***Code analysis***

**Line[1]:** importing of basic libraries for the project

**Line[2]:**Here we read the file from their paths and

Letter ‘r’ mentioned before file path is for reading the

File. Now variables spam\_folder and ham\_folder have

File path with instruction to read the file.

Now for spam\_filenames and ham\_filenames we use the os.listdir() with files path passed with action to be performed. This function gets the list of all files and directories in the specified path.

We have used comprehension list method for less time complexity

Hence we read all the files in sorted order from that given path which have their name length greater than 20 are taken and other are excluded. These files are in list format in ham\_filenmaes and spam\_filenames.

**Line[3]:**Checking length of each file and find their ratio

**Line[4]:**Email module used is for managing email messages and is not designed to do any sending. Overall structure has three components namely object\_model(EmailMessage API),parser and generator.

The parser takes the serialized version of an email message (a stream of bytes) and converts it into a tree of [EmailMessage](https://docs.python.org/3/library/email.message.html" \l "email.message.EmailMessage" \o "email.message.EmailMessage) objects. The generator takes an [EmailMessage](https://docs.python.org/3/library/email.message.html" \l "email.message.EmailMessage" \o "email.message.EmailMessage) and turns it back into a serialized byte stream.

Email.policy has a set of attributes and method for controlling and changing the behaviour of email package during use.

In this block we are making new files for ham mails and spam mails with their mail attached to their respective filenames

For this we have created a function named load\_email with two parametres 🡪 is\_spam which indicates if a filename is for ham or spam and second parameter is filename.

**1st line of function** 🡪 It is to set if the file is for spam or ham

**2nd line 🡪** In this we have used os.path.join which concatenates both files together as f

**3rd line🡪** here now we are reading the file through bytesparser which is here to use the RFC correct line separator characters when creating the binary string

**Line[5]:**We have taken testingmail and then performed certain action to show certain properties or attributes of the mail.

**Line[6]:**