ASSIGNMENT - 18

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

Ans: Text files store data as human-readable characters, while binary files contain encoded data in a format readable by machines. Text files typically store plain text, allowing easy human interpretation, whereas binary files store data in a more compact, structured, and machine-readable form, which might include images, videos, or executable code.

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?

Ans: Text files are preferable for storing configuration data, logs, or any content meant for human consumption due to their readability. Binary files are better suited for storing complex data structures, such as images, audio, or database files, where maintaining the exact structure and representation is crucial.

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

Ans: When directly reading or writing Python integers to disk using binary operations, issues arise due to differences in integer sizes and endianness between systems. This can cause compatibility problems when transferring files between platforms with different architectures.

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

Ans: Using the 'with' keyword in Python for file operations ensures automatic resource management by implicitly handling file closing. It guarantees that the file will be properly closed after its suite finishes, reducing the chance of memory leaks or other resource-related issues.

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?

Ans: When reading lines from a text file in Python, the trailing newline is preserved. When writing to a file using write() without adding a newline character (\n), Python won't append a newline at the end of the written content.

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

Ans: The seek() and tell() methods in Python file handling enable random-access operations. seek() moves the file pointer to a specific position, and tell() reports the current position of the file pointer.

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

Ans: The 'struct' package in Python is commonly used when dealing with binary data and raw byte manipulation, especially when working with binary file formats, network protocols, or low-level data representation tasks.

Q8. When is pickling the best option?

Ans: Pickling in Python is best suited for serializing and deserializing complex Python objects, making it convenient for storing and retrieving Python-specific data structures or objects without losing their inherent structure.

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

Ans: The 'shelve' package becomes useful when there's a need to store and retrieve Python objects like dictionaries persistently, offering a dictionary-like interface for persistent storage.

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

Ans: 'shelve' doesn't support simultaneous write access from multiple programs or threads due to its underlying use of the dbm module, which doesn't handle concurrent write operations well. This limitation is different from using other data dictionaries that might allow concurrent access with proper synchronization mechanisms.