**Assignment - 18**

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

Ans: Text files contain human-readable characters encoded in a specific format, such as ASCII or Unicode, and are typically used to store textual data like documents or configuration files. Binary files, on the other hand, store data in a format that is not human-readable and may include non-textual data, such as images, audio, or executable files. Binary files represent data in its raw form, using sequences of bytes, and are processed differently from text files, often requiring specialized handling for reading and writing.

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 scenarios where human readability and interoperability with different systems are important, such as storing configuration settings, logging information, or exchanging data with other programs. Binary files are preferred when dealing with non-textual data or when precise control over data representation and storage efficiency are required, such as storing images, videos, or serialized objects.

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

Ans: Some issues with using binary operations to read and write Python integers directly to disk include potential platform-dependent differences in byte order (endianness) and the need for manual conversion between Python integer objects and their binary representation. Additionally, directly writing binary data may introduce challenges in ensuring data portability and compatibility across different systems.

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

Ans: A benefit of using the with keyword instead of explicitly opening a file is that it automatically handles the closing of the file once the associated block of code is executed or an exception occurs. This ensures that system resources are properly released and file handles are closed, even in the event of errors or exceptions, leading to cleaner and more reliable code.

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: Python does not have a trailing newline while reading a line of text by default. However, when writing a line of text using the write() method, Python does not automatically append a newline character unless specified explicitly. It's up to the programmer to include or exclude newline characters as needed.

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

Ans: File operations such as seek() and tell() enable random-access operation, allowing you to move the file pointer to a specific position within the file (seek) and retrieve the current position of the file pointer (tell). This capability allows for efficient reading and writing of data at arbitrary locations within the file, rather than just sequentially from the beginning to the end.

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

Ans: The struct package is commonly used when dealing with binary data formats and low-level data manipulation tasks, such as parsing binary file formats, network protocols, or interfacing with hardware devices. It provides functions for converting between Python values and C structs represented as byte strings, making it useful for handling binary data with specific byte layouts and data types.

Q8. When is pickling the best option?

Ans: Pickling is often the best option when you need to serialize Python objects into a byte stream for storage or transmission, preserving their internal state and structure. Pickling allows for the serialization of complex data structures, including nested objects and custom classes, and supports various data types and objects not supported by other serialization formats.

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

Ans: The shelve package is best used when you need a persistent dictionary-like storage solution that can store Python objects directly to disk, preserving their internal structure and state between sessions. It provides a simple interface for storing and retrieving objects using key-value pairs, offering persistence across program executions without the need for manual serialization or deserialization.

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

Ans: A special restriction when using the shelve package, compared to other data dictionaries, is that keys and values must be pickleable objects. Since shelve internally uses pickling to serialize and deserialize objects, any objects stored as keys or values must be serializable using the pickle protocol. This limitation ensures that objects can be properly serialized and deserialized when stored in the shelve database.