Q1. Differences Between Text and Binary Files:

- Text files store data as plain text, using character encoding like ASCII or UTF-8, making them human-readable. Binary files, on the other hand, store data in a format not directly readable by humans, containing a sequence of bytes.

Q2. Scenarios for Text and Binary Files:

- Text Files: Suitable for storing human-readable data like configurations, scripts, or simple data interchange. Ideal when readability is essential.

- Binary Files: Better for storing complex data structures, multimedia files, or when raw data representation is crucial.

Q3. Issues with Binary Operations on Python Integer:

- Endianness: The system's endianness can impact the interpretation of binary data.

Q4. Benefit of 'with' Keyword:

- Context Management: 'with' ensures proper resource management (like file closure) even if an exception occurs, enhancing code readability and reducing boilerplate code.

Q5. Trailing Newline in Python:

- Reading: Python retains the newline character (`\n`) when reading a line of text.

- Writing: Python appends a newline by default when writing a line of text.

Q6. File Operations Enabling Random Access:

- Using the `seek()` method allows random access within a file.

Q7. Usage of the struct Package:

- The struct package is commonly used for packing and unpacking binary data, especially when dealing with C-style structures.

Q8. Best Option for Pickling:

- Pickling is best for serializing complex Python objects or data structures for storage or transmission.

Q9. Best Use of the shelve Package:

- The shelve package is beneficial when you need persistent storage for Python objects and want dictionary-like access.

Q10. Special Restriction with shelve Package:

- The keys in a shelve object must be strings, as the shelve package internally uses the keys as filenames.