File handling

Q 1. What is file handling?

File handling refers to the way in which a program can read from and write to files stored on the computer's hard drive or other external storage devices. In Python, file handling is done using the built-in `open()` function, which opens a file in a specified mode (read, write, append, etc.) and returns a file object.

Once a file is opened, the program can read data from the file, write new data to the file, or modify existing data in the file. After the program has finished working with the file, it must be closed using the `close()` method to ensure that any changes made to the file are saved.

File handling is a critical aspect of most software applications, as it enables programs to store data permanently and access it later. For example, a program may use file handling to save user settings, log data, or store output from previous runs of the program.

Q 2. Can you explain the difference modes of opening a file?

Yes, in Python, the open() function can be used to open a file in different modes, which determine how the file can be accessed. Here are the different modes

1. Read mode (“r”): This mode opens a file for reading only. If the file does not exist, an error will occur. This is the default mode for open().
   Example: file = open(“example.txt”, “r”)
2. Write mode (“w”): This mode opens a file for writing only. If the file already exists, it will be overwritten. If the file does not exist, a new file will be created.
   Example: file =open(“example.txt”, “w”)
3. Append mode (“a”): This mode opens a file for writing, but it does not overwrite the existing file. Instead, any new data that is written to the file is added to the end of the file. If the file does not exist, a new file will be created.
   Example: file = open(“example.txt”, “a”)
4. Read and write mode “r+”: This mode opens a file for both reading and writing. If the file does not exist, an error will occur in read mode.
   Example: file = open(“example.txt”, “r+”)
5. Read and write mode “w+”: This mode opens a file for both reading and writing. If the file does not exist new file will be created in write mode. But file is exist all data will be deleted
   Example: file = open(“example.txt”, “r+”)

6. Exclusive creation mode (“x”): This mode opens a file for exclusive creation, meaning that the file must not already exist. If the file exists, an error will occur.
   Example: file = open(“example.txt”, “x”)

Q 3. How do you create a text file?

In Python, you can create a text file using the built-in open() function in write mode (“w”). Here is an example code that creates a new file called “example.txt” and writes some text to it:

Syntax: open(“example.txt”, “w”)

Q 4. How to read and write to an existing file?

To read from an existing file in Python, you can use the open() function in read mode (“r”) and the read() method to read the contents of the file.

Syntax: file = open(“example.txt”, “r”)

To write to an existing file, you can use the open() function in write mode (“w”) the write() method to add new content to the file.

Syntax: file = open(“example.txt”, “w”)

Q 5. What are some important methods used for reading from a file?

Here are some important methods used for reading from a file in Python.

1. read( ) – is a method used for reading data from a file in Python. It is used to read a specified number of characters from a file, or the entire file if no size is specified. The read() method returns the contents of the file as a string.
2. Readline() – is a method used for reading data from a file in Python. It is used to read a single line from the file.
   The readline() method reads a single line from the file and returns it as a string. If the end of the file is reached, an empty string is returned.
3. Readlines() – is a method used for reading data from a file in Python. It is used to read all the lines of a file and return them as a list.
   The readlines() method reads all the lines of the file and returns them as a list of strings. Each string in the list represents a single line of the file, with the newline character at the end of each line included in the string.

Q 6. What are some common errors that can occur while working with files?

While working with files in Python, some common errors that can occur include.

1. FileNotFoundError: This error occurs when Python is unable to find the specified file. This can happen if the file path is incorrect or if the file does not exist.
2. PermissionError: This error occurs when Python does not have permission to access the specified file. This can happen if the file is locked or if the user does not have the necessary permissions.
3. IOError: This error occurs when there is an input/output error while working with the file. This can happen if the file is corrupted or if there is an issue with the file system.
4. ValueError: This error occurs when the mode used to open the file is invalid. For example, if you try to open a file in read mode when it does not exist, you may get a ValueError.
5. TypeError: This error occurs when the file object is not in the correct format. For example, if you try to write to a file that was opened in read mode, you may get a TypeError.
6. AttributeError: This error occurs when the file object does not have the attribute you are trying to access. For example, if you try to use the write() method on a file object that was opened in read mode, you may get an AttributeError.

It is important to handle these errors appropriately in your code to ensure that your program does not crash unexpectedly. This can be done using try-except blocks to catch and handle the errors that may occur.

Q 7. What is difference between text and binary files?

Text File	Binary File
1. Text files are stored and processed as a series of characters, typically encoded using a specific character encoding such as ASCII or UTF-8.	1. Binary files, on the other hand, are stored and processed as a series of bytes that represent raw data.
2. Text files can be read and edited using a text editor, and their content can be easily understood by humans.	2. Binary files are typically not meant to be read or edited directly by humans, and instead are used to store complex data structures such as images, audio files, video files, and executable programs.
3. When a text file is opened in Python, it is assumed to be in text mode by default, and the contents of the file are treated as a series of characters.	3. In contrast, when a binary file is opened in Python, it is assumed to be in binary mode, and the contents of the file are treated as a series of bytes.

Q 8. Which function allow us to check if we have reached the end of a file?

EOF stands for End of File allow us to check if we have reached the end of a file.

Q 9. List down the steps involved in a processing a large file?

There are 4 steps involved in a processing a large file.

1. Open a file
2. Read or Write (perform operations)
3. Append
4. Close the file

Q 10. What is the difference between write and append mode?

Write Mode	Append Mode
1. Whenever the file is opened in write mode, all the content which was previously present in the file will be overwritten due to the file pointer having it's position at the beginning of the file.	1. Whereas in append mode, the file pointer will be at the end of the file and any existing data won't be overwritten.
2. If the file doesn't exist, a new file is created. This means that if you open a file in write mode and write to it multiple times, only the last write operation will be reflected in the file.	2.If the file doesn't exist, a new file is created. This means that if you open a file in append mode and write to it multiple times, each write operation will be appended to the end of the file.

Q 11. What is the difference between read() ad read(n) functions?

read() Function	read(n) Function
1. The read() function reads the entire contents of the file and returns them as a single string.	1. The read(n) function reads the specified number of bytes (n) from the file and returns them as a string.
2. When read() is called without any arguments, it reads the entire file.	2. If the end of the file is reached before n bytes are read, the function returns the remaining bytes.

Q 12. Differentiate between absolute pathnames and relative pathnames?

Absolute Pathnames	Relative Pathnames
1. An absolute pathname is a complete path that specifies the exact location of a file or directory in the file system.	1. A relative pathname, on the other hand, specifies the location of a file or directory relative to the current working directory.
2. It begins with the root directory, denoted by a forward slash (/), and includes all subsequent subdirectories separated by forward slashes.	2.It does not begin with a forward slash and can only reference files or directories that are located within the current working directory or its subdirectories.
3. For example, /home/user/Documents/file.txt is an absolute pathname that specifies the exact location of the file.txt file in the Documents directory of the user's home directory.	3. For example, if the current working directory is /home/user, then Documents/file.txt is a relative pathname that specifies the location of the file.txt file in the Documents directory, which is located in the current working directory.

Q 13. Differentiate between file modes r+ and w+ with respect to python?

r+ Mode	w+ Mode
1.When you open a file in r+ mode, the file pointer is placed at the beginning of the file. This means that if you start writing to the file, it will overwrite the existing data from the beginning of the file. You can use the seek() method to move the file pointer to a specific location in the file before writing.	1.When you open a file in w+ mode, the file pointer is placed at the beginning of the file, just like in r+ mode. This means that if you start writing to the file, it will overwrite the existing data from the beginning of the file. However, since the file was truncated to zero length when it was opened, there is no existing data to overwrite. You can use the seek() method to move the file pointer to a specific location in the file before writing.
2. The r+ mode opens a file for reading and writing. If the file does not exist, it will raise an error.	2. The w+ mode opens a file for reading and writing as well, but it also truncates the file to zero length. If the file does not exist, it creates a new file.

Q 14. What is file mode? Name the default file mode.

In Python, file mode refers to the method in which a file is opened. It specifies whether the file should be read, written, or appended to, as well as whether it should be opened in binary or text mode.
The default file mode in Python is 'r', which stands for read mode.