Course Project in

“Password Management System”

——Group 10

Group members:

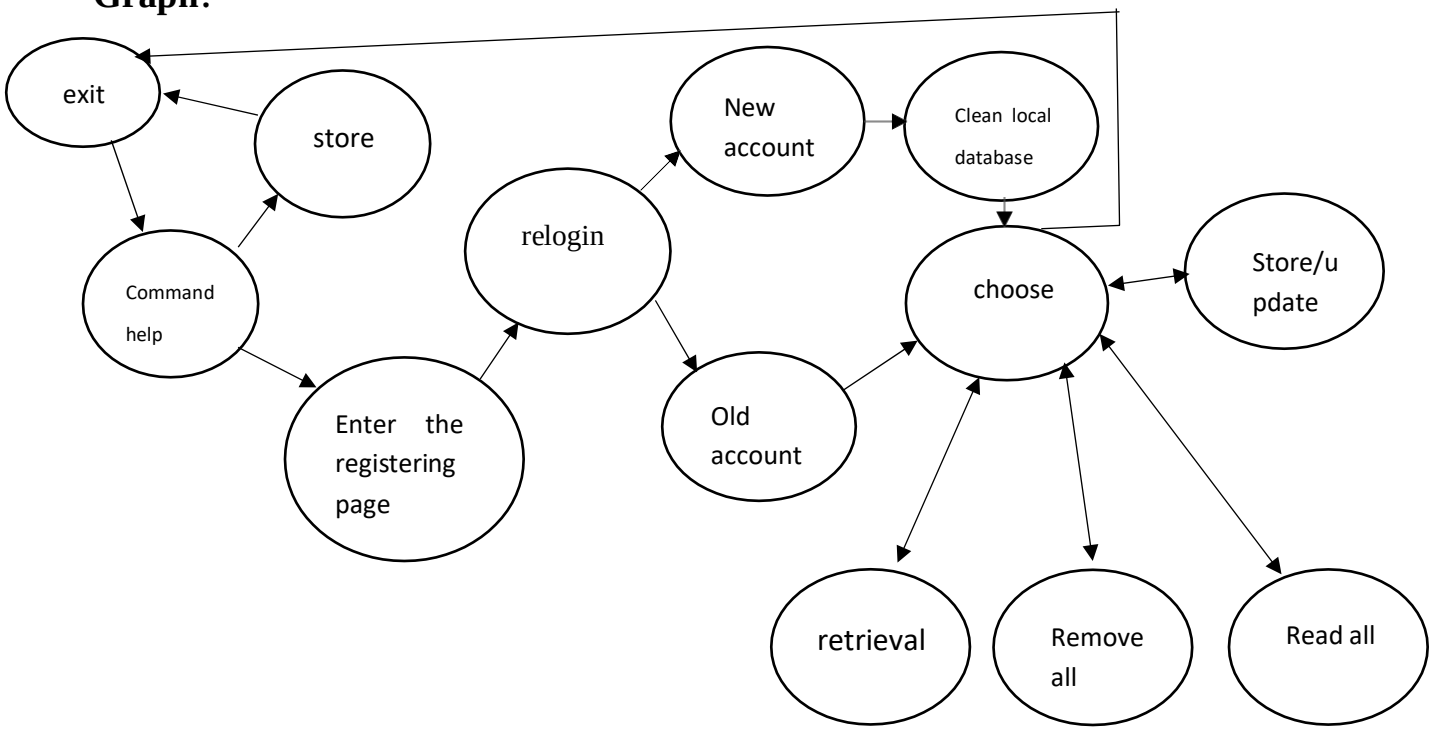
Name	Student ID
Letao ZHAO	21100685D
Zhoudao LU	21099695D
Jiang ZHENG	21095995D
Shuaining HUO	21100777D

The problem description

Security is the foundation on which our internet is built. A completed account system consists of administrators, users, and Guests. Since the ignorance of 'password strength might trigger the privacy problem,' the complexity of users' passwords has drawn significant attention in network security. To undermine peoples’ fear of being hacked, our group designed a simple password management system to help people comprehend the process of password recognition. In this project, we will simulate the operation of a password evaluation system and generate random passwords.

Data abstraction

Graph:



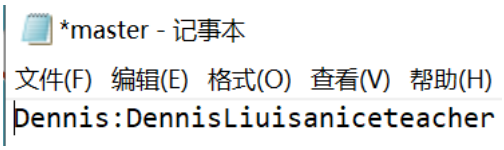
Proper tools：

Dictionary: Before storing in local or cloud, we used a dictionary to store account(s) and password (s). Every time users log in, there is an empty dictionary. When the store/update function is used, users will store the account and password in the dictionary.
Example:
{'emal': '_~i="VC} \\)<&+EG`P0Xa', 'whatsapp': 'K@TL>#/DvZo$Ym}x5%6a', 'banking': '9wT-4z:oraX.I&%j/?>N', 'shopping': 'gbdc<F!K|Ln"5m8pDJEt'}

Text files:

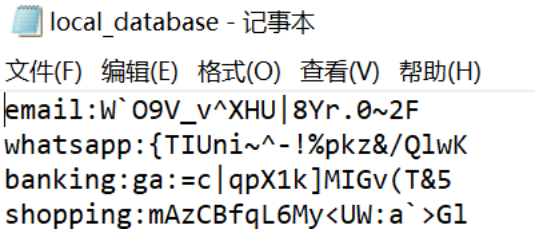
1. We use text files to store the main account and password of this system. Because we can store text files after exiting, every time user enters the registering page, the user should enter their account and password. After logging in, the read, remove and use retrieval functions.

   Example:

2. We use text files to store other accounts and passwords because we can store text files after exiting.

   Example:

3. Google Spreadsheets: We also use Google API to store other accounts(s) and password (s) when requested by the user. We came out of this idea when using another password manager and discovered that it could Sync accounts(s) and password (s) by using Google Drive. Therefore, we named this function "Cloud."

   Storing data in Google spreadsheet API is much more complicated than a text file. Firstly, we will have to create two separate lists called "accounts" and "passes" to store accounts and passwords along with the dictionary. We then use Google's database to store clients' datasets. In the process of 'Cloud,' we find it difficult to retrieval the last uploaded data from Google because removing data from a cloud database requires higher authority, which is very hard to manipulate in a python program. However, our project works not only for the client but also for servers. Thus, people can use other online accounts to refresh the cloud datasets. The setup guide of ‘Cloud’ is at the end of the report.

Python implementation of data types

1. We used a dictionary to store data at first. This system has a dictionary named 'passwords. For every time users use store/update function, passwords[account] = password, correspondingly.

2. We use a text file to store information locally. Before storing it in the file, we use a list and" \n" at the end of each couple. It makes the local database easy to read.

```
# local database
password_list = []
for i in passwords:
    password_list.append("{}:{}\n".format(i,passwords[i]))
pwd = open("local_database.txt", "a", encoding="utf-8")


pwd.writelines(password_list)
pwd.close()
print("Written Successful")
print("\nFinished\n")
break
```

3. We can also use Google spreadsheet API (cloud) to store information. Before storing in the cloud, we use a list to store all the account(s) and password(s), then we will need the codes below:

```
# Call the Sheets API
sheet = service.spreadsheets()

usernames = [accounts]

resource1 = {
"majorDimension": "ROWS",
"values": usernames
}

range1 = "A2:A";
service.spreadsheets().values().append(
spreadsheetId=SAMPLE_SPREADSHEET_ID,
range=range1,
body=resource1,
valueInputOption="USER_ENTERED"
).execute()

past = [passwords]

resource2 = {
"majorDimension": "ROWS",
"values": past
}

range2 = "B2:B";
service.spreadsheets().values().append(
spreadsheetId=SAMPLE_SPREADSHEET_ID,
range=range2,
body=resource2,
valueInputOption="USER_ENTERED"
).execute()
print("Written Successful")
```

Design and key functions

We have 7 different functions, random password (), analysis(password), choose (), store (), management (), update() and main().

Random password () is a function used to generate a random password. Users can set the length and get a random password consisting of an uppercase letter, lowercase letter, punctuation, and number.

Analysis () is used to analyze the strength of the password. The program will check the length of

the password every time a new password is generated. The rule is as below.

scoring rules:

Base Score:

1. Password length(pwdLen):
   (1) 0 points: less than or equal to four characters
   (2) 5 points: five to seven characters
   (3) 10 points: more than or equal to eight characters
2. Letters:
   (1) 0 points: no letters
   (2) 5 points: all in lowercase (capital) letters
   (3) 10 points: case-mixed letters
3. Numbers:
   (1) 0 points: no numbers
   (2) 5 points: one number
   (3) 10 points: more than or equal to three numbers
4. Special symbols:
   (1) 0 points: no special symbols
   (2) 5 points: one special symbol
   (3) 10 points: more than one symbol

Plus Score:

(1) 2 points: two types of passwords can be entered
(2) 3 points: three types of passwords can be entered
(3) 5 points: four types of passwords can be entered

Min Score:

If there are consecutively repeated characters of a single kind, each repetition is subtracted by one point

In Total:

50 points

Final grade:

Score>=40	Very strong
30<=Score<40	Strong
20<=Score<30	Medium
10<=Score<20	Commonly
Score<10	Weak

Choose () is used to make choices after registering. This function calls the user to make choices.

```
 What do you want?

1. store/update new account(s) and password(s)
2. read all account(s)/password(s) from local or cloud database
3. remove all account(s)/password(s) from local or cloud database
4. retrieval password from local database
5. Exit

Please Enter a number: |
```

Store () is used to generate and update account(s) and password(s). After creating or writing new password (s), users can choose to store account(s) and password(s) in the local or cloud.

Main () is used to call the first page. Users have two choices.

```
command helpers:
 a. enter the registering page
 b. store/update account(s) and password(s)

Please type a or b: |
```

If users type, a management () works. And they can make choices as choose () shows. Users cannot use the 2, 3, 4 functions if they type b. Before using these functions, users should re-logging for safety.

If users type b, update () works. Users can only store or update their password (s) for convenience.

Highlights

We want to highlight the advantages of our password management system.

At first, users have two choices as below.

```
command helpers:
 a. enter the registering page
 b. store/update account(s) and password(s)

Please type a or b: |
```

Why do we set two choices?

If users log in repeatedly, it will be more time-consuming. So, we have set up the b situation, users can continue to save and update the account(s) and password(s), but they cannot perform other operations to protect the safety of password (s).

If users want to do other operations, type a. Then, users should enter the account name of the

password management system. But if the account isn't the same as the account logged in last time, the local database will be cleaned up to protect users ' databases. We use getpass to prompt the user to enter a password without echoing; this way is safer. There are more complicated permissions of the cloud database, so we don't clean the cloud database simultaneously. Therefore, users must remember their main account and password. After they log in, they have four choices as below.

```
 What do you want?

1. store/update new account(s) and password(s)
2. read all account(s)/password(s) from local or cloud database
3. remove all account(s)/password(s) from local or cloud database
4. retrieval password from local database
5. Exit

Please Enter a number: |
```

If users enter 1, they can store or update account(s) and password (s).
If users enter 2, they can read all the information. But people should enter their main password for safety. We set this function to let people know comprehensive information about their password (s). For local, users can know how many different passwords and see the old password (s) they stored in the past. Especially because we have a b situation at first, and other people can update your password even if they don't know your primary password. So users can use the read function to see the original password they set if this situation happens. This function is selected both for safety and convenience.
If users enter 3, they can remove all their data from the local or cloud database.
If users enter 4, they can retrieval their password from a local database. If users have too many different accounts, the read function isn't convenient. However, because the permissions of the cloud are too complicated, we have not set up additional cloud retrieval functions. Users can see the passwords by the reading function. We can use this function as below.

```
Please Enter a number: 4

Do you save your password(s) in the local?y/n y

Which account's password do you want to extract?banking
This password is:=c|qpX1k]MIGv(T&5
```

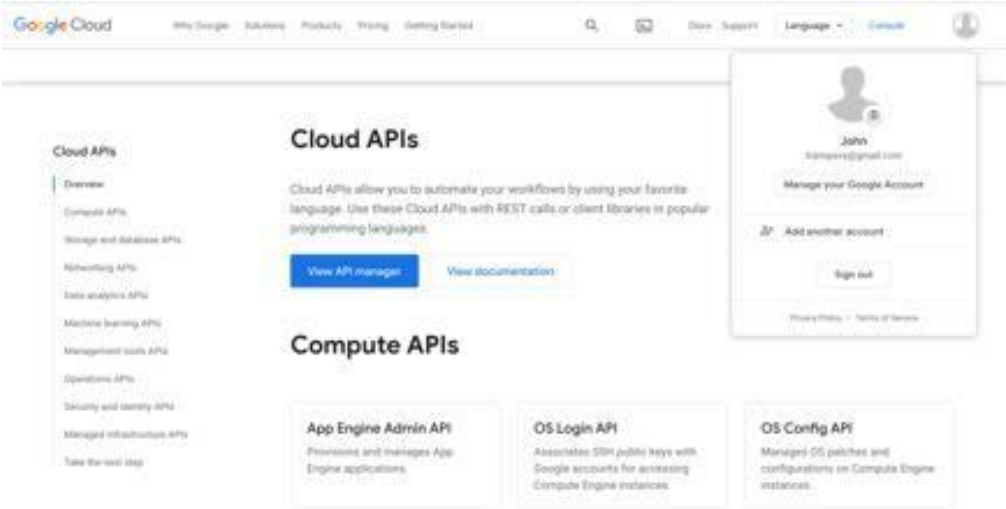
If users enter 5, they can come back to the command helper page.
By the way, we try our best to don't let the program report errors.

Conclusion

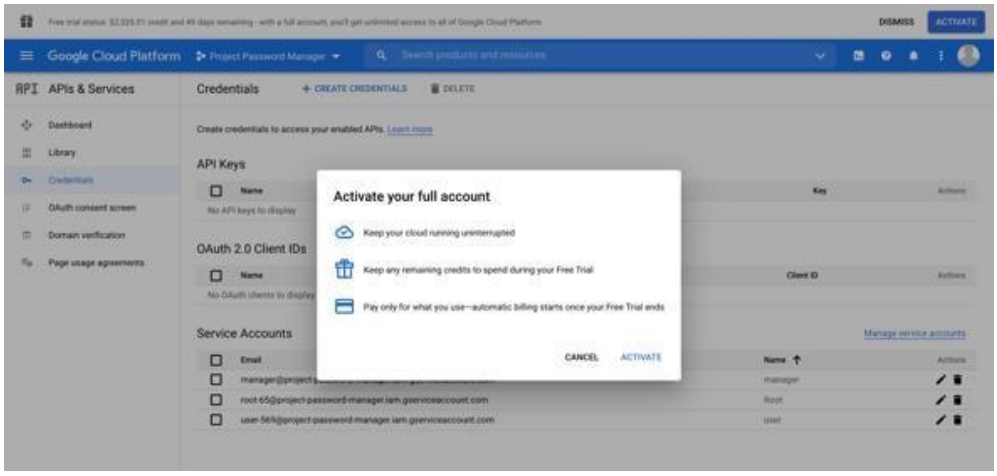
In this project, numerous extra functions are included, such as cloud storage, password encryption. The trickiest problem we encountered was uploading data because we had to load an additional package online, out of our current knowledge. We spent much time testing and finding bugs. The error caused by contraction repeatedly occurred due to the complexity of this project. On the default page, users are required to choose the mode between client and administrator. In the client mode, users can set new accounts and passwords, and in the administrators' mode, users' information can be read and modified. Besides, some essential functions- such as random password generator and password analyzing- can help users check the safety factor of passwords. By simulating the process of the password management system, we found that practice is comparatively essential to theoretical thinking.

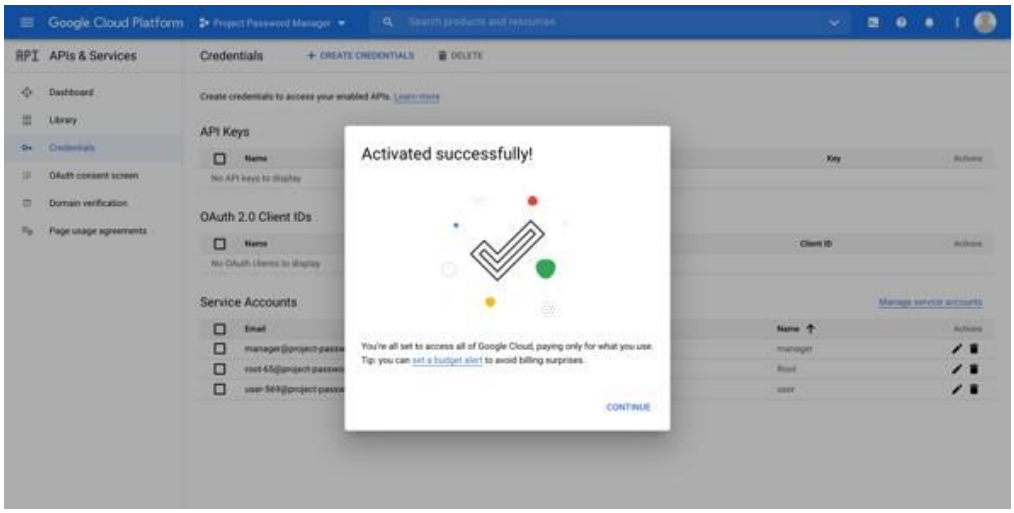
Setup guide

1. Create an API account with Google Account

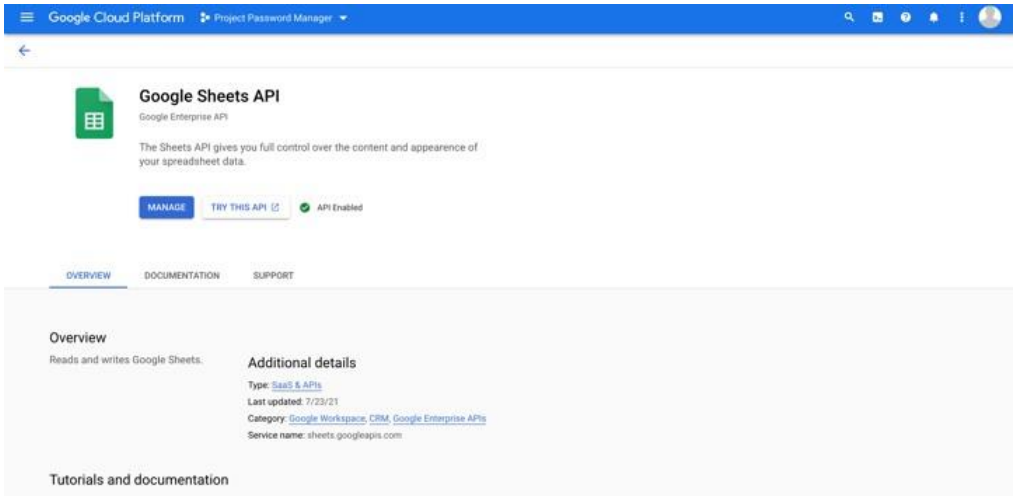


2. Activate Full Account

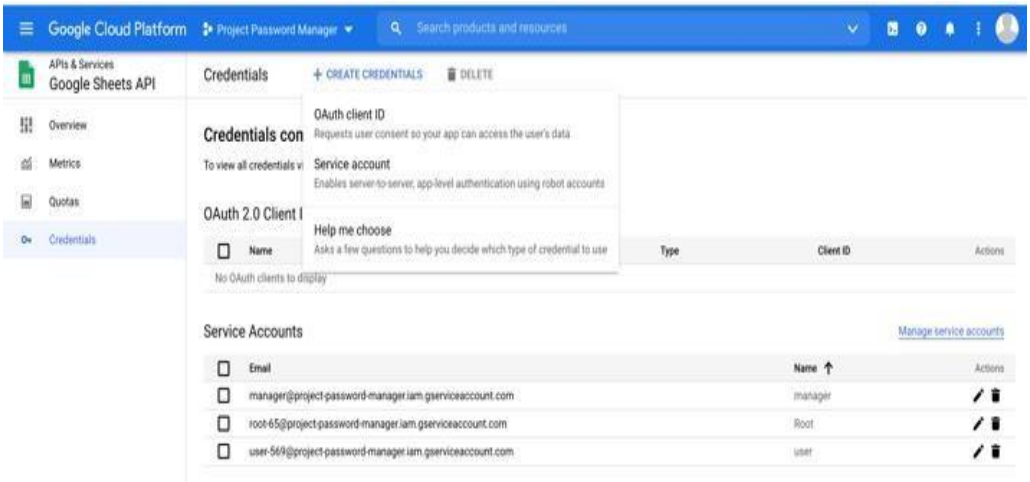


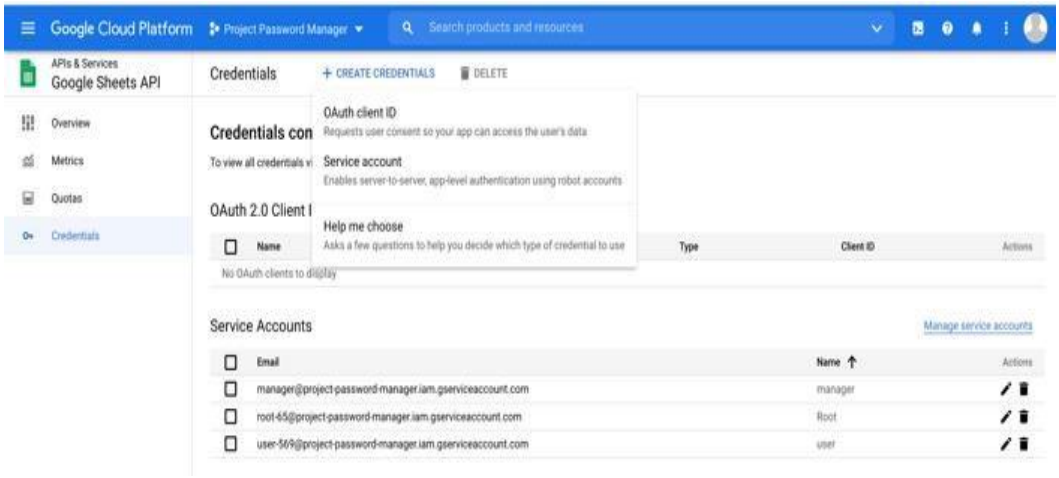


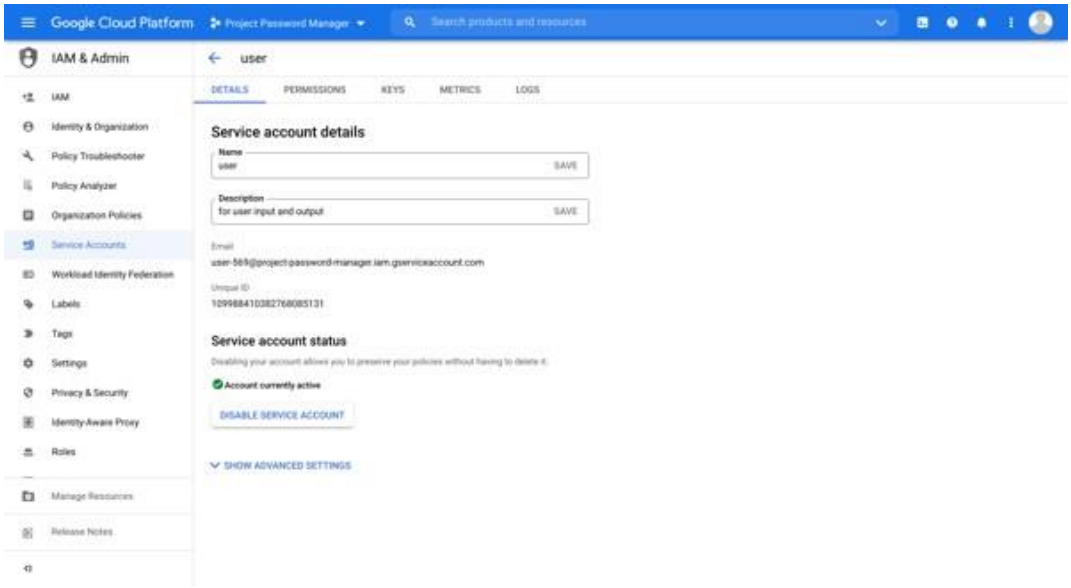
3. Enable Google Sheets API



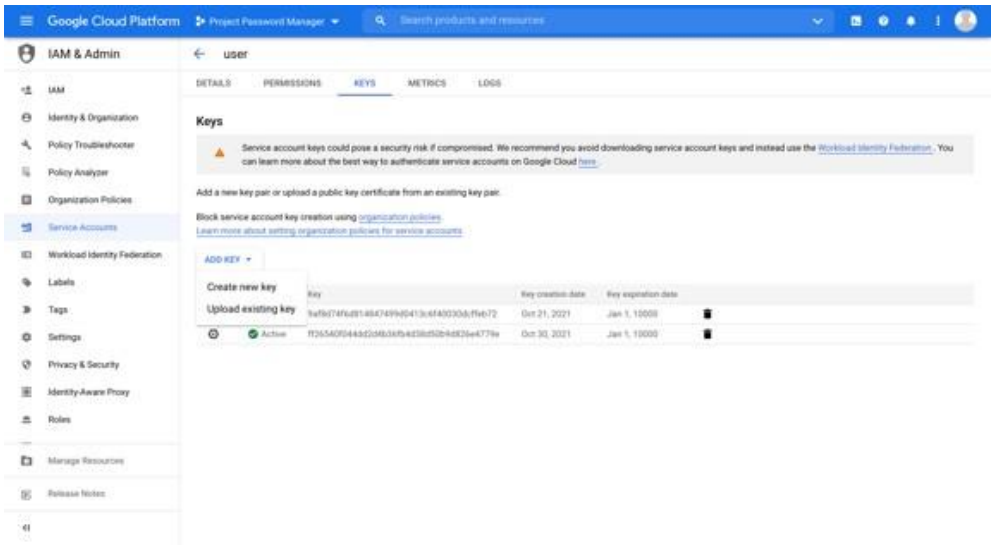
4. Create a Service Account

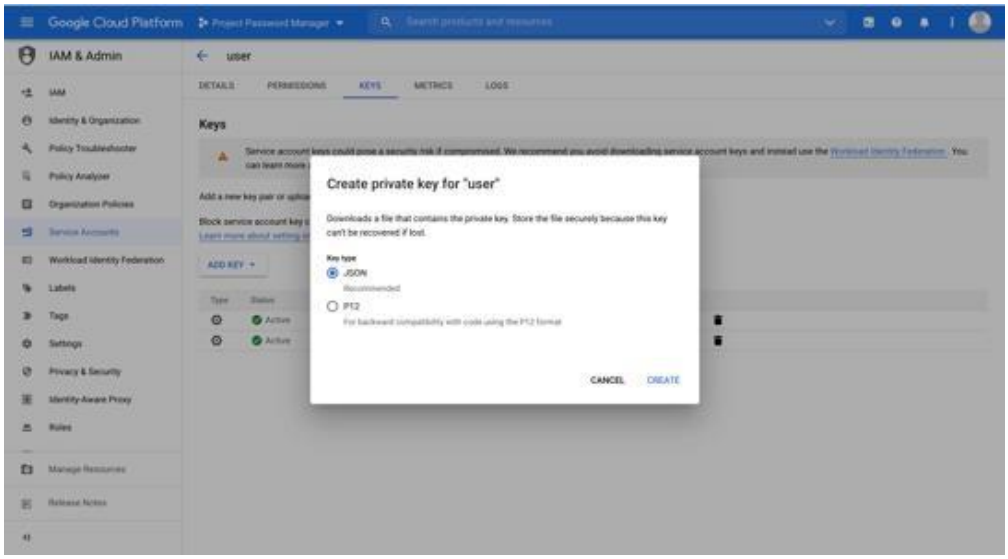


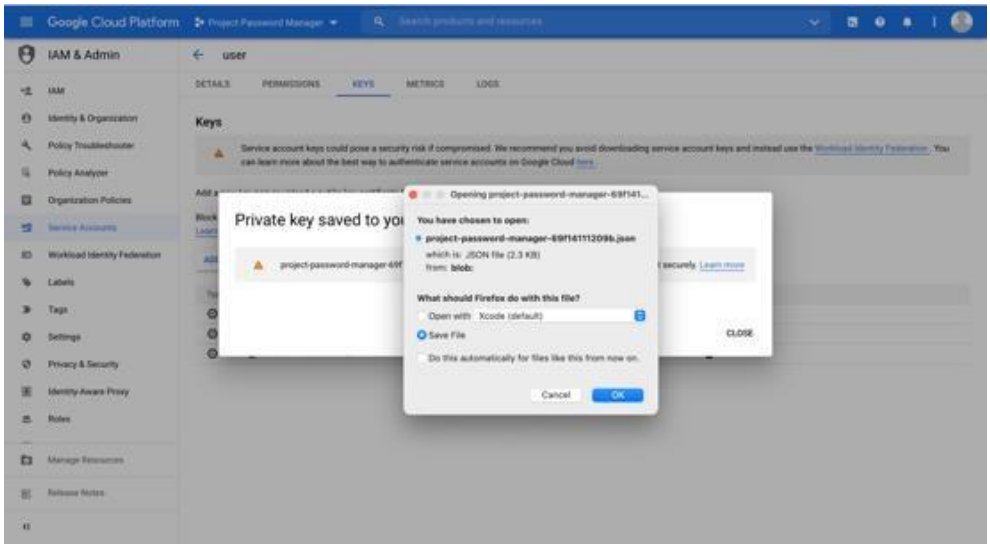




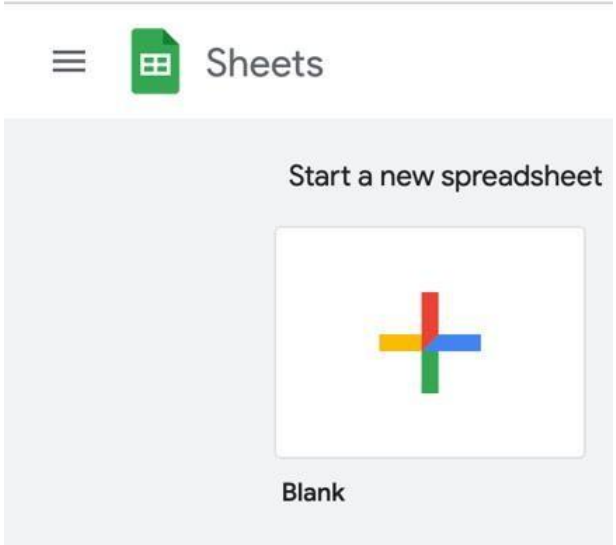
5. Add and save Json Keys to the same file with py. File and rename Json Keys to keys, JSON

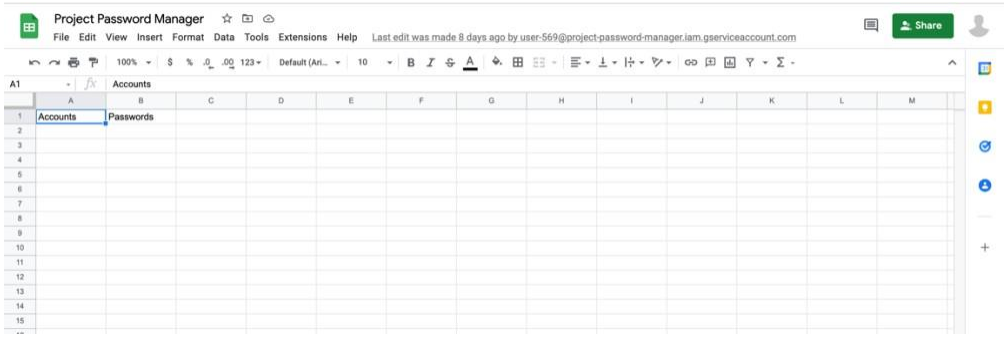

Google Cloud Platform
Project Password Manager
Search products and resources
IAM & Admin
IAM
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Federation
Labels
Tags
Settings
Privacy & Security
Identity-Aware Proxy
Roles
Manage Resources
Release Notes
user
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Keys
Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use the Workload Identity Federation. You can learn more about the best way to authenticate service accounts on Google Cloud here.
Add a new key pair or upload a public key certificate from an existing key pair.
Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts
ADD KEY
Create new key
Upload existing key
Key
Key creation date
Key expiration date
Oct 21, 2021
Jan 1, 10000
Active
Oct 30, 2021
Jan 1, 10000


Google Cloud Platform
Project Password Manager
IAM & Admin
user
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Keys
ADD KEY
Create private key for "user"
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.
Key type
JSON
Recommended
P12
For backward compatibility with code using the P12 format
CANCEL
CREATE

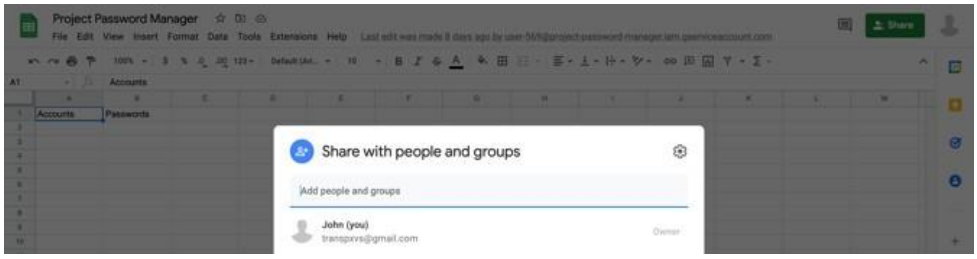

Google Cloud Platform
Project Password Manager
IAM & Admin
user
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Keys
Private key saved to you
CLOSE
Opening project-password-manager-69f141...
You have chosen to open:
project-password-manager-69f141112096.json
which is: JSON file (2.3 KB)
from: blob:
What should Firefox do with this file?
Open with Xcode (default)
Save File
Do this automatically for files like this from now on.
Cancel
OK

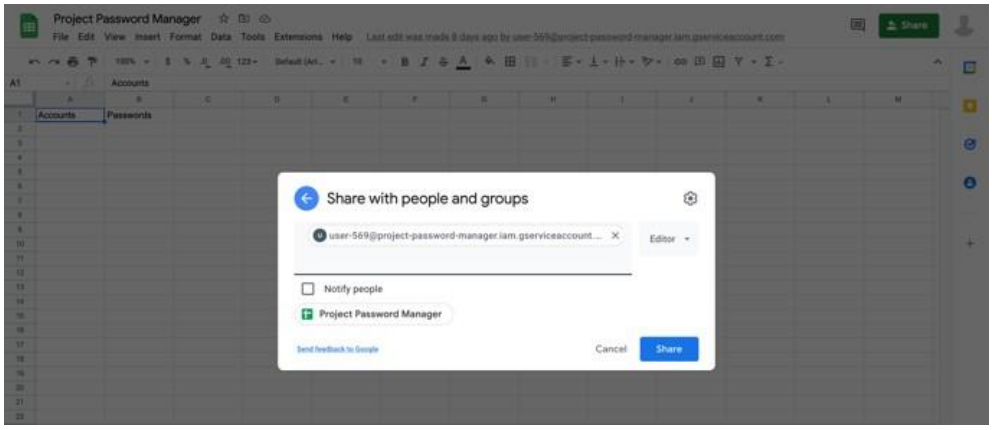
6. Start a new Spreadsheet with Google sheets and enter accounts to A1 and passwords to B1.

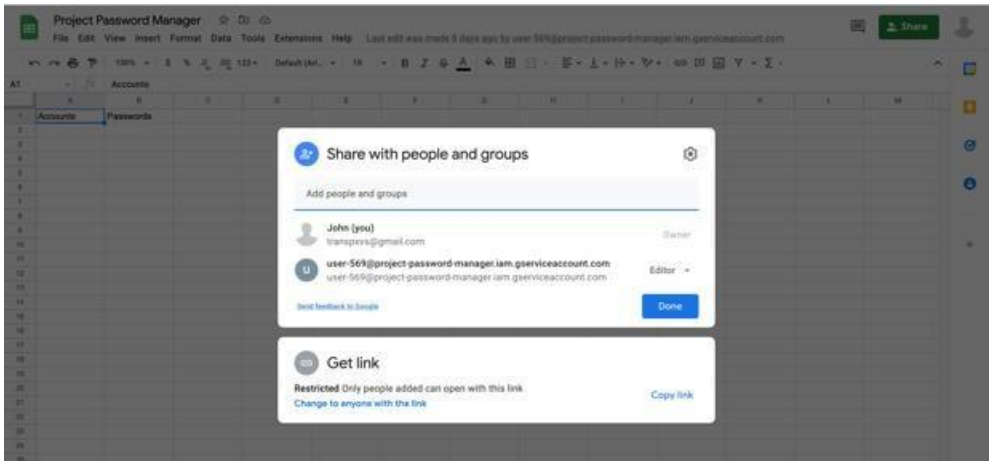




7. Click share, copy/paste service account to” Add people and groups," and then share spreadsheet to service account as editor.

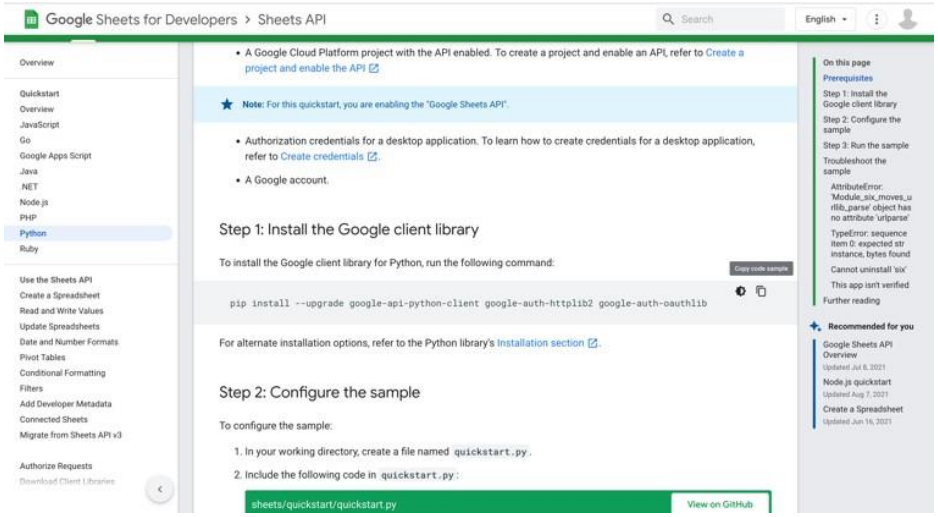




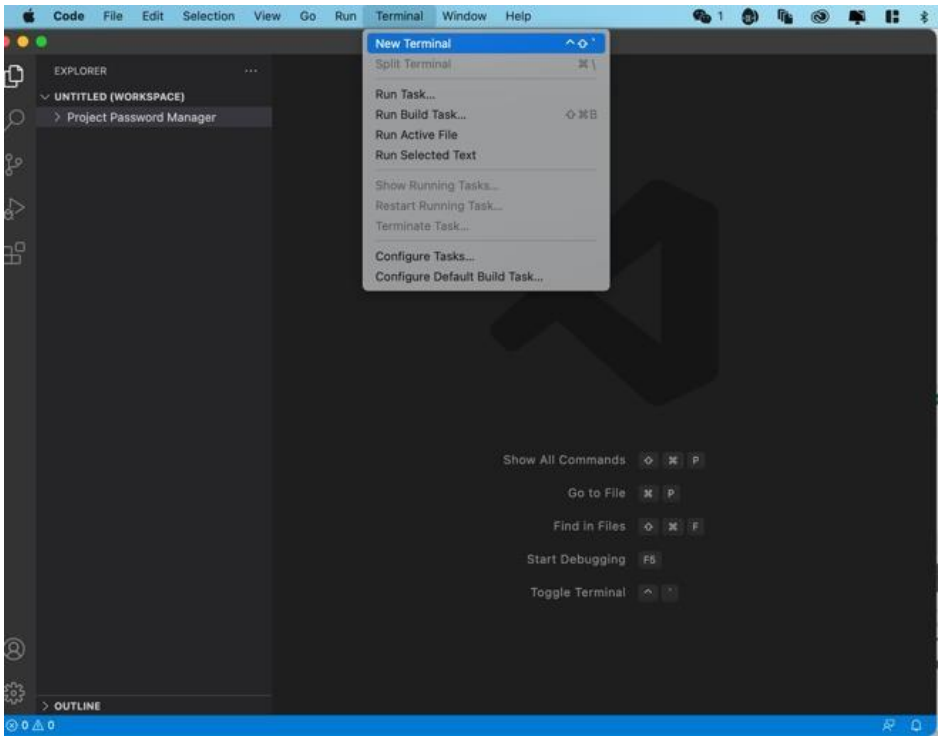


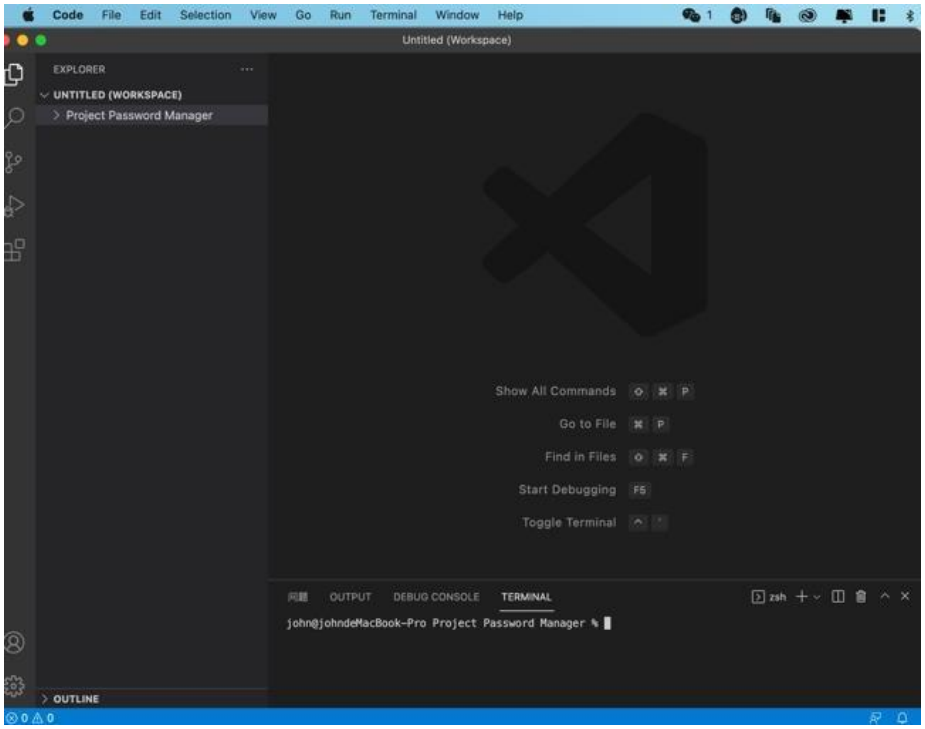
8. Download and Install Virtual Space Code

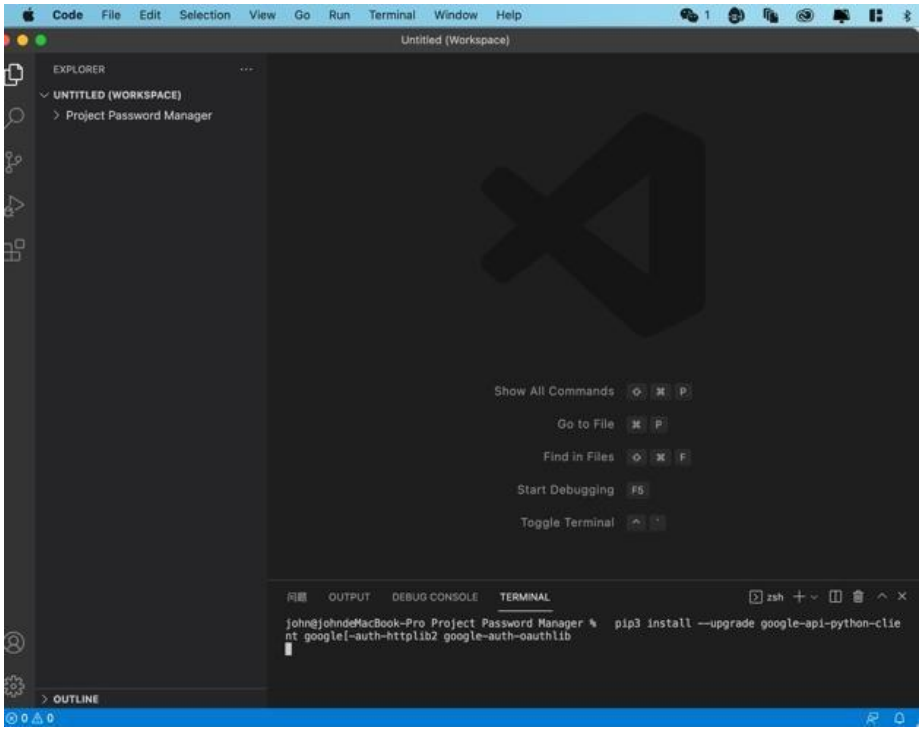
9. Go to https://developers.google.com/sheets/api/quickstart/python and copy the code for step one.

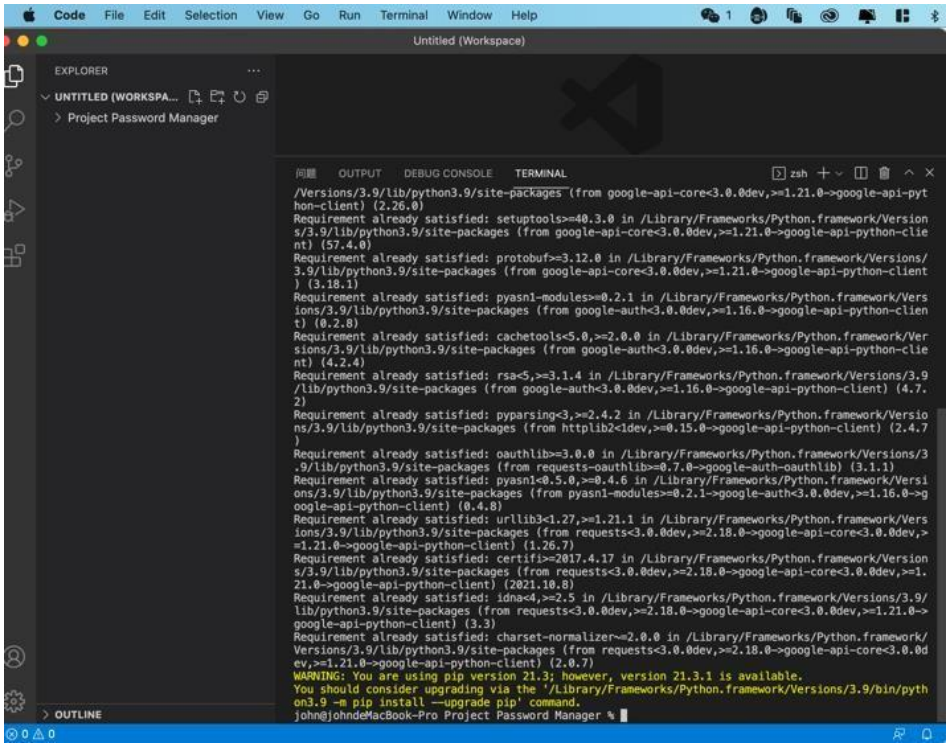


10. Open the new Terminal in Virtual Space Code, paste step one code inside the terminal, and then install an external library.

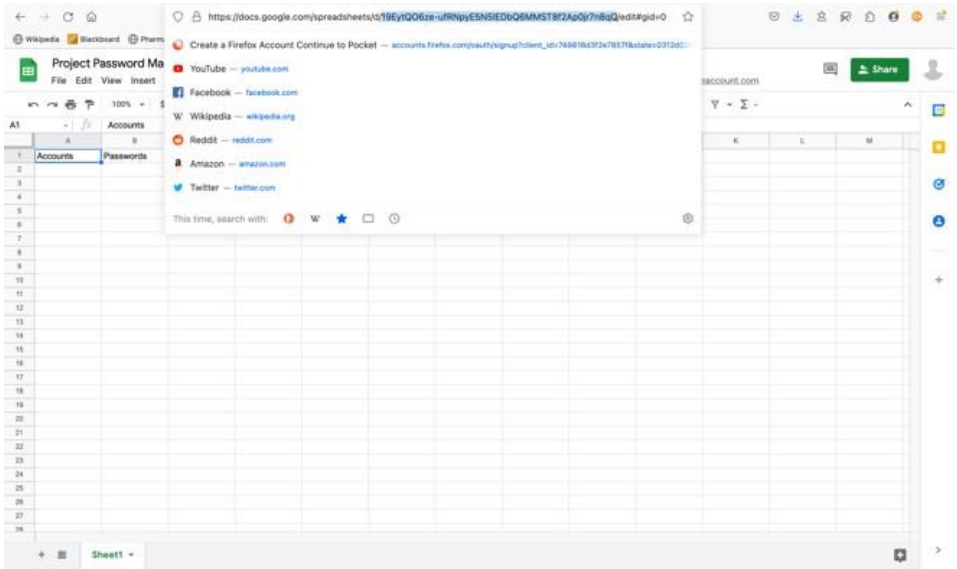



Code File Edit Selection View Go Run Terminal Window Help
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
Project Password Manager
Show All Commands
Go to File
Find in Files
Start Debugging
Toggle Terminal
問題 OUTPUT DEBUG CONSOLE TERMINAL
zsh
john@johndeMacBook-Pro Project Password Manager %
OUTLINE

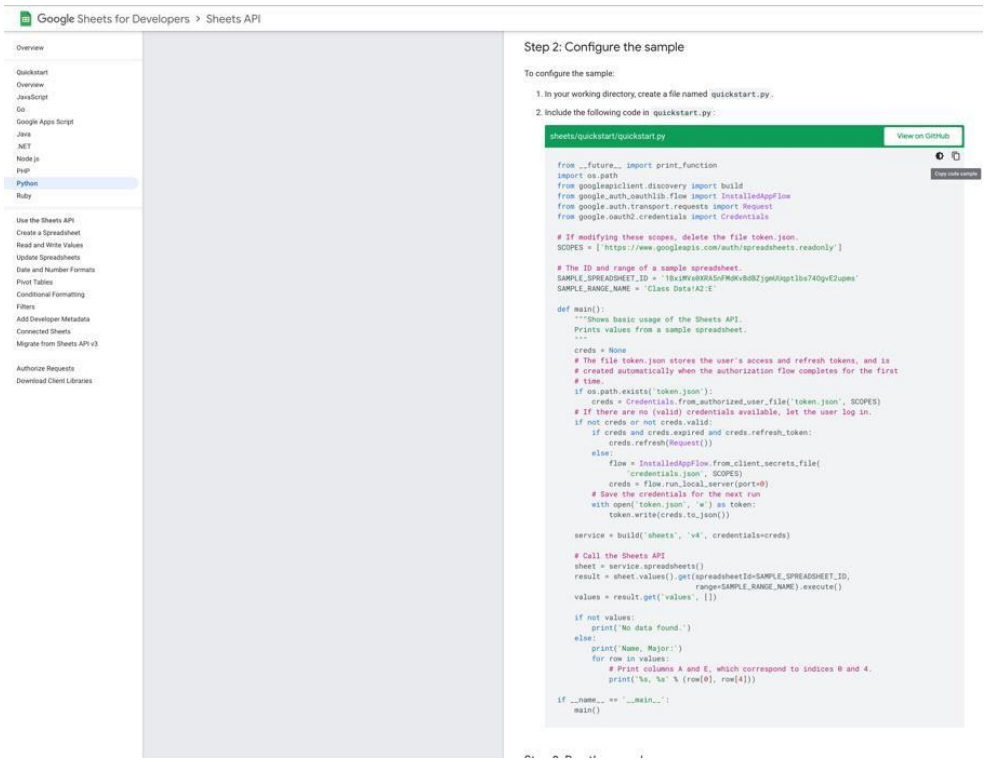

Code File Edit Selection View Go Run Terminal Window Help
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
Project Password Manager
Show All Commands
Go to File
Find in Files
Start Debugging
Toggle Terminal
問題 OUTPUT DEBUG CONSOLE TERMINAL
zsh
john@johndeMacBook-Pro Project Password Manager % pip3 install --upgrade google-api-python-client google-auth-oauthlib
OUTLINE



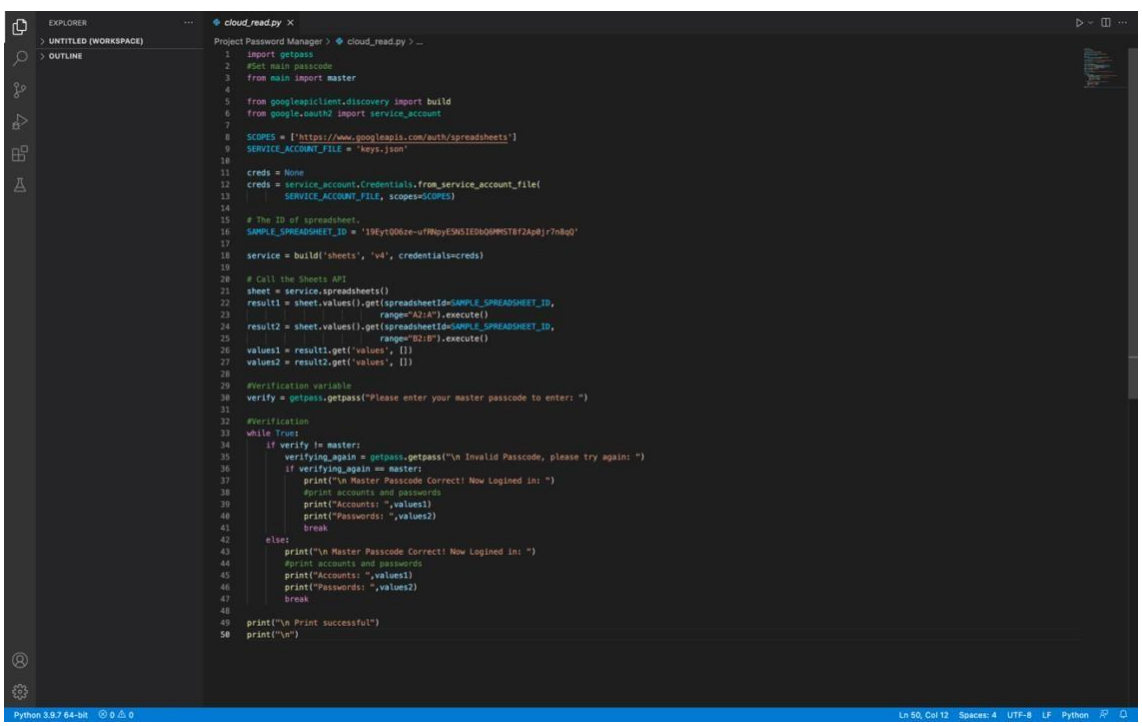
11. Copy Spreadsheet ID



12. Go to https://developers.google.com/sheets/api/quickstart/python. Copy and paste the python code for step two

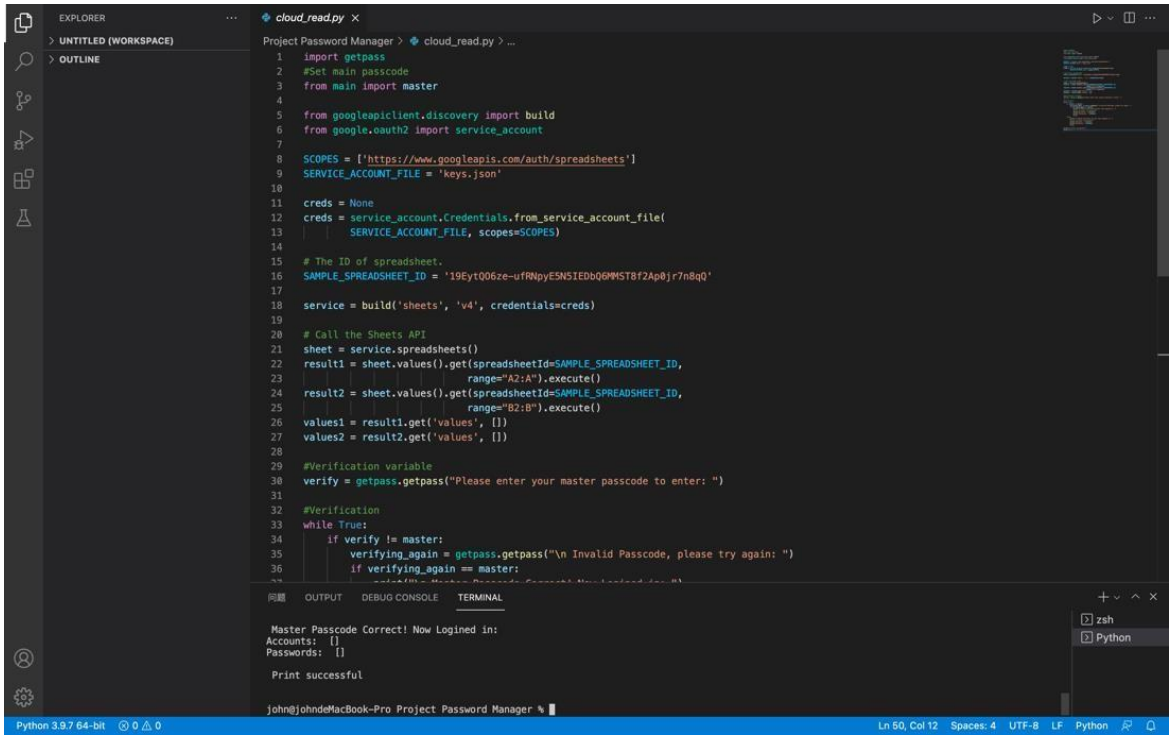


13. Modify the step 2 python code to the following code.



SERVICE_ACCOUNT_FILE = the JSON key file that you got from step 5

SAMPLE_SPREADSHEET ID = the spreadsheet ID that you got from step 11



Sources

[1]
https://developers.google.com/sheets/api/quickstart/python (accessed Nov. 29, 2021).

[2]
https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets.values

[3]
https://pythonexamples.org/python-read-text-file/

[4]
https://techeplanet.com/python-create-json-file/

[5]
https://medium.com/analytics-vidhya/create-a-random-password-generator-using-python-2fea485e9da9.

p.s.: these are websites we learned from. We understood how to store data in the cloud and so on but didn't copy them directly, so we set "Sources" instead of "references."