GOVERNMENT POLYTECHNIC AMRAVATI
DIPLOMA PROGRAMME IN COMPUTER ENGINEERING

COURSE : PROGRAMMING WITH PYTHON

COURSE CODE :CM5461

Unit 5
File Handling

RATIONALE

- Python is used for developing desktop GUI applications, websites and web applications. Also, as a high level programming language it allows you to focus on core functionality of the application by taking care of common programming tasks. This course is designed to help the students to understand fundamental syntactic information about ‘Python’. Also it will help the students to apply the basic concepts, program structure and principles of ‘Python’ programming paradigm to build given application. The course is basically designed to create a base to develop foundation skills of programming language.

COURSE OUTCOMES (COs)

At the end of this course, student will be able to: -

- Write and execute simple ‘Python’ programs.
- Write ‘Python’ programs using arithmetic expressions and control structure.
- Develop ‘Python’ programs using List, Tuples and Dictionary.
- Develop/Use functions in Python programs for modular programming approach.
- Develop ‘Python’ programs using File Input/output operations.
- Write ‘Python’ code using Classes and Objects.

UNIT 4 :- CONTENTS

1. Opening file in different modes.
2. Accessing file Contents using standard library functions.
3. Closing a file.
4.
5.

Files

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since Random Access Memory (RAM) is volatile (which loses its data when the computer is turned off), we use files for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

Files

- Hence, in Python, a file operation takes place in the following order:
- Open a file
- Read or write (perform operation)
- Close the file

Opening Files in Python

- Python has a built-in open() function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
f = open("test.txt") # open file in current directory

f = open("C:/Python38/README.txt") # specifying full path
```

- We can specify the mode while opening a file. In mode, we specify whether we want to read r, write w or append a to the file. We can also specify if we want to open the file in text mode or binary mode.

Opening Files in Python

The open Function

Before you can read or write a file, you have to open it using Python's built-in open() function. This function creates a file object, which would be utilized to call other support methods associated with it.

Syntax

file object = open(file_name [, access_mode][, buffering])

Opening Files in Python

- The default is reading in In this mode, we get strings when reading from the file. text mode.
- On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like images or executable files.

Opening Files in Python

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Opening Files in Python

```
f = open("test.txt") # equivalent to 'r' or 'rt‘
f = open("test.txt",'w') # write in text mode
f = open("img.bmp",'r+b') # read and write in binary mode
```

- Unlike other languages, the character a does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).

Moreover, the default encoding is platform dependent.

In windows, it is cp1252 but utf-8 in Linux.

Opening Files in Python

- So, we must not also rely on the default encoding or else our code will behave differently in different platforms.
- Hence, when working with files in text mode, it is highly recommended to specify the encoding type.
- f = open("test.txt", mode='r', encoding='utf-8')

Closing Files in Python

- When we are done with performing operations on the file, we need to properly close the file.
- Closing a file will free up the resources that were tied with the file. It is done using the close() method available in Python.
- Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.
- f = open("test.txt", encoding = 'utf-8') # perform file operations f.close()

Closing Files in Python

- This method is not entirely safe. If an exception occurs when we are performing some operation with the file, the code exits without closing the file.
- A safer way is to use a try...finally block.
- try:
- f = open("test.txt", encoding = 'utf-8') # perform file
- operations
- finally:
- f.close()

Closing Files in Python

- This way, we are guaranteeing that the file is properly closed even if an exception is raised that causes program flow to stop.
- The best way to close a file is by using the with statement. This ensures that the file is closed when the block inside the with statement is exited.
- We don't need to explicitly call the close() method. It is done internally.
- with open("test.txt", encoding = 'utf-8') as f:
- # perform file operations

Writing to Files in Python

```
with open("test.txt",'w',encoding = 'utf-8') as f:
f.write("my first file\n")
f.write("This file\n\n")
f.write("contains three lines\n")
```

- This program will create a new file named test.txt in the current directory if it does not exist. If it does exist, it is overwritten.
- We must include the newline characters ourselves to distinguish the different lines.

Reading Files in Python

- To read a file in Python, we must open the file in reading r mode.
- There are various methods available for this purpose. We can use the read(size) method to read in the size number of data. If the size parameter is not specified, it reads and returns up to the end of the file.
- We can read the text.txt file we wrote in the above section in the following way:

Reading Files in Python

- >>> f = open("test.txt",'r',encoding = 'utf-8')
- >>> f.read(4) # read the first 4 data
- 'This'
- >>> f.read(4) # read the next 4 data
- ' is ‘
- >>> f.read() # read in the rest till end of file 'my first file\nThis file\ncontains three lines\n'
- >>> f.read() # further reading returns empty sting

The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.Here is a list of all attributes related to file object −

Sr.No.	Attribute & Description
1	**file.closed** Returns true if file is closed, false otherwise.
2	**file.mode** Returns access mode with which file was opened.
3	**file.name** Returns name of the file.
4	**file.softspace** Returns false if space explicitly required with print, true otherwise.

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace
```

```
Name of the file:  foo.txt
Closed or not :  False
Opening mode :  wb
Softspace flag :  0
```

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace
```

```
Name of the file:  foo.txt
Closed or not :  False
Opening mode :  wb
Softspace flag :  0
```