Python Advanced Assignment 18

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

Ans: The differences between Text Files and Binary Files are:

Text files are special subset of binary files that are used to store human readable characters as a rich text document or plain text document. Text files also store data in sequential bytes but bits in text file represents characters. where as **Binary files** are those typical files that store data in the form of sequence of bytes grouped into eight bits or sometimes sixteen bits. These bits represent custom data and such files can store multiple types of data (images, audio, text, etc) under a single file.

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 less prone to get corrupted as any undesired change may just show up once the file is opened and then can easily be removed whereas binary files can be used instead of text files for image data/video/audio. For many types of data, binary representations are much more space-efficient. For instance, you could represent an image as a text file. But any color represented as a 24-bit RGB value in a binary file has to be represented as a six-character code (e.g. FFFFFF for white) in text, which is literally twice as many bits. Another reason for using binary formats is that they can map directly onto the internal representation of the data that the program is using. If my program uses a complex data structure that's full of integer, float, and byte values, when I read it in from a text file I have to parse every character of text and interpret it, spending processing time (and writing logic) to figure how to convert the 88 bits in "2983102931," into the 32-bit integer that the text represents. If I read it from a binary file, the program just copies those 32 bits from the file into memory.

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

Ans: When we read or write a python integer using binary operations

1. Binary operations deal with raw data
2. One needs to identify how many bytes one would read or write.

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

Ans: When a file is opened using the **with** keyword,it automatically closes the file if some exceptions occur after opening a file, or at the end of the file . Thereby not leaving a file in open mode and also there would no need to explicitly close 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?

Ans: Yes, Python have the trailing newline while reading a line of text. When we write, a newline has to be provided in python explicitly to append a newline. Hence Python does not append a newline implicitly

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

Ans: The file operations enable for random-access operation are **seek()** and **tell()**

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

Ans: The **struct** package is mostly used while converting a common python language types into **C** language types.

Q8. When is pickling the best option?

Ans: Pickling is best option for creating a new binary file using python. 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?

Ans: Shelve package is used to pickle data but treats the entire file as dictionary. 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.

Syntax: 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?

Ans: Only string data type can be used as key in this special dictionary (shelve package) object, whereas any pickle Python object can be used as value.