

4. Raise statement

- The raise statement allows the programmer to force a specified exception to occur.

Example:

```
>>> raise NameError('HiThere')
```

Output:

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: HiThere
```

- ✓ If you need to determine whether an exception was raised but don't intend to handle it, a simpler form of the raise statement allows you to **re-raise** the exception:

Example

```
try:
...   raise NameError('HiThere')
... except NameError:
...   print('An exception flew by!')
...   raise
```

Output:

```
An exception flew by! Traceback
(most recent call last):
  File "<stdin>", line 2, in <module>
NameError: HiThere
```

5. Try –Except-Finally

- ✓ A finally clause is always executed before leaving the try statement, whether an exception has occurred or not.
- ✓ The finally clause is also executed “on the way out” when any other clause of the try statement is left via a break, continue or return statement.

Syntax

```
try:
    statements
except:
    statements
finally:
    statements
```

Example

```
X=int(input("Enter the value of X"))
Y=int(input("Enter the value of Y"))
try:
    result = X / ( X - Y )
except Zero DivisionError:
    print("Division by Zero")
else:
    print("result=".result)
finally:
    print ("executing finally clause")
```

Output:1 Enter the value of X = 10 Enter the value of Y = 5 Result = 2 executing finally clause	Output : 2 Enter the value of X = 10 Enter the value of Y = 10 Division by Zero executing finally clause
--	---

5. MODULES IN PYTHON

- A python module is a file that consists of python definition and statements. A module can define functions, classes and variables.
- It allows us to logically arrange related code and makes the code easier to understand and use.

1.Import statement:

- An import statement is used to import python module in some python source file.

Syntax: import module1 [, module2 [...module]]

Example:

```
>>>import math
>>>print (math.pi)
3.14159265
```

2.Import with renaming:

The import a module by renaming it as follows,

```
>>>import math as a
>>>print("The value of pi is ",a.pi)
The value of pi is 3.14159265
```

Writing modules:

- Any python source code file can be imported as a module into another python source file. For example, consider the following code named as support.py, which is python source file defining two function add(), display().

Support.py:

```
def add(a,b):
    print("The result is ",a+b)
    return
def display(p):
    print("welcome ",p)
    return
```

The support.py file can be imported as a module into another python source file and its functions can be called from the new files as shown in the following code:

3. Import file name

```
import support          #import module support
support.add(3,4)        #calling add() of support module with two integers
support.add (3.5,4.7)  #calling add() of support module with two real values
support.add ('a','b')  #calling add() of support module with two character values
support.add ("yona","alex")#calling add() of support module with two string values
support.display ('fleming') #calling display() of support module with a string value
```

Output:

The result is 7
 The result is 8.2
 The result is ab
 The result is yonaalex
 Welcome, fleming

4. from.....import statement:

- ✓ It allows us to import specific attributes from a module into the current namespace.

Syntax: from modulename import name1 [, name2[,.....nameN]]

```
from support import add          #import module support
support.add(3,4)                 #calling add() of support module with two integers
support.add(3.5,4.7)            #calling add() of support module with two real values
support.add('a','b')            #calling add() of support module with two character values
support.add("yona","alex")      #calling add() of support module with two string values
support.display('fleming')      #calling display() of support module with a string value
```

Output:

The result is 7
 The result is 8.2
 The result is ab
 The result is yonaalex
 Welcome, fleming

5.OS Module

- ✓ The OS module in python provide function for interacting with operating system
- ✓ To access the OS module have to import the OS module in our program

import os

method	example	description
name	Os.name 'nt'	This function gives the name of the operating system
getcwd()	Os.getcwd() ,C:\\Python34'	Return the current working directory(CWD)of the file used to execute the code
mkdir(folder)	Os.mkdir("python")	Create a directory(folder) with the given name
rename(oldname,newname)	Os.rename("python","pspp")	Rename the directory or folder
remove("folder")	Os.remove("pspp")	Remove (delete)the directory or folder

getuid()	Os.getuid()	Return the current process's user id
environ	Os.nviron	Get the users environment

6.Sys Module

- ✓ Sys module provides information about constant, function and methods
- ✓ It provides access to some variables used or maintained by the interpreter

import sys

methods	example	description
sys.argv	sys.argv sys.argv(0) sys.argv(1)	Provides the list of command line arguments passed to a python script Provides to access the file name Provides to access the first input
sys.path	sys.path	It provide the search path for module
sys.path.append()	sys.path.append()	Provide the access to specific path to our program
sys.platform	sys.platform 'win32'	Provide information about the operating system platform
sys.exit	sys.exit <built.in function exit>	Exit from python

Steps to Create the Own Module

- ✓ Here we are going to create a calc module ; our module contains four functions

i.e add(),sub(),mul(),div()

Program for calculator module	output
<pre>Module name ;calc.py def add(a,b); print(a+b) def sub(a,b); print(a-b) def mul(a,b); print(a*b) def div(a,b); print(a/b)</pre>	<pre>import calculator calculator.add(2,3) Outcome >>>5</pre>

6. PACKAGES IN PYTHON

- ✓ A package is a collection of python module. Module is a single python file containing function definitions
- ✓ A package is a directory(folder)of python module containing an additional init py file, to differentiate a package from a directory
- ✓ Packages can be nested to any depth, provided that the corresponding directories contain their own init py file.
- ✓ `__init.py` file is a directory indicates to the python interpreter that the directory should be treated like a python package `init.py` is used to initialize the python package

Steps to Create a Package_

Step1: create the package directory

- ✓ Create the directory (folder)and give it your packages name
- ✓ Here the package name is calculator

Name	Data modified	Type
1. <code>pycache__</code>	05-12-2017	File folder
2.calculator	08-12-2017	File folder
3. DLLs	10-12-2017	File folder

Step2: write module for calculator directory add save the module in calculator directory

- ✓ Here four module have create for calculator directory

Local Disk (C)>Python34>Calculator

Name	Data modified	Type	Size
1. add	08-12-2017	File folder	1KB
2. div	08-12-2017	File folder	1KB
3. mul	08-12-2017	File folder	1KB
4. sub	08-12-2017	File folder	1KB

add.py	div.py	mul.py	sub.py
<code>def add(a,b); print(a+b)</code>	<code>def div(a,b); print(a/b)</code>	<code>def mul(a,b); print(a*b)</code>	<code>def sub(a,b); print(a-b)</code>

Step3: add the `__init.py` file in the calculator directory

- ✓ A directory must contain the file named `__init.py` in order for python to consider it as a package

Add the following code in the_init_.py file

```
from * add import add
from * sub import sub
from * mul import mul
from * div import div
```

Local Disk (C):/Python34>Calculator

Name	Data modified	Type	Size
1. init_____	08-12-2017	File folder	1KB
2. add	08-12-2017	File folder	1KB
3. div	08-12-2017	File folder	1KB
4. mul	08-12-2017	File folder	1KB
5. sub	08-12-2017	File folder	1KB

Step4: To test your package

- ✓ Import calculator package in your program and add the path of your package in your program by using sys.path.append()

Example

```
import calculator
import sys
sys.path.append("C:/Python34")
print ( calculator.add(10,5))
print ( calculator.sub(10,5))
print ( calculator.mul(10,5))
print ( calculator.div(10,5))
```

Output :

```
>>> 15
      5
      50
      2
```

Two marks:

1. Why do we go for file?

File can a persistent object in a computer. When an object or state is created and needs to be persistent, it is saved in a non-volatile storage location, like a hard drive.

2. What are the three different mode of operations of a file?

The three mode of operations of a file are,

- i. Open – to open a file to perform file operations
- ii. Read – to open a file in read mode
- iii. Write – to open a file in write mode

3. State difference between read and write in file operations.

Read	Write
A "Read" operation occurs when a computer program reads information from a computer file/table (e.g. to be displayed on a screen). The "read" operation gets information out of a file.	A "Write" operation occurs when a computer program adds new information, or changes existing information in a computer file/table.
After a "read", the information from the file/table is available to the computer program but none of the information that was read from the file/table is changed in any way.	After a "write", the information from the file/table is available to the computer program but the information that was read from the file/table can be changed in any way.

4. Differentiate error and exception.

Errors

- Error is a mistake in python also referred as bugs .they are almost always the fault of the programmer.
- The process of finding and eliminating errors is called debugging
 - Types of errors
 - Syntax error or compile time error
 - Run time error
 - Logical error

Exceptions

An exception is an error that happens during execution of a program. When that Error occurs

5. Give the methods of exception handling.

1. try –except
2. try –multiple except
3. try –except-else
4. raise exception
5. try –except-finally

6. State the syntax for try...except block

The try and except statements are used to handle runtime errors

Syntax:

```
try :  
    statements  
except:  
    statements
```

7. Write a program to add some content to existing file without effecting the existing content.

```
file=open("newfile.txt",'a')  
file.write("hello")
```

newfile.txt Hello!!World!!!	newfile.txt(after updating) Hello!!!World!!!hello
---------------------------------------	---

8. What is package?

- A package is a collection of python module. Module is a single python file containing function definitions
- A package is a directory(folder)of python module containing an additional_ init_ py file, to differentiate a package from a directory
- Packages can be nested to any depth, provided that the corresponding directories contain their own __init py file

9. What is module?

A python module is a file that consists of python definition and statements. A module can define functions, classes and variables. □ It allows us to logically arrange related code and makes the code easier to understand and use.

10. Write the snippet to find the current working directory.

```
Import os  
print(os.getcwd())
```

Output:

C:\\Users\\Mano\\Desktop

11. Give the use of format operator

The argument of write has to be a string, so if we want to put other values in a file, we have to convert them to strings. The easiest way to do that is with str:

```
>>> x = 52
>>> fout.write(str(x))
```

An alternative is to use the format operator, %. When applied to integers, % is the modulus operator. But when the first operand is a string, % is the format operator. The first operand is the format string, which contains one or more format sequences, which specify how the second operand is formatted. The result is a string. For example, the format sequence '%d' means that the second operand should be formatted as an integer (d stands for “decimal”):

```
>>> camels = 42
>>> '%d' % camels '42'
```

The result is the string '42', which is not to be confused with the integer value 42.

12. Write the snippet to find the absolute path of a file.

```
import os
os.path.abspath('write.py')
```

Output:

```
'C:\\Users\\Mano\\Desktop\\write.py'
```

13. What is the use of os.path.isdir() function.

os.path.isdir() is a function defined in the package os. The main function of isdir(“some input”) function is to check whether the passed parameter is directory or not. isdir() function will only **return** only **true or false**.

14. What is the use of os.path.isfile() function.

os.path.isfile () is a function defined in the package os. The main function of isfile (“some input”) function is to check whether the passed parameter is file or not. isfile () function will only **return** only **true or false**.

15. What is command line argument?

sys.argv is the list of command line arguments passed to the Python program. Argv represents all the items that come along via the command line input, it's basically an array holding the command line arguments of our program.