```
from argparse import ArgumentParser
import re
import sys
"""
Assignment:HW 3-Enron Analysis
Driver/Navigator:Vineet Ravichandran
Professor:Daniel Pauw
Date:04/06/23
Challenges Encountered:-------------
"""







"""
Server Class:


This class stores all the data for the emails in the data set


Attributes:
emails: list of email objects

"""
class Server:

    """
    opens the file specified by the path and set the emails attribute to a list of
email objects. each instance of an email
    should correspond to an email in the file

    params:
    self:instance

    path: path of the file to be parsed


    """
    def __init__(self,path):
        self.emails=[] #empty list of email objects
        with open(path,'r') as fil: #open the file in read mode
            content=fil.read() #read all of the contents of the file
            mailList=re.split(r'End Email', content) #separates individual emails
by 'End Email'
            mailList=mailList[:-1]#removes empty string from the list(last element)
            for mail in mailList: #go through each email in the list
                messageId=re.search(r'Message-ID: <(.*?)>',mail)

                """
                each of these uses regular expressions to extract different
components of the email and uses if statements to
                find if they are present in the dataset, and if they aren't
present, None is returned. However, if the component is present
                it is stored in the respective variable using the .group() method

                """
                if messageId:
                    messageId=messageId.group(1)

                else:
                    messageId=None


                date=re.search(r'Date: (.*?)\n',mail)

                if date:
                    date=date.group(1)
                else:
                    date=None


                emailSubject=re.search(r'Subject: (.*?)\n',mail)

                if emailSubject:
                    emailSubject.group(1)

                else:
                    emailSubject=None
                emailSender=re.search(r'Sender: (.*?)\n',mail)

                if emailSender:
                    emailSender=emailSender.group(1)
                else:
                    emailSender=None

                emailReceiver=re.search(r'To: (.*)', mail)
                if emailReceiver:
                    emailReceiver=emailReceiver.group(1)
                else:
                    emailReceiver=None

                emailText=re.search(r'\n\n(.*?)$',mail)
                if emailText:
                    emailText=emailText.group(1)

                else:
                    emailText=None

self.emails.append(Email(messageId,date,emailSubject,emailSender,emailReceiver,emai
lText)) #appends the extracted data as an email object



"""
Email Class:
Stores all the details of the email messages
"""
class Email:
    """
    initializes all of the variables

    params:
    self=instance
    messageId=unique Id for each email
    date=date of email that was sent
    emailSubject=subject of the email
    emailSender=sender of the email
    emailReceiver:person who received the email
    emailText=the body of the email

    all of these are strings

    initializes all of these variables
    """
    def __init__(self, 
messageId,date,emailSubject,emailSender,emailReceiver,emailText):
        self.messageId=messageId
        self.date=date
        self.emailSubject=emailSubject
        self.emailSender=emailSender
        self.emailReceiver=emailReceiver
        self.emailText=emailText

"""
params:
path: the path of the file that will be parsed
"""
def main(path):
    sPath=Server(path) #creases a new instance of a Server object using the path 
parameter
    print(len(sPath.emails))
    return(len(sPath.emails))   #returns the length of the emails list




"""
parses command line args

Params:
args_list: list of strings with the command line arguments

"""

def parse_args(args_list):
    parse=ArgumentParser() #creates an ArgumentParser object
    parse.add_argument("path",type=str)
    return parse.parse_args(args_list) #returns the parsed args of of the command 
line args




if __name__ == "__main__":
    args=parse_args(sys.argv[1:])#calls the parse_args function with command line 
argumetns that are passed to the script
    main(args.path) #calls the main function
```