendar of December 2018
print(cal)
```

Output:

December 2018
Mo Tu We Th Fr Sa Su
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31

Using dir() function

The dir() function returns a sorted list of names defined in the passed module. This list contains all the sub-modules, variables and functions defined in this module.
Consider the following example.

Example

```
import json
List = dir(json)
print(List)
```

Output:

['JSONDecoder', 'JSONEncoder', '__all__', '__author__', '__builtins__', '__cached__', '__doc__',
'__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_default_decoder', '_default_encoder', 'decoder', 'dump', 'dumps', 'encoder', 'load', 'loads', 'scanner']

EXERCISE

VERY SHORT ANSWER TYPE QUESTIONS

Q.1. What is module?

Ans. A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module.

Q.2. What is the extension of module file?

Ans. .py

Q.3. What is import statement?

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

Q.4. Write the Names of standard modules.

Ans. Sys, math, time, datetime, calendar.

Q.5. What is dir?

Ans. The dir() function returns a sorted list of names defined in the passed module. This list contains all the sub-modules, variables and functions defined in this module.

SHORT ANSWER TYPE QUESTIONS :

1. How module is imported in Python file?
2. Explain the math module with example.
3. What are various time function?
4. Explain sys module.
5. How can you show calendar?

LONG ANSWER TYPE QUESTIONS:

1. What is module? How it is created and used in Python? What is its role?
2. Explain various built in modules in Python with example.

□□□