UNIT V

FILES, MODULES, PACKAGES

1. FILE AND ITS OPERATION

- File is a collection of record.
- A file stores related data, information, settings or commands in secondary storage device like magnetic disk, magnetic tape, optical disk, flash memory.

File Type

1. Text file
2. Binary file

Text file	Binary file
Text file is a sequence of characters that can be sequentially processed by a computer in forward direction Each line is terminated with a special character called the E0L or end of line character	A binary files store the data in the binary format(i.e .0's and 1's) It contains any type of data (pdf,images,word doc,spreadsheet,zip files,etc)

Mode in File

Module	Description
r	Read only
w	mode Write
a	only Appending
r+	only Read and write only

Differentiate write and append mode:

Write mode	Append mode
• It is used to write a string in a file • If file is not exist it creates a new file • If file is exit in the specified name, the existing content will overwrite in a file by the given string	• It is used to append (add) a string into a file • If file is not exist it creates a new file • It will add the string at the end of the old file

File Operation:

- ✓ Open a file
- ✓ Reading a file
- ✓ Writing a file
- ✓ Closing a file

1. Open ( ) function:

- Pythons built in open function to get a file object.
- The open function opens a file.
- It returns a something called a file object.
- File objects can turn methods and attributes that can be used to collect

Syntax:

file_object=open(“file_name” , ”mode”)

Example:

fp=open(“a.txt”,”r”)

Create a text file

fp=open (“text.txt”,”w”)

2. Read ( ) function

Read functions contains different methods

- read() – return one big string
- readline() – return one line at a time
- readlines() – return a list of lines

Syntax:

file_name.read ()

Example:

```
fp=open(“a.txt”,”w”)
print(fp.read())
print(fp.read(6))
print (fp.readline())
print (fp.readline(3))
print (fp.readlines())
```

a.txt

A file stores related data, information, settings or commands in secondary storage device like magnetic disk, magnetic tape, optical disk, flash memory.

Output

Reading file using looping:

- Reading a line one by one in given file

```
fp=open("a.txt","r")
for line in fp:
        print(line)
```

3. Write ( ) function

This method is used to add information or content to existing file.

Syntax:

```
file_name.write( )
```

Example:

```
fp=open("a.txt","w")
fp.write("this file is a.txt")
fp.write("to add more lines")
fp.close()
```

Output: a.txt

```
A file stores related data, information, settings or commands in secondary storage device like magnetic disk, magnetic tape, optical disk, flash memory.
this file is a.txt to
add more lines
```

4. Close ( ) function

It is used to close the file.

Syntax:

```
File name.close()
```

Example:

```
fp=open("a.txt","w")
fp.write("this file is a.txt")
fp.write("to add more lines")
fp.close()
```

Splitting line in a text line:

```
fp=open("a.txt","w")
for line in fp:
    words=line.split()
print(words)
```

2. Write a program for one file content copy into another file:

```
source=open("a.txt","r")
destination=open("b.txt","w")
for line in source:
    destination.write(line)
source. close()
destination.close()
```

Output:

Input a.txt	Output b.txt
A file stores related data, information, settings or commands in secondary storage device like magnetic disk, magnetic tape, optical disk, flash memory	A file stores related data, information, settings or commands in secondary storage device like magnetic disk, magnetic tape, optical disk, flash memory

3. Write a program to count number of lines, words and characters in a text file:

```
fp = open("a.txt","r")
line =0
word = 0
character = 0
for line in fp:
        words = line . split ( )
        line = line + 1
        word = word + len(words)
        character = character +len(line)
print("Number of line", line)
print("Number of words", word)
print("Number of character", character)
```

Output:

```
Number of line=5
Number of words=15
Number of character=47
```

4. ERRORS,EXCEPTION HANDLING

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

Syntax errors

- Syntax errors are the errors which are displayed when the programmer do mistakes when writing a program, when a program has syntax errors it will not get executed
  - ✓ Leaving out a keyword
  - ✓ Leaving out a symbol, such as colon, comma, brackets
  - ✓ Misspelling a keyword
  - ✓ Incorrect indentation

Runtime errors

- If a program is syntactically correct-that is ,free of syntax errors-it will be run by the python interpreter
- However, the program may exit unexpectedly during execution if it encounters a runtime error.
- When a program has runtime error it will get executed but it will not produce output
  - ✓ Division by zero
  - ✓ Performing an operation on incompatible types
  - ✓ Using an identifier which has not been defined
  - ✓ Trying to access a file which doesn't exit

Logical errors

- Logical errors are the most difficult to fix
- They occur when the program runs without crashing but produces incorrect result
  - ✓ Using the wrong variable name
  - ✓ Indenting a blocks to the wrong level
  - ✓ Using integer division instead of floating point division
  - ✓ Getting operator precedence wrong

Exception handling

Exceptions

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

Errors in python

- IO Error-If the file cannot be opened.
- Import Error -If python cannot find the module
- Value Error -Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value
- Keyboard Interrupt -Raised when the user hits the interrupt
- EOF Error -Raised when one of the built-in functions (input() or raw_input()) hits an end-of-file condition (EOF) without reading any data

Exception Handling Mechanism

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

1. Try –Except Statements

- The try and except statements are used to handle runtime errors

Syntax:

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

The try statement works as follows:-

- ✓ First, the try clause (the statement(s) between the try and except keywords) is executed.
- ✓ If no exception occurs, the except clause is skipped and execution of the try statement is finished.
- ✓ If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then if its type matches the exception named after the except keyword, the except clause is executed, and then execution continues after the try statement.

Example:

```
X=int(input("Enter the value of X"))
Y=int(input("Enter the value of Y"))
try:
        result = X / ( X – Y )
        print("result=".result)
except ZeroDivisionError:
        print("Division by Zero")
```

Output:1	**Output : 2**
Enter the value of X = 10 Enter the value of Y = 5 Result = 2	Enter the value of X = 10 Enter the value of Y = 10 Division by Zero

2. Try – Multiple except Statements

- o Exception type must be different for except statements

Syntax:

```
try:
        statements
except errors1:
        statements
except errors2:
        statements
except errors3:
        statements
```

Example

```
X=int(input("Enter the value of X"))
Y=int(input("Enter the value of y"))
try:
        sum = X + Y
        divide = X / Y
        print (" Sum of %d and %d = %d", %(X,Y,sum))
        print (" Division of %d and %d = %d", %(X,Y,divide))
except NameError:
        print(" The input must be number")
except ZeroDivisionError:
        print("Division by Zero")
```

Output:1	Output 2:	Output 3:
Enter the value of X = 10 Enter the value of Y = 5 Sum of 10 and 5 = 15 Division of 10 and 5 = 2	Enter the value of X = 10 Enter the value of Y = 0 Sum of 10 and 0 = 10 Division by Zero	Enter the value of X = 10 Enter the value of Y = a The input must be number

3. Try –Except-Else

- The else part will be executed only if the try block does not raise the exception.
- Python will try to process all the statements inside try block. If value error occur, the flow of control will immediately pass to the except block and remaining statements in try block will be skipped.

Syntax:

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

Example

```
X=int(input("Enter the value of X"))
Y=int(input("Enter the value of Y"))
try:
        result = X / ( X – Y )
except ZeroDivisionError:
        print("Division by Zero")
else:
        print("result=".result)
```

Output:1	**Output : 2**
Enter the value of X = 10 Enter the value of Y = 5 Result = 2	Enter the value of X = 10 Enter the value of Y = 10 Division by Zero

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	**Output : 2**
Enter the value of X = 10	Enter the value of X = 10
Enter the value of Y = 5	Enter the value of Y = 10
Result = 2	Division by Zero
executing finally clause	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	Osname '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

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)
	import calculator
	calculator.add(2,3)
	
	Outcome
	>>>5

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. pycache___	05-12-2017	File folder
2.calculater	**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
def add(a,b); print(a+b)	def div(a,b); print(a/b)	def mul(a,b); print(a*b)	def sub(a,b); print(a-b)

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
importsys
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	**newfile.txt(after updating)**
Hello!!World!!!	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 anydepth, 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('w
rite.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.