

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://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.

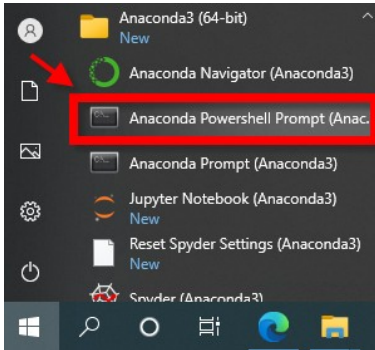
Download, Installation, and Licensing

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.

```

Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\chris.pham> pip install twilio
Collecting twilio
  Downloading twilio-6.62.1.tar.gz (486 kB)
    | 486 kB 3.3 MB/s
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from twilio) (1.15.0)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from twilio) (2021.1)
Collecting PyJWT==1.7.1
  Downloading PyJWT-1.7.1-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: requests>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from twilio) (2.25.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->twilio) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->twilio) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->twilio) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->twilio) (2020.12.5)
Building wheels for collected packages: twilio
  Building wheel for twilio (setup.py) ... done
  Created wheel for twilio: filename=twilio-6.62.1-py2.py3-none-any.whl size=1287004 sha256=5fd4dc4cf0d489c73ca3d631a8165f7a826a1f1d4d94ae2ee9af238a30255e20
  Stored in directory: c:\users\chris.pham\appdata\local\pip\cache\wheels\82\65\ae\1b1a1d38b53f22e67536ff35dda39900b570128418ece9cd00
Successfully built twilio
Installing collected packages: PyJWT, twilio
Successfully installed PyJWT-1.7.1 twilio-6.62.1
(base) PS C:\Users\chris.pham>
```

1. Procedure to create a free-trial Twilio Account

Overview

WITH TWILIO YOU CAN BUILD:

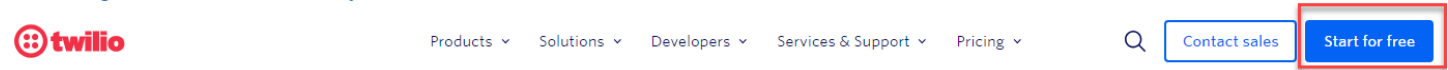
- SMS marketing
- Omnichannel contact center
- Call tracking
- Web chat
- Push notifications

- 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:



Christopher Pham,

To activate your Twilio Account, please verify your email address.
Your account will not be created until your email address is confirmed.

[Confirm Your Email](#)

Or, copy and paste the following URL into your browser:

<https://www.twilio.com/console/activate?key=%2BJHXVC1IY%2FAEu5VsOtXLzNKucww3uW150Vdrft1c6k9X3UllbXvpbi1ZivMj6Qe%2BaXZ6l30y9lCbmrG2m2vQ1aep77ja52jCWnVHY9yS2C9hql%2BAAt8AVkQbn9Mo2umLhyQlrJ9y1m6fhODLSMR7rsa8Sa9GZYHcLGMWD8SLASK0%3D>

Then you can log in

Log in

Email

christopher.h.pham01@sjsu.edu

Next

Don't have an account yet? [Sign up for free.](#)


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

Verify you're a human to start your free trial

Verify Email

Verify Phone Number

NUMBER



▼

+1

Phone Number

Why verify a phone number?

Verify

We will contact you at the number above with a verification code

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

Verify you're a human to start your free trial

Verify Email

Verify Phone Number

Please enter the verification code we sent to

393318

Submit

Want to verify with a Call instead of SMS?

Didn't receive a code?

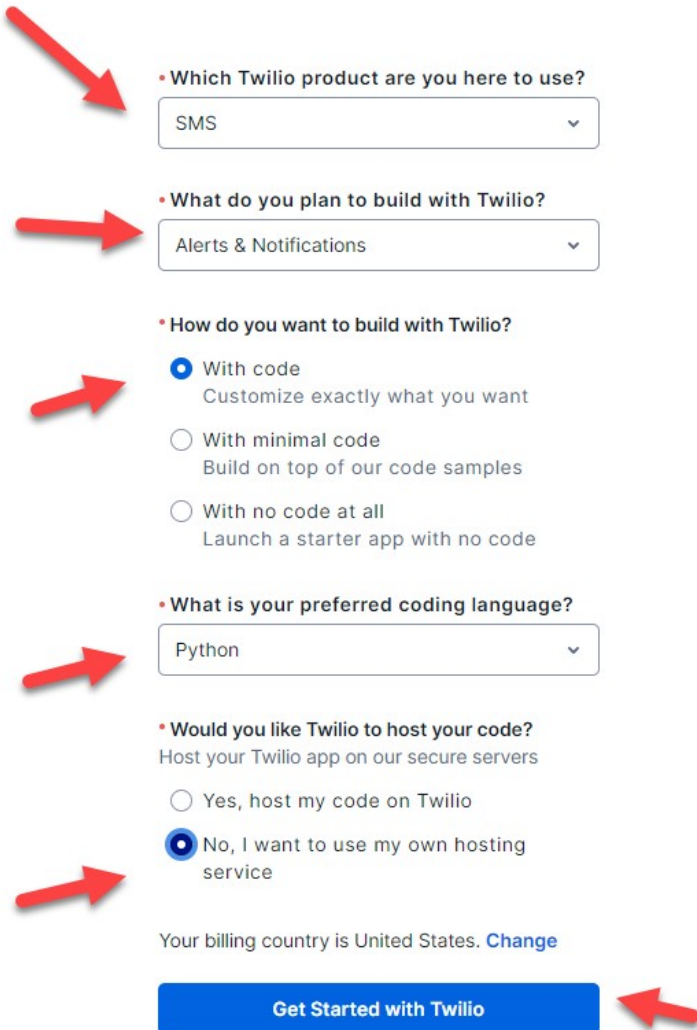
[Resend Code](#)

[Change phone number](#)

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

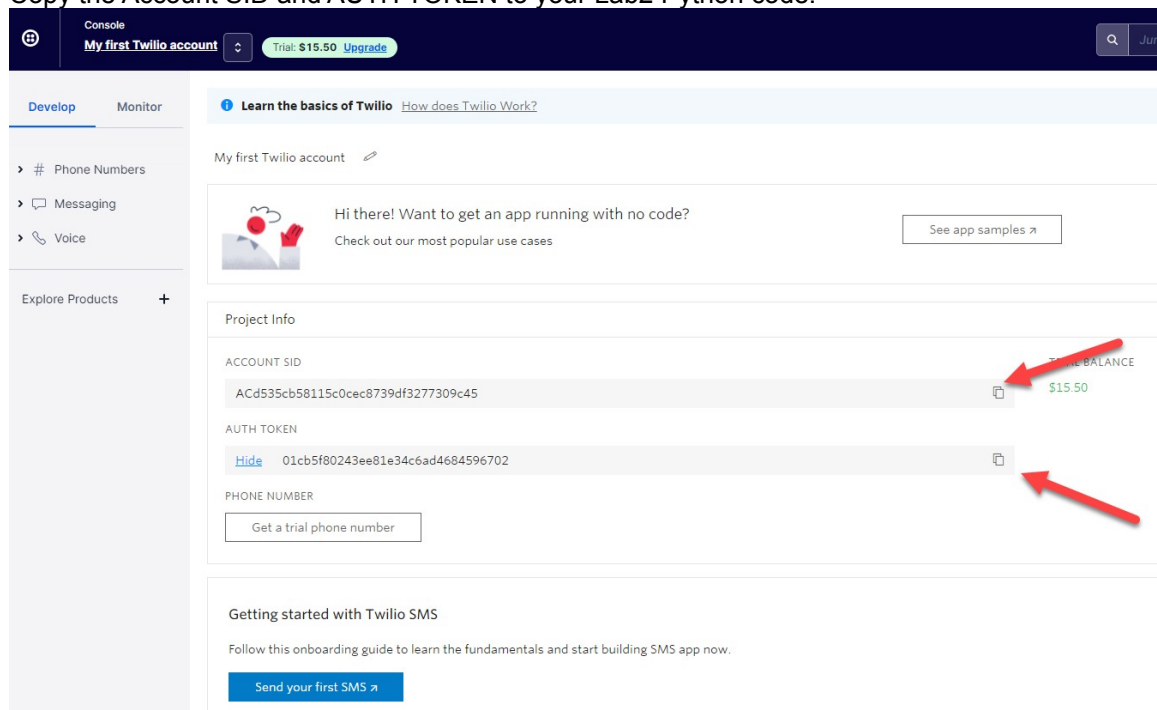
☐ Yes, host my code on Twilio

☒ No, I want to use my own hosting service

Your billing country is United States. [Change](#)

Get Started with Twilio

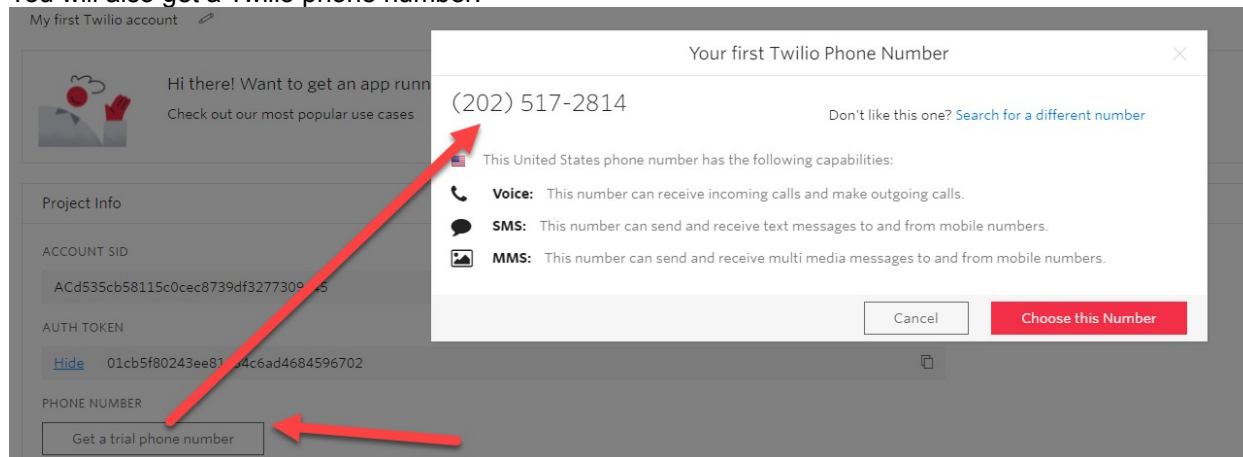
After you click on the blue button above, you will see this screen
Copy the Account SID and AUTH TOKEN to your Lab2 Python code:



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

```
54
55 Twilio_Number="+15625804763", #this is your own Twilio number
56 account_sid = 'ACd535cb58115c0cec8739df3277309c45' #this is your account SID
57 auth_token = '4432c9f31598f093a77af87b3e4e34b5' #this is your own auth_token
58
59 from_address = "ee104sjsu@gmail.com" #this is your own gmail account
60 app_password = 'watrmvxhvuycxuus' # a token for gmail, this is the app password from Gmail Security
61
```

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



TRIAL BALANCE

\$14.50

AUTH TOKEN

[Hide](#) 01cb5f80243ee81e34c6ad4684596702



PHONE NUMBER

+12025172814



[Upgrade to buy more numbers](#)

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:

Try SMS

Send an SMS with a Messaging Service

To phone number: [Redacted] **Your phone #**

From Messaging Service SID: [Redacted]

Body Text: [Redacted]

Send test SMS View Docs

This request may cost money. [Learn more about pricing information](#)

curl Java Ruby PHP **Python** C# Node.js

[Download the Python Helper Library](#)

☐ Show Auth Token **Copy**

```
from twilio.rest import Client

account_sid = 'ACd535cb58115c0cec8739d[Redacted]'
auth_token = '[AuthToken]'
client = Client(account_sid, auth_token)

message = client.messages.create(
    to='+1408[Redacted]'
)

print(message.sid)
```

Response

Response will appear here after you make request

Press Copy to copy the code to your editor & modify accordingly

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.

Lab2 - Twilio > twilio

Name

TestTwilio

```
1 from twilio.rest import Client
2
3 account_sid = 'ACd535cb58115c0cec8739d[Redacted]' #this is your account SID
4 auth_token = '4432c9f31598f093a77af[Redacted]' #this is your own auth_token
5 client = Client(account_sid, auth_token)
6
7 message = """Subject: Second COVID-19 Vaccination Reminder
8
9 Hello {First} {Last} your second COVID 19 vaccination is coming up on 09/17/2022."""
10
11 FirstName = 'Christopher'
12 LastName = 'Pham'
13
14
15
16 message = client.messages.create(
17     body=message.format(
18         First=FirstName, Last=LastName
19     ),
20     from_="+1[Redacted]", #this is your own Twilio number
21     to="+1408[Redacted]" #this is your own phone number to receive the text message
22 )
23
24 print(message.sid)
```

Console 6/A

```
In [5]: runfile('C:/Users/chris.pham/Desktop/EE104F22/Labs/Lab2 - Twilio/twilio/TestTwilio.py', wdir='C:/Users/chris.pham/Desktop/EE104F22/Labs/Lab2 - Twilio/twilio')
SM6485fe59d1d047c218d4384cbc083976
In [6]:
```

success

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

```
54 Twilio_Number="+15625804763", #this is your own Twilio number
55 account_sid = 'ACd535cb58115c0cec8739df3277309c45' #this is your account SID
56 auth_token = '4432c9f31598f093a77af87b3e4e34b5' #this is your own auth_token
57
58
59 from_address = "ee104sjsu@gmail.com" #this is your own gmail account
60 app_password = 'watrmvxhvuycxuus' # a token for gmail, this is the app password from Gmail Security
61
```

That's it! Enjoy!

2. Set up a Gmail account for test emails

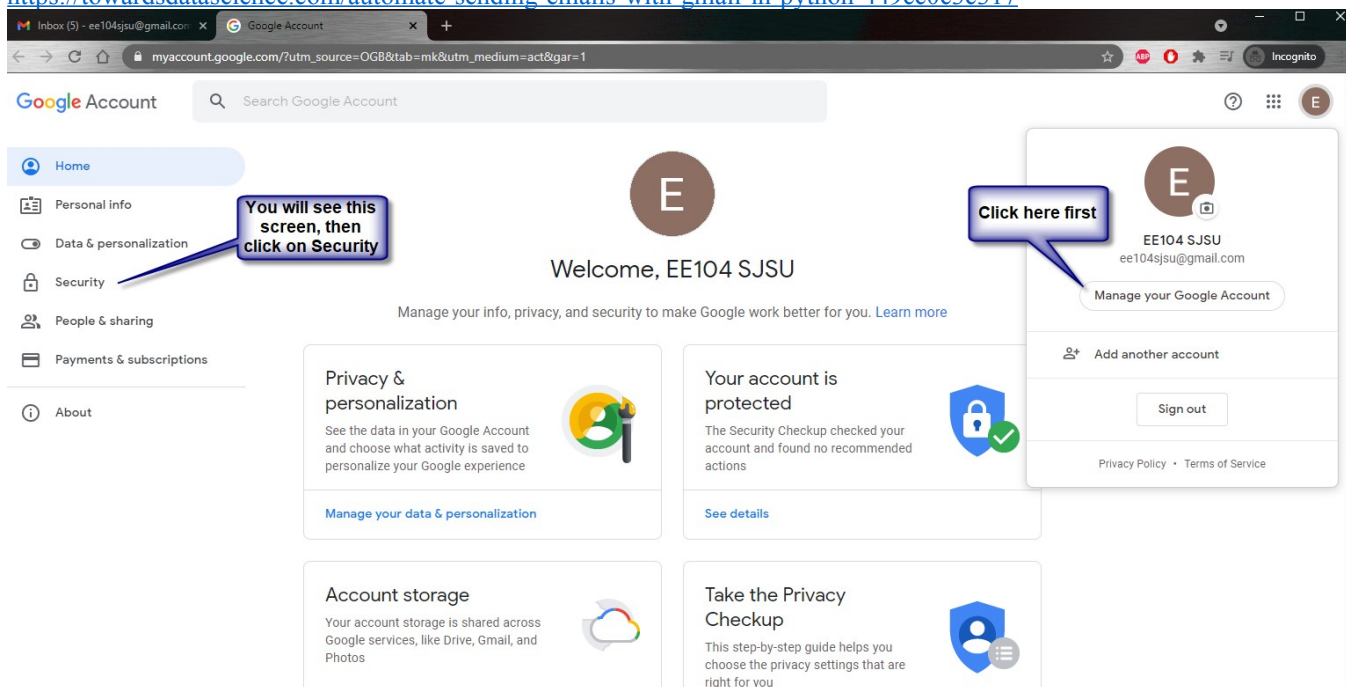
Follow the instruction from this URL to set up a Gmail account for development.

<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>



- 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.



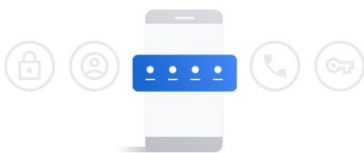
Security made easy

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.

**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](#)

← 2-Step Verification

Confirm that it works

Google just sent a text message with a verification code to [redacted]

Enter the code

685647

Didn't get it? [Resend](#)

[BACK](#) Step 2 of 3 [NEXT](#)

Then click Next on the confirmation screen.

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

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 >

2-Step Verification ☒ On >

App passwords None >

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

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Select app Select device

Mail

Calendar

Contacts

YouTube

Other (Custom name)

[GENERATE](#)

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

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Python

×

GENERATE

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.

Generated app password

Email

securesally@gmail.com

Password

••••••••••

Your app password for your device

watr mvxh vujc xuus


How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

DONE

You will see 1 password in your screen below:

Signing in to Google



Password	Last changed 5:50 PM	>
2-Step Verification	<input checked="" type="checkbox"/> On	>
App passwords	1 password	>

Now pip install yagmail




```
Anaconda Prompt (anaconda3)

(base) C:\Users\chris.pham>pip install yagmail
Collecting yagmail
  Downloading yagmail-0.15.285-py2.py3-none-any.whl (17 kB)
Collecting premailer
  Downloading premailer-3.10.0-py2.py3-none-any.whl (19 kB)
Requirement already satisfied: requests in c:\users\chris.pham\anaconda3\lib\site-packages (from premailer->yagmail) (2.27.1)
Requirement already satisfied: cssselect in c:\users\chris.pham\anaconda3\lib\site-packages (from premailer->yagmail) (1.1.0)
Requirement already satisfied: lxml in c:\users\chris.pham\anaconda3\lib\site-packages (from premailer->yagmail) (4.8.0)
Collecting cssutils
  Downloading cssutils-2.6.0-py3-none-any.whl (399 kB)
Requirement already satisfied: cachetools in c:\users\chris.pham\anaconda3\lib\site-packages (from premailer->yagmail) (4.2.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\chris.pham\anaconda3\lib\site-packages (from requests->premailer->yagmail) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\chris.pham\anaconda3\lib\site-packages (from requests->premailer->yagmail) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\chris.pham\anaconda3\lib\site-packages (from requests->premailer->yagmail) (3.3)
Requirement already satisfied: charset-normalizer~2.0.0 in c:\users\chris.pham\anaconda3\lib\site-packages (from requests->premailer->yagmail) (2.0.4)
Installing collected packages: cssutils, premailer, yagmail
Successfully installed cssutils-2.6.0 premailer-3.10.0 yagmail-0.15.285

(base) C:\Users\chris.pham>
```

Download the content from the folder yagmail from Canvas

io > yagmail

Name	Date modified	Type
 cat	8/27/2022 6:42 PM	JPG File
 test	8/27/2022 6:39 PM	PNG File
 TestYagagmail	8/27/2022 6:46 PM	PY File

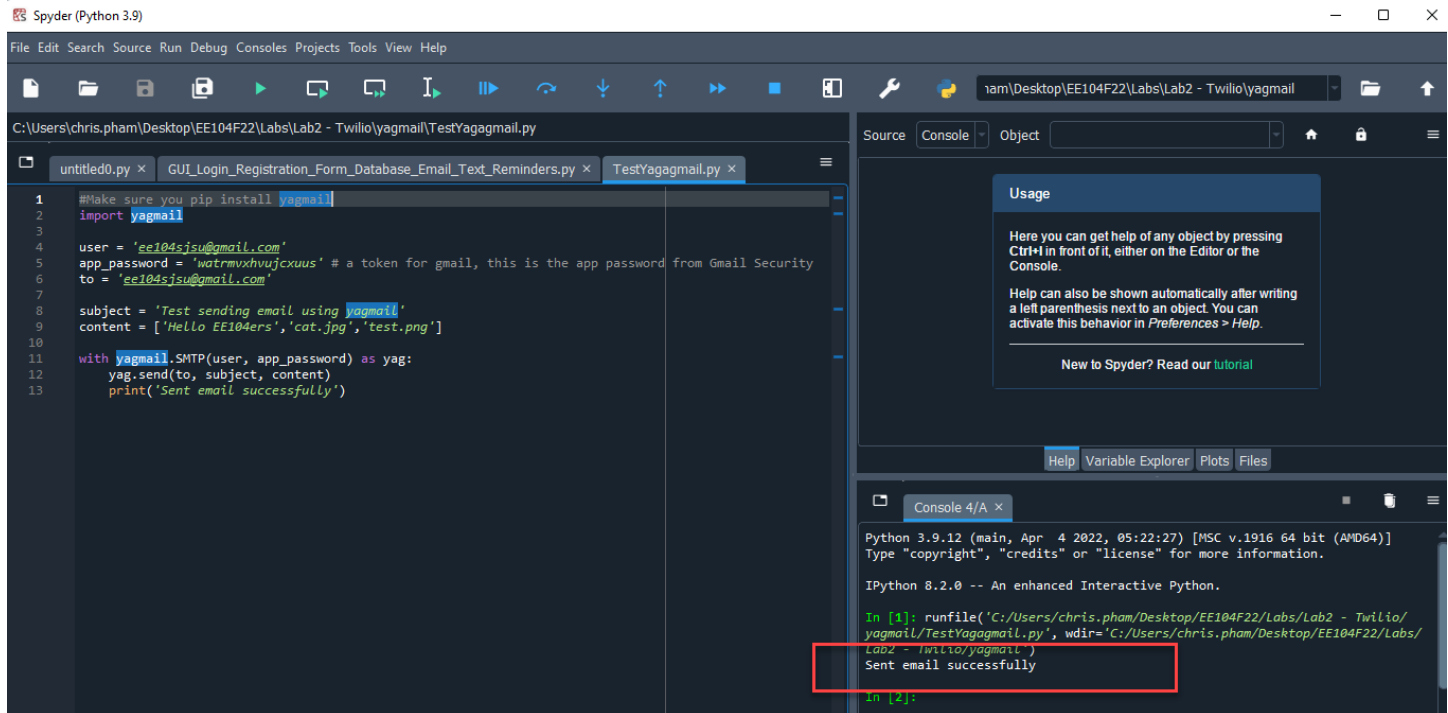
Make change accordingly using your own email and app password:

```

1 #Make sure you pip install yagmail
2 import yagmail
3
4 user = 'ee104sjsu@gmail.com' #this is your own gmail account
5 app_password = 'watrmvxhvuujcxuus' # a token for gmail, this is the app password from Gmail Security
6 to = 'ee104sjsu@gmail.com' #send test to another email or the same email is OK
7
8 subject = 'Test sending email using yagmail' #modify the subject line anyway you like
9 content = ['Hello EE104ers', 'cat.jpg', 'test.png'] #you can have different email and attachment
10
11 with yagmail.SMTP(user, app_password) as yag:
12     yag.send(to, subject, content)
13     print('Sent email successfully') #you can have different success message

```

And execute the file in Spyder or from your command line



The screenshot shows the Spyder Python IDE. The editor window displays the script for sending an email using yagmail. The console window on the right shows the output of running the script: 'Sent email successfully'.

From the command line:

```

(base) C:\Users\chris.pham>cd C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail

(base) C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail>
(base) C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail>
(base) C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail>dir
Volume in drive C is OSDisk
Volume Serial Number is 18E1-A514

Directory of C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail

08/27/2022  06:46 PM    <DIR>          .
08/27/2022  06:48 PM    <DIR>          ..
08/27/2022  06:42 PM             2,224,388  cat.jpg
08/27/2022  06:39 PM             34,266  test.png
08/27/2022  06:46 PM              433  TestYagagmail.py
               3 File(s)      2,259,087 bytes
               2 Dir(s)      628,313,985,024 bytes free

(base) C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail>python TestYagagmail.py
Sent email successfully
(base) C:\Users\chris.pham\Desktop\EE104F22\Labs\Lab2 - Twilio\yagmail>

```

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

Test sending email using yagmail Inbox x



ee104sjsu@gmail.com <ee104sjsu@gmail.com>

to me ▼

Hello EE104ers

2 Attachments



ee104sjsu@gmail.com <ee104sjsu@gmail.com>

to me ▼

Hello EE104ers

2 Attachments



↩ Reply

➦ Forward

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 1 st vaccination date, calculate the 2 nd date for the 2 nd dose, and send text and/or email to remind the patron

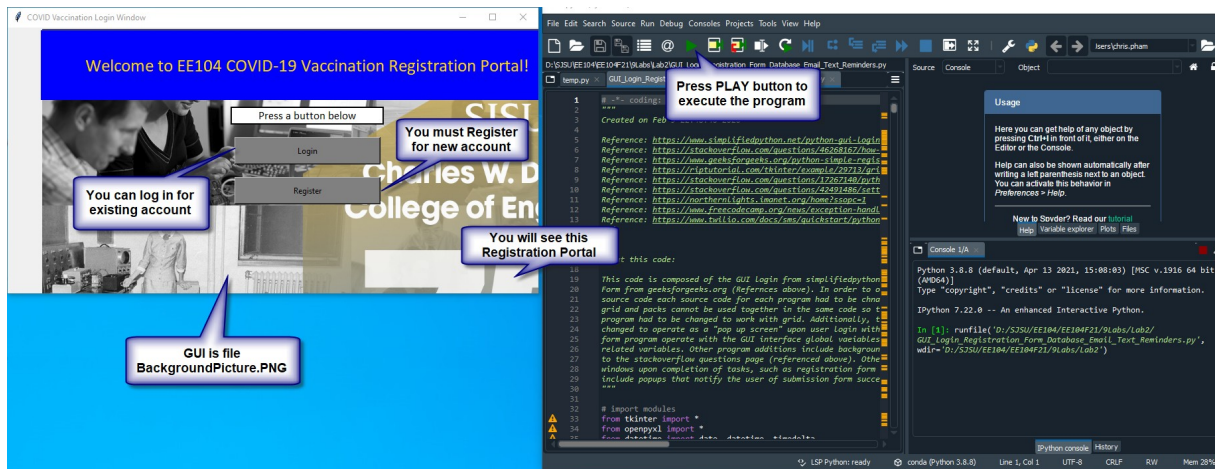
GUI File

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

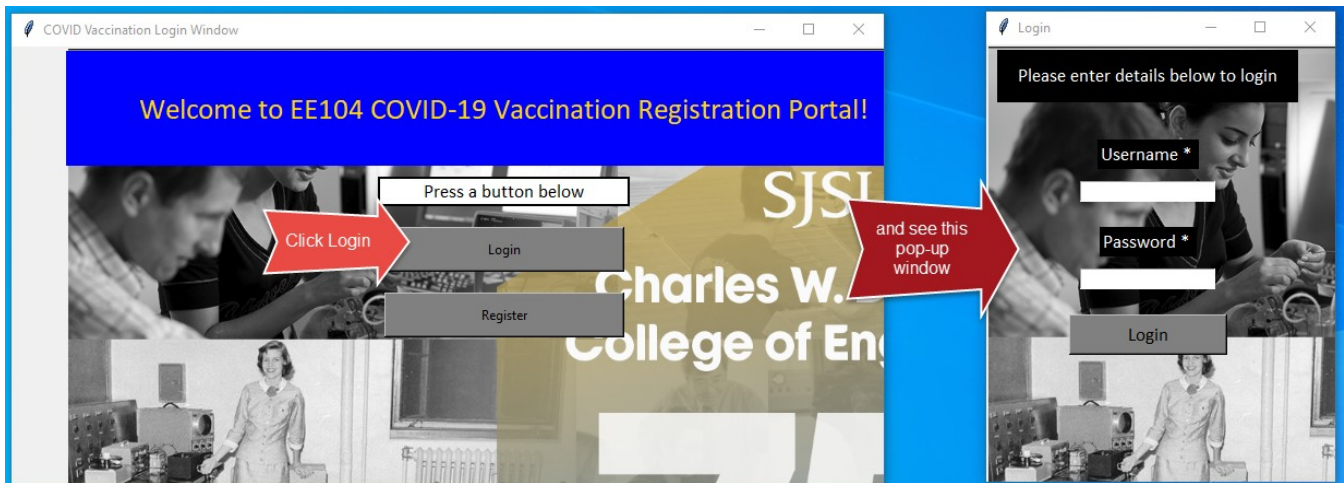
The file **GUI_Login_Registration_Form_Database_Email_Text_Reminders.py** is provided to you. Here are some highlights.

Sequence of execution and related Python code:



Code to produce the GUI above:

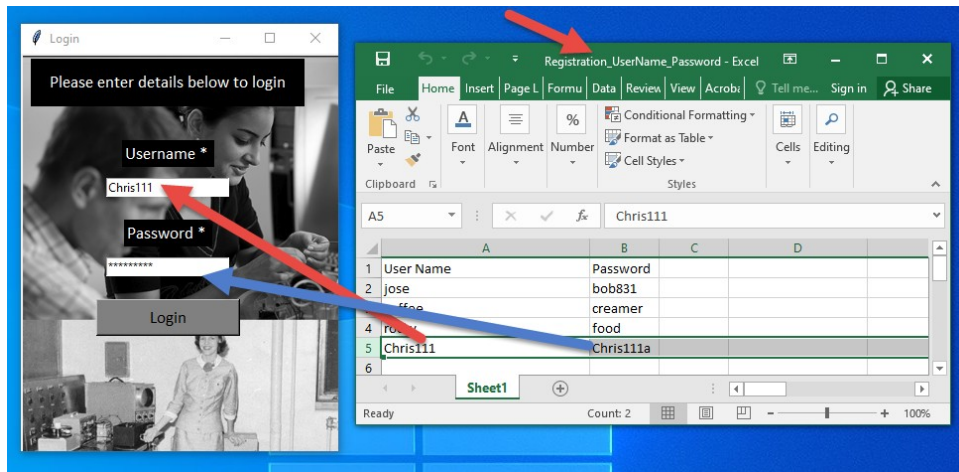
```
996 # Designing Main(first) window
997 def main_account_screen():
998     global main_screen
999     global df
1000
1001     # Generate GUI
1002     main_screen = Tk()
1003
1004     # Resize main screen
1005     main_screen.geometry("800x400")
1006     main_screen.title("COVID Vaccination Login Window")
1007
1008     # Import and set ground of main screen
1009     bg = PhotoImage(file="BackgroundPicture.png", height=400, width=800)
1010     bg_label = Label(main_screen, image=bg)
1011     bg_label.image = bg # keep a reference!
1012     bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)
1013
1014     Label(
1015         text="Welcome to EE104 COVID-19 Vaccination Registration Portal!",
1016         fg="gold",
1017         bg="#0000FF", #all RGB = 0000FF
1018         width="57",
1019         height="3",
1020         font=("Calibri", 20),
1021     ).grid(row=0, column=2, padx=50)
1022
1023     Label(
1024         main_screen,
1025         relief="solid",
1026         text="Press a button below",
1027         width="25",
1028         height="1",
1029         fg="black",
1030         bg="white",
1031         font=("Calibri", 13),
1032     ).grid(row=1, column=2, padx=50)
1033
1034     Button(text="Login", height="2", width="30", bg="grey", command=login).grid(
1035         row=3, column=2
1036     )
1037
1038     Button(text="Register", height="2", width="30", bg="grey", command=register).grid(
1039         row=5, column=2
1040     )
1041
1042     main_screen.mainloop()
1043
1044
1045 main_account_screen()
```

```

737 # Designing window for login
738 def login():
739     global login_screen
740     login_screen = Toplevel(main_screen)
741     login_screen.title("Login")
742     login_screen.geometry("320x400")
743
744     bg = PhotoImage(file="BackgroundPicture.png", height=400, width=320)
745     bg_label = Label(login_screen, image=bg)
746     bg_label.image = bg # keep a reference!
747     bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)
748
749     Label(
750         login_screen,
751         text="Please enter details below to login",
752         fg="white",
753         bg="black",
754         width="30",
755         height="2",
756         font=("Calibri", 13),
757     ).grid(row=0, column=0, padx="10")
758
759     global username_verify
760     global password_verify
761
762     username_verify = StringVar()
763     password_verify = StringVar()
764
765     global username_login_entry
766     global password_login_entry
767
768     Label(
769         login_screen, text="Username * ", fg="white", bg="black", font=("Calibri", 13)
770     ).grid(row=3, column=0)
771     username_login_entry = Entry(login_screen, textvariable=username_verify)
772     username_login_entry.grid(row=4, column=0)
773
774     Label(
775         login_screen, text="Password * ", fg="white", bg="black", font=("Calibri", 13)
776     ).grid(row=6, column=0)
777     password_login_entry = Entry(login_screen, textvariable=password_verify, show="*")
778     password_login_entry.grid(row=7, column=0)
779
780     Button(
781         login_screen,
782         text="Login",
783         width=15,
784         height=1,
785         bg="grey",
786         font=("Calibri", 13),
787         command=login_verify,
788     ).grid(row=9, column=0)

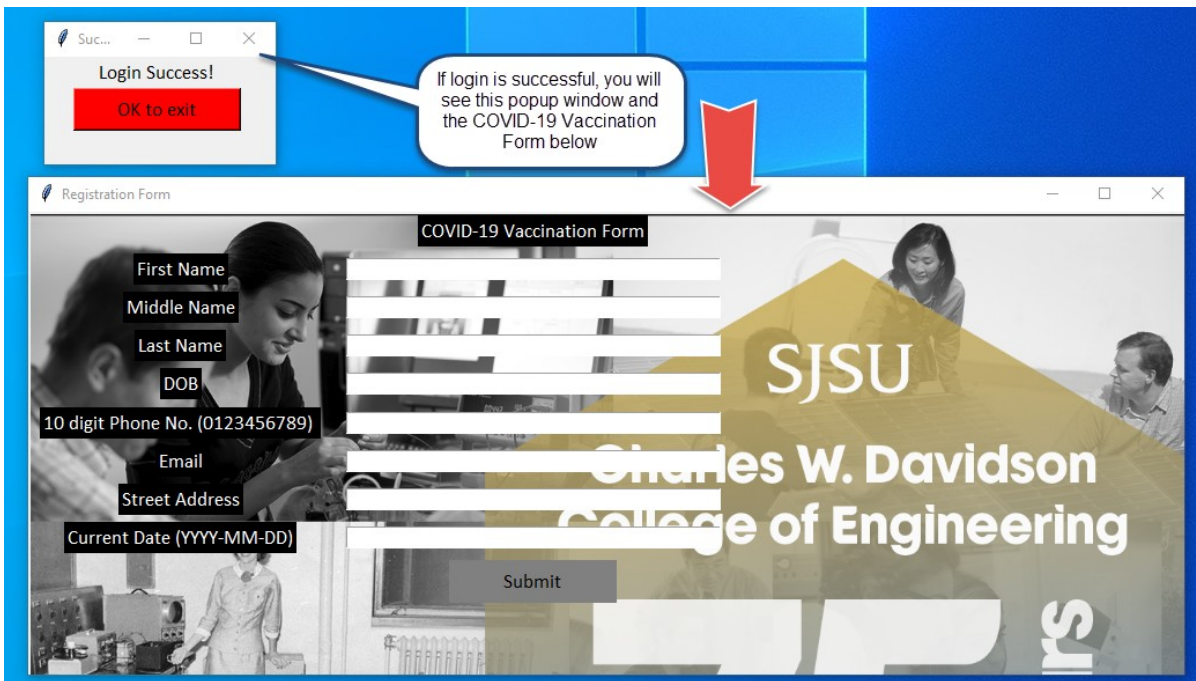
```



```

738 def login():
739     global login_screen
740     login_screen = Toplevel(main_screen)
741     login_screen.title("Login")
742     login_screen.geometry("320x400")
743
744     bg = PhotoImage(file="BackgroundPicture.png", height=400, width=320)
745     bg_label = Label(login_screen, image=bg)
746     bg_label.image = bg # keep a reference!
747     bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)
748
749     Label(
750         login_screen,
751         text="Please enter details below to login",
752         fg="white",
753         bg="black",
754         width="30",
755         height="2",
756         font=("Calibri", 13),
757     ).grid(row=0, column=0, padx=10)
758
759     global username_verify
760     global password_verify
761
762     username_verify = StringVar()
763     password_verify = StringVar()
764
765     global username_login_entry
766     global password_login_entry
767
768     Label(
769         login_screen, text="Username * ", fg="white", bg="black", font=("Calibri", 13)
770     ).grid(row=3, column=0)
771     username_login_entry = Entry(login_screen, textvariable=username_verify)
772     username_login_entry.grid(row=4, column=0)
773
774     Label(
775         login_screen, text="Password * ", fg="white", bg="black", font=("Calibri", 13)
776     ).grid(row=6, column=0)
777     password_login_entry = Entry(login_screen, textvariable=password_verify, show="*")
778     password_login_entry.grid(row=7, column=0)
779
780     Button(
781         login_screen,
782         text="Login",
783         width=15,
784         height=1,
785         bg="grey",
786         font=("Calibri", 13),
787         command=login_verify,
788     ).grid(row=9, column=0)
789
790
791     # Write user name and password to spreadsheet
792     # Use to modify Excel file
793     Registration_UserName_Password.xlsx
794     def excel2():
795
796         # resize the width of columns in
797         # excel spreadsheet
798         sheet2.column_dimensions["A"].width = 30
799         sheet2.column_dimensions["B"].width = 10
800
801         # write given data to an excel spreadsheet
802         # at particular location
803         # Columns for user name and password
804         sheet2.cell(row=1, column=1).value = "User Name"
805         sheet2.cell(row=1, column=2).value = "Password"

```



```

853 # Implementing event on login button
854 def login_verify():
855     username = username_verify.get()
856     password1 = password_verify.get()
857     username_login_entry.delete(0, END)
858     password_login_entry.delete(0, END)
859
860     # create name list to search from for login dat
861     # create passwords list to search from
862     # https://stackoverflow.com/questions/51800122/using-openpyxl-to-find-rows-that-contain-cell-with-sr
863     names = []
864     passwords = []
865
866     for i in range(2, sheet2.max_row + 1):
867         if sheet2[i][0].value:
868             names.append(sheet2[i][0].value)
869             passwords.append(sheet2[i][1].value)
870
871     # exempt ValueError when user inputs a username not in file
872     # https://www.freecodecamp.org/news/exception-handling-pyhton/
873     try:
874         index = names.index(username)
875     except ValueError:
876         print("")
877
878     # if password and names are saved allow user to register_form
879     # otherwise no user found or incorrect password
880     # Check to see if the index number of the username matches with its corresponding user password
881     if username in names:
882         if password1 == passwords[index]:
883             login_success()
884             register_form()
885         else:
886             password_not_recognised()
887     else:
888         user_not_found()
889
890 # Designing popup for login succes
891 def login_success():
892     global login_success_screen
893     login_success_screen = Toplevel(login_screen)
894     login_success_screen.title("Success")
895     login_success_screen.geometry("200x100")
896
897 # display Login success to user
898 Label(login_success_screen, text="Login Success!", font=("Calibri", 13)).grid(
899     )
900
901 # close login success window when pressing button
902 Button(
903     ).grid(row=1, column=0, padx=25)

```

Annotations in the code:

- A red arrow points to the `login_verify()` function definition.
- A speech bubble labeled '"get" methods' points to the `username_verify.get()` and `password_verify.get()` lines.
- A red arrow points to the `login_success()` and `register_form()` calls within the `login_verify()` function.
- A green arrow points from the `login_success()` call to the `login_success()` function definition.
- Red arrows point to the `Label(login_success_screen, text="Login Success!", font=("Calibri", 13)).grid()` line and the `Button()` line.

```

501 def register_form():
502     global root
503     # keep screen on top
504     root = Toplevel()
505     root.title("Registration Form")
506     root.geometry("1000x400")
507
508     # set background image to registration form main screen
509     # Reference: https://stackoverflow.com/questions/42451486/setting-an-image-as-a-tkinter-window-background
510     # import image and set as background
511     bg = PhotoImage(file="BackgroundPicture.png", height=400, width=1000)
512     bg_label = Label(root, image=bg)
513     bg_label.image = bg # keep a reference!
514     bg_label.grid(row=0, column=0, columnspan=20, rowspan=20)
515
516     # create global variables for registration form inputs to successfully
517     # implement with GUI program across the rest of the program
518     global Name_in
519     global course_in
520     global sem_in
521     global grad_in
522     global phone_in
523     global email_in
524     global address_in
525     global current_date_in
526
527     global name_field
528     global middle_field
529     global last_field
530     global DOB_field
531     global phone_field
532     global email_field
533     global address_field
534     global current_date_field
535
536     # Widget text variables
537     # https://stackoverflow.com/questions/51783852/what-is-the-difference-between-a-variable-and-stringvar-of-tkinter/51785046
538     Name_in = StringVar()
539     course_in = StringVar()
540     sem_in = StringVar()
541     grad_in = StringVar()
542     phone_in = StringVar()
543     email_in = StringVar()
544     address_in = StringVar()
545     current_date_in = StringVar()
546
547     excel()
548
549     # create a Form label
550     heading = Label(
551         root,
552         text="COVID-19 Vaccination Form",
553         fg="white",
554         bg="black",
555         font=("Calibri", 13),
556     )
557
558     # create a Name label
559     name = Label(root, text="First Name", bg="black", fg="white", font=("Calibri", 13))
560
561     # create a Course label
562     course = Label(
563         root, text="Middle Name", bg="black", fg="white", font=("Calibri", 13)
564     )
565
566     # create a Semester label
567     sem = Label(root, text="Last Name", bg="black", fg="white", font=("Calibri", 13))
568
569     # create a graduation year. label
570     grad = Label(root, text="DOB", bg="black", fg="white", font=("Calibri", 13))
571
572     # create a phone No. label
573     phone = Label(
574         root,
575         text="10 digit Phone No. (0123456789) ",
576         bg="black",
577         fg="white",
578         font=("Calibri", 13),
579     )
580
581     # create a Email label
582     email = Label(root, text="Email", bg="black", fg="white", font=("Calibri", 13))
583
584     # create a address label
585     address = Label(
586         root, text="Street Address", bg="black", fg="white", font=("Calibri", 13)
587     )
588
589     # create a current date label
590     c_date = Label(
591         root,
592         text="Current Date (YYYY-MM-DD)",
593         bg="black",
594         fg="white",
595         font=("Calibri", 13),
596     )
597
598     # headings position on the screen with the use of grid.
599     heading.grid(row=0, column=1)
600     name.grid(row=1, column=0)
601     course.grid(row=2, column=0)
602     sem.grid(row=3, column=0)
603     grad.grid(row=4, column=0)
604     phone.grid(row=5, column=0)
605     email.grid(row=6, column=0)
606     address.grid(row=7, column=0)
607     c_date.grid(row=8, column=0)
608
609     name_field = Entry(root, textvariable=Name_in)
610     middle_field = Entry(root, textvariable=course_in)
611     last_field = Entry(root, textvariable=sem_in)
612     DOB_field = Entry(root, textvariable=grad_in)
613     phone_field = Entry(root, textvariable=phone_in)
614     email_field = Entry(root, textvariable=email_in)
615     address_field = Entry(root, textvariable=address_in)
616     current_date_field = Entry(root, textvariable=current_date_in)
617
618     name_field.bind("<Return>", focus1)
619
620     # whenever the enter key is pressed
621     # then call the focus2 function
622     middle_field.bind("<Return>", focus2)
623
624     # whenever the enter key is pressed
625     # then call the focus3 function
626     last_field.bind("<Return>", focus3)
627
628     # whenever the enter key is pressed
629     # then call the focus4 function
630     DOB_field.bind("<Return>", focus4)

```



```

631 # whenever the enter key is pressed
632 # then call the focus5 function
633 phone_field.bind("<Return>", focus5)
634
635 # whenever the enter key is pressed
636 # then call the focus6 function
637 email_field.bind("<Return>", focus6)
638
639 address_field.bind("<Return>", focus7)
640
641 current_date_field.bind("<Return>", focus7)
642
643 # position text entry boxes with grid function in their respected rows/columns
644 # column 1 is used for text boxes for user input
645 name_field.grid(row=1, column=1, ipadx="100")
646 middle_field.grid(row=2, column=1, ipadx="100")
647 last_field.grid(row=3, column=1, ipadx="100")
648 DOB_field.grid(row=4, column=1, ipadx="100")
649 phone_field.grid(row=5, column=1, ipadx="100")
650 email_field.grid(row=6, column=1, ipadx="100")
651 address_field.grid(row=7, column=1, ipadx="100")
652 current_date_field.grid(row=8, column=1, ipadx="100")
653
654 # submission button with red background and font change i used to submit user information
655 # into excel file
656 submit = Button(
657     root,
658     text="Submit",
659     fg="Black",
660     relief="flat",
661     bg="grey",
662     width=15,
663     height=1,
664     font=("Calibri", 13),
665     command=insert
666 )
667 submit.grid(row=9, column=1)
668

```


Simple_Registration_Database - Excel

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do...

Clipboard Font Alignment

Form field values are captured to the Excel columns

FirstName	MiddleName	LastName	DOB	PhoneNumber	Email	StreetAddress	CurrentDate	FutureDate	Sent	SendText
Chris	H	Pham	01011980	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

```

440 # Write user 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 (
446         name_field.get() == ""
447         or last_field.get() == ""
448         or DOB_field.get() == ""
449         or phone_field.get() == ""
450         or email_field.get() == ""
451         or address_field.get() == ""
452         or current_date_field.get() == ""
453     ):
454         Empty_Input()
455     else:
456         # assigning the max row and max column
457         # value upto which data is written
458         # in an excel sheet to the variable
459         current_row = sheet.max_row
460         current_column = sheet.max_column
461
462         # Recive user input from text entry box and submit to excel sheet
463         sheet.cell(row=current_row + 1, column=1).value = name_field.get()
464         sheet.cell(row=current_row + 1, column=2).value = middle_field.get()
465         sheet.cell(row=current_row + 1, column=3).value = last_field.get()
466         sheet.cell(row=current_row + 1, column=4).value = DOB_field.get()
467         sheet.cell(row=current_row + 1, column=5).value = phone_field.get()
468         sheet.cell(row=current_row + 1, column=6).value = email_field.get()
469         sheet.cell(row=current_row + 1, column=7).value = address_field.get()
470
471         # Recive date input and add a 5 inute timedelta to avoid datetime errors
472         sheet.cell(row=current_row + 1, column=8).value = datetime.strptime(
473             current_date_field.get(), "%Y-%m-%d"
474         ) + timedelta(minutes=5)
475
476         # Add 21 days yo the original date input
477         sheet.cell(row=current_row + 1, column=9).value = datetime.strptime(
478             current_date_field.get(), "%Y-%m-%d"
479         ) + timedelta(days=21)
480         sheet.cell(row=current_row + 1, column=10).value = 1
481         # add +1 to phone number input
482         sheet.cell(row=current_row + 1, column=11).value = "+1" + phone_field.get()
483         # sheet.cell(row=current_row + 1, column=9).value = future_date_field.get()
484
485         # save the file
486         wb.save(".\Simple_Registration_Database.xlsx")
487
488         df_csv = pd.read_excel("Simple_Registration_Database.xlsx")
489         df_csv.to_csv("COVID_Vaccine_Database.csv", index=None, header=True)
490
491         # set focus on the name_field box
492         name_field.focus_set()
493         clear()
494         submission_success(), first_vaccine(),
495         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()
```

```

204 # Design the date check popup tool screen
205 def current_date_check():
206     global date_check_screen
207     date_check_screen = Toplevel()
208     global date_check_entry
209     global date_check_in
210     date_check_in = StringVar()
211     # keep submission popup ontop
212     date_check_screen.title("Date Check Screen (Developer Testing Tool)")
213     date_check_screen.geometry("800x180")
214
215     # Display label with the first vaccination date and second vaccination date
216     display_submission()
217     Label(
218         date_check_screen,
219         text="Submit Date To Ccheck System!(format: YYYY-MM-DD)",
220         font=("Calibri", 13),
221     ).grid(row=0, column=0, padx=25)
222     date_check_entry = Entry(date_check_screen, textvariable=date_check_in)
223     date_check_entry.grid(row=0, column=1)
224
225     Button(
226         date_check_screen,
227         text="Submit",
228         width=15,
229         height=1,
230         bg="red",
231         font=("Calibri", 13),
232         command=date_verify,
233     ).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.

```

236 # Verify that a second email will be sent to recipient that just registered
237 # 21 days from the first vaccine date
238 def first_vaccine():
239     df = pd.read_csv("COVID_Vaccine_Database.csv")
240
241     # initiate text messaging sever from twilio (Reference listed above importe modules)
242     client = Client(
243     )
244     rows = df.shape[0]
245
246     # format the dates to be sent to user with strptime and strftime
247     first_vac = df.iloc[rows - 1, 7]
248     first_vac = datetime.strptime(first_vac, "%Y-%m-%d %H:%M:%S")
249     first_vac = first_vac.strftime("%B %d, %Y")
250     sec_vac = df.iloc[rows - 1, 8]
251     sec_vac = datetime.strptime(sec_vac, "%Y-%m-%d %H:%M:%S")
252     sec_vac = sec_vac.strftime("%B %d, %Y")
253
254     Email = df.iloc[rows - 1, 5]
255     LastName = df.iloc[rows - 1, 2]
256     FirstName = df.iloc[rows - 1, 0]
257     text = df.iloc[rows - 1, 10]
258
259     message = """Subject: First COVID-19 Vaccination Received
260
261     from_address = "eel04.jose.r@gmail.com"
262     password = "dyl1hrmmahxvzxwu"
263
264     # initiate email server for messages (Obtained from class files for eel04 sjsu)
265     # section of the code was formatted
266     context = ssl.create_default_context()
267     with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
268         server.login(from_address, password)
269
270         server.sendmail(
271         )
272         # Send text message to from twilio account that is called my the client.messages function
273         try:
274             message2 = client.messages.create(
275                 body=message.format(
276                 ),
277                 from_="+13852157449",
278                 to=text,
279             )
280             message2.sid
281
282         except Exception as e:
283             server.quit()

```

replace with your own ee104 email and password here

replace with your own phone number to receive a text here

```

306 def second_vaccine():
307     df2 = pd.read_csv("COVID_Vaccine_Database.csv")
308
309     # connect to twilio account
310     client = Client(
311         "ACalddba3fa9157b86845046af637d928b", "2bd4819314c7dc3ad9a3c3f405a8b70d"
312     )
313
314     message = ""Subject: Reminder for Second COVID-19 Vaccination
315
316     Hello {First} {Last} your second COVID-19 vaccination is on coming up on {Vaccination2}""
317
318     # time delta of 3 days and datetime.now() for current time
319     # helps check the current date and second vaccination date
320     check = timedelta(days=3)
321     check = check.days
322     now = datetime.now()
323
324     # number of rows in datframes df2 (COVID_Vaccine_Database)
325     rows = df2.shape[0]
326     # second vaccination date
327     sec_vac = df2.iloc[rows - 1, 8]
328
329     from_address = "ee104sjsu@gmail.com"
330     password = "EE104F2021"
331
332     # open messaging sever
333     # modified loop to read and import a dataframe object (df2)
334     # https://stackoverflow.com/questions/59631659/i-dont-understand-why-i-get-a-too-many-values-to-unpack-error
335     context = ssl.create_default_context()
336     with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
337         server.login(from_address, password)
338         for i, r in df2.iterrows():
339             FD = datetime.strptime(r["FutureDate"], "%Y-%m-%d %H:%M:%S")
340             FD2 = FD.strftime("%B %d, %Y")
341             # check to see if the current date is 3days or less from second vaccination date
342             if (
343                 abs(datetime.strptime(r["FutureDate"], "%Y-%m-%d %H:%M:%S") - now).days
344                 <= check
345             ):
346
347                 # check to see if a second vaccination notification has been sent already
348                 # if message has not been sent send the sendond vaccination date and update the sent
349                 # value to 0.
350                 if r["Sent"] == 1:
351                     server.sendmail(
352                         from_address,
353                         r["Email"],
354                         message.format(
355
356
357                     ),
358                 )
359                 # send second text message for second vaccination
360                 try:
361                     message2 = client.messages.create(
362                         body=message.format(
363                             First=r["FirstName"],
364                             Last=r["LastName"],
365                             Vaccination2=FD2,
366                         ),
367                         from_="+13852157449",
368                         to=r["SendText"],
369                     )
370                     message2.sid
371                 except Exception as e:
372                     # update database, excel file and csv
373                     df2.loc[i, "Sent"] = 0
374                     df2.to_excel("Simple_Registration_Database.xlsx", index=False)
375                     df2.to_csv("COVID_Vaccine_Database.csv", index=None, header=True)

```

replace with your ee104 gmail here


```

378 # verify the date to be less than three days from second vaccination for the
379 # professor tool
380 def date_verify():
381     date_input = datetime.strptime(date_check_entry.get(), "%Y-%m-%d") + timedelta(
382     )
383     submission_success2()
384     df = pd.read_csv("COVID_Vaccine_Database.csv")
385     # start text messaging server from twilio account
386     client = Client(
387     )
388     # time delta of 3 days
389     check = timedelta(days=3)
390     check = check.days
391     rows = df.shape[0]
392
393     # second vaccination date from dataframe
394     # obtaine values from dataframe through their corresponding index Location
395     sec_vac = df.iloc[rows - 1, 8]
396     sec_vac2 = datetime.strptime(sec_vac, "%Y-%m-%d %H:%M:%S") + timedelta(minutes=5)
397     sec_vac_send = sec_vac2.strftime("%B %d, %Y")
398
399     Email = df.iloc[rows - 1, 5]
400     LastName = df.iloc[rows - 1, 2]
401     FirstName = df.iloc[rows - 1, 0]
402     text = df.iloc[rows - 1, 10]
403
404     message = """Subject: Second COVID-19 Vaccination Reminder
405
406     from_address = "eel04.jose.r@gmail.com"
407     password = "dyl1hrmmahxvzxwu"
408
409     context = ssl.create_default_context()
410     with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
411         server.login(from_address, password)
412         if abs(date_input - sec_vac2).days <= check:
413             server.sendmail(
414                 from_address,
415                 Email,
416                 message.format(
417                 ),
418             )
419
420         try:
421             message2 = client.messages.create(
422                 body=message.format(
423                     First=FirstName, Last=LastName, Vaccination2=sec_vac_send
424                 ),
425                 from_="+13852157449",
426                 to=text,
427             )
428             message2.sid
429
430         except Exception as e:
431             server.quit()
432
433
434
435
436
437
438

```

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