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

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?

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

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

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?

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

Q7. When do you think you&#39;ll use the struct package the most?

Q8. When is pickling the best option?

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

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

A1. Text files contain human-readable text data, while binary files contain non-text data that is not human-readable. Text files consist of characters, such as letters, numbers, and symbols, that can be easily edited with a text editor, whereas binary files consist of binary data, such as machine code, images, and sound, that require specific software or tools to interpret and modify.

A2. Text files are better suited for scenarios where the data needs to be human-readable and editable, such as storing configuration files, log files, and source code. Binary files are ideal for storing non-text data that needs to be processed by specific applications, such as images, videos, and executable files. Binary files are also useful for storing large amounts of data efficiently, such as in scientific or financial applications.

A3. One issue with using binary operations to read and write a Python integer directly to disc is that the byte order, or endianness, of the integer may be different on different systems. This can result in unexpected behavior when reading or writing the integer. Another issue is that binary files are not human-readable, which can make debugging and troubleshooting difficult.

A4. The with keyword provides a cleaner and more concise way to open and close files in Python. It ensures that the file is closed automatically after use, even if an error occurs. This helps to prevent resource leaks and improves the readability of the code.

A5. When reading a line of text, Python includes the trailing newline character, if present. When writing a line of text, Python does not append a newline character by default, but it can be added using the newline parameter of the write() method.

A6. The seek() and tell() file operations enable random-access operation, allowing you to read or write data at any position in the file. The seek() method moves the file pointer to a specified position, while the tell() method returns the current position of the file pointer.

A7. The struct package is most commonly used in scenarios where binary data needs to be packed and unpacked in a specific format, such as in network protocols, file formats, and data serialization.

A8. Pickling is the best option when you need to serialize complex Python objects, such as classes, functions, and instances, into a byte stream that can be stored or transmitted. Pickling preserves the state of the objects, including their data and behavior, making it easy to restore them later.

A9. The shelve package is best used when you need to store and retrieve persistent objects, such as dictionaries, lists, and instances, in a key-value database format. The shelve module provides a simple interface to a powerful database backend, allowing you to store and retrieve objects as if they were in memory.

A10. The shelve package has a special restriction that the keys in the database must be strings. This means that you cannot use objects, such as integers or tuples, as keys directly. Instead, you need to convert them to strings first, using a suitable encoding scheme.