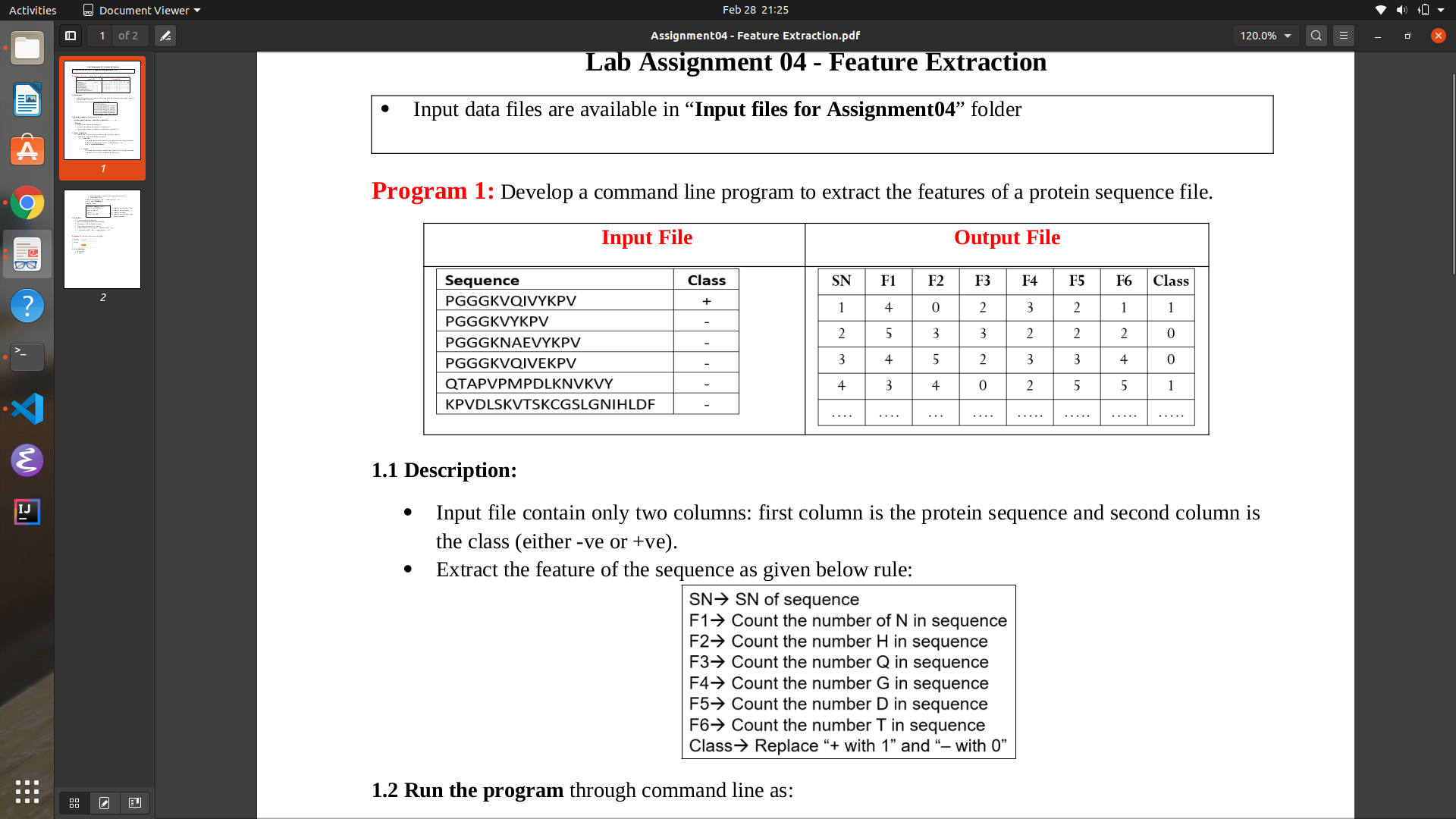
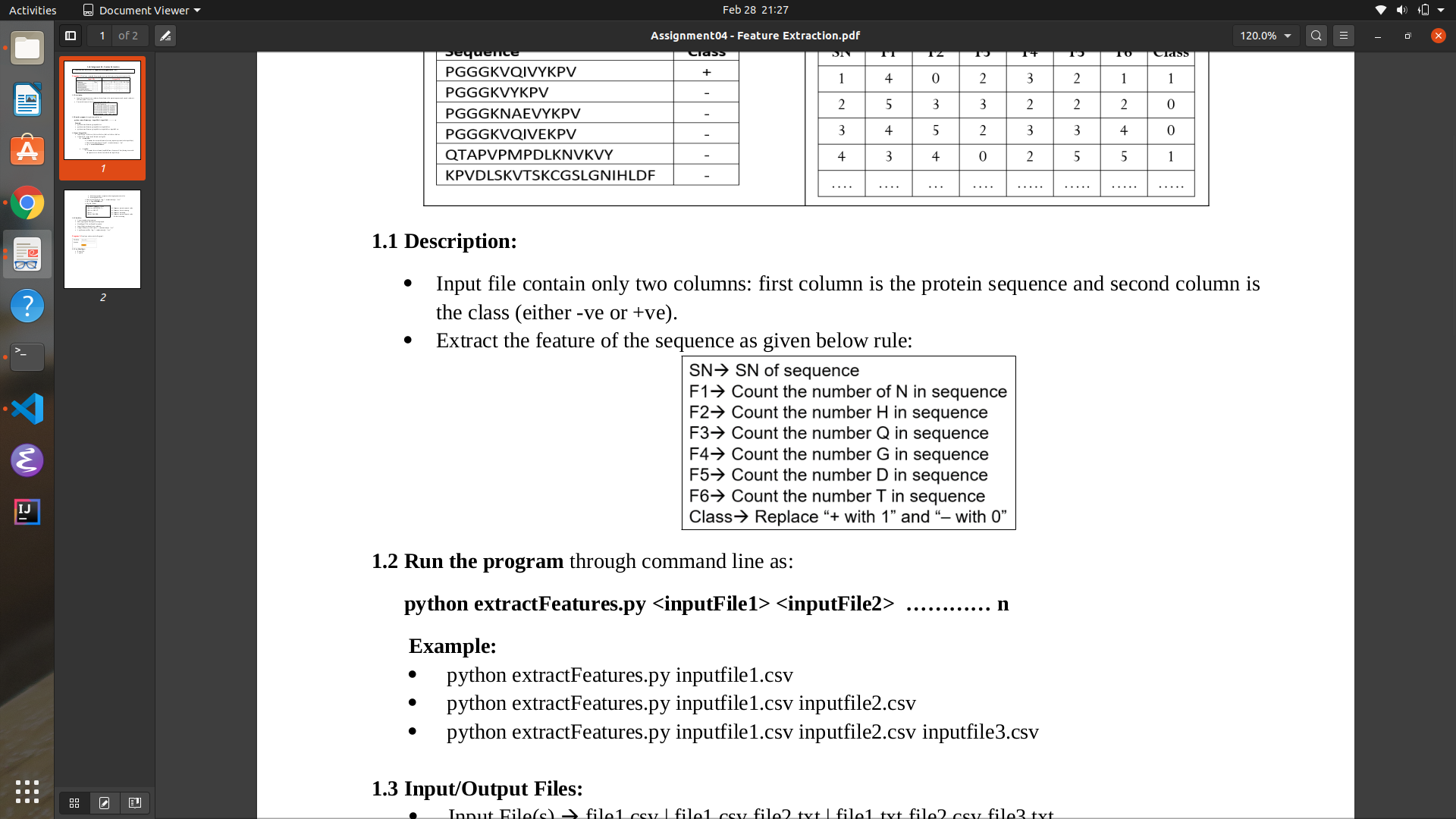
**Experiment 4**

Program 1: Develop a command line program to extract the features of a protein sequence file.



1.1 Description:

* Input file contains only two columns: first column is the protein sequence and second column is the class (either -ve or +ve).
* Extract the feature of the sequence as given below rule:



1.2 Run the program through command line as:

***python extractFeatures.py <inputFile1> <inputFile2> ............ n***

Example:

* python extractFeatures.py inputfile1.csv
* python extractFeatures.py inputfile1.csv inputfile2.csv
* python extractFeatures.py inputfile1.csv inputfile2.csv inputfile3.csv

1.3 Input/Output Files:

* Input File(s) file1.csv | file1.csv file2.txt | file1.txt file2.csv file3.txt
* Output Files : One result file and one log file
* Result file: It contains the extracted features for every sequence present in the input file(s).

Name of the result file *“result-” + str(time.time()) + “.csv”*

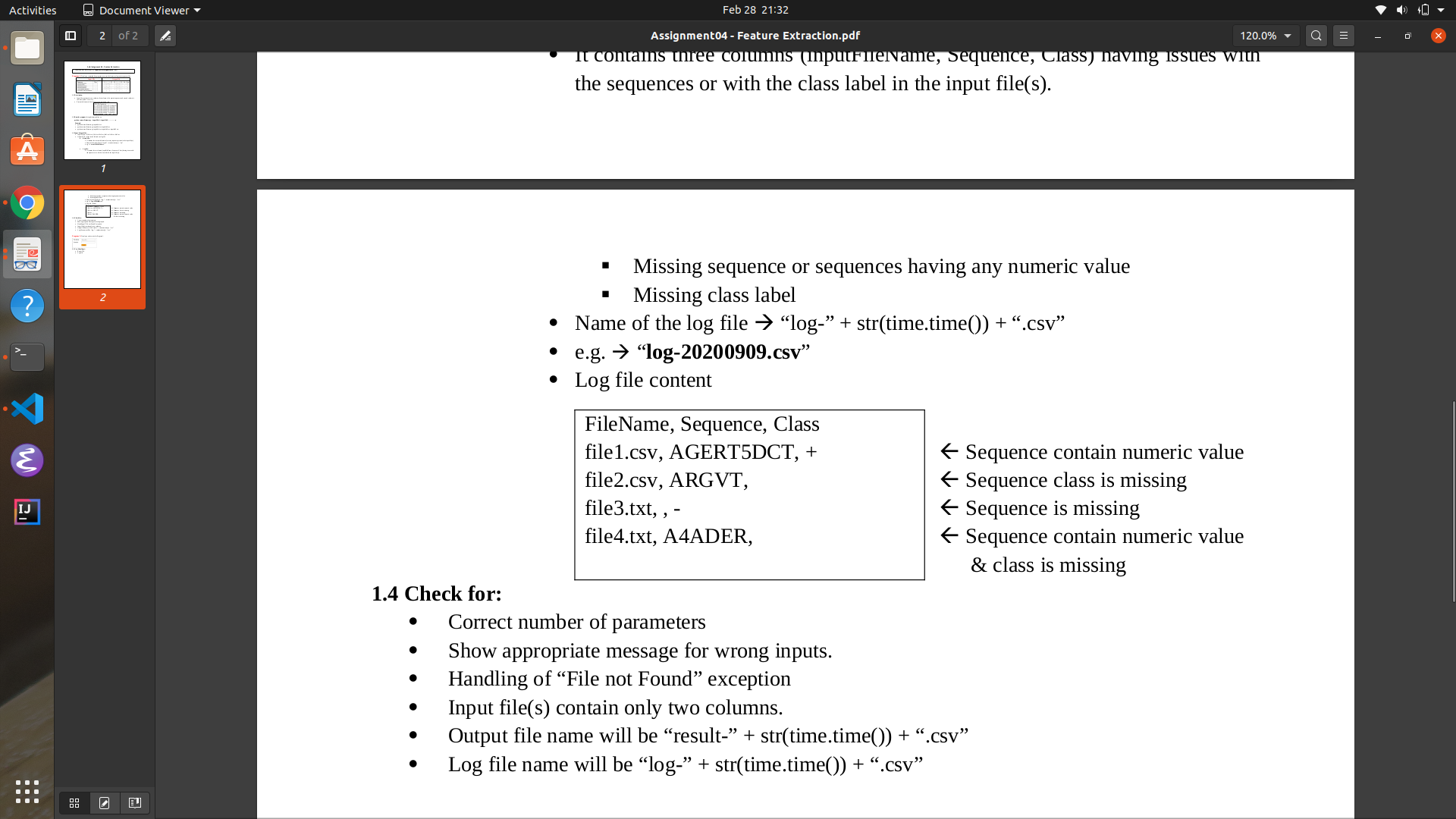
e.g. “*result-20202109.csv*”

* Log file: It contains three columns (inputFileName, Sequence, Class) having issues with the sequences or with the class label in the input file(s). Missing sequence or sequences having any numeric value Missing class label

Name of the log file *“log-” + str(time.time()) + “.csv”*

e.g. *“log-20200909.csv”*

**Log file content:**



1.4 Check for:

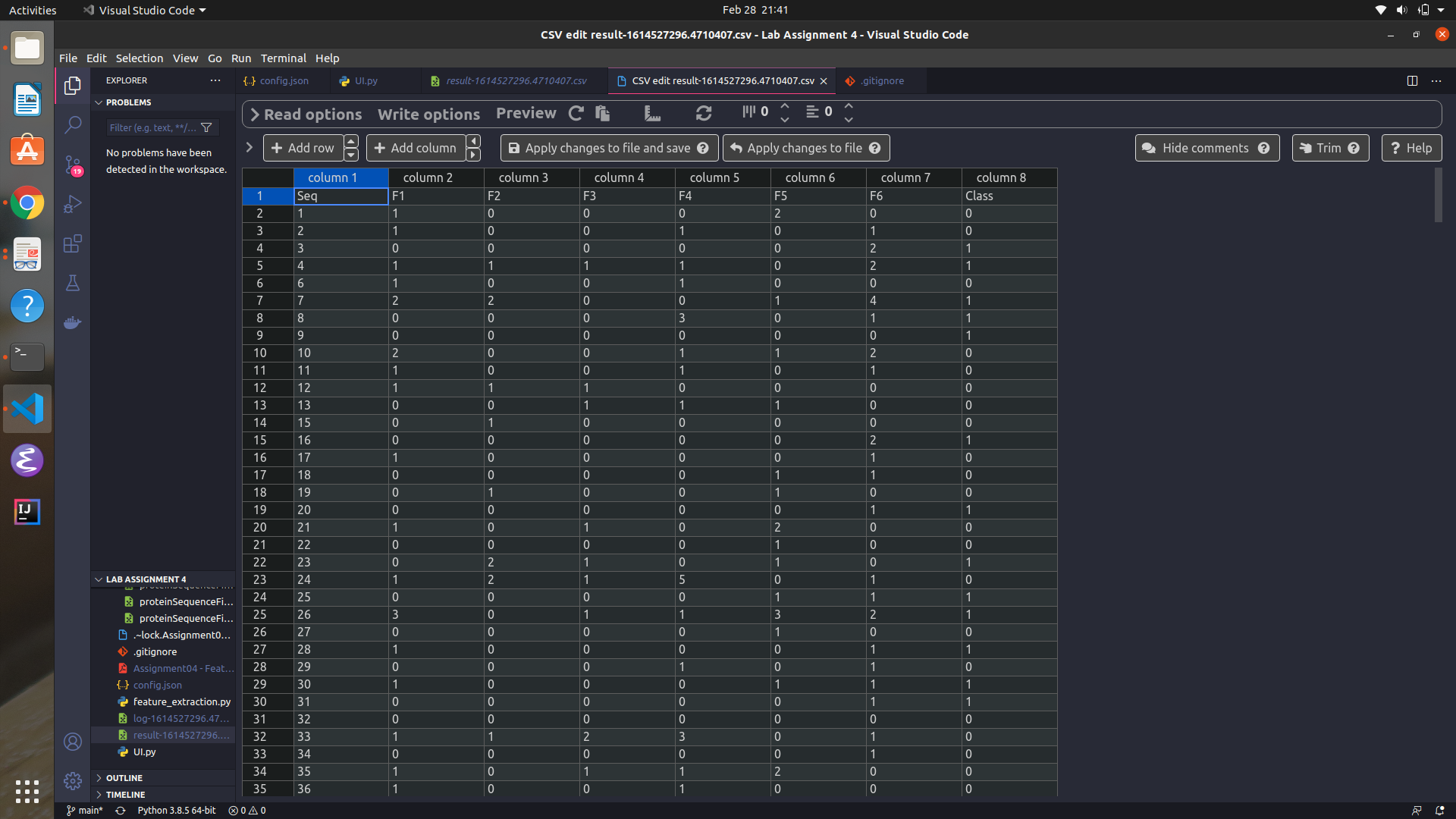
* Correct number of parameters
* Show appropriate message for wrong inputs.
* Handling of “File not Found” exception
* Input file(s) contain only two columns.
* Output file name will be “result-” + str(time.time()) + “.csv”
* Log file name will be “log-” + str(time.time()) + “.csv”

Code:

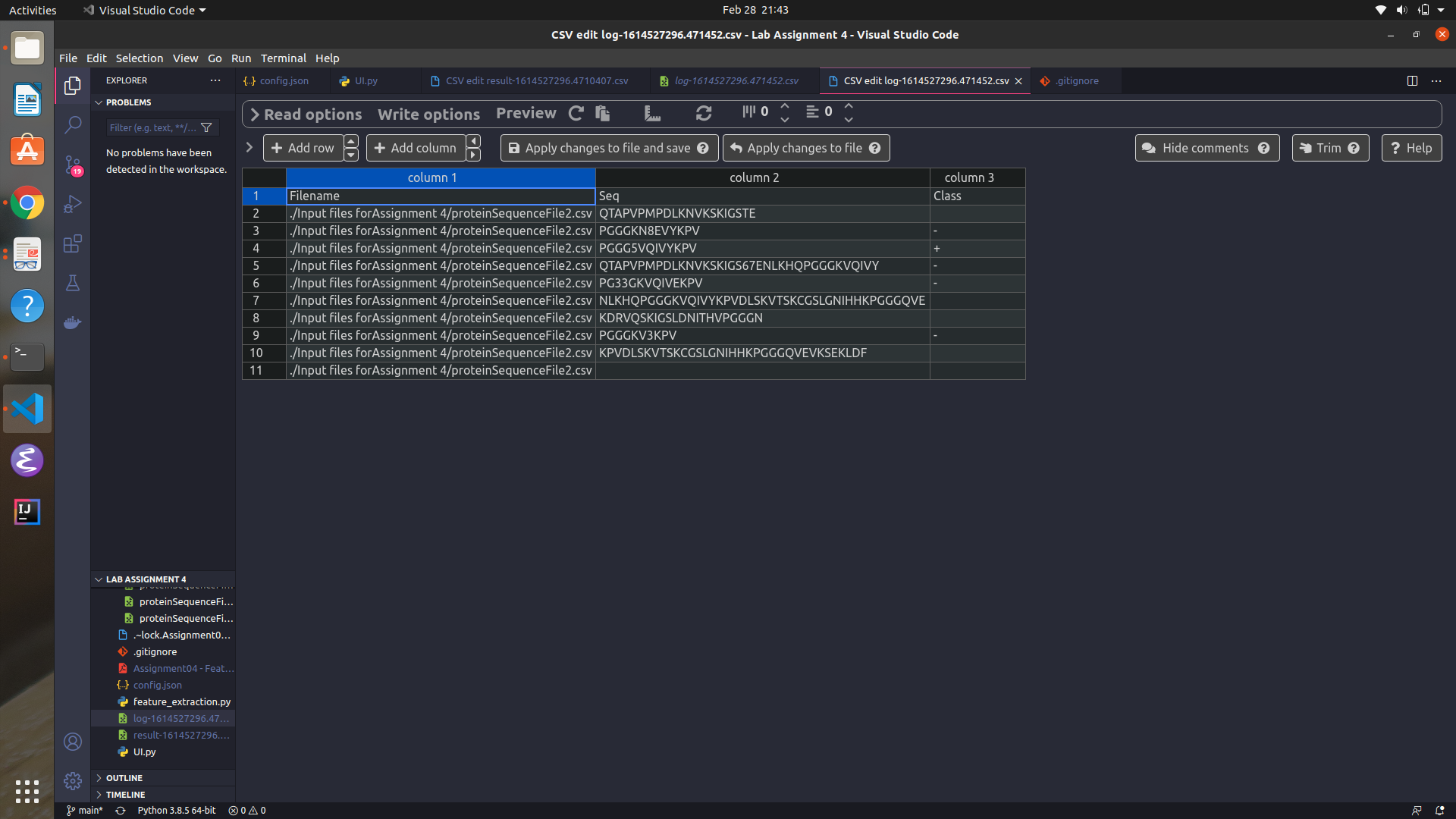
|  |
| --- |
| **import** **sys**  **import** **time**  **import** **csv**  **import** **logging**  **import** **os**  **def** create\_file(filetype):  *'''*  *This function takes a filetype and create a file named base on the filetye*  *@return : filename*  *'''*  f\_name = f'{filetype}-{time.time()}.csv'  file\_ = open(f\_name, 'w')  file\_.close()  **return** f\_name  **def** process\_seq(class\_, seq):  input\_config = {  'F1' : 0,  'F2' : 0,  'F3' : 0,  'F4' : 0,  'F5' : 0,  'F6' : 0,  'Class' : 0  }  resultant = {  'N' : 'F1',  'H' : 'F2',  'Q' : 'F3',  'G' : 'F4',  'D' : 'F5',  'T' : 'F6'  }  input\_config['Class'] = class\_  digits\_of\_sequence = [char **for** char **in** seq **if** char.isdigit()]  is\_seq\_valid = (len(digits\_of\_sequence) == 0)  **if** is\_seq\_valid == **False**:  **return** **False**, {}    **for** char **in** seq:    **if** char **in** resultant.keys():  input\_config[resultant[char]] += 1    **return** **True**, input\_config  **def** build\_config(line):  *'''*  *@What -> This function takes a line from file as an input*  *Check if the line is valid*  *A line is valid if --> 1) It has 2 columns*  *No digit in first column*  *+/- in second column*  *if Line is valid, build a config and return*  *else*  *return {} and Invalid line*  *'''*    is\_line\_valid = **False** **if** (len(line) < 2) **else** **True**    seq, class\_ = '', ''    is\_class\_mentioned = **False**  **if** is\_line\_valid:  seq, class\_ = line.split(',')[0], line.split(',')[1]  is\_line\_valid = is\_class\_mentioned = class\_ **in** ['+', '-']  class\_res\_values = {'+' : 1, '-' : 0}  **if** is\_line\_valid:  class\_ = class\_res\_values[class\_]  is\_line\_valid, input\_config = process\_seq(seq = seq, class\_=class\_)    **if** is\_line\_valid:    **return** is\_line\_valid, input\_config    **else**:    **if** is\_class\_mentioned:    class\_ = '-' **if** (class\_ == 0) **else** '+'  *# print(f'Seq : {seq}\tClass : {class\_}')*    **return** is\_line\_valid, (seq, class\_)  **def** process\_file(filename, seq\_no):  *'''*  *@Input -> Filename, seq\_no*  *@returns ->*  *1. Updated seq\_no*  *2. List of Result file entries*  *3. List of Log file entries*  *'''*  result\_entries = []  log\_entries = []  **with** open(filename, 'r+') **as** in\_file:  file\_content = in\_file.read().split('**\n**')  **for** line **in** file\_content[1:]:    is\_line\_valid, config = build\_config(line)  **if** is\_line\_valid:  config['seq\_no'] = seq\_no  new\_res\_entry = [  config['seq\_no'],  config['F1'],  config['F2'],  config['F3'],  config['F4'],  config['F5'],  config['F6'],  config['Class']  ]  result\_entries.append(new\_res\_entry)  **else**:  seq, class\_ = config  new\_log\_entry = [filename, seq, class\_]  log\_entries.append(new\_log\_entry)  seq\_no += 1    **return** seq\_no, result\_entries, log\_entries  **def** write\_entries\_in\_file(filename, entries, header):    **with** open(filename, 'r+') **as** file:  file.write(f'{header}**\n**')  **for** entry **in** entries:    *# Convert values of entries to strings*  entry = [str(value) **for** value **in** entry]    *# join using ','*  entry = ','.join(entry)  file.write(f'{entry}**\n**')  file.close()  **def** main():  *########## CHECK FOR WRONG PARAMS ########*  **try**:  input\_filenames = sys.argv[1:]  **if** len(input\_filenames) == 0:  **raise** **Exception**('No file provided')  seq\_no = 1  *# result\_writer, log\_writer = generate\_result\_and\_log\_writer()*  result\_filename = create\_file(filetype='result')    log\_filename = create\_file(filetype='log')  result\_file\_entries = []  log\_file\_entries = []    **for** filename **in** input\_filenames:  seq\_no, res\_entries, log\_entries = process\_file(filename, seq\_no=seq\_no)  result\_file\_entries = result\_file\_entries + res\_entries  log\_file\_entries = log\_file\_entries + log\_entries  *# Now take the result filename*  write\_entries\_in\_file(result\_filename, result\_file\_entries, header='Seq,F1,F2,F3,F4,F5,F6,Class')  write\_entries\_in\_file(log\_filename, log\_file\_entries, header='Filename,Seq,Class')    **except** **OSError** **as** osError:  print(f'OSError : {osError.\_\_str\_\_()}')  **except** **Exception** **as** invalid\_param\_exception:  print(f'Error : {invalid\_param\_exception.\_\_str\_\_()}')      **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

Output:

Result File:



Log File:



Program 2: Develop a web service for Program 1.

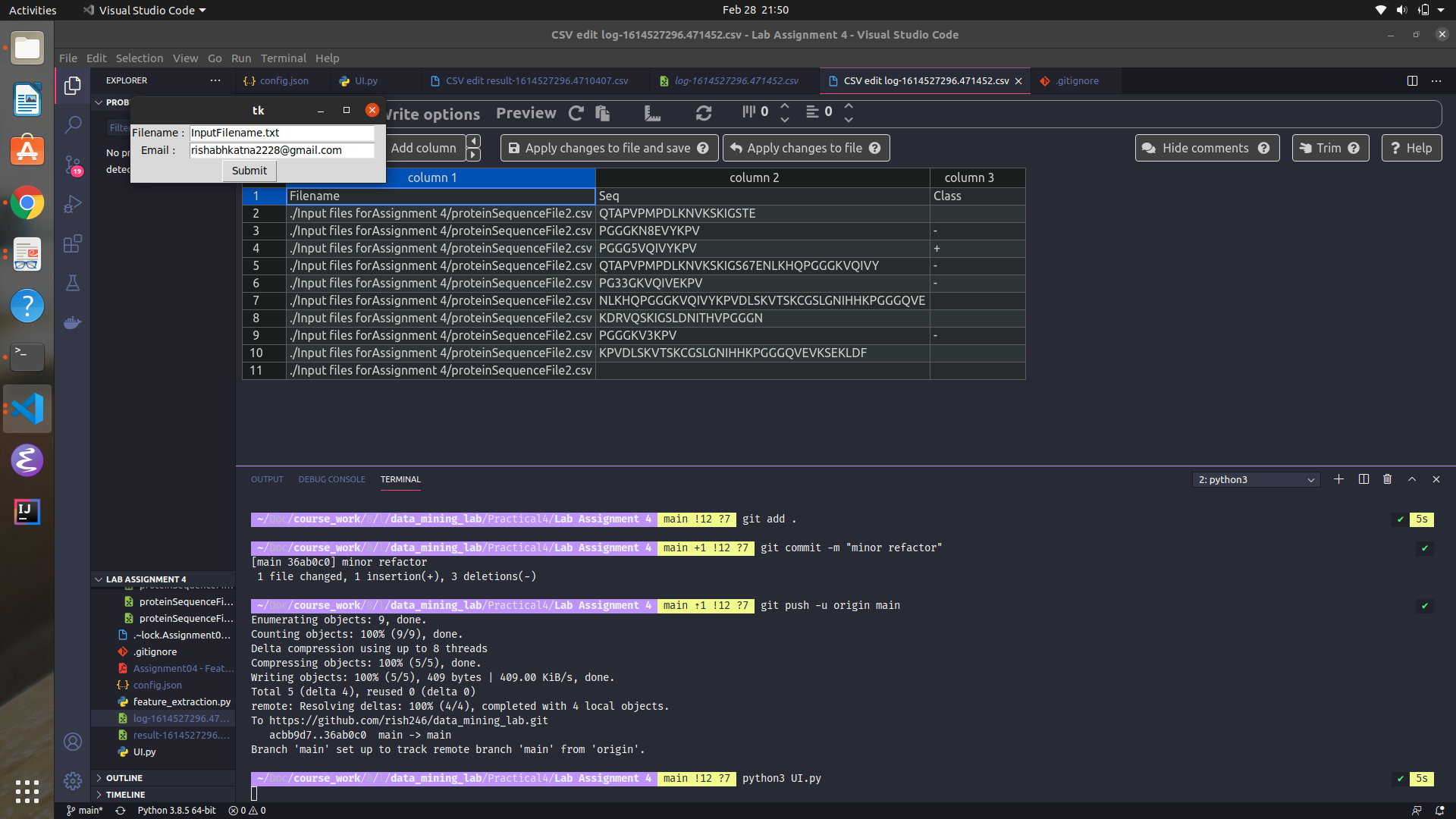
2.1 User Should get:

* Result File
* Log File

Code:

|  |
| --- |
| **import** **sys**  **import** **smtplib**  **import** **json**  **import** **tkinter** **as** **tk**  **from** **email.message** **import** EmailMessage  **from** **time** **import** sleep  **from** **tkinter.constants** **import** HIDDEN  **from** **feature\_extraction** **import** process\_file, create\_file, write\_entries\_in\_file  *##########################################################################*  **def** get\_email\_and\_password(config\_file\_path):  config\_file = open(config\_file\_path)    config\_file\_content = config\_file.read()  credentials = json.loads(config\_file\_content)  MY\_EMAIL = credentials['email']    MY\_PASSWORD = credentials['password']  **return** MY\_EMAIL, MY\_PASSWORD  *########################################################################*  **def** construct\_email\_message(from\_ ,to\_, result\_filename : str, log\_filename : str) -> EmailMessage:  msg = EmailMessage()  msg['Subject'] = "This is the subject of new message"  msg['From'] = from\_  msg['To'] = to\_  **with** open(result\_filename, 'rb') **as** result\_file:  *# send the file*  content = result\_file.read()  *# # add attatchment to email*  msg.add\_attachment(content, maintype='application', subtype='octet-stream', filename = result\_filename)  **with** open(log\_filename, 'rb') **as** log\_file:  content = log\_file.read()  *# # add attatchment to email*  msg.add\_attachment(content, maintype='application', subtype='octet-stream', filename = log\_filename)  **return** msg  *############# Only problem left is that the files are getting deleted and do not hold any content ##############*  *############ Why is it happeing ###############################################################################*  **def** send\_email(recipient\_email, result\_filename : str, log\_filename : str):  *# Read email and password from config*  MY\_EMAIL, MY\_PASSWORD = get\_email\_and\_password('config.json')    **with** smtplib.SMTP\_SSL('smtp.gmail.com', 465) **as** smtp:  *# Have an smtp connection*  code, status = smtp.login(MY\_EMAIL, MY\_PASSWORD)  **if** code == 235:  print('Successfully logged in to google')    msg = construct\_email\_message(MY\_EMAIL, recipient\_email, result\_filename, log\_filename)  smtp.send\_message(msg)  smtp.close()  *#1 -> Login using your email and password*  **def** get\_user\_input():  window = tk.Tk()  user\_email, filename = '', ''    *# Label and Entry*  *# 1-> Filepath label*  file\_path\_label = tk.Label(window, text="Filename : ")  file\_path\_label.grid(row=0, column=0)  *# Input field -> ENtry*  file\_path\_input = tk.Entry(window, width=30, borderwidth=2, border=1)  file\_path\_input.grid(row=0, column=1, columnspan=2)  *# Similarly create fields for Email*  user\_email\_label = tk.Label(window, text="Email : ")  user\_email\_label.grid(row=1, column=0)  *# Input field -> ENtry*  user\_email\_input = tk.Entry(window, width=30, borderwidth=2, border=1)  user\_email\_input.grid(row=1, column=1, columnspan=2)  **def** returnParams():  **nonlocal** user\_email, filename  *# Get the Filename and user\_email*  filename = file\_path\_input.get()  user\_email = user\_email\_input.get()  **if** len(filename) == 0 **or** len(user\_email) == 0:  print('Please enter both filename and user\_email')  **else**:    window.destroy()  **return**  *# make a submit button*  submit\_button = tk.Button(window, text="Submit", command=returnParams)  submit\_button.grid(row=2, column=1)    window.mainloop()  **return** filename, user\_email  **def** main():  *# Take a file and an email from the command line*  **try**:  filename, user\_email = get\_user\_input()  **if** len(filename) == 0 **or** len(user\_email) == 0:  **raise** **Exception**('No filename or email provided')    *# print("Successfully processed files")*  result\_filename = create\_file(filetype='result')  log\_filename = create\_file(filetype='log')  *# We created log and result files*  seq\_no, res\_file\_entries, log\_file\_entries = process\_file(filename, seq\_no=1)  write\_entries\_in\_file(result\_filename, res\_file\_entries, header='Seq,F1,F2,F3,F4,F5,F6,Class')  write\_entries\_in\_file(log\_filename, log\_file\_entries, header='Filename,Seq,Class')  *# Send Email*  send\_email(user\_email, result\_filename, log\_filename)    **except** **OSError** **as** osError:  print(f'OSError : {osError.\_\_str\_\_()}')  **except** **Exception** **as** invalid\_param\_exception:  print(f'Error : {invalid\_param\_exception.\_\_str\_\_()}')  **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

Output:



Mail Images:

