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

**Text File:**

* Bits represent character.
* Less prone to get corrupt as changes reflect as soon as the file is opened and can easily be undone.
* Can store only plain text in a file.
* Widely used file format and can be opened using any simple text editor.
* Mostly .txt and .rtf are used as extensions to text files.
* Easy for human to read it.

**Binary File:**

* Bits represent a custom data.
* Can easily get corrupted, even a single bit change may corrupt the file.
* Can store different types of data (image, audio, text) in a single file.
* Developed especially for an application and may not be understood by other applications.
* Can have any application defined extension.

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?

**Text file:** When we wan to store some information in such a way that every one can easily understand. Means readable and understood by the human not to the machine then in that case we prefer using text file. We can store any kind of information by using just simply writing the characters. We do not write the information in the form of bits and all. So that if any thing is written wrong in that and cause some erro then it can be easily handle.

**Binary file:** When we want to make some kind of a file which can only read by the machine not by the human then we use binary file. With this file we can store the information in bits. We can only be able to see the final functionality of the file like pdf file, audio file, video files, etc. We can only see the fina function but do not see the available data inside that which helped in maing that application.

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

While reading,read method returns a sequence of bytes as a string. With using binary operaions i.e. using bitwise operator the integers are first converted into binary and then operations are performed on bit by bit. This is a very lengthy task from usual steps. More over it is not possible for a reader to read the integers in binary format.

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

**with** keyword is very useful while working with a file. This gives u a privellage that u dnt need to worry about closing the file after the work is completed.

Sometimes it happen that we have written a long codes and we somehow we forgot to close the file. This may lead to error with the file so we use **with** keyword.

The with statement itself ensures proper acquisition and release of resources

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?

**read line:** Yes while reading a line using object.read() function everynew line will be readed

**write line:** No it does not append a new line. It will just write the new data side by side. If we use **'w'** as a mode then this will **truncate** the everyline if available there in the file and write the new text which we have given. And again if we give the new text more than one time in the same code, it will get append to the next right.

**with** open('d.txt') **as** r:

d**=**r**.**read()

print(d)

hi my name is rahul

hello

world

india

In [16]:

**with** open('s.txt','a',newline**=**'\n') **as** w:

w**.**write('rahul')

w**.**write('ineuron')

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

**RandomAccess File** class is used for reading and writing to random access file.

**read** and **write** file handling operation enable random access operations. With these operations, random access file allows you to read and write a small chunk of data at any position in the file, using a file pointer.

**Following are some of the file operations:**

* close(): It closes this random access fle stream and releases any system resources associated with the stream.
* write(): It writes the specifed byte to this fle
* read(): It reads a byte of data from this fle.
* length(): It returns the length of this fle
* seek(long pos): moves the file pointer to a specified position in the file. The offset is measured in bytes from the beginning of the file. At this position, the next read or write occurs.

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

Whenever suppose we need to convert the data types (like string and number) into string of bytes we use **struct** module.

When we need to parse the binary files of data stored in C into python we generally used this module

Q8. When is pickling the best option?

pickling is storing some data in the binary form to further functionality or operation of application. The data gets converted into binary format by suing pickle module. It best when we are creating some application like AI application and then we need to load it for deployment.

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

When we need to store the data in constant/continuos manner such that we do not require the relational data base, then we use shelve package. The shelf object defined in this module is dictionary-like object which is persistently stored in a disk file. This creates a file similar to dbm database. Only string data type can be used as key in this special dictionary object.

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

One Restriction of shelve objects is that the keys to the objects must be strings, but the values stored in a shelve object can be any Python object, as long as it can be written with the pickle module