FILE UPLOAD APP INTO ORACLE CLOUD

The file upload process has various sections which addresses the different requirements as stated in the objectives. The program is designed to be flexible and parameter driven, albeit it can still be run with bare bone inputs.

Upon start, the first check is done to get the size of the file. If the size is less than 5 MB(or whatever is optimum for the network or the server) then the system proceeds to upload it on to the server without any chunking.

If however the size of the file is more than 5 MB, then the size per chunk is calculated based on the split parameter. If it is less than 5 MB then the system overrides it and sets it as 5 MB.

It then proceeds to encrypt the file. The encryption routine used in this application is a collection of cryptographic modules implementing various algorithms and protocols. More details can be found here https://pypi.python.org/pypi/crypto/1.4.1.
(Note: This encryption requires a key. For the purpose of this assignment a random key was chosen. However in an actual working environment a key can be generated on the fly and passed on to the server to decrypt the file once it is received on the other end).

Once it is able to successfully encrypt the file it then reads the chunk one by one and assigns it to a thread **asyncronously.** The purpose of this mode is so that the process does not wait for one upload to end before it begins the other. The fact that each chunk is loaded into the server independently ensures a concurrent and speedy approach to the inherent issue of uploading big files to the server. It uses a special python module to split up the file (real time) into segments. The module is called **filechunkIO** and more details can be found at this link https://pypi.python.org/pypi/filechunkio. This module allows the benefit of splitting it up without the need to actually write the segments, which in turn helps to reduce the load on memory and speed up performance.

Once it finishes loading all the segments into the server it originates a REST API call with the filename, the original file size and a key(for encryption). The program siting on the server side reads the chunks and ‘stiches’ them back together to recreate the original file. It then deletes the folder in which the segments were loaded and copies the file into its parent folder. It also matches the size of that file to the one it receives from the client to determine if it was successfully recreated. It also uses the key to decrypt the file.

FILE UPLOAD APP INTO ORACLE CLOUD

A **wrapper** was also created in order to facilitate calling the REST API for generating the token and uploading content, creating folders and/ deleting the same. It is kept in the same folder and will also be included in the final delivery. The wrapper is called **file-upload**.

Fault Tolerance and Resume mechanism:

In order to maintain the persistence of the segments that are uploaded, a key value db is used. It is called pickleDB and is inspired by redis. More information can be found here
https://pythonhosted.org/pickleDB/

This is the mechanism used to resume file upload.

Start of Application :

1. Lookup the file used to track the progress of the file upload.
2. if file not found, then it means a new upload.
3. If 2 is true then create a ‘dict’ and start the upload(skip 4)
4. If 2 is not true (which means that there is a previous upload)
5. if step 4 is true, get the dict, and then upload only the segments which are not marked as uploaded.
6. As each segments are uploaded the file is updated to indicate that it has been uploaded successfully.
7. If all the segments are uploaded then delete the key value pair(No need to keep track of the file upload)