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

There are mainly two types of data files — text file and binary file. A text file consists of human readable characters, which can be opened by any text editor. On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

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?

The text looks readable to a human or at least moderately sane. Even if it contains a heavy proportion of punctuation symbols (like HTML, RTF, and other markup formats), there is some visible structure and it’s not seemingly random garbage.

The data format is usually line-oriented. Each line could be a separate command, or a list of values could put each item on a different line, etc. The maximum number of characters in each line is usually a reasonable value like 100, not like 1000.

Non-printable ASCII characters are discouraged or disallowed. Examples include the NUL byte (0x00), DEL byte (0x7F), and most of the range 0x01 to 0x1F (except tab, carriage return, newline, etc.). Some text editors silently convert or discard these bytes, which is why binary files should never be edited in a text editor.

A binary file always needs a matching software to read or write it. For example, an MP3 file can be produced by a sound recorder or audio editor, and it can be played in a music player or audio editor. But an MP3 file cannot be played in an image viewer or a database software.

Some binary formats are popular enough that a wide variety of programs can produce or consume it. Image formats like JPEG are the best example – not only can they be used in image viewers and editors, they can be viewed in web browsers, audio players (for album art), and document software (such as adding a picture into a Word doc). But other binary formats, especially for niche proprietary software, might have only one program in the world that can read and write it. For example, a high-end video editing software might let you save your project to a file, but this software is the only one that can understand its own file format; the binary file will never be useful anywhere else.

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

It is important to note that, in our case, binary data happens to contain printable characters, like alphabets, newline etc. However, this will not be the case most of the time. It means that with binary data we can't reliably use readline() and file object (as an iterator) to read the contents of a file because might be no newline character in a file. The best way to read binary data is to read it in chunks using the read() method.

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

Using with means that the file will be closed as soon as you leave the block. This is beneficial because closing a file is something that can easily be forgotten and ties up resources that you no longer need.

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?

Python readline() is a file method that helps to read one complete line from the given file. It has a trailing newline (“\n”) at the end of the string returned.

The new line character in Python is \n . It is used to indicate the end of a line of text. You can print strings without adding a new line with end = <character> , which <character> is the character that will be used to separate the lines.

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

A file is a container in computer storage devices used for storing data.

When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order:

Open a file

Read or write (perform operation)

Close the file

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

It is used mostly for handling binary data stored in files or from network connections, among other sources. This process needs to be done at the start of the program.

Q8. When is pickling the best option?

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it’s the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. The pickled byte stream can be used to re-create the original object hierarchy by unpickling the stream. This whole process is similar to object serialization in Java or .Net

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

A “shelf” is a persistent, dictionary-like object. The difference with “dbm” databases is that the values (not the keys!) in a shelf can be essentially arbitrary Python objects — anything that the pickle module can handle. This includes most class instances, recursive data types, and objects containing lots of shared sub-objects. The keys are ordinary strings.

shelve.open(filename, flag='c', protocol=None, writeback=False)

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

The shelf dictionary has certain restrictions. Only string data type can be used as key in this special dictionary object, whereas any picklable Python object can be used as value. This is the base class for shelf implementations.