Q1. Describe the differences between text and binary files in a single paragraph.

Answer: Text files are organized around lines, each of which ends with a newline character ('\n'). The source code files are themselves text files. A binary file is the one in which data is stored in the file in the same way as it is stored in the main memory for processing.

Q2. What are some scenarios where using text files will be the better option? When would you like to use binary files instead of text files?

Answer: Text files are less prone to get corrupted as any undesired change may just show up once the file is opened and then can easily be removed whereas binary files can be used instead of text files for image data/video/audio. For many types of data, binary representations are much more space-efficient. For instance, you could represent an image as a text file. But any color represented as a 24-bit RGB value in a binary file has to be represented as a six-character code (e.g. FFFFFF for white) in text, which is literally twice as many bits. Another reason for using binary formats is that they can map directly onto the internal representation of the data that the program is using. If my program uses a complex data structure that's full of integer, float, and byte values, when I read it in from a text file I have to parse every character of text and interpret it, spending processing time (and writing logic) to figure how to convert the 88 bits in "2983102931," into the 32-bit integer that the text represents. If I read it from a binary file, the program just copies those 32 bits from the file into memory.A binary file is the one in which data is stored in the file in the same way as it is stored in the main memory for processing. It is stored in binary format instead of ASCII characters. It is normally used for storing numeric information (int, float, double).

Q3. What are some of the issues with using binary operations to read and write a Python integer directly to disc?

Answer: To open a file in binary format, add 'b' to the mode parameter. Hence the "rb" mode opens the file in binary format for reading, while the "wb" mode opens the file in binary format for writing. Unlike text files, binary files are not human-readable. When opened using any text editor, the data is unrecognizable

Q4. Describe a benefit of using the with keyword instead of explicitly opening a file.

Answer: The with keyword in Python is used as a context manager. As in any programming language, the usage of resources like file operations or database connections is very common. But these resources are limited in supply. Therefore, the main problem lies in making sure to release these resources after usage. If they are not released, then it will lead to resource leakage and may cause the system to either slow down or crash.

As we know, the open() function is generally used for file handling in Python. But it is a standard practice to use context managers like with keywords to handle files as it will automatically release files once its usage is complete.

Q5. Does Python have the trailing newline while reading a line of text? Does Python append a newline when you write a line of text?

Answer: Here, are important characteristics of Python read line: Python readline() method reads only one complete line from the file given. It appends a newline ("\n") at the end of the line. If you open the file in normal read mode, readline() will return you the string. The default value of the end parameter of the built-in print function is \n , so a new line character is appended to the string

Q6. What file operations enable for random-access operation?

Answer: Random access file in C enables us to read or write any data in our disk file without reading or writing every piece of data before it. ftell() is used to find the position of the file pointer from the starting of the file. rewind() is used to move the file pointer to the beginning of the file.vRandom access files permit nonsequential, or random, access to a file's contents. To access a file randomly, you open the file, seek a particular location, and read from or write to that file. This functionality is possible with the SeekableByteChannel interface.

Q7. When do you think you'll use the struct package the most?

Answer: The struct module in Python is used to convert native Python data types such as strings and numbers into a string of bytes and vice versa. What this means is that users can parse binary files of data stored in C structs in Python.

It is used mostly for handling binary data stored in files or from network connections, among other sources. struct.pack() is the function that converts a given list of values into their corresponding string representation. It requires the user to specify the format and order of the values that need to be converted.

Q8. When is pickling the best option?

Answer: Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

Q9. When will it be best to use the shelve package?

Answer: The shelve module in Python's standard library is a simple yet effective tool for persistent data storage when using a relational database solution is not required. The shelf object defined in this module is dictionary-like object which is persistently stored in a disk file.

Q10. What is a special restriction when using the shelve package, as opposed to using other data dictionaries?

Answer:
The shelf dictionary has certain restrictions. Only string data type can be used as key in this special dictionary object, whereas any picklable Python object can be used as value