San Jose State University
Department of Electrical Engineering
EE104, Fall 2023, Pham

Laboratory Assignment #2

Objectives

This lab introduces you the art of integration a few tools that you are already familiar to provide a business solution that you can use in the real life application.

In this lab, we will implement an appointment registration GUI using a new method, a simple database application using Excel, an email method to send a reminder emails for the 2nd follow-up visitation, and a text method to send a reminder text message to a given phone number for the 2nd visitation.

Grading

Refer to the section **Python Programming** for grading criteria.

Bibliography

I would like to acknowledge the Python open-source community and respective suppliers for making the material available. Jose Estrada Ramirez from EE104 Spring 2021 contributed some parts to this lab.

References:
https://www.simplifiedpython.net/python-gui-login/
https://stackoverflow.com/questions/46268167/how-to-search-for-data-in-an-xlsx-file-using-python-3
https://pythonguides.com/python-tkinter-label/
https://stackoverflow.com/questions/42491486/setting-an-image-as-a-tkinter-window-background
https://www.geeksforgeeks.org/python-simple-registration-form-using-tkinter/?ref=rp
https://riptutorial.com/tkinter/example/29713/grid--#:~:text=tkinter%20grid()&text=The%20grid()%20geometry%20manager,%2C%20row%20%2C%20rowspan%20and%20sticky%20.
https://stackoverflow.com/questions/17267140/python-pack-and-grid-methods-together
https://northernlights.imanet.org/home?ssopc=1
https://www.freecodecamp.org/news/exception-handling-python/
https://www.twilio.com/docs/sms/quickstart/python
https://medium.com/paul-zhao-projects/sending-emails-with-python-c084b55a2857

High-level Process

The example below follows a COVID-19 vaccination process, but the idea is expandable to any appointment platform that has two or more follow-up visitations.

It is time for a person to receive a COVID19 vaccination. The following is a typical process in the Santa Clara County.

1. IT staff creates 2 databases. One for medical staff admins, and one for patient records.
2. IT staff creates an admin record for a medical staff and save the information to the staff database.
3. The medical staff creates a patient record and saves in the patient database with name, DOB, phone number, email, etc.
4. The medical staff administer a vaccination shot for the patient, and log in today's date.
    a. At this time, the patient database saves the first shot record, and automatically schedule the 2nd shot for 21 days later.
    b. An IT automated messaging service sends the first text message to the patient congratulating on the 1st vaccination, and show the date for the 2nd shot.
    c. An IT automated email service sends the first email to the patient congratulating on the 1st vaccination, and show the date for the 2nd shot.
5. Three days before the 2nd appointment:
    a. An IT automated messaging service sends the reminding text message to the patient for the 2nd shot date.
    b. An IT automated email service sends the reminding email to the patient for the 2nd shot date

Of course the process continues for the 3rd and 4th shots, but we will stop at Step 5 above because when you know how to do it to this point, you can extend the program to add any number of steps.

Because of the text messaging and email services, you will need to leverage 3rd party tools. Continue to the next sections below.

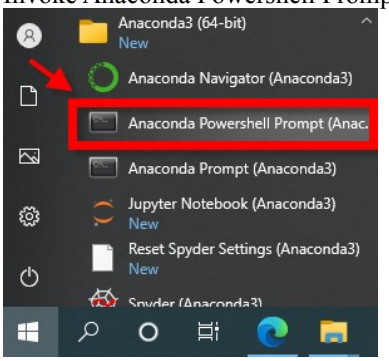
Download, Installation, and Licensing

1. Install necessary Python packages

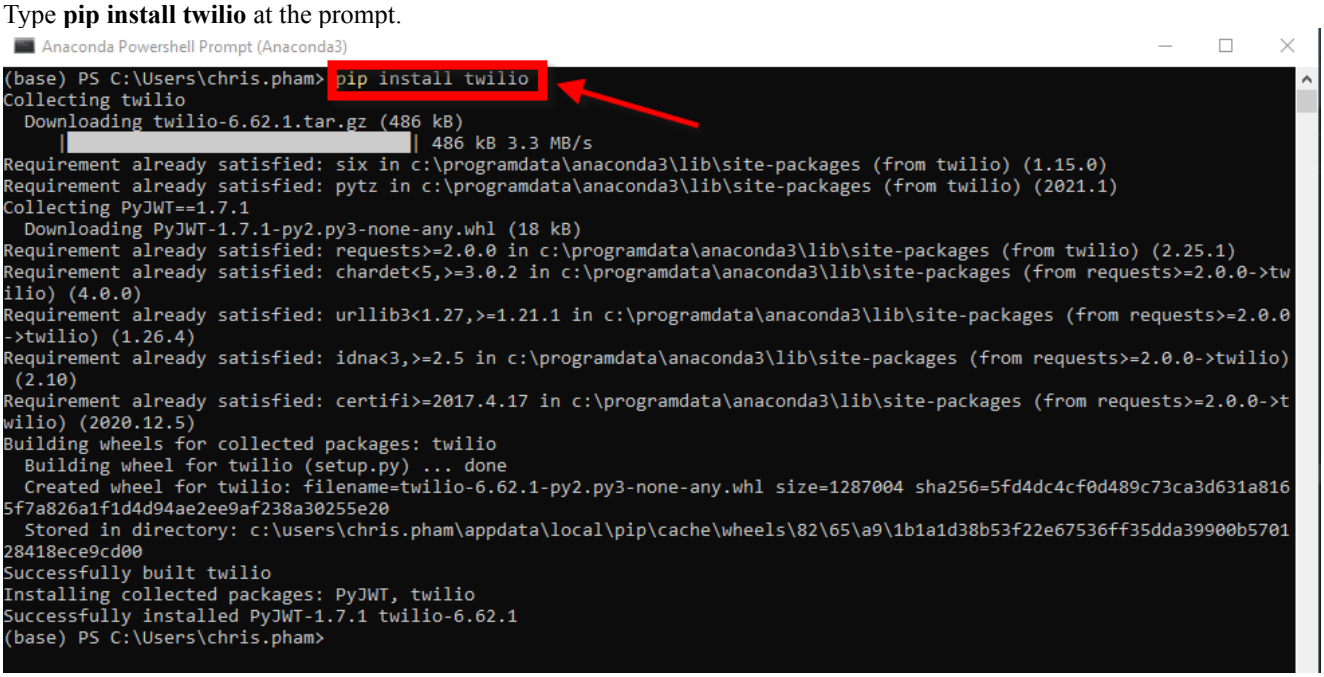
You will need to PIP INSTALL the followings: **pip install twilio**

Reference to https://www.twilio.com/the-current/what-is-twilio-how-does-it-work, we will use Twilio to send SMS phone messages.

Invoke Anaconda Powershell Prompt



Type **pip install twilio** at the prompt.



1. Procedure to create a free-trial Twilio Account

Overview

WITH TWILIO YOU CAN BUILD:

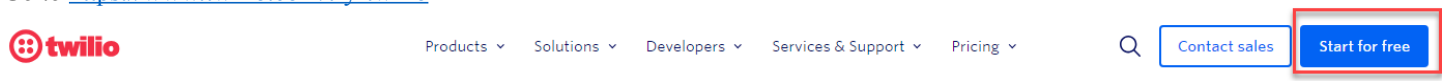
- SMS marketing
- Omnichannel contact center
- Call tracking
- Web chat
- Push notifications

- Alerts and notifications
- Phone verification

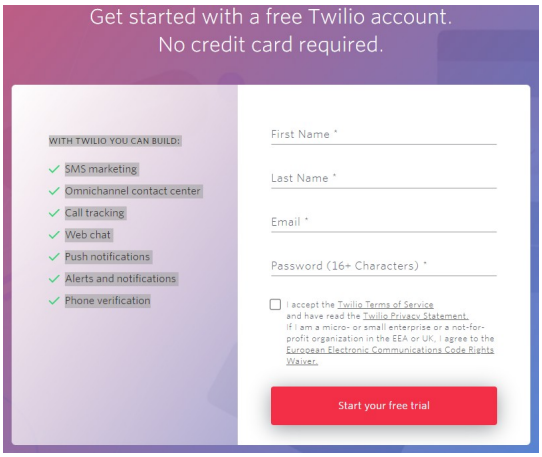
Application: Lab 2 to send SMS text messages

Procedure to create a free-trial Twilio account

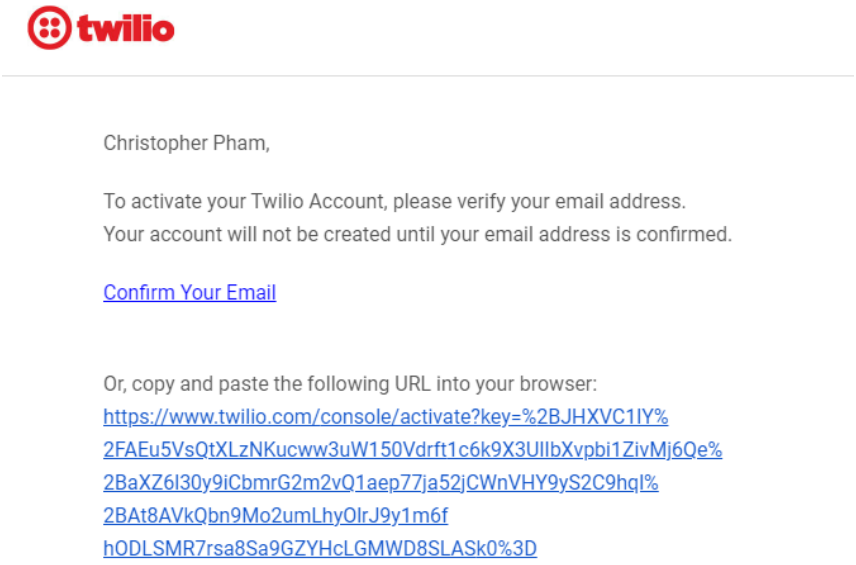
Go to https://www.twilio.com/try-twilio



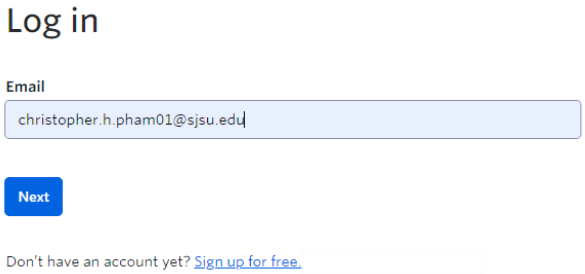
and fill out the form:



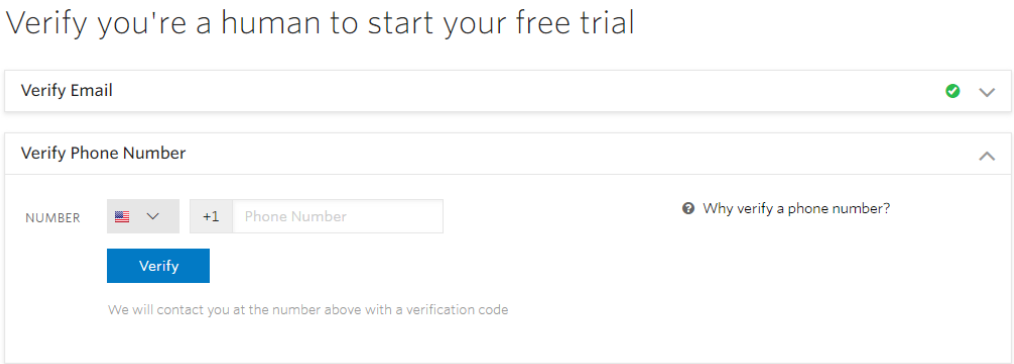
You will receive an email with a link to confirm your email. Click on the link or copy and paste the URL into a browser to confirm:



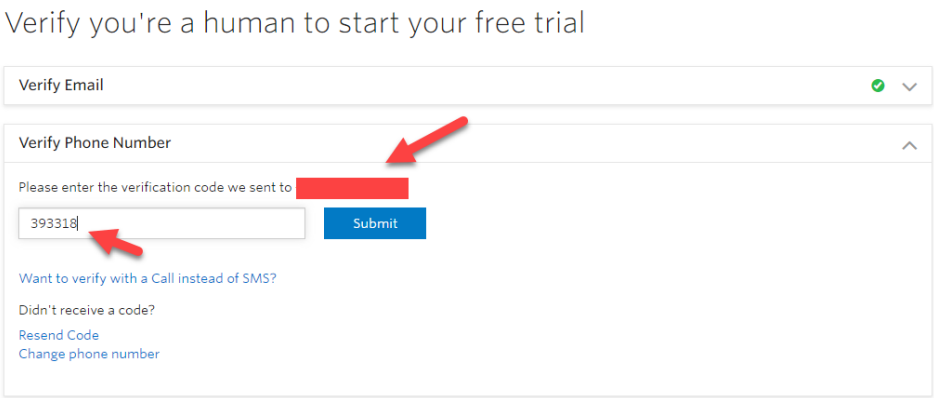
Then you can log in



Now you must provide a phone number to start the free trial:



Enter the number Twilio texts to your phone and submit to start the trial.



On the next screen, select the choices as

Ahoy Christopher Pham, welcome to Twilio!

Tell us a bit about yourself so we can personalize your experience. You will have access to all Twilio products.

• **Which Twilio product are you here to use?**

SMS

• **What do you plan to build with Twilio?**

Alerts & Notifications

• **How do you want to build with Twilio?**

- (●) With code
  Customize exactly what you want
- ( ) With minimal code
  Build on top of our code samples
- ( ) With no code at all
  Launch a starter app with no code

• **What is your preferred coding language?**

Python

• **Would you like Twilio to host your code?**
Host your Twilio app on our secure servers

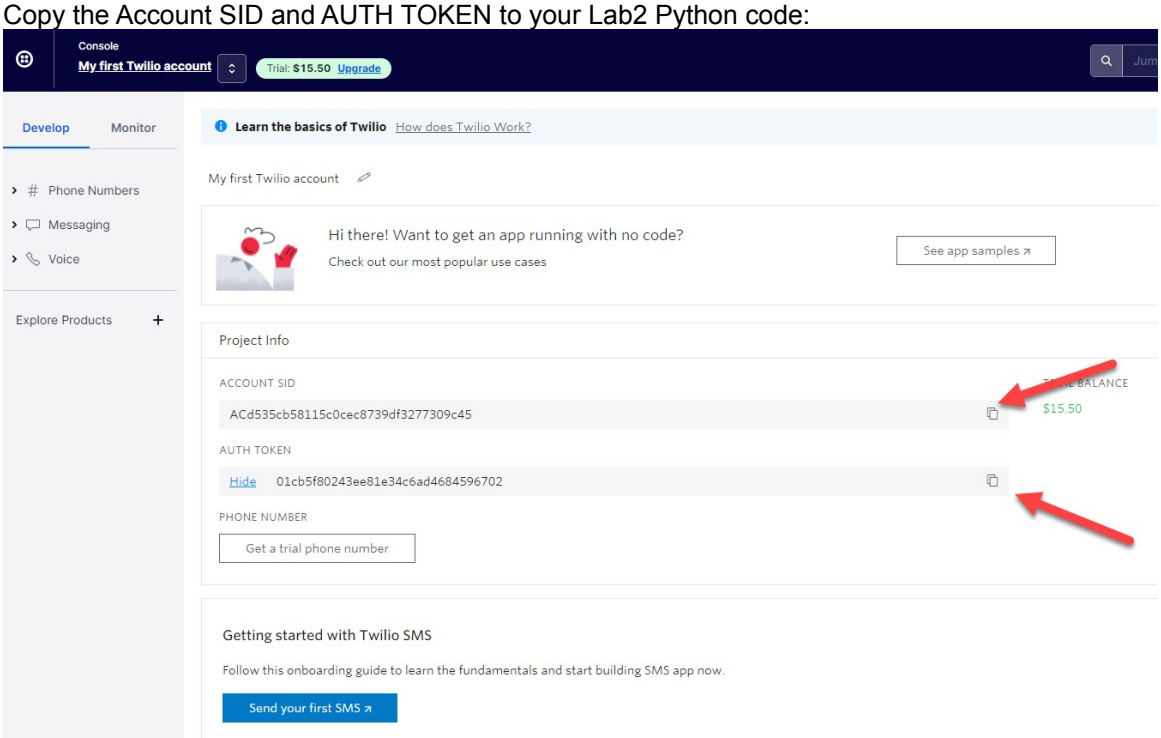
- ( ) Yes, host my code on Twilio
- (●) No, I want to use my own hosting service

Your billing country is United States. **Change**

Get Started with Twilio

 Privacy Policy | Terms of Service

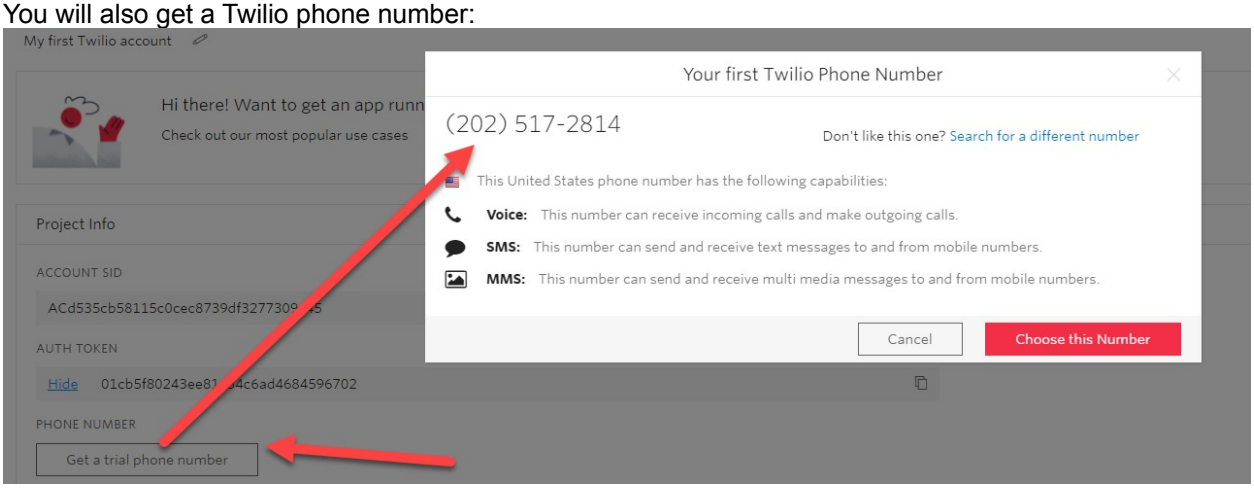
After you click on the blue button above, you will see this screen
Copy the Account SID and AUTH TOKEN to your Lab2 Python code:



There are one location in the code that you must change to your own SID and AUTH TOKEN:

```
Twilio_Number="+15625804763", #this is your own Twilio number
account_sid = 'ACd535cb58115c0cec8739df3277309c45'  #this is your account SID
auth_token = '4432c9f31598f093a77af87b3e4e34b5'     #this is your own auth_token

from_address = "ee104sjsu@gmail.com" #this is your own gmail account
app_password = 'watrmvxhvujcxuus' # a token for gmail, this is the app password from Gmail Security
```

You will also get a Twilio phone number:



Press the button Choose this Number

Congratulations!

Your new Phone Number is **+12025172814**

For help building your Twilio application, check out the resources on the getting started page.
Once you've built your application, you can configure this phone number to send and receive calls and messages.

Done

My first Twilio account

Hi there! Want to get an app running with no code?

Check out our most popular use cases

See app samples

Project Info

ACCOUNT SID

ACd535cb58115c0cec8739df3277309c45

AUTH TOKEN

Hide 01cb5f80243ee81e34c6ad4684596702

PHONE NUMBER

+12025172814

Upgrade to buy more numbers

TRIAL BALANCE

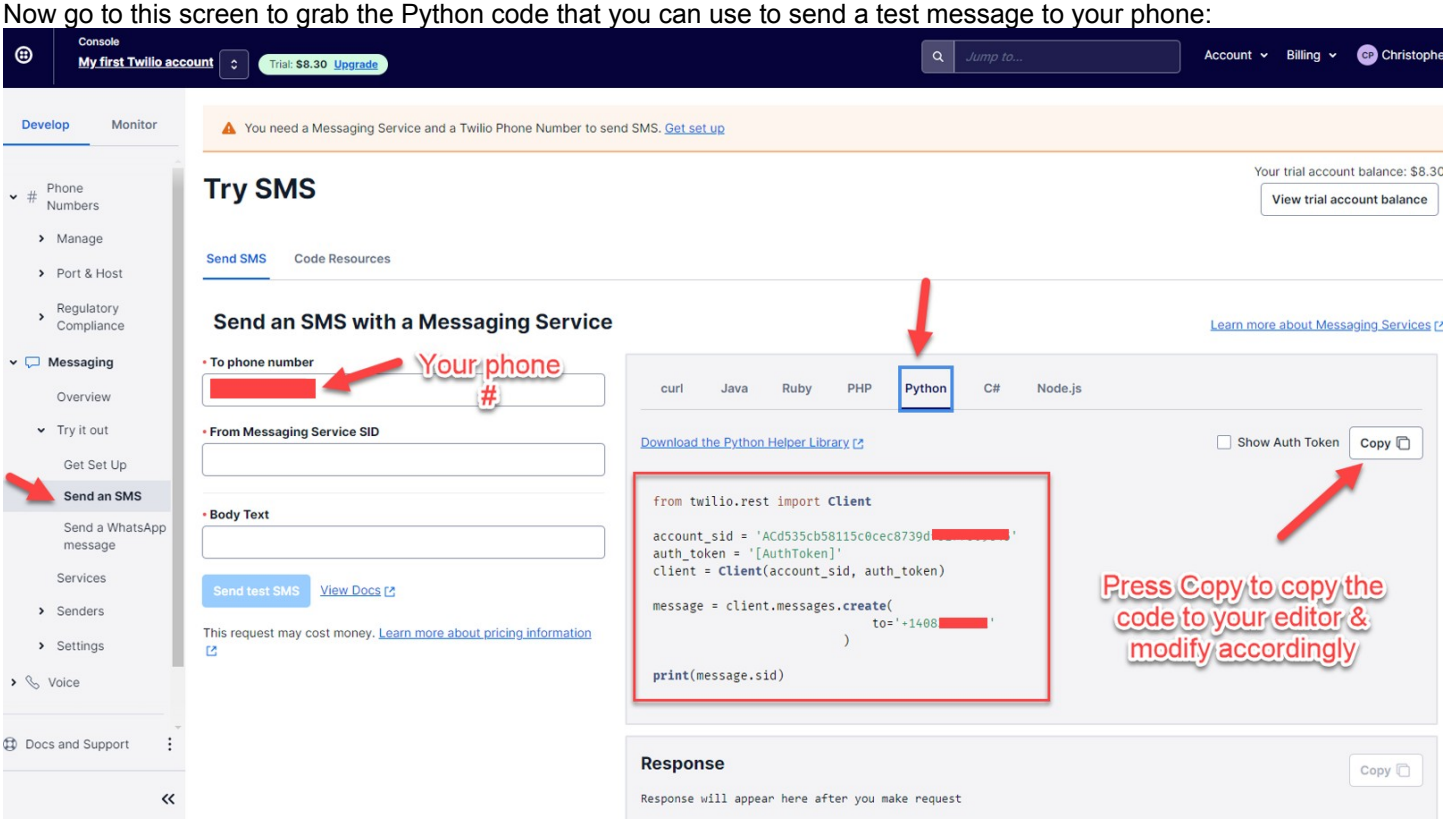
$14.50

Getting started with Twilio SMS

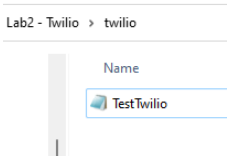
Follow this onboarding guide to learn the fundamentals and start building SMS app now.

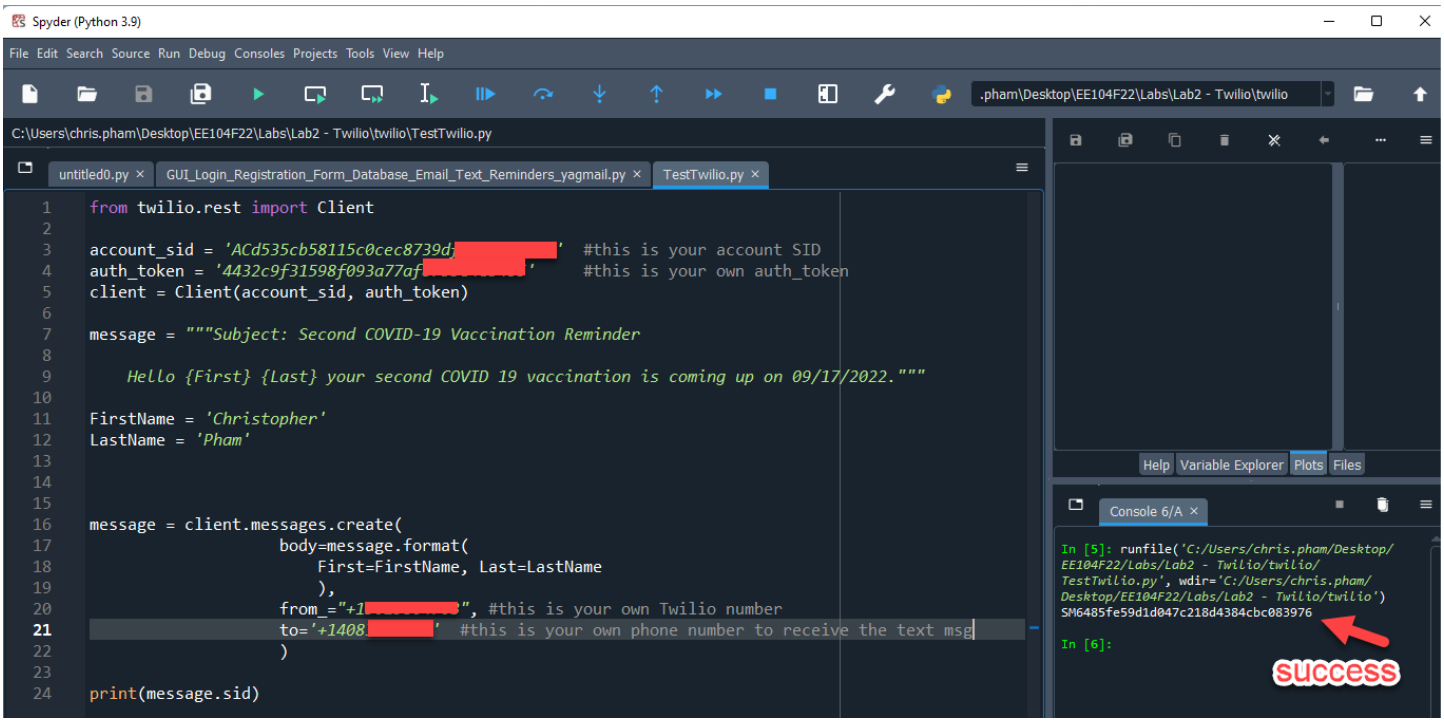
Send your first SMS

Now go to this screen to grab the Python code that you can use to send a test message to your phone:



I have created a sample file for you to modify and send a test text message to your phone. Download subfolder Twilio from Canvas and modify the source code accordingly to send a test message to your phone.





Then use that number in your Lab2 code. There is one location in the code that you need to replace the phone number:

```
Twilio_Number="+15625804763", #this is your own Twilio number
account_sid = 'ACd535cb58115c0cec8739df3277309c45'  #this is your account SID
auth_token = '4432c9f31598f093a77af87b3e4e34b5'     #this is your own auth_token

from_address = "ee104sjsu@gmail.com" #this is your own gmail account
app_password = 'watrmvxhvujcxuus' # a token for gmail, this is the app password from Gmail Security
```

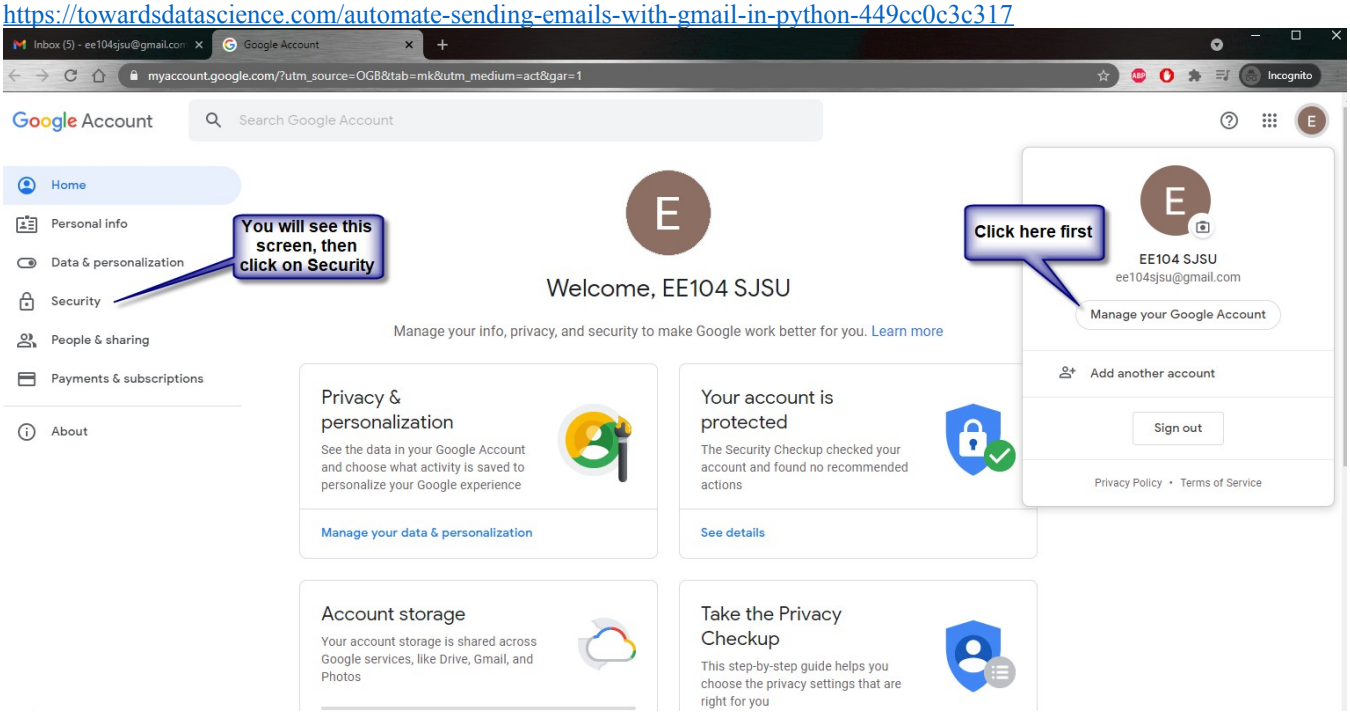
That's it! Enjoy!

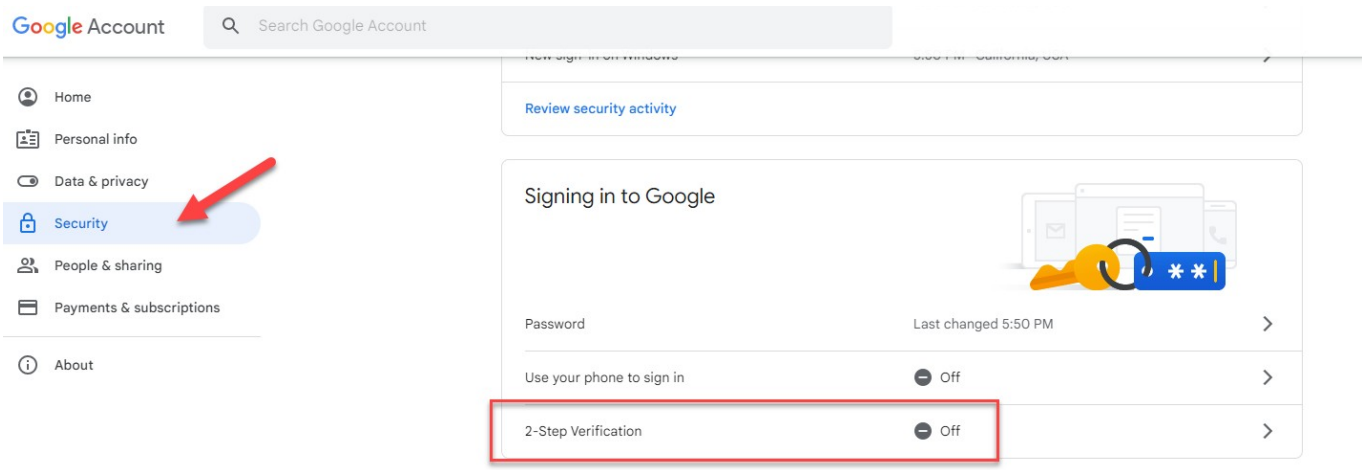
2. Set up a Gmail account for test emails

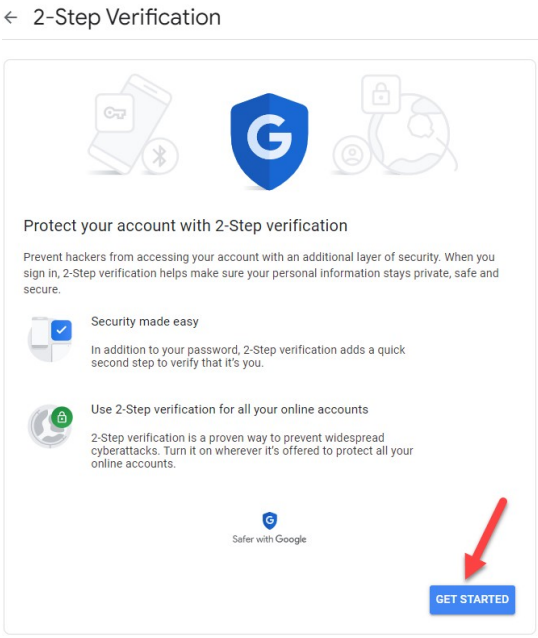
Follow the instruction from this URL to set up a Gmail account for development.
https://medium.com/paul-zhao-projects/sending-emails-with-python-c084b55a2857

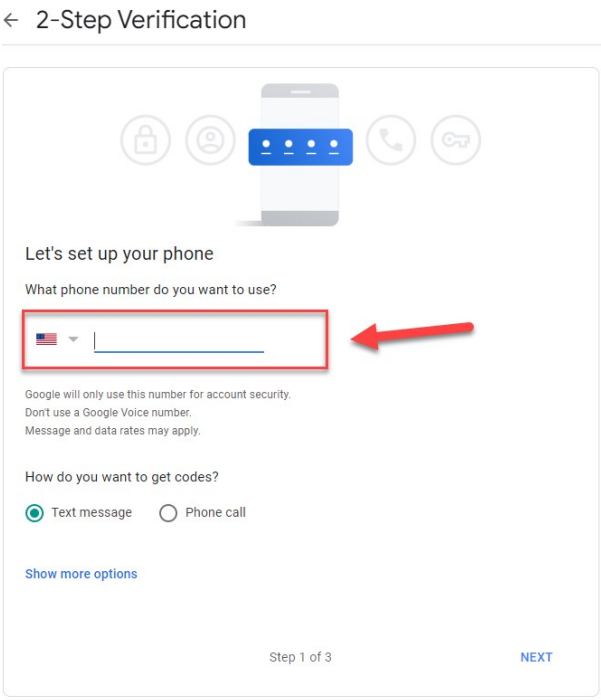
Create a Gmail account as regular. You can call it anything you want. This email will be used for the EE104 lab purposes only.

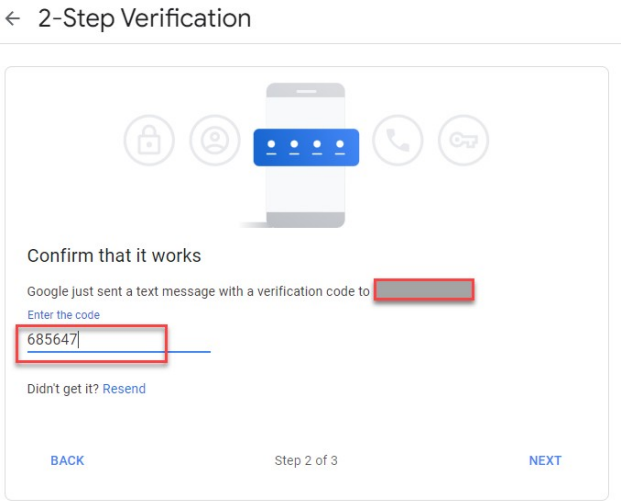
In order for Python to send email from your Gmail account, you will follow the steps from here:
https://towardsdatascience.com/automate-sending-emails-with-gmail-in-python-449cc0c3c317




Google Account
Search Google Account
Home
Personal info
Data & privacy
Security
People & sharing
Payments & subscriptions
About
Review security activity
Signing in to Google
Password
Last changed 5:50 PM
Use your phone to sign in
Off
2-Step Verification
Off

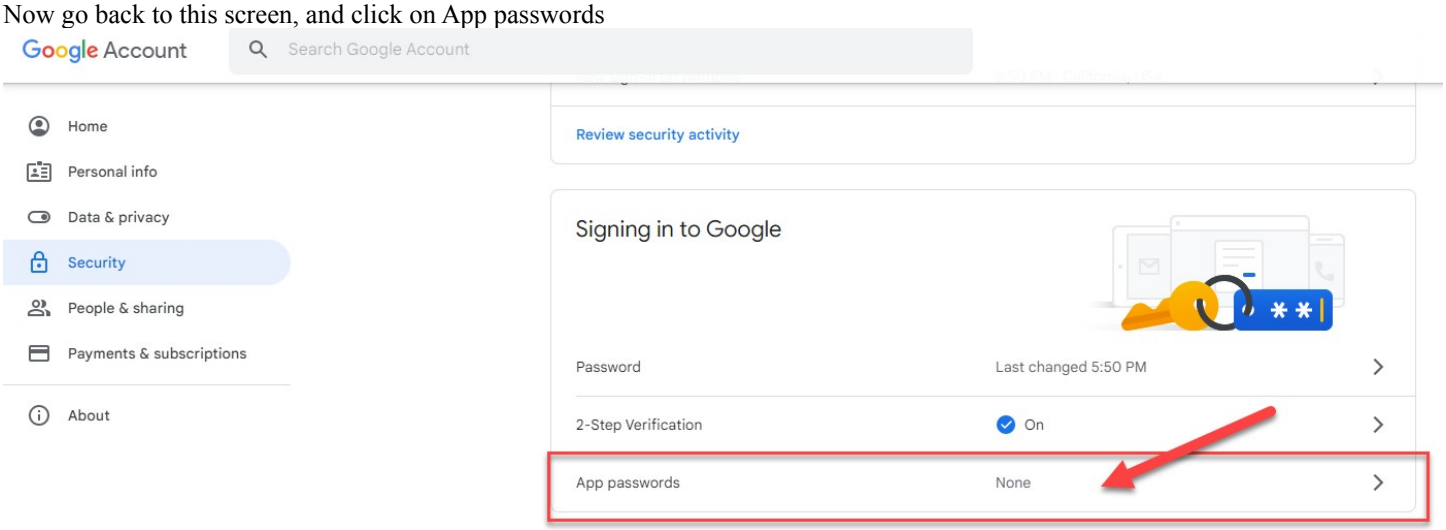

2-Step Verification
Protect your account with 2-Step verification
Prevent hackers from accessing your account with an additional layer of security. When you sign in, 2-Step verification helps make sure your personal information stays private, safe and secure.
Security made easy
In addition to your password, 2-Step verification adds a quick second step to verify that it's you.
Use 2-Step verification for all your online accounts
2-Step verification is a proven way to prevent widespread cyberattacks. Turn it on wherever it's offered to protect all your online accounts.
Safer with Google
GET STARTED


2-Step Verification
Let's set up your phone
What phone number do you want to use?
Google will only use this number for account security.
Don't use a Google Voice number.
Message and data rates may apply.
How do you want to get codes?
Text message
Phone call
Show more options
Step 1 of 3
NEXT

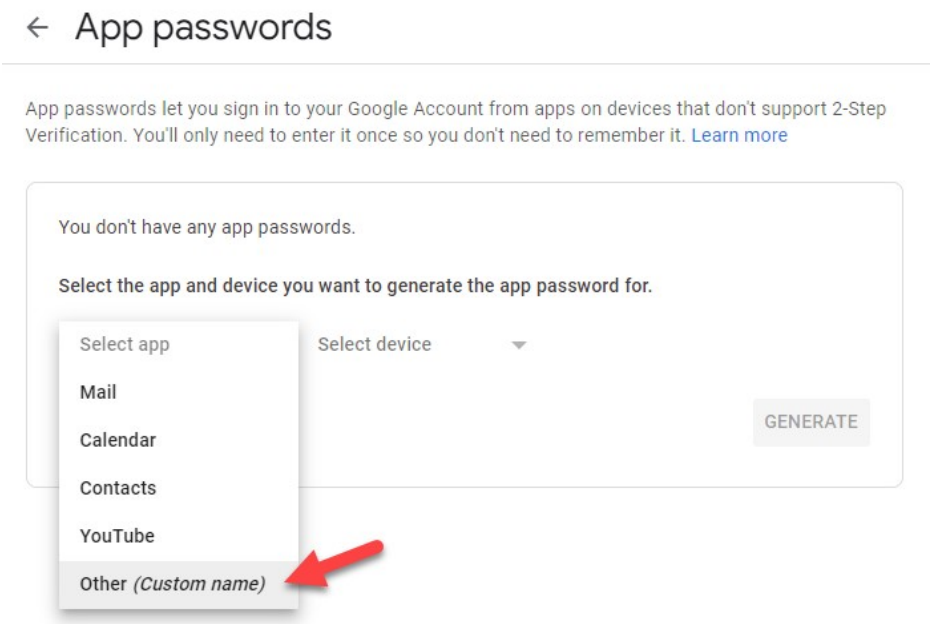


Then click Next on the confirmation screen.

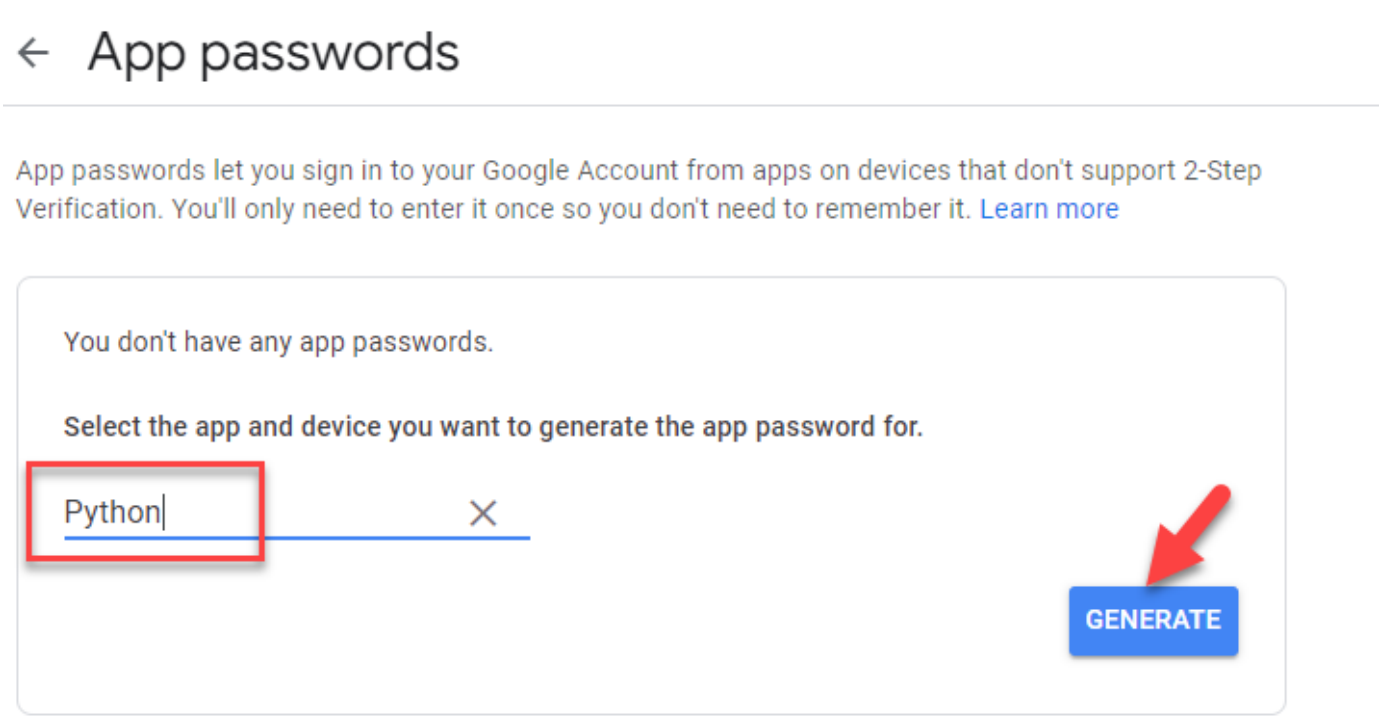
Now go back to this screen, and click on App passwords



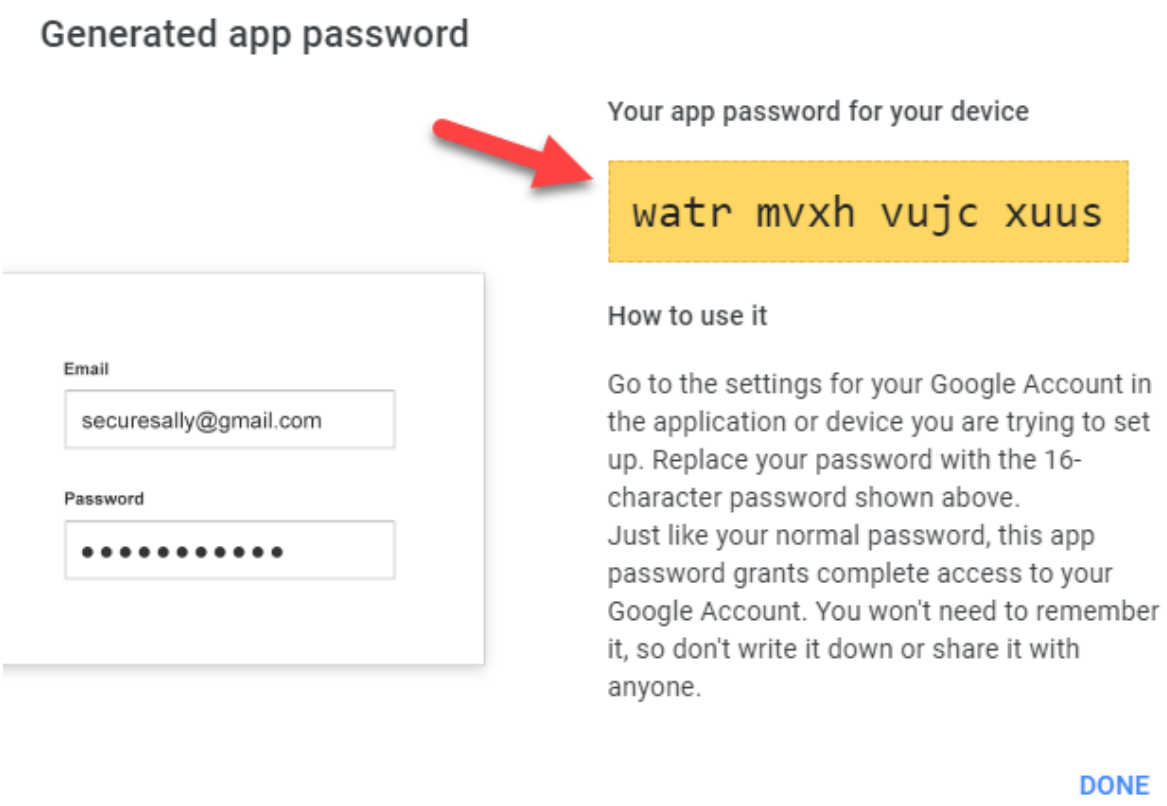
Authenticate as being requested, and you will see this screen, click on **Other (Custom name)**



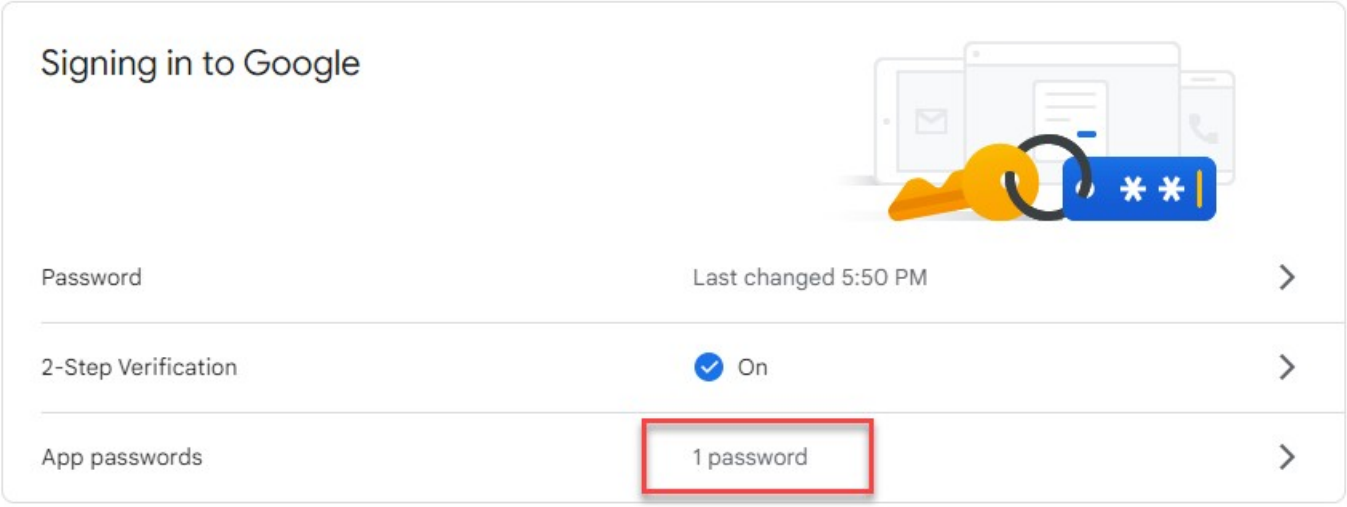
Name it **Python**, then click the **GENERATE** button.



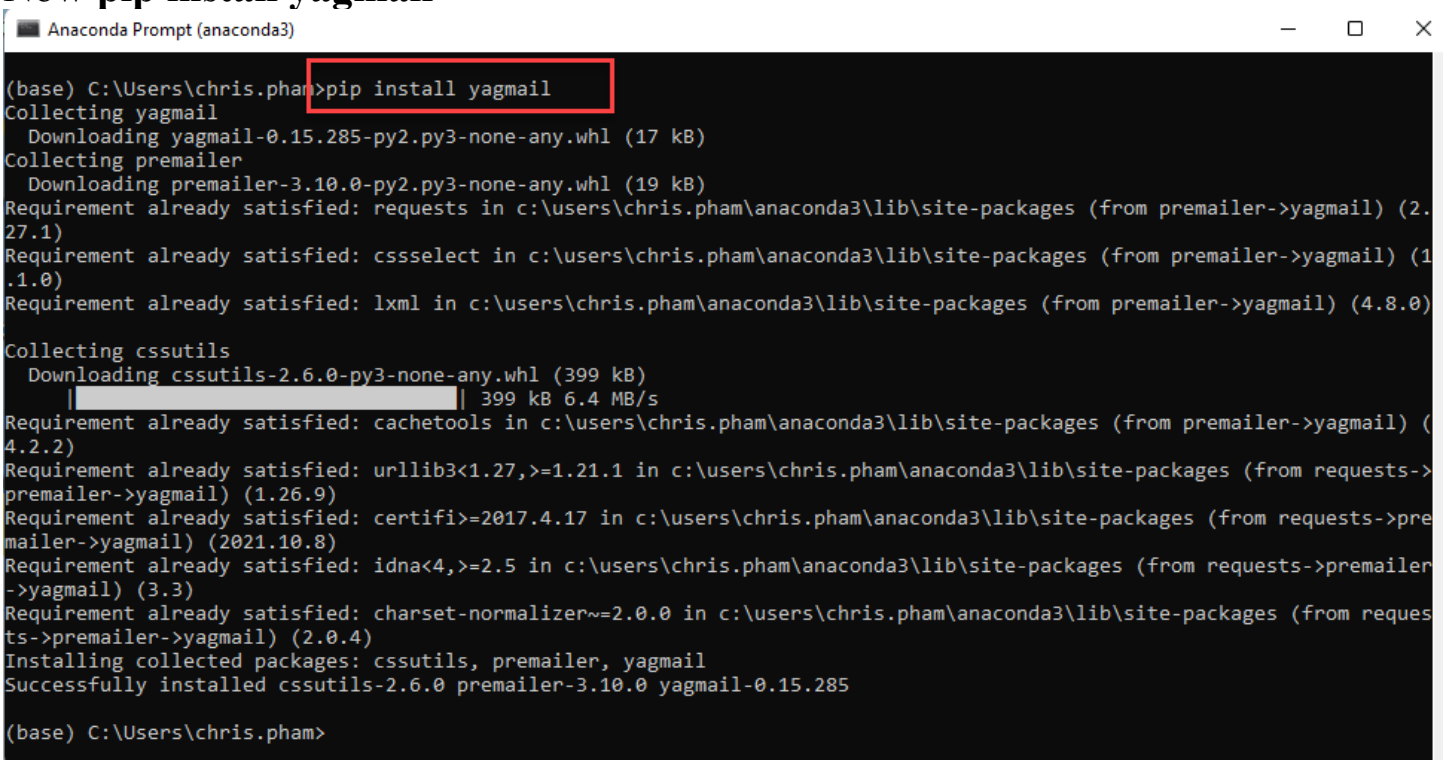
Then you will get a new app password. Copy and save the 16-character password without space, e.g. **watrmvxhvujcxuus** from the screenshot below, to use in your Python script.



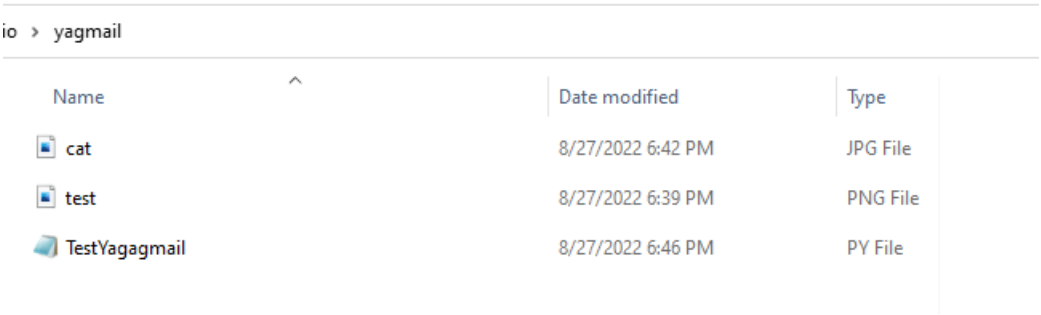
You will see 1 password in your screen below:



Now **pip install yagmail**



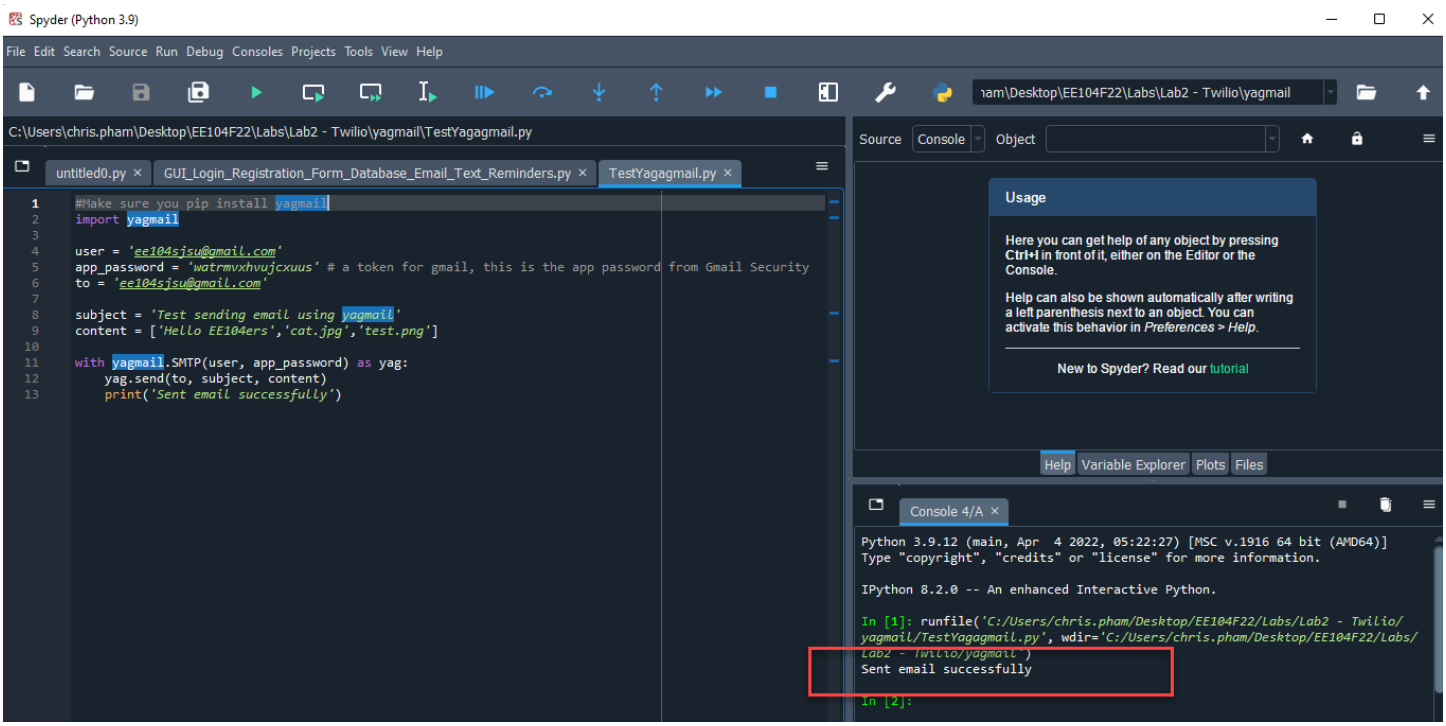
Download the content from the folder yagmail from Canvas



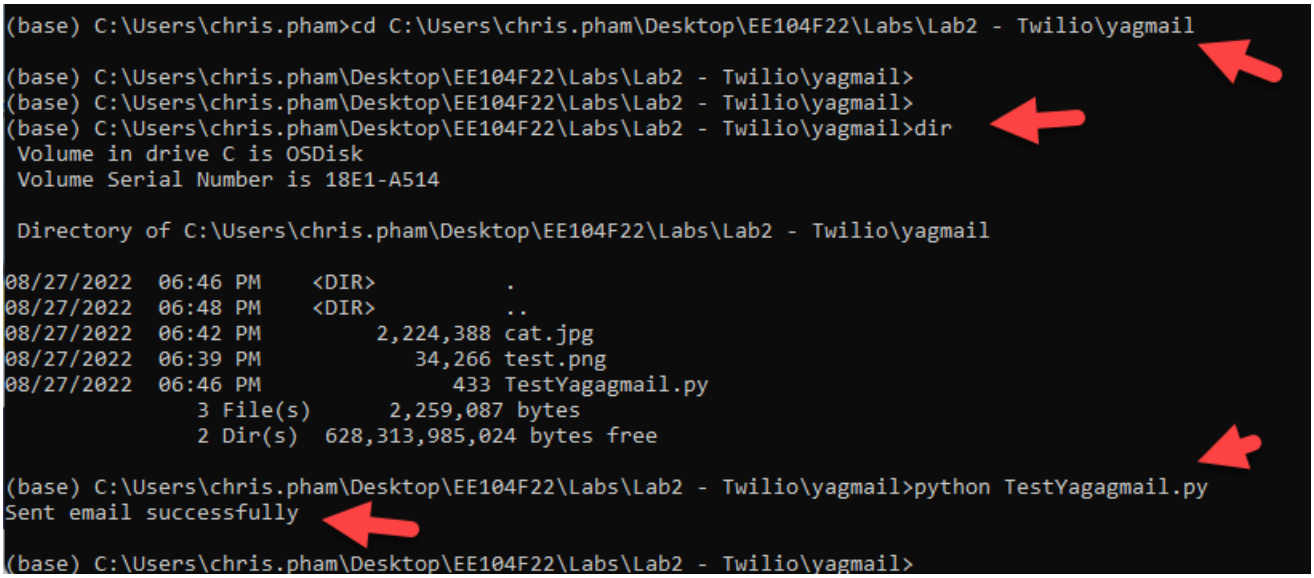
Make change accordingly using your own email and app password:

```python
#Make sure you pip install yagmail
import yagmail

user = 'ee104sjsu@gmail.com' #this is your own gmail account
app_password = 'watrmvxhvujcxuus' # a token for gmail, this is the app password from Gmail Security
to = 'ee104sjsu@gmail.com'   #send test to another email or the same email is OK

subject = 'Test sending email using yagmail'  #modify the subject line anyway you like
content = ['Hello EE104ers','cat.jpg','test.png']  #you can have different email and attachment

with yagmail.SMTP(user, app_password) as yag:
    yag.send(to, subject, content)
    print('Sent email successfully')  #you can have different success message
```

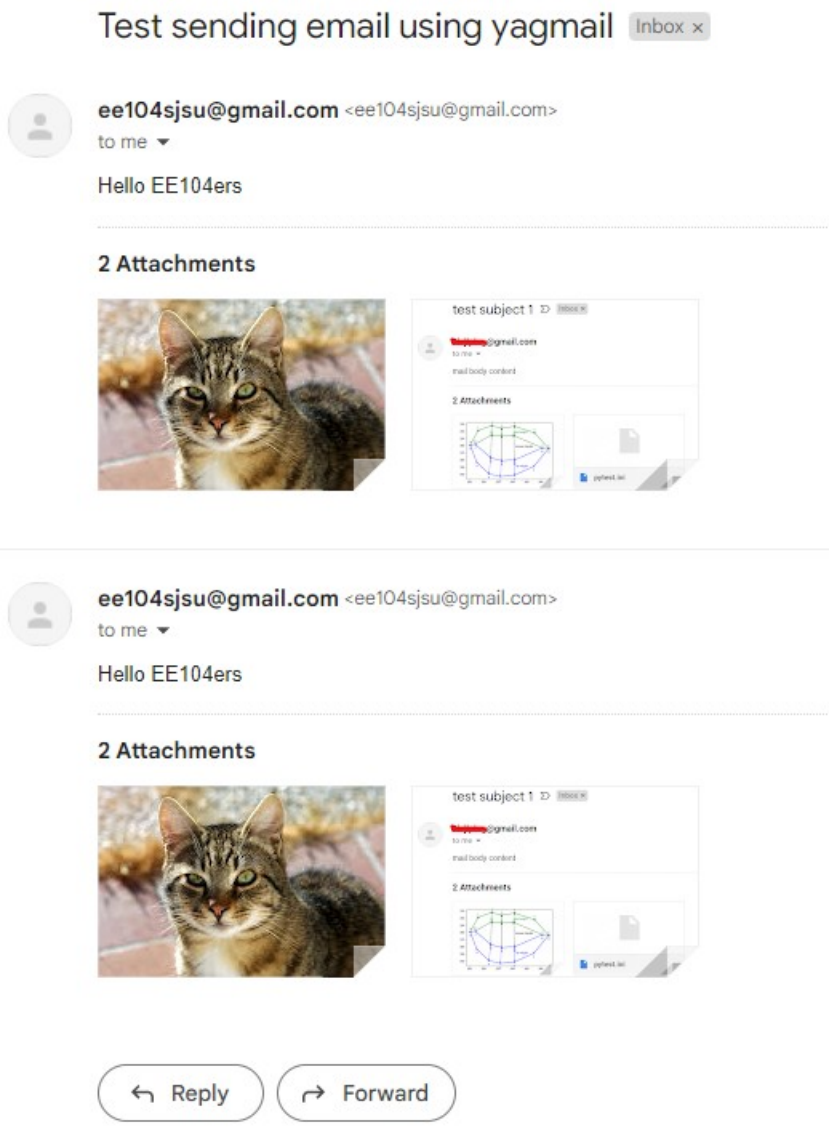
And execute the file in Spyder or from your command line



From the command line:



Validate that you get 2 emails, one from Spyder, and one from the command line execution:



OTHER METHODS

There is also another method using OAuth2 authorization framework. You can read for yourself here: https://developers.google.com/gmail/api/quickstart/python

Sample Excel Files

There are three Excel files:

File Name	File Type	Purpose
Simple_Registration_Database.xlsx	Microsoft Excel Worksheet	Capture personal information of vaccine patron
Registration_UserName_Password.xlsx	Microsoft Excel Worksheet	Store the UserName and Password of the vaccine patron
COVID_Vaccine_Database.csv	Microsoft Excel Comma Separated Value File	For Python to record the 1st vaccination date, calculate the 2nd date for the 2nd dose, and send text and/or email to remind the patron

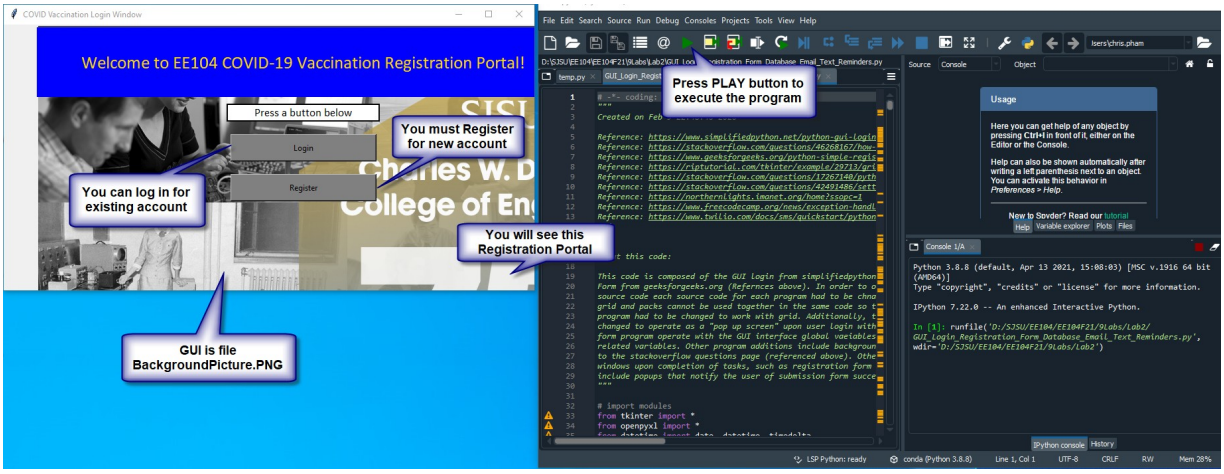
GUI File

There is only one background picture file for Graphic User Interface (GUI). The file name is BackgroundPicture.PNG. You actually can have it in any other picture format such as JPG, etc. Try it out for yourself.

Sample Python Program

The file **GUI_Login_Registration_Form_Database_Email_Text_Reminders.py** is provided to you. Here are some highlights.

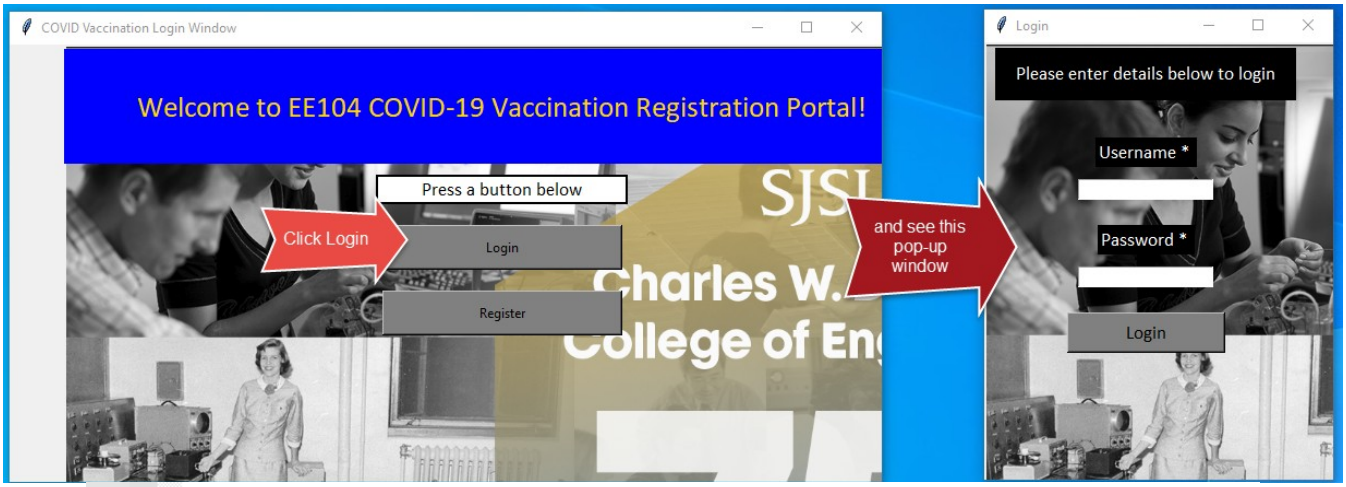
Sequence of execution and related Python code:



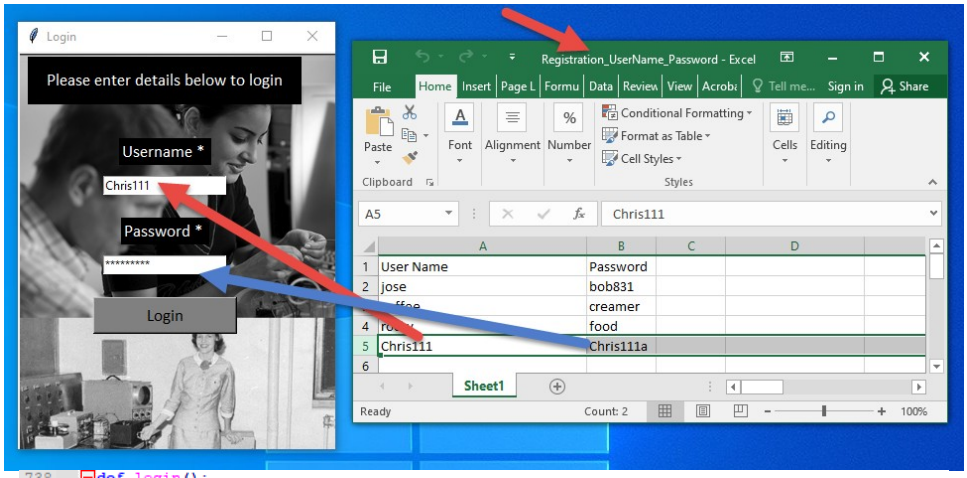
Code to produce the GUI above:

```python
# Designing Main(first) window
def main_account_screen():
    global main_screen
    global df

    # Generate GUI
    main_screen = Tk()

    # Resize main screen
    main_screen.geometry("800x400")
    main_screen.title("COVID Vaccination Login Window")

    # Import and set ground of main screen
    bg = PhotoImage(file="BackgroundPicture.png", height=400, width=800)
    bg_label = Label(main_screen, image=bg)
    bg_label.image = bg  # keep a reference!
    bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)

    Label(
        text="Welcome to EE104 COVID-19 Vaccination Registration Portal!",
        fg="gold",
        bg="#0000FF", #all RGB = 0000FF
        width="57",
        height="3",
        font=("Calibri", 20),
    ).grid(row=0, column=2, padx=50)

    Label(
        main_screen,
        relief="solid",
        text="Press a button below",
        width="25",
        height="1",
        fg="black",
        bg="white",
        font=("Calibri", 13),
    ).grid(row=1, column=2, padx=50)

    Button(text="Login", height="2", width="30", bg="grey", command=login).grid(
        row=3, column=2
    )

    Button(text="Register", height="2", width="30", bg="grey", command=register).grid(
        row=5, column=2
    )

    main_screen.mainloop()


main_account_screen()
```

This is how we call the function main_account_screen()




```python
# Designing window for login
def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("320x400")

    bg = PhotoImage(file="BackgroundPicture.png", height=400, width=320)
    bg_label = Label(login_screen, image=bg)
    bg_label.image = bg  # keep a reference!
    bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)

    Label(
        login_screen,
        text="Please enter details below to login",
        fg="white",
        bg="black",
        width="30",
        height="2",
        font=("Calibri", 13),
    ).grid(row=0, column=0, padx="10")

    global username_verify
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
    global password_login_entry

    Label(
        login_screen, text="Username * ", fg="white", bg="black", font=("Calibri", 13)
    ).grid(row=3, column=0)
    username_login_entry = Entry(login_screen, textvariable=username_verify)
    username_login_entry.grid(row=4, column=0)

    Label(
        login_screen, text="Password * ", fg="white", bg="black", font=("Calibri", 13)
    ).grid(row=6, column=0)
    password_login_entry = Entry(login_screen, textvariable=password_verify, show="*")
    password_login_entry.grid(row=7, column=0)

    Button(
        login_screen,
        text="Login",
        width=15,
        height=1,
        bg="grey",
        font=("Calibri", 13),
        command=login_verify,
    ).grid(row=9, column=0)
```

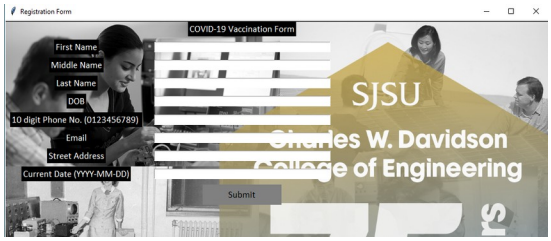



```python
def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("320x400")

    bg = PhotoImage(file="BackgroundPicture.png", height=400, width=320)
    bg_label = Label(login_screen, image=bg)
    bg_label.image = bg  # keep a reference!
    bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)

    Label(
        login_screen,
        text="Please enter details below to login",
        fg="white",
        bg="black",
        width="30",
        height="2",
        font=("Calibri", 13),
    ).grid(row=0, column=0, padx="10")

    global username_verify
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
    global password_login_entry

    Label(
        login_screen, text="Username * ", fg="white", bg="black", font=("Calibri", 13)
    ).grid(row=3, column=0)
    username_login_entry = Entry(login_screen, textvariable=username_verify)
    username_login_entry.grid(row=4, column=0)

    Label(
        login_screen, text="Password * ", fg="white", bg="black", font=("Calibri", 13)
    ).grid(row=6, column=0)
    password_login_entry = Entry(login_screen, textvariable=password_verify, show="*")
    password_login_entry.grid(row=7, column=0)

    Button(
        login_screen,
        text="Login",
        width=15,
        height=1,
        bg="grey",
        font=("Calibri", 13),
        command=login_verify,
    ).grid(row=9, column=0)


# Write user name and password to spreadhseet
#   Use to modify Excel file
 Registration_UserName_Password.xlsx
def excel2():

    # resize the width of columns in
    # excel spreadsheet
    sheet2.column_dimensions["A"].width = 30
    sheet2.column_dimensions["B"].width = 10

    # write given data to an excel spreadsheet
    # at particular location
    # Columns for user name and password
    sheet2.cell(row=1, column=1).value = "User Name"
    sheet2.cell(row=1, column=2).value = "Password"
```


```
# Implementing event on login button
def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    # create name list to search from for login dat
    # create passwords list to search from
    # https://stackoverflow.com/questions/51800122/using-openpyxl-to-find-rows-that-contain-cell-with-sp
    names = []
    passwords = []

    for i in range(2, sheet2.max_row + 1):
        if sheet2[i][0].value:
            names.append(sheet2[i][0].value)
            passwords.append(sheet2[i][1].value)

    # exempt ValueError when user inputs a username not in file
    # https://www.freecodecamp.org/news/exception-handling-python/
    try:
        index = names.index(username1)
    except ValueError:
        print("")

    # if password and names are saved allow user to register_form
    # otherwize no user found or incorrect password
    # Check to see if the index number of the username matches with its corresponding user password
    if username1 in names:
        if password1 == passwords[index]:
            login_sucess()
            register_form()
        else:
            password_not_recognised()
    else
        user_not_found()


# Designing popup for login succes
def login_sucess():
    global login_success_screen
    login_success_screen = Toplevel(login_screen)
    login_success_screen.title("Success")
    login_success_screen.geometry("200x100")

    # display Login success to user
    Label(login_success_screen, text="Login Success!", font=("Calibri", 13)).grid(
    )

    # close login success window when pressing button
    Button(
    ).grid(row=1, column=0, padx=25)
```

"get"methods

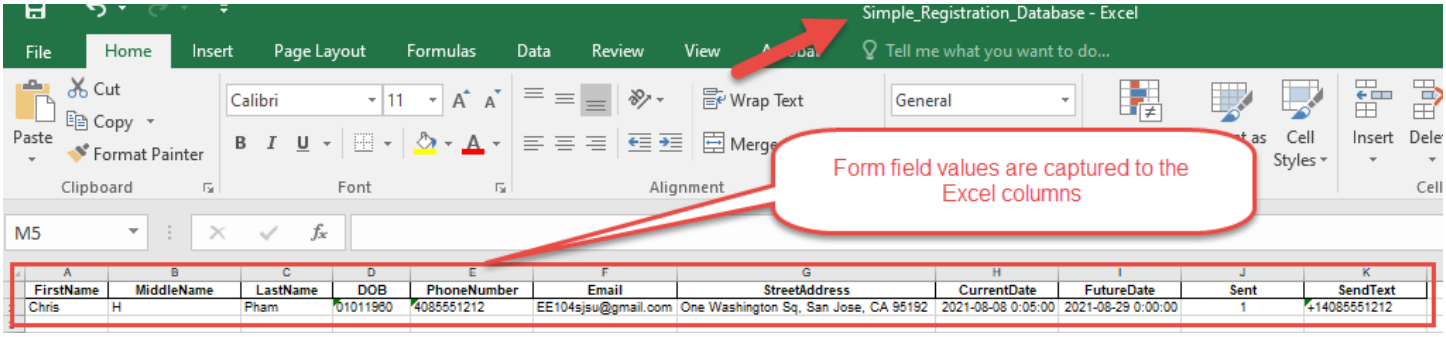
```python
def register_form():
    global root
    # keep screen on top
    root = Toplevel()
    root.title("Registration Form")
    root.geometry("1000x400")

    # set background image to registration form main_screen
    # Reference: https://stackoverflow.com/questions/42491486/setting-an-image-as-a-tkinter-window-background
    # import image and set as backgroung
    bg = PhotoImage(file="BackgroundPicture.png", height=400, width=1000)
    bg_label = Label(root, image=bg)
    bg_label.image = bg  # keep a reference!
    bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)

    # create global variables for registration form inputs to successfully
    # implimnet with GUI program across the rest of the program
    global Name_in
    global course_in
    global sem_in
    global grad_in
    global phone_in
    global email_in
    global address_in
    global current_date_in

    global name_field
    global middle_field
    global last_field
    global DOB_field
    global phone_field
    global email_field
    global address_field
    global current_date_field

    # Widget text variables
    # https://stackoverflow.com/questions/51783852/what-is-the-difference-between-a-variable-and-stringvar-of-tkinter/51785046
    Name_in = StringVar()
    course_in = StringVar()
    sem_in = StringVar()
    grad_in = StringVar()
    phone_in = StringVar()
    email_in = StringVar()
    address_in = StringVar()
    current_date_in = StringVar()

    excel()

    # create a Form label
    heading = Label(
        root,
        text="COVID-19 Vaccination Form",
        fg="white",
        bg="black",
        font=("Calibri", 13),
    )

    # create a Name label
    name = Label(root, text="First Name", bg="black", fg="white", font=("Calibri", 13))

    # create a Course label
    course = Label(
        root, text="Middle Name", bg="black", fg="white", font=("Calibri", 13)
    )
```




```python
    # create a Semester label
    sem = Label(root, text="Last Name", bg="black", fg="white", font=("Calibri", 13))

    # create a graduation year. lable
    grad = Label(root, text="DOB", bg="black", fg="white", font=("Calibri", 13))

    # create a phone No. label
    phone = Label(
        root,
        text="10 digit Phone No. (0123456789) ",
        bg="black",
        fg="white",
        font=("Calibri", 13),
    )

    # create a Email label
    email = Label(root, text="Email", bg="black", fg="white", font=("Calibri", 13))

    # create a address label
    address = Label(
        root, text="Street Address", bg="black", fg="white", font=("Calibri", 13)
    )

    # create a current date label
    c_date = Label(
        root,
        text="Current Date (YYYY-MM-DD)",
        bg="black",
        fg="white",
        font=("Calibri", 13),
    )

    # headings postion on the scree with the use of grid.
    heading.grid(row=0, column=1)
    name.grid(row=1, column=0)
    course.grid(row=2, column=0)
    sem.grid(row=3, column=0)
    grad.grid(row=4, column=0)
    phone.grid(row=5, column=0)
    email.grid(row=6, column=0)
    address.grid(row=7, column=0)
    c_date.grid(row=8, column=0)

    name_field = Entry(root, textvariable=Name_in)
    middle_field = Entry(root, textvariable=course_in)
    last_field = Entry(root, textvariable=sem_in)
    DOB_field = Entry(root, textvariable=grad_in)
    phone_field = Entry(root, textvariable=phone_in)
    email_field = Entry(root, textvariable=email_in)
    address_field = Entry(root, textvariable=address_in)
    current_date_field = Entry(root, textvariable=current_date_in)

    name_field.bind("<Return>", focus1)

    # whenever the enter key is pressed
    # then call the focus2 function
    middle_field.bind("<Return>", focus2)

    # whenever the enter key is pressed
    # then call the focus3 function
    last_field.bind("<Return>", focus3)

    # whenever the enter key is pressed
    # then call the focus4 function
    DOB_field.bind("<Return>", focus4)
```

```python
    # whenever the enter key is pressed
    # then call the focus5 function
    phone_field.bind("<Return>", focus5)

    # whenever the enter key is pressed
    # then call the focus6 function
    email_field.bind("<Return>", focus6)

    address_field.bind("<Return>", focus7)

    # current_date_field.bind("<Return>", focus7)

    # position text entry boxes with grid function in their respected rows/columns
    # column 1 is used ofr text boxes for user input
    name_field.grid(row=1, column=1, ipadx="100")
    middle_field.grid(row=2, column=1, ipadx="100")
    last_field.grid(row=3, column=1, ipadx="100")
    DOB_field.grid(row=4, column=1, ipadx="100")
    phone_field.grid(row=5, column=1, ipadx="100")
    email_field.grid(row=6, column=1, ipadx="100")
    address_field.grid(row=7, column=1, ipadx="100")
    current_date_field.grid(row=8, column=1, ipadx="100")

    # submission button with red background and font change i sused to submit user information
    # into excel file
    submit = Button(
        root,
        text="Submit",
        fg="Black",
        relief="flat",
        bg="grey",
        width=15,
        height=1,
        font=("Calibri", 13),
        command=insert,
    )
    submit.grid(row=9, column=1)
```

Notice the focuses



FirstName	MiddleName	LastName	DOB	PhoneNumber	Email	StreetAddress	CurrentDate	FutureDate	Sent	SendText
Chris	H	Pham	01011960	4085551212	EE104sjsu@gmail.com	One Washington Sq, San Jose, CA 95192	2021-08-08 0:05:00	2021-08-29 0:00:00	1	+14085551212

```python
# Write unser input into excel sheet2
# if either text entry box is empty the user will be notified
def insert():
    # if user not fill any or one entry box
    # then pop up empty input screen
    if (
        name_field.get() == ""
        or last_field.get() == ""
        or DOB_field.get() == ""
        or phone_field.get() == ""
        or email_field.get() == ""
        or address_field.get() == ""
        or current_date_field.get()
    ):
        Empty_Input()

    else:

        # assigning the max row and max column
        # value upto which data is written
        # in an excel sheet to the variable
        current_row = sheet.max_row
        current_column = sheet.max_column

        # Recive uer input from text entry box and submit to excel sheet
        sheet.cell(row=current_row + 1, column=1).value = name_field.get()
        sheet.cell(row=current_row + 1, column=2).value = middle_field.get()
        sheet.cell(row=current_row + 1, column=3).value = last_field.get()
        sheet.cell(row=current_row + 1, column=4).value = DOB_field.get()
        sheet.cell(row=current_row + 1, column=5).value = phone_field.get()
        sheet.cell(row=current_row + 1, column=6).value = email_field.get()
        sheet.cell(row=current_row + 1, column=7).value = address_field.get()

        # Recive date input and add a 5 inute timedelta to avoid datetime errors
        sheet.cell(row=current_row + 1, column=8).value = datetime.strptime(
            current_date_field.get(), "%Y-%m-%d"
        ) + timedelta(minutes=5)

        # Add 21 days yo the original date input
        sheet.cell(row=current_row + 1, column=9).value = datetime.strptime(
            current_date_field.get(), "%Y-%m-%d"
        ) + timedelta(days=21)
        sheet.cell(row=current_row + 1, column=10).value = 1
        # add +1 to phone number input
        sheet.cell(row=current_row + 1, column=11).value = "+1" + phone_field.get()
        # sheet.cell(row=current_row + 1, column=9).value = future_date_field.get()

        # save the file
        wb.save(".\Simple_Registration_Database.xlsx")

        df_csv = pd.read_excel("Simple_Registration_Database.xlsx")
        df_csv.to_csv("COVID_Vaccine_Database.csv", index=None, header=True)

        # set focus on the name_field box
        name_field.focus_set()
        clear()
        submission_success(), first_vaccine(),
        second_vaccine(), current_date_check()
```

Detect empty insertion

Keep track of the last row and column

Insert a new row for the new user, fill out all columns

Just a trick for time keeping

Add 21 days from the day of the first vaccination

By now you should get the idea and can trace down the GUI actions to the code using the same method by searching for texts and match those in the code.

The key actions after registering a user and entering his/her personal information and the first date of vaccination is to send the email and/or text to remind this person to come 21 days later for the second shot.

Here are the functions.

```
440  # Write unser input into excel sheet2
441  # if either text entry box is empty the user will be notified
442  def insert():
443      # if user not fill any or one entry box
444      # then pop up empty input screen
445      if (
453      ):
455
456      else:
457
458          # assigning the max row and max column
459          # value upto which data is written
460          # in an excel sheet to the variable
461          current_row = sheet.max_row
462          current_column = sheet.max_column
463
464          # Recive uer input from text entry box and submit to excel sheet
465          sheet.cell(row=current_row + 1, column=1).value = name_field.get()
466          sheet.cell(row=current_row + 1, column=2).value = middle_field.get()
467          sheet.cell(row=current_row + 1, column=3).value = last_field.get()
468          sheet.cell(row=current_row + 1, column=4).value = DOB_field.get()
469          sheet.cell(row=current_row + 1, column=5).value = phone_field.get()
470          sheet.cell(row=current_row + 1, column=6).value = email_field.get()
471          sheet.cell(row=current_row + 1, column=7).value = address_field.get()
472
473          # Recive date input and add a 5 inute timedelta to avoid datetime errors
474          sheet.cell(row=current_row + 1, column=8).value = datetime.strptime(
476          ) + timedelta(minutes=5)
477
478          # Add 21 days yo the original date input
479          sheet.cell(row=current_row + 1, column=9).value = datetime.strptime(
481          ) + timedelta(days=21)
482          sheet.cell(row=current_row + 1, column=10).value = 1
483          # add +1 to phone number input
484          sheet.cell(row=current_row + 1, column=11).value = "+1" + phone_field.get()
485          # sheet.cell(row=current_row + 1, column=9).value = future_date_field.get()
486
487          # save the file
488          wb.save(".\Simple_Registration_Database.xlsx")
489
490          df_csv = pd.read_excel("Simple_Registration_Database.xlsx")
491          df_csv.to_csv("COVID_Vaccine_Database.csv", index=None, header=True)
492
493          # set focus on the name_field box
494          name_field.focus_set()
495          clear()
496          submission_success(), first_vaccine(),
497          second_vaccine(), current_date_check()
```

```python
# Design the date check popup tool screen
def current_date_check():
    global date_check_screen
    date_check_screen = Toplevel()
    global date_check_entry
    global date_check_in
    date_check_in = StringVar()
    # keep submittion popup ontop
    date_check_screen.title("Date Check Screen (Developer Testing Tool)")
    date_check_screen.geometry("800x180")

    # Display label with the first vaccination date and second vaccination date
    display_submission()
    Label(
        date_check_screen,
        text="Submit Date To Ckeck System!(format: YYYY-MM-DD)",
        font=("Calibri", 13),
    ).grid(row=0, column=0, padx=25)
    date_check_entry = Entry(date_check_screen, textvariable=date_check_in)
    date_check_entry.grid(row=0, column=1)

    Button(
        date_check_screen,
        text="Submit",
        width=15,
        height=1,
        bg="red",
        font=("Calibri", 13),
        command=date_verify,
    ).grid(row=3, column=1, padx=30)
```

This function is used for testing. Tester will enter "today's date" so s/he can validate that if the date is near 21 days then the patron will get a reminder text and/or email for the 2nd shot.

```python
# Verify that a second email will be sent to recipient that just registered
# 21 days from the first vaccine date
def first_vaccine():
    df = pd.read_csv("COVID_Vaccine_Database.csv")

    # initiate text messaging sever from twilio (Reference listed above importe modules)
    client = Client(
    )
    rows = df.shape[0]

    # format the dates to be sent to user with strptime and strftime
    first_vac = df.iloc[rows - 1, 7]
    first_vac = datetime.strptime(first_vac, "%Y-%m-%d %H:%M:%S")
    first_vac = first_vac.strftime("%B %d, %Y")
    sec_vac = df.iloc[rows - 1, 8]
    sec_vac = datetime.strptime(sec_vac, "%Y-%m-%d %H:%M:%S")
    sec_vac = sec_vac.strftime("%B %d, %Y")

    Email = df.iloc[rows - 1, 5]
    LastName = df.iloc[rows - 1, 2]
    FirstName = df.iloc[rows - 1, 0]
    text = df.iloc[rows - 1, 10]

    message = """Subject: First COVID-19 Vaccination Recived

    from_address = "ee104.jose.r@gmail.com"
    password = "dyllhrmmahxvzxwu"

    # initiate email server for messeges (Obtained from class files for ee104 sjsu)
    # section of the code was formated
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(from_address, password)

        server.sendmail(
        )
        # Send text message to from twilo account that is called my the client.messages function
        try:
            message2 = client.messages.create(
                body=message.format(
                ),
                from_="+13852157449",
                to=text,
            )
            message2.sid

        except Exception as e:
        server.quit()
```

replace with your own ee104 email and password here

replace with your own phone number to receive a text here

```python
def second_vaccine():
    df2 = pd.read_csv("COVID_Vaccine_Database.csv")

    # connect to twilio account
    client = Client(
        "ACalddba3fa9157b86845046af637d928b", "2bd4819314c7dc3ad9a3c3f405a8b70d"
    )

    message = """Subject: Reminder for Second COVID-19 Vaccination

    Hello {First} {Last} your second COVID-19 vaccination is on coming up on {Vaccination2}"""

    # time delta of 3 days and datetime.now() for current time
    # helps check the current date and second vaccination date
    check = timedelta(days=3)
    check = check.days
    now = datetime.now()

    # number of rwos in datframes df2 (COVID_Vaccine_Database)
    rows = df2.shape[0]
    # second vaccination date
    sec_vac = df2.iloc[rows - 1, 8]

    from_address = "ee104sjsu@gmail.com"
    password = "EE104F2021"

    # open messaging sever
    # modified loop to read and import a datframe object (df2)
    # https://stackoverflow.com/questions/59631659/i-dont-understand-why-i-get-a-too-many-values-to-unpack-error
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(from_address, password)
        for i, r in df2.iterrows():
            FD = datetime.strptime(r["FutureDate"], "%Y-%m-%d %H:%M:%S")
            FD2 = FD.strftime("%B %d, %Y")
            # check to see if the current date is 3days or less from second vaccination date
            if (
                abs(datetime.strptime(r["FutureDate"], "%Y-%m-%d %H:%M:%S") - now).days
                <= check
            ):

                # check to see if a seccond vaccination notifaction has been sent already
                # if message has not been sent send the sendond vaccination date and update the sent
                # value to 0.
                if r["Sent"] == 1:
                    server.sendmail(
                        from_address,
                        r["Email"],
                        message.format(
                        ),
                    )
                    # send second text message for second vaccination
                    try:
                        message2 = client.messages.create(
                            body=message.format(
                                First=r["FirstName"],
                                Last=r["LastName"],
                                Vaccination2=FD2,
                            ),
                            from_="+13852157449",
                            to=r["SendText"],
                        )
                        message2.sid
                    except Exception as e:
                    # update database, excel file and csv
                    df2.loc[i, "Sent"] = 0
                    df2.to_excel("Simple_Registration_Database.xlsx", index=False)
                    df2.to_csv("COVID_Vaccine_Database.csv", index=None, header=True)
```

replace with your ee104 gmail here

```python
# verify the date to be less than three days from second vaccination for the
# professor tool
def date_verify():
    date_input = datetime.strptime(date_check_entry.get(), "%Y-%m-%d") + timedelta(
    )
    submission_success2()
    df = pd.read_csv("COVID_Vaccine_Database.csv")
    # start text messaging server from twilio account
    client = Client(
    )
    # time delta of 3 days
    check = timedelta(days=3)
    check = check.days
    rows = df.shape[0]

    # second vaccination date from dataframe
    # obtaine values from datframe through their corresponding index Location
    sec_vac = df.iloc[rows - 1, 8]
    sec_vac2 = datetime.strptime(sec_vac, "%Y-%m-%d %H:%M:%S") + timedelta(minutes=5)
    sec_vac_send = sec_vac2.strftime("%B %d, %Y")

    Email = df.iloc[rows - 1, 5]
    LastName = df.iloc[rows - 1, 2]
    FirstName = df.iloc[rows - 1, 0]
    text = df.iloc[rows - 1, 10]

    message = """Subject: Second COVID-19 Vaccination Reminder

    from_address = "ee104.jose.r@gmail.com"
    password = "dyllhrmmahxvzxwu"

    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(from_address, password)
        if abs(date_input - sec_vac2).days <= check:
            server.sendmail(
                from_address,
                Email,
                message.format(
                ),
            )

            try:
                message2 = client.messages.create(
                    body=message.format(
                        First=FirstName, Last=LastName, Vaccination2=sec_vac_send
                    ),
                    from_="+13852157449",
                    to=text,
                )
                message2.sid

            except Exception as e:
            server.quit()
```

This function will send a reminder within 3 days of the 2nd shot.

We can go over the entire operation in the lab session. You should not miss the lab if you cannot read the code yourself yet.

Python Programming

Lab Submission

Once you learn the process and the code associate with each step in the process, you will be able to customize the program to do the followings.

Program or Requirement	Use Case	Earned Score / Max Score
Lab Report	**Turn in lab report (using the group report template) and the video recordings of all your work with your own voice narration for each requirement below and Submit Lab#_TeamName.Zip file on time to Canvas**	_____ / 10
Minimum 3 different background pictures	Design and use your own background for important GUI steps.	_____ / 15
Customized Gmail	To send and receive emails from Python	_____ / 15
Text messaging	Send text successfully to a designated phone number	_____ / 15
Customize the Excel database	Add a minimum 3 more fields to the User Database	_____ / 15
End-to-end execution	Successful end-to-end execution of your modified program and a successful video recording showing clearly the code being executed, the outputs from all receiving media such as the computer, the cell phone, etc., professional and clear voice narration. You can post it on a YouTube and share the link if you want to do so.	_____ / 30
	TOTAL	**100%**

That's all for this lab. Hopefully you found it useful and increase your interest in the Python world! See you in Lab 3.

Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. Turn in your archive to Canvas along with a narrated video capturing the screen of your computer and your phone running your program demonstration. You are also required to submit an archive of your project in the form of a ZIP file. Use 7-Zip option to create the ZIP file. Name the archive lab#_TeamName.zip. Refer to Lab 1 for detail instructions.

You will submit your zip file to the instructor through Canvas by the due date and time. Turn in your archive to Canvas along with a narrated YouTube video capturing the screen of your computer running your program demonstration. If your program is not completely functional by the due date, you should demonstrate and turn in what you have accomplished to receive partial credit. See the syllabus for the late penalty guideline