A class named **“Email”** is defined, which is used to create an email body with the required subject **(\_init\_(self, subject))** and body **(htmladd)** of the email. This class also has functions to add attachments to the email.

The purpose of **logger()** function is to write log entries to a CSV file, and use the ‘os’ module to check if the file exists and the ‘datetime’ module to get the current date and time. The **logger()** function writes log entries to a CSV file. The function takes in three arguments: name, email, and att i.e. attachments. The function uses the ‘datetime’ module to retrieve the current date and time and assigns it to the variable ‘now’. Then the code retrieves a list of files in the directory "../RosterLog/" using the ‘os.listdir()’ function and assigns it to the variable files. The code then checks if the file "log.csv" exists in the directory using an if statement. If it does, the code opens the file in append mode using the ‘with open’ statement and writes a new line to the file using the f.write() method. The line contains the values of now, name, email, and att separated by commas. If the file "log.csv" does not exist in the directory, the code opens the file in write mode using the ‘with open’ statement and writes a new line to the file with the values of now, name, email, and att separated by commas.

The purpose of the **get\_emails()** function is to retrieve email addresses from a CSV file, store them in a dictionary, and return the dictionary to be used by other parts of the program. It uses the Pandas library to read and manipulate the contents of the CSV file. The code initializes an empty dictionary ‘email\_dict’ that will store the email addresses. Then the function retrieves a list of files in the directory "RosterLog/" using the ‘os.listdir()’ function and assigns it to the variable ‘files’. The code then checks if the file "emails.csv" exists in the directory using an ‘if’ statement. If it does, the code uses the ‘pd.read\_csv’ function from the Pandas library to read the contents of the file into a DataFrame ‘df’. The DataFrame is then modified by adding a new column "Fullname" that concatenates the values of the "Name" and "Lastname" columns separated by a space. The code then iterates over the rows of the DataFrame using the ‘df.iterrows()’ method and adds each email address to the ‘email\_dict’ dictionary with the key being the full name and the value being the email address. Finally, the function returns the ‘email\_dict’ dictionary. If the file "emails.csv" does not exist in the directory, the code prints an error message and returns ‘None’.

The **send(email, recipients)** function is used to send emails to recipients. The function takes two arguments:

email: an object that holds the sender's email address, the subject of the email, and any attachments.

recipients: a dictionary where the key is a recipient's name and each value is the recipient's email address.

A MIME (Multipurpose Internet Mail Extension) is constructed using the sender’s email address, subject, and message body. If there are attachments, they are added to the message. An SMTP\_SSL connection is established to the Gmail smtp server and then the sender logs in using the sender’s email address and password and sends the email to each recipient. The function then logs the recipient's name, email address, and attachments.

The **main\_func()** function of a program sends emails with attachments. It performs the following steps:

Calls a function ‘get\_emails’ to retrieve a dictionary of recipients and their email addresses.

Checks if the ‘email\_dict’ is None. If it's None, it prints a message saying that the program will not work without a roster and returns it.

Lists all files in the "Attachments" folder and checks if there are any files. If there are no files, it prints a message saying that the program cannot run without attachments and returns.

Changes the working directory to the "Attachments" folder.

Creates an Email object with the subject "Reports for" and the current date.

Adds all the attachments from the "Attachments" folder to the email object.

Calls the send function with the Email object and the ‘email\_dict’ as arguments to send the emails.

The **schedule** library is used to run the **main\_func()**function every day at 12:00. The while loop runs the pending tasks in the schedule and waits for 1 second between each check.