# envelope

*By Vedant Srinivasan of class 12 Sc.D*

Reminders and Scheduled Messaging

Computer Project

# THE INDIAN HIGH SCHOOL, DUBAI

## CERTIFICATE

This is to certify that the work in this project is the bonafide work of Master ______________________________ of Class _____ Div ______ Roll No. ______________________ recorded in the Computer Laboratory during the academic year

2017 to 2018.

Date: Teacher in-charge:

__________________ ________________________

__________________ _______________________

Examiner 1 Examiner 2

# TABLE *of* CONTENTS

# Acknowledgement

I wish to convey my gratitude to all those who have helped me complete this project. It could not have been done without the support of my grade 12 Computer Science Teacher Mrs. Viji Venugopal, my 11th grade Computer Science Teacher Mrs. Elsy K Varghese, and the encouragement from my Friends and Family.

I thank the school for providing us with this creative outlet and for letting us ignite the developer within.

# Why I chose this project?

One simple reason. I wanted to make something that I could use every day.

I always wanted to be able to schedule a message on Gmail but that isn't possible without third-party applications. To solve this problem, I created envelope which is a free and easy-to-use message scheduler and reminder app.

The reminders sent through this app are cleanly formatted and can be sent to people to notify them of something, for example, you could invite someone for dinner through this app. You can specify the place and a link to the corresponding Google Maps location would be attached to the mail automatically.

Seeing the possible use of this app, I undertook this project.

# Installing Python

Python 2.7.x is required in order to run envelope. To check if you already have Python installed, open a command prompt window and type in `python` and then hit enter.

If you see Python's version number, and something along the lines of `Type "help", "copyright", "credits" or "license" for more information`, you are ready to run envelope on your computer.

If you receive a message saying that `python is not recognized as an internal or external command`, Python isn't installed on your computer and you will need to carry out the following steps:

1. Visit Python's downloads site, `python.org/downloads` on your browser. Under `Download the latest version for Windows`, click `Download Python 2.7.13`. (If you see another version of Python whose version number begins with 2.7, feel free to click that. Note that Python 3.x is not backwards compatible with Python 2.7 programs)
2. You should see a file downloading with extension `.msi.` This is the installer for Python. Open the installer once it has been downloaded.
3. A confirmation screen should popup asking you to grant permission to run the installer. Please click `Run`.
4. Continue to the next screens until you reach `Customize Python 2.7.13.` Scroll down in the window and find the `Add Python.exe to Path` and click on the small red x. Choose the `Will be installed on local hard drive` option then press `Next.`
5. A new window will popup asking you to click `Finish`. Click `Finish`. You have successfully installed Python.
6. Once you have successfully installed Python, it is time to add it to the System Path Variable. Doing this will allow Python to run scripts on your computer without any conflicts of problems.
7. Begin by opening the start menu and typing in "environment" and select the option called "Edit the system environment variables."

8. When the “System Properties” window appears, click on “Environment Variables.
9. Once you have the “Environment Variables” window open, direct your focus to the bottom half. You will notice that it controls all the “System Variables” rather than just this associated with your user. Click on “New…” to create a new variable for Python.
10. Simply enter a name for your Path and the code shown below.
11. The string that you will need to enter is: `C:\Python27\;C:\Python27\Scripts;`
12. Press “OK,” then “OK,” then “OK,” then the red “X” to accept all changes and exit the “System Properties” window.

## System Requirements

### OS:

*Minimum: Windows XP*

*Recommended: Windows 10*

### Memory:

*Minimum: 4 GB RAM*

*Recommended: 8 GB RAM*

### Storage:

*Minimum: 2 GB*

*Recommended: 10 GB*

### Processor:

*Minimum: Intel i3-3xxxx*

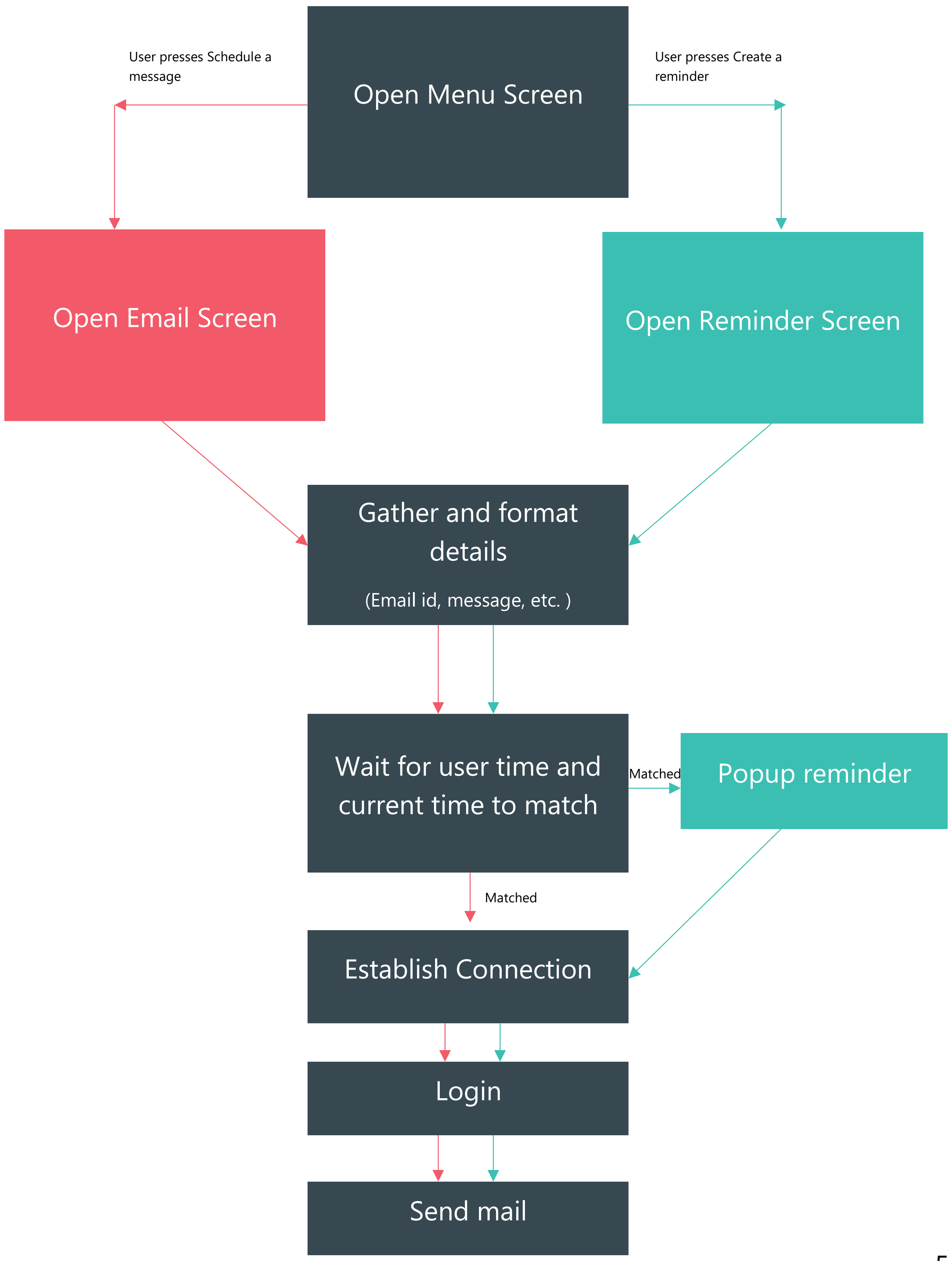
*Recommended: Intel i5-6xxxx*

### Others:

*Internet Connection*

*Desktop or Laptop computer*

# System Flowchart


Open Menu Screen
User presses Schedule a message
User presses Create a reminder
Open Email Screen
Open Reminder Screen
Gather and format details
(Email id, message, etc. )
Wait for user time and current time to match
Matched
Popup reminder
Matched
Establish Connection
Login
Send mail

# Functions and Modules

| Module Name | Object Name | Purpose |
|---|---|---|
| *Tkinter (GUI package)* | `Tk()` | Class that creates a top-level window |
| | `root.Label(parent,  option1, ,option2, ...)` | Displays one or more lines of text in the same style, or a bitmap or image |
| | `root.Button(parent, option1, ...)` | Creates button widget |
| | `root.Entry(parent, option1, ...)` | Creates text entry widget |
| | `StringVar()` | Variable Class that stores text. |
| | `StringVar().get()` | Method that can be used to read the text contained in a variable class. |
| | `StringVar().set()` | Method to set the value of a variable class. |
| | `root.mainloop()` | An infinite while loop that updates the window. |
| *datetime* | `datetime` | Class that contains date and time |
| | `now()` | Method that contains current date and time. |
| *smtplib* | `SMTP` | Class that contains connection details. |
| | `ehlo()` | Identifies your computer to the SMTP server |
| | `starttls()` | Begins TLS encryption to secure email contents. |
| | `login()` | Logs in to SMTP server |
| | `sendmail()` | Sends the mail |
| | `quit()` | Closes the connection |
| *pickle* | `load()` | Read a string from the open file |
| | `dump()` | Store a string in an open file |
| *os* | `remove()` | Remove a file from the computer |
| | `rename()` | Rename a file in the computer |

## *Functions and Classes:*

| **Function/Class name** | ***Purpose*** |
|---|---|
| login_and_send(function) | Responsible for logging in to smtp server and sending the message. |
| popup(function) | Popup message to display reminder to user |
| time_is_right(function) | Waits until current time and user time are same and then calls login_and_send. |
| menu_screen(class) | Class that creates the main menu |
| settings(class) | Class that creates settings window to change email id |
| about(class) | Class that creates about window |
| maillist_screen(class) | Class that allows user to create mailing lists |
| editing(class) | Class that allows user to edit mailing lists |
| schedule_screen(class) | Class that allows user to type in a message and submit |
| reminder_screen(class) | Class that allows user to set a reminder |

# Source Code

```
import Tkinter as tk
import tkMessageBox
import tkFileDialog
import smtplib
import datetime
import pickle
import os

# Colours
bgcolour = '#232830'
bglight_colour = '#2b313b'
button_colour = '#E68C55'
title_colour = '#F4F7E2'
sub_colour = '#eff2e1'
field_colour = '#302d2d'
field_text_colour = '#6BB279'
env_colour = 'white'
green = '#53DB8B'
red = '#ED766C'
white = 'white'

# Fonts
title_font = ('segoe ui', 26) # a font is defined as a tuple with font face and
size in pixels as elements
sub_font = ('segoe ui', 12)
sub2_font = ('segoe ui italic', 14)
button_font = ('segoe ui bold', 12)
spin_font = ('segoe ui', 11)
footer_font = ('segoe ui italic', 11)
env_font = ('century gothic bold', 64)

def login_and_send(to, msg):
    '''
    Responsible for logging in to gmail's smtp server and sending the message.
    '''

    try:
        f = open('C:/Users/vedant/Desktop/Projects/Programming/envelope/envelope
2.0/eap.envelope', 'rb') # file containing a dictionary with the email and
password of sender email id.
        d = pickle.load(f) # said dictionary. Keys: from address, port number,
and smtp server address.
        from_addr = d['from address']
        password = d['password']
        port_no = int(d['Port number'])
        smtp_server = d['SMTP server']
        f.close()
        spb = smtplib.SMTP(smtp_server, port_no) # 587 is the port that should be
used if TLS has to be used.
        spb.ehlo() # identify yourself to the server. Say hello :)
        spb.starttls() # begins TLS encryption to secure connection
        spb.ehlo() # this has to be done twice
        spb.login(from_addr, password)
        spb.sendmail(from_addr, to, msg)
        spb.quit() # disconnect from server
    except:
        # if there is any problem with sending the email. most likely due to user
giving incorrect credentials on settings page.
        msgbox = tkMessageBox.showinfo(title = 'Problem with settings', message =
'We have encountered a problem with the settings. Please enter required details
in the settings page')
        s = settings()
        s.create_widgets()


def popup(user_id, msg):
    '''
    A popup window that reminds the user to do the given task.
    '''
    if len(user_id) != 0: # check if user has input anything in email field and
if so, login and send mail to the email id.
        login_and_send(user_id, msg)

    popup_window = tk.Tk() # class that creates a window

    popup_window.title('Reminder') # text in title bar
    popup_window.configure(background = bgcolour) # change bg colour

    popup_text = tk.Label(popup_window, text = msg, font = sub_font, bg =
bgcolour, fg = 'white', padx = 15, pady = 15).pack() # content of the window

    popup_window.mainloop() # an infinite loop which updates the window and stops
when the window is closed.


def time_is_right(hour, minute, to, msg):
    '''
    Waits until current time and user time are same and then calls
login_and_send.
    '''

    if hour == 'HRS': # if user does not enter any time
        hour = datetime.datetime.now().hour # default time is current time
    if minute == 'MINS':
        minute = datetime.datetime.now().minute

    while True:

        current_time = datetime.datetime.now()

        if current_time.hour > int(hour): # you know that you can't go back to
the past, right?
            m = tkMessageBox.showinfo(title = 'Oh oh', message = 'We have not
found a way to send messages to the past. Please try again.')
            menu_screen() # go back to menu screen
            break

        elif int(hour) <= current_time.hour and int(minute) ==
current_time.minute  and 'REMINDER\n' not in msg:
            login_and_send(to, msg)
            break

        elif int(hour) <= current_time.hour and int(minute) ==
current_time.minute and 'REMINDER\n' in msg:
            popup(to, msg)
            break




class menu_screen:
    '''
    This is the main window
    '''

    def __init__(self):

        self.root = tk.Tk()
        self.root.title('envelope')
        self.root.config(background = bgcolour)
        self.root.resizable(width = False, height = False)

        self.master = tk.Frame(self.root)

    def create_widgets(self):

        var = tk.StringVar(self.master)
        var.set('')

        option_menu = tk.OptionMenu(self.master, var, 'About', 'Settings',
'Quit', command=self.optmenu_choice)
        option_menu.config(relief='flat', bg=bgcolour, fg=bgcolour,
font=sub_font, activeforeground=title_colour, activebackground=field_colour,
width=1, padx=5, pady=5, highlightthickness=0)
        option_menu['menu'].config(bg=bgcolour, fg=title_colour, font=sub_font,
activeforeground=title_colour, activebackground=field_colour)

        window_title = tk.Label(self.master, text='envelope', font=env_font,
pady=10, bg=bgcolour, fg=env_colour, padx=40)
        menu_title = tk.Label(self.master, text='What do you want to do?',
font=title_font, bg=bgcolour, fg=sub_colour, pady=15, padx=20)
        button1 = tk.Button(self.master, text='SCHEDULE A MESSAGE',
relief='flat', font=button_font,bg=bgcolour, fg=button_colour,
activeforeground=title_colour, activebackground = field_colour, padx = 10, pady =
10, command = self.schedule_on_click)
        button2 = tk.Button(self.master, text='CREATE A REMINDER', relief='flat',
bg=bgcolour, fg=button_colour, activeforeground=title_colour, activebackground =
field_colour, font = button_font, pady = 10, padx = 10, command =
self.reminder_on_click)
        button3 = tk.Button(self.master, text='CREATE A MAILING LIST',
relief='flat', bg=bgcolour, fg=button_colour, activeforeground=title_colour,
activebackground = field_colour, font = button_font, pady = 10, padx = 10,
command = self.maillist_on_click)
```

```
        footer = tk.Label(self.master, text='Vedant Srinivasan and Dhruv Jain 
2017', font=footer_font, bg=bgcolour, fg=title_colour, pady=15, padx=10)

        self.master.config(background = bgcolour, padx = 5, pady = 5)
        self.master.grid()

        option_menu.grid(column = 1, row = 7)
        window_title.grid(column = 1, row = 1)
        menu_title.grid(column = 1, row = 2)
        button1.grid(column = 1, row = 3)
        button2.grid(column = 1, row = 4)
        button3.grid(column = 1, row = 5)
        footer.grid(column = 1, row = 6)
        self.root.mainloop()

    def maillist_on_click(self):
        self.root.destroy()
        t = maillist_screen()
        t.create_widgets()

    def reminder_on_click(self):
        self.root.destroy()
        r = reminder_screen()
        r.create_widgets()

    def schedule_on_click(self):
        self.root.destroy()
        s = schedule_screen()
        s.create_widgets()

    def optmenu_choice(self, choice):
        print 'Clicked', choice
        if choice == 'Quit':
            self.root.destroy()
        elif choice == 'Settings':
            s = settings()
            s.create_widgets()
        elif choice == 'About':
            a = about()
            a.create_widgets()

class settings:
    '''
    Settings menu accessed from option menu.
    Change default email id.
    '''
    def __init__(self):
        self.root = tk.Tk()
        self.root.title('Settings')
        self.root.config(background = bgcolour)
        self.root.resizable(width = False, height = False)

        self.master = tk.Frame(self.root)

    def create_widgets(self):
        menu_title = tk.Label(self.master, text='Settings', font=title_font,
bg=bgcolour, fg=sub_colour, pady=15, padx=20)
        self.master.config(bg = bgcolour, padx = 10, pady = 10)
        frame1 = tk.Frame(self.master, bg = bglight_colour, padx = 10, pady = 10)
        subtitle = tk.Label(frame1, text = 'Change Email ID', bg =
bglight_colour, font = sub2_font, fg = sub_colour, padx = 10, pady = 20)

        label1=tk.Label(frame1,text="Email", font = sub_font, fg = sub_colour, bg
= bglight_colour, padx = 10, pady = 10)
        self.x = tk.StringVar(frame1)
        entry1 = tk.Entry(frame1, textvariable=self.x, font=sub_font,
fg=sub_colour, bg=field_colour, highlightcolor = button_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)

        label2=tk.Label(frame1,text="Password", font = sub_font, fg = sub_colour,
bg = bglight_colour, padx = 10, pady = 10)
        self.y = tk.StringVar(frame1)
        entry2 = tk.Entry(frame1, textvariable=self.y, show = '*', font=sub_font,
fg=sub_colour, bg=field_colour, highlightcolor = button_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)

        submit = tk.Button(frame1, text = 'SUBMIT', font = button_font, fg =
button_colour, bg = bglight_colour, padx = 10, pady = 10, relief = 'flat',
command = self.insert)

        self.smtp_str = tk.StringVar(frame1)
        self.port_str = tk.StringVar(frame1)

        label3 = tk.Label(frame1, text = 'SMTP Server', font = sub_font, fg =
sub_colour, bg = bglight_colour, padx = 10, pady = 10)
        label4 = tk.Label(frame1, text = 'Port number', font = sub_font, fg =
sub_colour, bg = bglight_colour, padx = 10, pady = 10)
        entry3 = tk.Entry(frame1, textvariable = self.smtp_str, font=sub_font,
fg=sub_colour, bg=field_colour, highlightcolor = button_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        entry4 = tk.Entry(frame1, textvariable = self.port_str, font = sub_font,
fg=sub_colour, bg=field_colour, highlightcolor = button_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)



        menu_title.grid(row = 0, columnspan = 2)

        subtitle.grid(row = 1, column = 1, columnspan = 2)
        label1.grid(row = 3, column = 0)
        entry1.grid(row = 3, column = 1)
        label2.grid(row = 4, column = 0)
        entry2.grid(row = 4, column = 1)
        label3.grid(row = 5, column = 0)
        entry3.grid(row = 5, column = 1)
        label4.grid(row = 6, column = 0)
        entry4.grid(row = 6, column = 1)
        submit.grid(row = 7, column = 1)
        frame1.grid(row = 1)
        self.master.grid()
        menu_title.grid(row = 0, column = 0)

        self.root.mainloop()

    def insert(self):
        m = tkMessageBox.askyesno('Are you sure?', 'Are you sure you want to 
change the default email id and password')
        if m:
            self.root.destroy()
            f1 = 
open("C:/Users/vedant/Desktop/Projects/Programming/envelope/envelope 
2.0/eap.envelope", "wb")
            email_id = self.x.get()
            password = self.y.get()
            smtp_server = self.smtp_str.get()
            port_no = self.port_str.get()
            d = {}
            d['from address'] = email_id
            d['password'] = password
            d['Port number'] = port_no
            d['SMTP server'] = smtp_server
            pickle.dump(d,f1)
            f1.close()


class about:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title('About')
        self.root.config(background = bgcolour)
        self.root.resizable(width = False, height = False)
        self.master = tk.Frame(self.root)

    def create_widgets(self):
        frame1 = tk.Frame(self.master, bg = bglight_colour, padx = 20, pady = 20)
        menu_title = tk.Label(self.master, text='About', font=title_font, 
bg=bgcolour, fg=sub_colour, pady=15, padx=20)
        body_text = '''MADE BY VEDANT SRINIVASAN AND DHRUV JAIN OF 12 SC.D

This is an app to schedule emails and set reminders.
It is an attempt made to learn to make a UI with both form and function,
use classes effectively, and make something that could be used everyday.
'''

        body = tk.Label(frame1, text=body_text, font=sub_font, bg=bglight_colour, 
fg=sub_colour, pady=15, padx=20)
        self.master.config(bg = bgcolour, padx = 20, pady = 20)
        self.master.grid()
        menu_title.grid(row = 0, column = 0)
        frame1.grid(row = 1, column = 0)
        body.grid(row = 1, column = 0)
        self.root.mainloop()
```

```
class maillist_screen:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title('Create a Mailing List')
        self.master = tk.Frame(self.root)
        self.root.config(bg = bgcolour)
        self.master.config(bg = bgcolour, padx = 20, pady = 20)
        self.root.resizable(height = False, width = False)
        self.contact_dict = {} # dictionary with names and email ids of contacts.

    def create_widgets(self):
        self.txt0 = tk.StringVar(self.master)
        self.txt0.set('Enter name of mailing list here')

        self.txt1 = tk.StringVar(self.master)
        self.txt1.set('Enter email id here') # email id of contact

        self.txt2 = tk.StringVar(self.master)
        self.txt2.set('Enter name here') # name of contact

        frame1 = tk.Frame(self.master, bg=bglight_colour, padx=30, pady = 20)
        title = tk.Label(self.master, text='Mailing List', font=title_font,
fg=title_colour, bg=bgcolour, pady=10)
        entry0 = tk.Entry(frame1, textvariable=self.txt0, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        entry1 = tk.Entry(frame1, textvariable=self.txt1, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        entry2 = tk.Entry(frame1, textvariable=self.txt2, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        submit = tk.Button(frame1, text='SUBMIT', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.submit_on_click)
        self.list_of_contacts = tk.Listbox(self.master, font = sub_font, bg =
field_colour, highlightcolor = button_colour, fg = sub_colour, selectbackground =
'#353535', relief = 'flat', height = 16, width = 40, highlightthickness = 2)
        delete_all = tk.Button(frame1, text = 'delete all'.upper(),
relief='flat', bg=bglight_colour, fg=button_colour, font=button_font, padx=20,
pady=10, command = self.delall_on_click)
        delete = tk.Button(frame1, text = 'delete contact'.upper(),
relief='flat', bg=bglight_colour, fg=button_colour, font=button_font, padx=20,
pady=10, command = self.delete_on_click)

        create_button = tk.Button(frame1, text = 'CREATE MAILING LIST',
relief='flat', bg=bglight_colour, fg=button_colour, font=button_font, padx=20,
pady=10, command = self.create_on_click)
        edit = tk.Button(frame1, text = 'EDIT MAILING LIST', relief='flat',
bg=bglight_colour, fg=button_colour, font=button_font, padx=20, pady=10, command
= self.edit_on_click)
```

```
        self.no_contacts_label = tk.Label(self.master, text = 'No contacts', font
= sub_font, fg = 'gray', bg = field_colour)

        title.grid(row = 0, column = 0, columnspan = 2)
        entry0.grid(row = 0, column = 0, columnspan = 2)
        entry1.grid(row = 1, column = 0)
        entry2.grid(row = 3, column = 0)

        submit.grid(row = 4, column = 0)
        edit.grid(row = 9, column = 0)

        delete.grid(row = 6, column = 0)
        delete_all.grid(row = 7, column = 0)
        create_button.grid(row = 8, column = 0)
        frame1.grid(row = 1, column = 0)
        self.list_of_contacts.grid(row = 1, column = 1)
        self.no_contacts_label.grid(row = 1, column = 1)


        self.master.grid()
        self.root.mainloop()

    def edit_on_click(self):
        self.root.destroy()
        t = editing()
        t.edit()

    def delete_on_click(self):
        del self.contact_dict[self.list_of_contacts.get('active').strip('- ')]
        self.list_of_contacts.delete('active')
        print self.contact_dict
        if self.list_of_contacts.size() == 0:
            self.no_contacts_label = tk.Label(self.master, text = 'No contacts',
font = sub_font, fg = 'gray', bg = field_colour)
            self.no_contacts_label.grid(row = 1, column = 1)

    def delall_on_click(self):
        self.list_of_contacts.delete(0, 'end')
        self.contact_dict = {}
        print self.contact_dict
        self.no_contacts_label = tk.Label(self.master, text = 'No contacts', font
= sub_font, fg = 'gray', bg = field_colour)
        self.no_contacts_label.grid(row = 1, column = 1)


    def submit_on_click(self, n = 'end'):

        if self.txt1.get() not in ['Enter email id here', '']:
            if self.txt2.get() in ['Enter name here', '']:
                self.txt2.set(self.txt1.get())
            self.contact_dict[self.txt2.get()] = self.txt1.get()
            print self.contact_dict
            string = '- ' + self.txt2.get()
            self.txt1.set('Enter email id here')
            self.txt2.set('Enter name here')
            self.no_contacts_label.destroy()
            self.list_of_contacts.insert(n, string)
            self.list_of_contacts.itemconfig(n, bg = '#292929')

    def create_on_click(self):
        f = open(self.txt0.get() + '.mailinglist', 'ab')
        pickle.dump(self.contact_dict, f)
        f.close()


class editing:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title('Editing a Mailing List')
        self.master = tk.Frame(self.root)
        self.root.config(bg = bgcolour)
        self.master.config(bg = bgcolour, padx = 20, pady = 20)
        self.root.resizable(height = False, width = False)

    def edit(self):
        self.mailing_list = tk.StringVar(self.master)
        self.mailing_list.set('Enter name of mailing list here')

        frame1 = tk.Frame(self.master, bg=bglight_colour, padx=30, pady = 20)
        title = tk.Label(self.master, text='Editing a Mailing List',
font=title_font, fg=title_colour, bg=bgcolour, pady=10)
        entry0 = tk.Entry(frame1, textvariable=self.mailing_list, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)

        append = tk.Button(frame1, text='Append', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.append_on_click)
        modify = tk.Button(frame1, text='Modify', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.modify_on_click)
        delete =tk.Button(frame1, text='Delete', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.delete_on_click)

        title.grid(row = 0, column = 0, columnspan = 3)
        entry0.grid(row = 1, column = 0, columnspan = 3)
        append.grid(row = 2, column = 0)
        modify.grid(row = 2, column = 1)
        delete.grid(row = 2, column = 2)
        frame1.grid(row = 1, column = 0)

        self.master.grid()
        self.root.mainloop()

    def append_on_click(self):
        self.txt1 = tk.StringVar(self.master)
        self.txt1.set('Enter email id here')

        self.txt2 = tk.StringVar(self.master)
        self.txt2.set('Enter name here')

        frame2 = tk.Frame(self.master, bg=bglight_colour, padx=30, pady = 20)
        entry1 = tk.Entry(frame2, textvariable=self.txt1, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        entry2 = tk.Entry(frame2, textvariable=self.txt2, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        submit = tk.Button(frame2, text='SUBMIT', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.submit_on_click_append)

        entry1.grid(row = 3, column = 0)
        entry2.grid(row = 4, column = 0)
        submit.grid(row = 5, column = 0)
        frame2.grid(row = 3, column = 0)

    def modify_on_click(self):
        self.txt1 = tk.StringVar(self.master)
        self.txt1.set('Enter email id here')

        self.txt2 = tk.StringVar(self.master)
        self.txt2.set('Enter name here')

        frame2 = tk.Frame(self.master, bg=bglight_colour, padx=30, pady = 20)
        entry1 = tk.Entry(frame2, textvariable=self.txt1, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        entry2 = tk.Entry(frame2, textvariable=self.txt2, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        submit = tk.Button(frame2, text='SUBMIT', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.submit_on_click_modify)

        frame2.grid(row = 3, column = 0)
        entry1.grid(row = 3, column = 0)
        entry2.grid(row = 4, column = 0)
        submit.grid(row = 5, column = 0)

    def delete_on_click(self):
        self.txt2 = tk.StringVar(self.master)
        self.txt2.set('Enter name here')

        frame2 = tk.Frame(self.master, bg=bglight_colour, padx=30, pady = 20)
        entry2 = tk.Entry(frame2, textvariable=self.txt2, font=sub_font,
highlightcolor = button_colour, fg=sub_colour, bg=field_colour, selectbackground
= '#353535', relief='flat', width=40, highlightthickness=2)
        submit = tk.Button(frame2, text='SUBMIT', relief='flat',
font=button_font,fg=button_colour, bg=bglight_colour, padx=20 ,pady=20,
command=self.submit_on_click_delete)

        frame2.grid(row = 3, column = 0)
        entry2.grid(row = 3, column = 0)
        submit.grid(row = 4, column = 0)

    def submit_on_click_append(self):
        try:
            f = open(self.mailing_list.get() + '.mailinglist', 'rb')
            f_temp = open('temp.dat','wb')
            self.contact_dict = pickle.load(f)

            if self.txt1.get() not in ['Enter email id here', ''] and '@' in
self.txt1.get():
                if self.txt2.get() in ['Enter name here', '']:
                    self.txt2.set(self.txt1.get())
                self.contact_dict[self.txt2.get()] = self.txt1.get()
                print self.contact_dict
            else:
                tkMessageBox.showerror("Error","Enter a valid Email address")

            pickle.dump(self.contact_dict, f_temp)
            f.close()
            f_temp.close()
            os.remove(self.mailing_list.get() + '.mailinglist')
            os.rename('temp.dat',self.mailing_list.get() + '.mailinglist')

        except IOError:
            tkMessageBox.showerror("Error","Mailing List not found")

    def submit_on_click_modify(self):
        try:
            f = open(self.mailing_list.get() + '.mailinglist', 'rb')
            f_temp = open('temp.dat','wb')
            self.contact_dict = pickle.load(f)

            if self.txt1.get() not in ['Enter email id here', ''] and '@' in
self.txt1.get():
                if self.txt2.get() in ['Enter name here', '']:
                    self.txt2.set(self.txt1.get())

                if self.txt2.get() in self.contact_dict.keys():
                    self.contact_dict[self.txt2.get()] = self.txt1.get()
                    print self.contact_dict
                else:
                    tkMessageBox.showerror("Error","Name not found in the
dictionary")
            else:
                tkMessageBox.showerror("Error","Enter a valid Email address")

            pickle.dump(self.contact_dict, f_temp)
            f.close()
            f_temp.close()
            os.remove(self.mailing_list.get() + '.mailinglist')
            os.rename('temp.dat',self.mailing_list.get() + '.mailinglist')

        except IOError:
            tkMessageBox.showerror("Error","Mailing List not found")
    def submit_on_click_delete(self):
        try:
            f = open(self.mailing_list.get() + '.mailinglist', 'rb')
            f_temp = open('temp.dat','wb')
            self.contact_dict = pickle.load(f)

            try:
                del self.contact_dict[self.txt2.get()]
            except KeyError:
                tkMessageBox.showerror("Error","Name not found in the
dictionary")

            pickle.dump(self.contact_dict, f_temp)
            f.close()
            f_temp.close()
            os.remove(self.mailing_list.get() + '.mailinglist')
            os.rename('temp.dat',self.mailing_list.get() + '.mailinglist')

        except IOError:
            tkMessageBox.showerror("Error","Mailing List not found")



class schedule_screen:
    def __init__(self):
        self.root = tk.Tk()
        self.master = tk.Frame(self.root)
        self.root.title('Schedule a message')

    def create_widgets(self):
        self.master.config(bg = bgcolour, padx = 20, pady = 20)
        title = tk.Label(self.master, text = 'Enter your message', font =
title_font, fg = title_colour, bg = bgcolour, pady = 5, padx = 20)
        self.frame1 = tk.Frame(self.master, bg = bglight_colour, padx = 17, pady
= 15)
        frame2 = tk.Frame(self.master, bg = bglight_colour, padx = 15, pady = 15)
        frame3 = tk.Frame(self.master, bg = bglight_colour, padx = 190, pady =
15)
        scroll = tk.Scrollbar(frame2)
        self.to_txt = tk.StringVar(self.frame1)
        to_label = tk.Label(self.frame1, text = 'To', font = sub_font, fg =
sub_colour, bg = bglight_colour, padx = 13, pady = 5)
        to_entry = tk.Entry(self.frame1, textvariable = self.to_txt,
font=sub_font, highlightcolor = button_colour, fg=sub_colour, bg=field_colour,
selectbackground = '#353535', relief='flat', width=72, highlightthickness=2)

        ml_button = tk.Button(self.frame1, text = 'MAILING LIST', command =
self.ml_on_click, bg = bglight_colour, fg = button_colour, font = button_font,
relief = 'flat')
        self.mailinglist = 0

        self.sub_txt = tk.StringVar(self.frame1)
        sub_label = tk.Label(self.frame1, text = 'Subject', font = sub_font, fg =
sub_colour, bg = bglight_colour, padx = 13, pady = 5)
```

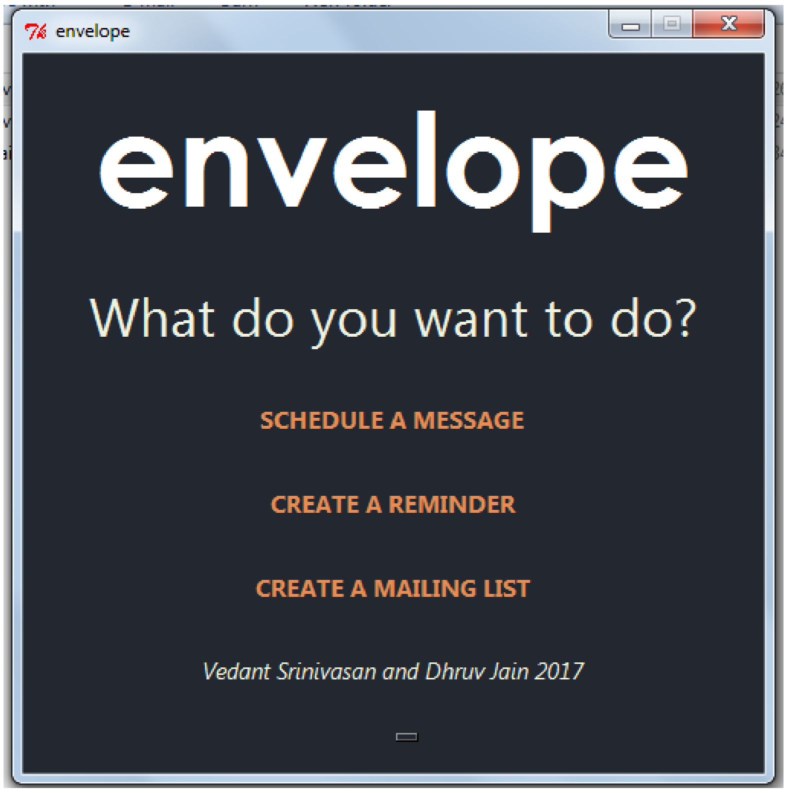
```
        sub_entry = tk.Entry(self.frame1, textvariable = self.sub_txt, 
font=sub_font, highlightcolor = button_colour, fg=sub_colour, bg=field_colour, 
selectbackground = '#353535', relief='flat', width=72, highlightthickness=2)
        self.box = tk.Text(frame2, font=sub_font, fg=sub_colour, bg=field_colour, 
highlightcolor = button_colour, height = 15, width = 80, yscrollcommand = 
scroll.set, selectbackground = '#353535', relief='flat', highlightthickness=2)
        button = tk.Button(self.master, text = 'SUBMIT', command = self.on_click, 
bg = bglight_colour, fg = button_colour, font = button_font, relief = 'flat', 
padx=350, pady=20)

        scroll.config(command = self.box.yview)

        self.time_strhour = tk.IntVar(frame3, value = 'HRS')
        self.time_strmin = tk.IntVar(frame3, value = 'MINS')
        time_label = tk.Label(frame3, text = 'Send message at:', font = sub_font, 
bg = bglight_colour, fg = 'white', pady = 8, padx = 20)

        hrs_label = tk.Label(frame3, text = 'hrs', font = sub_font, bg = 
bglight_colour, fg = 'white', pady = 8, padx = 20)
        min_label = tk.Label(frame3, text = 'mins', font = sub_font, bg = 
bglight_colour, fg = 'white', pady = 8, padx = 20)
        time_hour= tk.Spinbox(frame3, from_ = 00, to = 23, textvariable = 
self.time_strhour, bg = field_colour, fg = 'gray', font = sub_font, relief = 
'flat')
        time_min = tk.Spinbox(frame3, from_ = 00, to = 59, textvariable = 
self.time_strmin, bg = field_colour, bd=2, fg = 'gray', font = sub_font, relief = 
'flat')

        title.grid(row = 0, column = 0)
        self.box.grid(row = 2, column = 0)
        sub_label.grid(row = 2, column = 0)
        sub_entry.grid(row = 2, column = 1)
        scroll.grid(row = 2, column = 1, sticky = 'ns')
        to_label.grid(row = 1, column = 0)
        to_entry.grid(row = 1, column = 1)
        ml_button.grid(row = 0, column = 1)
        time_label.grid(row = 3, column = 0, columnspan = 2)
        hrs_label.grid(row = 4, column = 0)
        time_hour.grid(row = 5, column = 0)
        min_label.grid(row = 4, column = 1)
        time_min.grid(row = 5, column = 1)
        button.grid(row = 6, column = 0)
        self.frame1.grid(row = 1 , column = 0)
        frame2.grid(row = 2, column = 0)
        frame3.grid(row = 3, column = 0)
        self.master.grid()
        self.root.mainloop()

    def ml_on_click(self):


        explorer = tk.Tk()
        # to limit user to choose only .mailinglist files filetypes kwarg is 
passed
```

```
        self.src_path = str(tkFileDialog.askopenfilename(title = 'Choose mailing
list', filetypes = (('Mailing lists','*.mailinglist'),("all files","*.*"))))
        explorer.destroy()
        if self.src_path not in ['', None]:
            self.mailinglist = 1 # flag variable. if 1 mailing list was chosen by
the user
        a = self.src_path.split('/')[-1]
        file_name_label = tk.Label(self.frame1, text = a, font = sub_font, fg =
sub_colour, bg = bglight_colour, padx = 13, pady = 5)
        file_name_label.grid(row = 0, column = 1)


    def on_click(self):
        to = []
        if self.to_txt.get() not in ['', ' ']:
            # if anything is entered in the to field make it a list of ids
            alist = self.to_txt.get().split(';')
            for e in alist:
                to += [[e, e]] # email id and name are same for those entered in
to field
            # email ids are seperated by semicolons
            # a list is created from the string
        if self.mailinglist:
            f = open(self.src_path, 'rb')
            d = pickle.load(f)
            f.close()
            for key in d:
                to += [[d[key], key]]
        sub = self.sub_txt.get()
        msg = 'Subject: ' + sub + '\n' + self.box.get('1.0','end') # get all the
text within the box from beginning to end.
        print to
        hrs = self.time_strhour.get()
        mins = self.time_strmin.get()
        try:
            # check if the hrs and mins entered are valid numbers are not
            a = (hrs == int(hrs))
            b = (mins == int(mins))

            if a and b:
                message = 'Are you sure you want to send this message to ' +
str(to) + ' at ' + str(hrs) + ':' + str(mins)
                m = tkMessageBox.askyesno('Are you sure?', message)

                # smtplib can take multiple email ids in the form of a list
                if m:
                    self.root.destroy()
                    for e in range(len(to)):
                        msg1 = msg.replace('%name', to[e][1])
                        time_is_right(hrs, mins, to[e][0], msg1)


        except ValueError:
            m = tkMessageBox.showerror(title = 'Error', message = 'Invalid Time
of message')
```

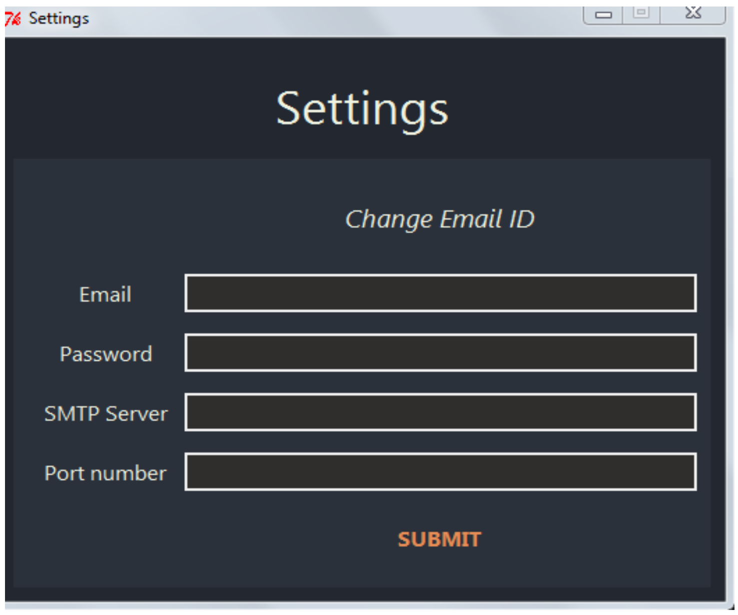
```
##              time_is_right(hrs, mins, to, msg)

class reminder_screen:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title('Create a reminder')
        self.root.config(background = bgcolour)
        self.root.resizable(width = False, height = False)
        self.master = tk.Frame(self.root)

    def create_widgets(self):


        window_title = tk.Label(self.root, text = 'Create a reminder', font =
title_font, bg = bgcolour, fg = 'white', pady = 8, padx = 30).pack()

        note_label = tk.Label(self.root, text = 'Note:', font = sub_font, bg =
bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        self.note_string = tk.StringVar(self.root)
        note = tk.Entry(self.root, textvariable = self.note_string, font =
sub_font, fg = bgcolour).pack()

        people_label = tk.Label(self.root, text = 'People:', font = sub_font, bg
= bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        self.people_string = tk.StringVar(self.root)
        people = tk.Entry(self.root, textvariable = self.people_string, font =
sub_font, fg = bgcolour).pack()

        place_label = tk.Label(self.root, text = 'Place:', font = sub_font, bg =
bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        self.place_string = tk.StringVar(self.root)
        place = tk.Entry(self.root, textvariable = self.place_string, font =
sub_font, fg = bgcolour).pack()

        time_of_event_label = tk.Label(self.root, text = 'Event at:', font =
sub_font, bg = bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        self.time_of_event_string = tk.StringVar(self.root)
        time_of_event = tk.Entry(self.root, textvariable =
self.time_of_event_string, font = sub_font, fg = bgcolour).pack()

        self.time_strhour = tk.StringVar(self.root, value = 'HRS')
        self.time_strmin = tk.StringVar(self.root, value = 'MINS')
        time_label = tk.Label(self.root, text = 'Be reminded at:', font =
sub_font, bg = bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        time_hour = tk.Entry(self.root, textvariable = self.time_strhour, font =
sub2_font, fg = bgcolour, width = 5).pack()
        time_min = tk.Entry(self.root, textvariable = self.time_strmin, font =
sub2_font, fg = bgcolour, width = 5).pack()

        self.user_id_string = tk.StringVar(self.root)
        user_id_label = tk.Label(self.root, text = 'Your email id:', font =
sub_font, bg = bgcolour, fg = 'white', pady = 8, padx = 20).pack()
        user_id = tk.Entry(self.root, textvariable = self.user_id_string, font =
sub_font, fg = bgcolour).pack()
```

```
        submit = tk.Button(self.root, text = 'SUBMIT', font = button_font, bg =
bgcolour, fg = button_colour, command = self.s, relief = 'flat', pady =
10).pack()

        self.root.mainloop()


    def s(self):

        note = self.note_string.get()
        people = self.people_string.get()
        place = ''
        for i in self.place_string.get():
            if not i.isspace():
                place = place + i
            else:
                place = place + '+'
        place = self.place_string.get() + ' (Directions: maps.google.com/?q=' +
place + ')'
        user_id = self.user_id_string.get()
        hour = self.time_strhour.get()
        minute = self.time_strmin.get()
        try:
            if hour != 'HRS':
                a = int(hour)
            if minute != 'MINS':
                b = int(minute)
            time_of_event = self.time_of_event_string.get()
            msg = 'REMINDER\n\n' + note + '\n\nPEOPLE: ' + people + '\nPLACE: ' +
place + '\nTIME: ' + time_of_event + '\n\n\n\n\nThis message was sent by
envelopebot.'
            self.root.destroy()
            time_is_right(hour, minute, user_id, msg)

        except ValueError:
            m = tkMessageBox.showerror(title = 'Error', message = 'Invalid Time
of reminder')




# main

p1 = menu_screen()
p1.create_widgets()
```

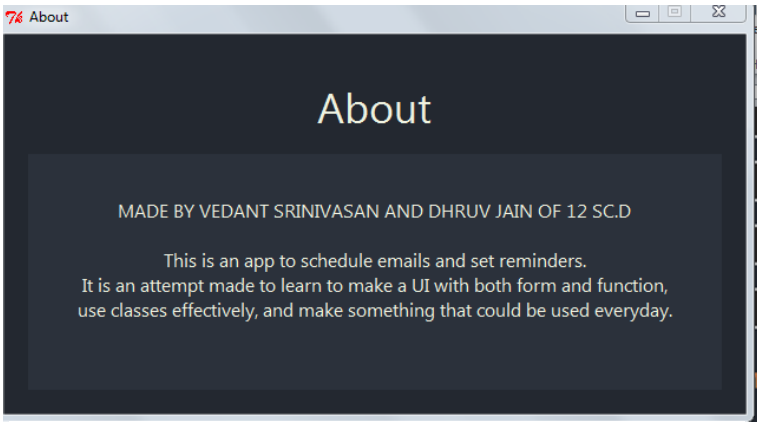
## Screenshots



*The main menu where the user can choose what to do.*



*Settings page allows user to change the default email-id from which the message will be sent*

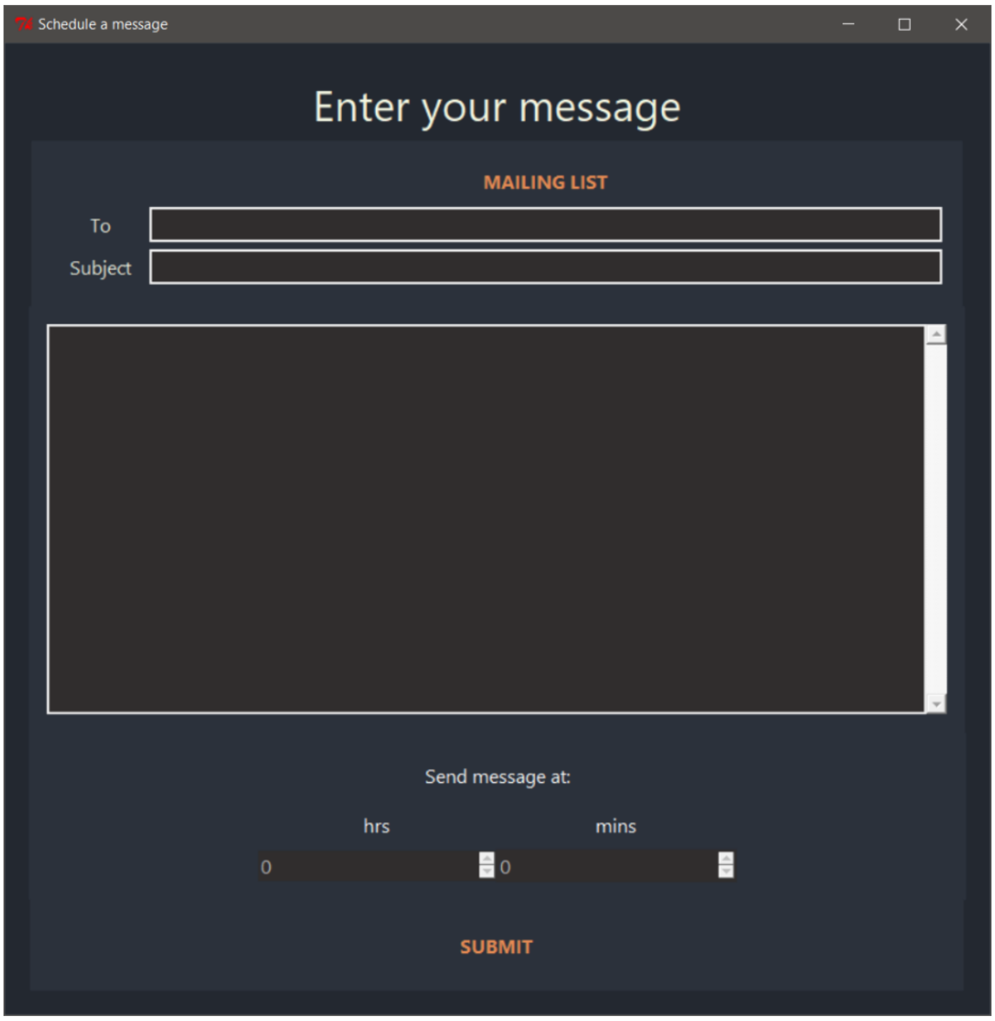

About

About

MADE BY VEDANT SRINIVASAN AND DHRUV JAIN OF 12 SC.D

This is an app to schedule emails and set reminders.
It is an attempt made to learn to make a UI with both form and function,
use classes effectively, and make something that could be used everyday.

*About page*


Schedule a message

Enter your message

MAILING LIST

To

Subject

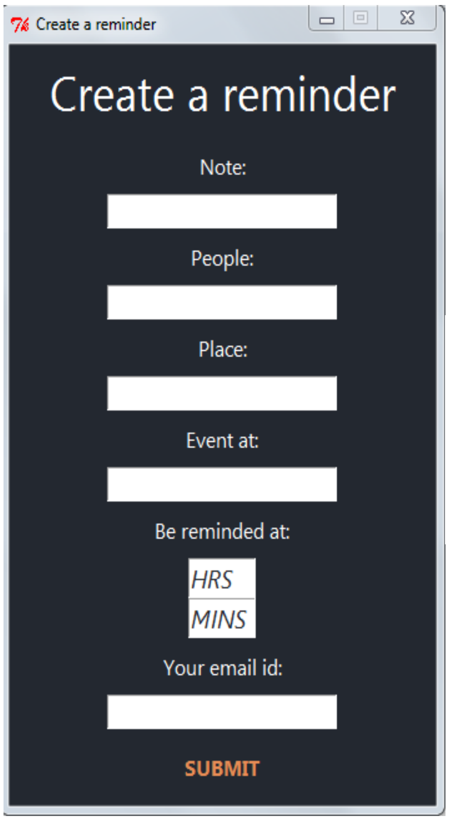
Send message at:

hrs 0

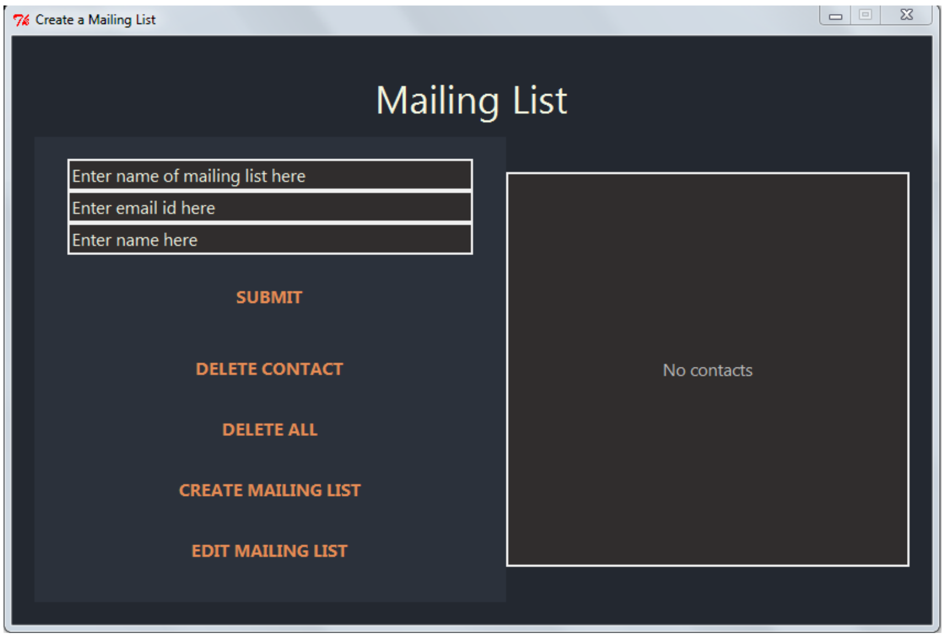
mins 0

SUBMIT

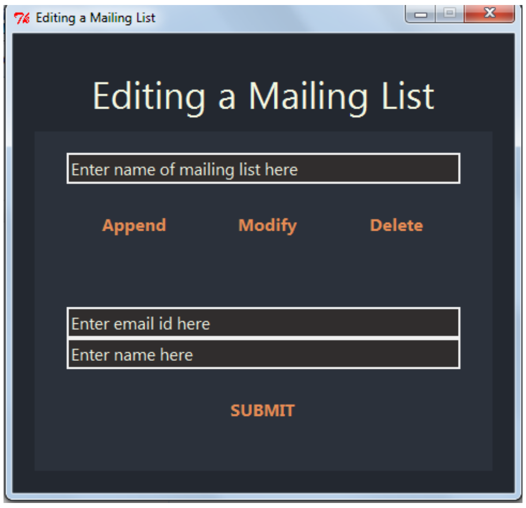
*Window which allows user to enter a message to be sent to all email ids specified in 'To' field and/or the mailing list email ids at a specified time.*



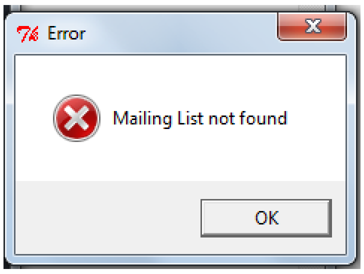
*Window which allows user to specify common details for a reminder. If user enters email id, an email version of reminder is sent.*



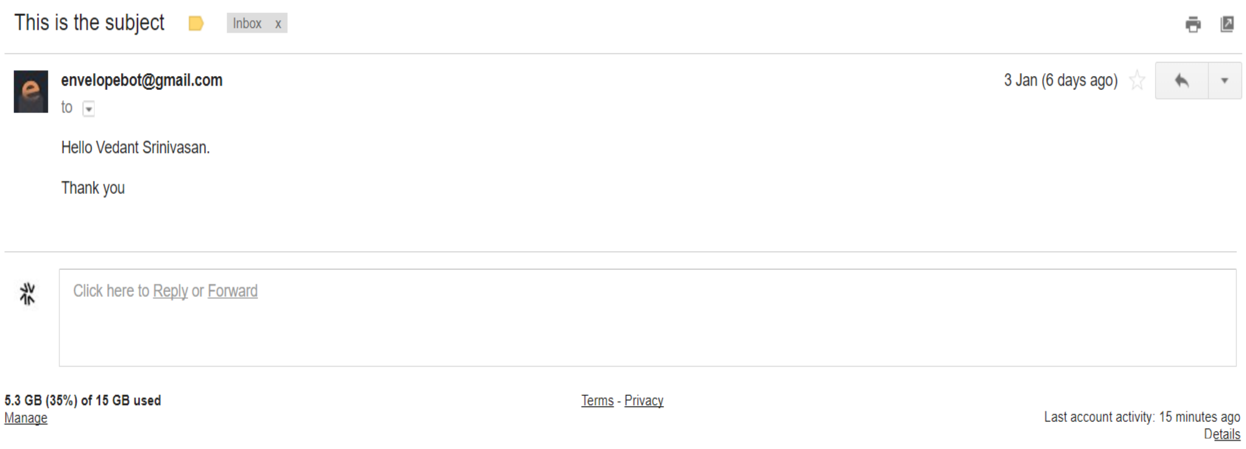
*Window which allows user to edit or create a mailing list.*



*Window which allows user to make changes to an existing mailing list*



*Example of how an error is reported to user. In this case, user has tried to modify a mailing list that doesn't exist.*



This is the subject Inbox x

envelopebot@gmail.com

to

3 Jan (6 days ago)

Hello Vedant Srinivasan.

Thank you

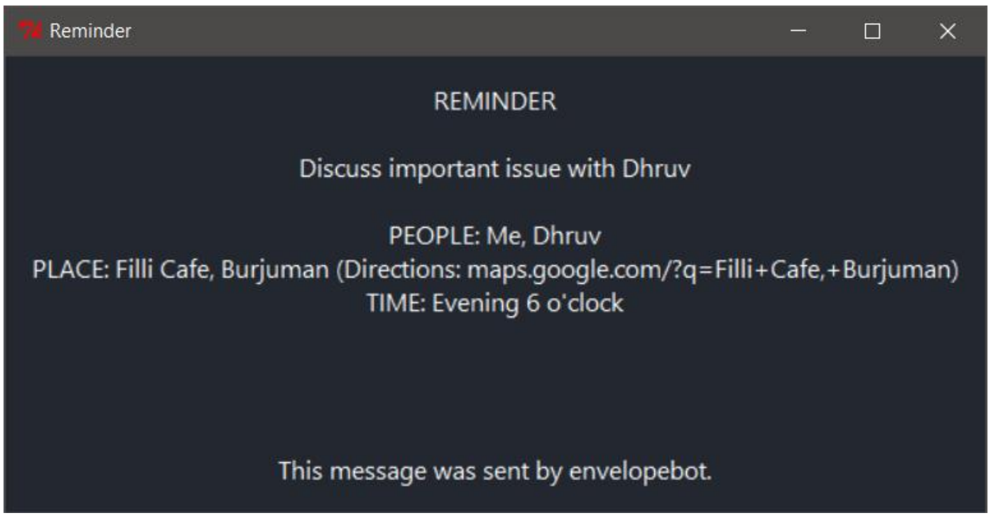
Click here to Reply or Forward

5.3 GB (35%) of 15 GB used
Manage

Terms - Privacy

Last account activity: 15 minutes ago
Details

*Example of a message sent through envelope. When sending an email to multiple people from mailing list, the user can use '%name' as place-holder and it is replaced by the contact name in the mail.*



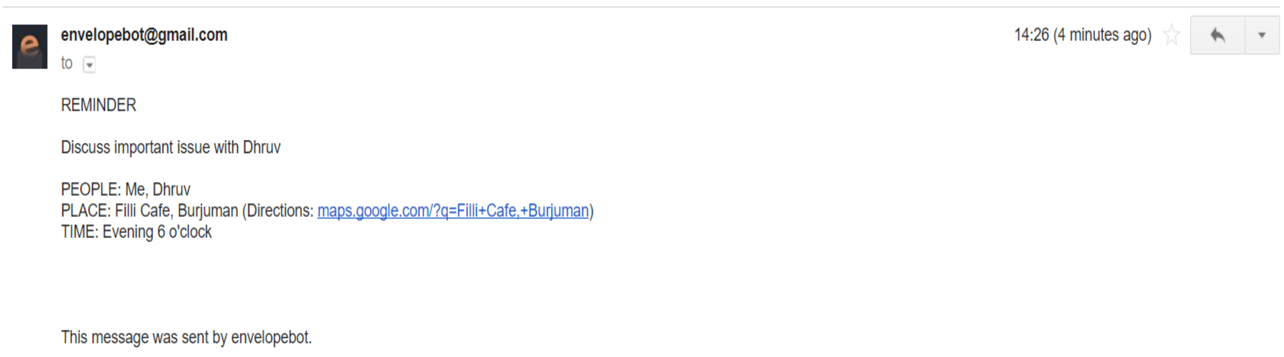
REMINDER

Discuss important issue with Dhruv

PEOPLE: Me, Dhruv
PLACE: Filli Cafe, Burjuman (Directions: maps.google.com/?q=Filli+Cafe,+Burjuman)
TIME: Evening 6 o'clock

This message was sent by envelopebot.

*Example of a popup reminder.*



envelopebot@gmail.com
to

14:26 (4 minutes ago)

REMINDER

Discuss important issue with Dhruv

PEOPLE: Me, Dhruv
PLACE: Filli Cafe, Burjuman (Directions: maps.google.com/?q=Filli+Cafe,+Burjuman)
TIME: Evening 6 o'clock

This message was sent by envelopebot.

*Example of a reminder sent through email.*

## Bugs Encountered

Since envelope is a .pyw file, errors are not reported to the user in the console. Some uncommon errors may just close the program without any messages. Common errors are handled and reported using message boxes.

1. Invalid email id. The program will send the message to all the valid email ids and ignore the others.
2. You cannot schedule messages for the next day as the program will be closed as soon as you shut down your computer. A possible workaround is to store the program online. In that way your computer's state of activity will not matter.
3. When you change the default email id for the first time, you might not able to send messages. To do so, enable 'less secure apps' from your gmail account settings.
4. envelope cannot run on mobile devices during this period of time where everyone uses mobile phones to send messages.

## Future Modifications

1. Ability to send attachments like pictures, videos and other files.
2. WhatsApp, SMS support.
3. Ability to retrieve messages.
4. Conversion into an .exe file so that people without Python can use it.

# Bibliography

1. Learning Python by Mark Lutz
2. Python Documentation (https://docs.python.org/2/)
3. Stackoverflow (stackoverflow.com)
4. Effbot (effbot.org/Tkinterbook)