Module 11: Python Files

File:

- File is a named location on disk to store related information.
- It is used to permanently store data in a non-volatile memory (e.g. hard disk).

File Types:

- Python can handle two types of files:
  - 1) Text files
  - 2) Binary files

File Operations:

- It takes place in the following order:
  - Open a file.
  - Perform file operation (read/write/append)
  - Close the file.

Opening a File:

f = open("filename", "mode", "buffering")

open() ➔ this functions opens a file and returns a file object(file handle).

mode:

w ➔ write data into file.

- It creates new file and stores the data.
- If the file already exists then it would overwrite the data.

r ➔ read data from the file.

- It reads the data from exisiting file
- If the file does not exists then it throws FileNotFoundError

a➔ append data to file.

- It appends the data to the end of the file.
- If the file does not exists then it creates new file.

+ in suffix if we want to read and write/append.

b in suffix if we want to work with binary data

For buffering any positive integer can be used. (default to 4096)

File Operations:

Write Operations:

- **write(s)** ➔ write string s to the file and return the number of characters written.
  - Note 1: string can be text or binary data.
  - Note 2: write method does not add a newline character to the end of the string.
- **writelines():** ➔ it writes a List of string elements in the file.

Read Operations:

- **read()** ➔ returns the read bytes in the form of string.
  - If n is specified then read atmost n characters from the file.
- **readline()** ➔ reads and returns one line from the file.
- **readlines()** ➔ reads and return a list of lines from the file.

Closing a File:

- **close()** will flush out any unwritten information and close the file, after which no more writing can be done.
- On closed file any operation would lead to exception: ValueError: I/O operation on closed file.
- If an exception occurs when we are performing some operation with file, the code exits without closing the file.
- Two Options for safe closure:
  - Use try..finally block or use with statement

---

os module:

- The os module provides a big range of useful methods to manipulate files and directories:
  - **remove():** ➔ it is used to delete file.
  - **rename():** ➔ it is used for renaming the current file name with new file name.
- The os.path is a module which provides ways of manipulating paths:
  - **isfile():** ➔ return True if the file exists.
  - **exists():** ➔ returns True if dir/file exists.

Module 11: Python Files – Test

Q1) What is the result? Assuming ft1.txt does not exists

```
f = open('ft1.txt')
f.close()
```

Options:

a) FileDoesNotExistsError
b) New file named ft1.txt would be created.
c) FileNotFoundError
d) New file name ft1 would be created

Solution:

Q2) What is the result? Asuming ft1.txt does not exists

```
f = open('ft1.txt', "w")
f.write('how are you')
f.close()
```

Options:

a) Nothing is added to file system.
b) New file named ft1.txt would be created containing "how are you".
c) Empty file named ft1.txt would be created.
d) New file name ft1 would be created containing "how are you".

Solution:

Q3) What is the result. Assuming f1.txt exists and contains a line "hey der"

```
f = open('f1.txt')
f.write('how are you')
f.close()
```

Options:

a) File now contains "hey der how are you"
b) File now contains "how are you"
c) File contains "hey der"
d) io.UnsupportedOperation is thrown.

Solution:

Q4) What is the result if we run following code two times?

```
f = open('ft1', 'a')
f.write('hey der ')
f.close()
```

Options:

a) File now contains "hey der".

b) File now contains "hey der hey der".

c) io.UnsupportedOperation is thrown.

d) File contains no data.

Solution:

Q5) What is the result?

```
f = open('pk1.txt', 'w')
f.write('hello ')
f.close()
f.write('hru ')
f.close()
```

Options:

a) File now contains "hello".

b) File now contains "hru".

c) File now contains "hello hru".

d) Exception is thrown.

Solution:

Q6) you develop a python application for your school. You need to read and write data to a text file. If the file does not exist, it must be created. If the file has content, the content must be removed. Which code should you use?

Options:

A) open('local_data', 'r')

B) open('local_data', 'r+')

C) open('local_data', 'w+')

D) open('local_data', 'w')

Solution:

Q7) you are writing a function that works with files. You need to ensure that the function returns None if the file does not exist. If file does exist, the function must return the first line. You write the following code:

```
import os
def get_first_line(filename, mode):
```

In which order would you arrange the code segments to complete the function.

A) if os.path.isfile(filename):
B) return file.readline()
C) with open(filename, 'r') as file:
D) return None
E) else:

Solution:

Q8) You are creating a function that reads a data file and prints each line of the file. You write the following code. Line numbers are included for reference only.

```
01 import os
02 def read_file(file):
03     line = None
04     if os.path.isfile(file):
05         data = open(file,'r')
06     while line != '':
07         line = data.readline()
08         print(line)
```

The code attempts to read the file even if the file does not exist.
You need to correct the code. Which three lines have indentation problems?

Options:

A. Line 01	B. Line 02	C. Line 03	D. Line 04
E. Line 05	F. Line 06	G. Line 07	H. Line 08

Solution: