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

Text files store data as human-readable text, with lines delimited by newline characters, and may include character encoding. Binary files store data in a raw format, preserving exact bytes without any character encoding, allowing for non-textual data.

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?

Text files are preferable for storing and reading human-readable data like configuration files, logs, and textual information. Binary files are suitable for handling non-textual data like images, audio, video, and other complex data structures.

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

Using binary operations to read/write Python integers directly to disk can lead to endianness issues on different platforms, where the byte order might not match, potentially resulting in incorrect data interpretation.

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

Using the with keyword (context manager) to open files offers automatic resource management. It ensures proper file closure and exception handling, reducing the risk of resource leaks and enhancing code readability.

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?

When reading a line of text, Python includes the trailing newline character (\n) unless it's the last line. When writing a line of text, Python does append a newline character by default unless specified otherwise.

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

Random-access operations are enabled by the seek() and tell() methods, which allow you to move the file pointer to a specific position and determine its current position, respectively.

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

The struct package is often used for handling binary data, such as parsing and constructing binary file formats, network protocols, or interacting with hardware interfaces.

Q8. When is pickling the best option?

Pickling is a good option when you need to serialize Python objects to a format that preserves their data and structure, allowing for easy storage and later retrieval, especially for complex data types like custom classes or data structures.

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

The shelve package is best suited when you need to store and retrieve Python objects persistently using a dictionary-like interface, particularly for large datasets where memory consumption might be an issue.

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

A special restriction with the shelve package is that the keys used to access stored data must be strings. Unlike other data dictionaries, where keys can be of various types, shelve requires keys to be strings for efficient storage and retrieval.