

# APU Timetable to Google Calendar Project

The system built in this project can automatically export the APU timetable to Google Calendar by utilizing various tools. By following this step-by-step guide, you will be able to build the system from scratch!

## Pre-requisite:

Please install the necessary libraries before you start building the system:

1. Download the script, from <https://bootstrap.pypa.io/get-pip.py>.
2. Open a terminal/command prompt, cd to the folder containing the get-pip.py file and run:

Windows user:

```
1 py get-pip.py
```

MacOS user:

```
1 python get-pip.py
```

3. After installed pip, you can start installing libraries. Open a new terminal/command prompt, run:

```
1 pip install requests bs4 google-api-python-client python-dotenv
```

4. When the libraries installation is completed, you are ready to start!

**\*\*Note:** For all libraries import in the script, please insert it together at the top of the script.

```
1 from datetime import date, timedelta, datetime
2 import requests
3 from bs4 import BeautifulSoup
4
5 from googleapiclient.discovery import build
6 from google.oauth2 import service_account
7
8 import os.path
9 from dotenv import load_dotenv
10 import smtplib
11 import ssl
12 from email.message import EmailMessage
```

## Web Scraping

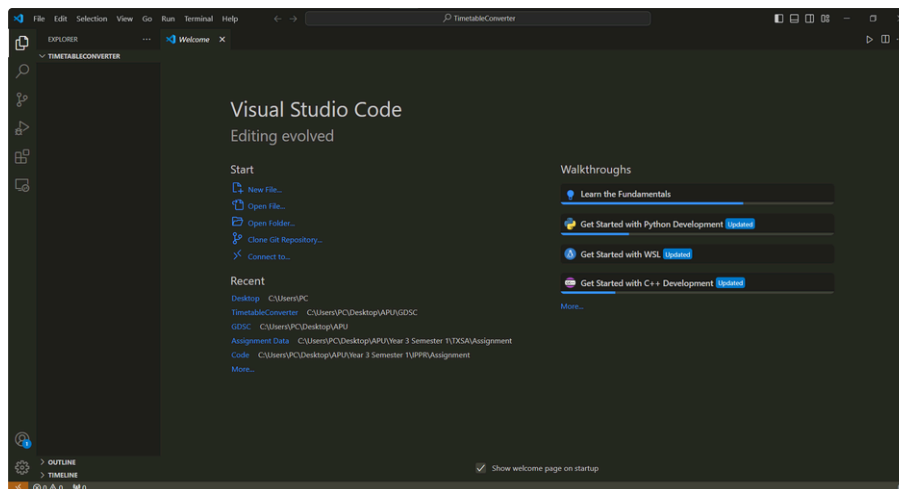
1. First, you will need to navigate to your timetable in APSpace.
2. Click on the "print" logo in the timetable page, you will be redirected to the web page that specific to a week. Example:  
[https://api.apiit.edu.my/timetable-print/index.php?Week=2024-04-29&Intake=APD3F2402CS\(AI\)&Intake\\_Group=G1&print\\_request=print\\_tt](https://api.apiit.edu.my/timetable-print/index.php?Week=2024-04-29&Intake=APD3F2402CS(AI)&Intake_Group=G1&print_request=print_tt)

APD3F2402CS(AI)					
Week 29-Apr-2024					
DATE	TIME	CLASSROOM	LOCATION	SUBJECT/MODULE	LECTURER
Mon, 29-Apr-2024	08:30 - 10:30	E-08-03	APU CAMPUS	CT107-3-3-TXSA-L-1	RAHEEM MAFAS
Mon, 29-Apr-2024	11:00 - 12:00	B-05-10	APU CAMPUS	CT032-3-3-FAI-T-1	DR. KUAN YIK JIUNN
Tue, 30-Apr-2024	10:45 - 12:45	B-04-04	APU CAMPUS	CT036-3-3-IPPR-L-1	DR. V. SIVAKUMAR
Tue, 30-Apr-2024	13:45 - 14:45	B-05-03	APU CAMPUS	CT027-3-3-EPDA-L-1	KAU GUAN KIAT
Tue, 30-Apr-2024	15:00 - 16:30	D-07-08	APU CAMPUS	CT050-3-3-PRMGT-T-7	TS. JERRY CHONG CHEAN FUH
Tue, 30-Apr-2024	16:45 - 17:45	B-06-03	APU CAMPUS	CT036-3-3-IPPR-LAB-1	DR. V. SIVAKUMAR
Thu, 02-May-2024	08:30 - 10:30	B-04-05	APU CAMPUS	CT052-3-3-L-2	DHASON PADMAKUMAR
Thu, 02-May-2024	10:45 - 12:45	Tech Lab 4-05	APU CAMPUS	CT027-3-3-EPDA-LAB-1	KAU GUAN KIAT
Thu, 02-May-2024	13:30 - 15:30	ONLMCO3-10	APU CAMPUS	CT050-3-3-PRMGT-L-1 (Online)	VIJAYARAJ A/L C. VIJAYASINGAM
Fri, 03-May-2024	11:15 - 12:15	Tech Lab 6-14	APU CAMPUS	CT107-3-3-TXSA-LAB-1	RAHEEM MAFAS
Fri, 03-May-2024	15:30 - 17:30	Auditorium 2 @ Level 6	APU CAMPUS	CT065-3-3-AIG-L-1	DR. VAZEERUDEEN ABDUL HAMEED

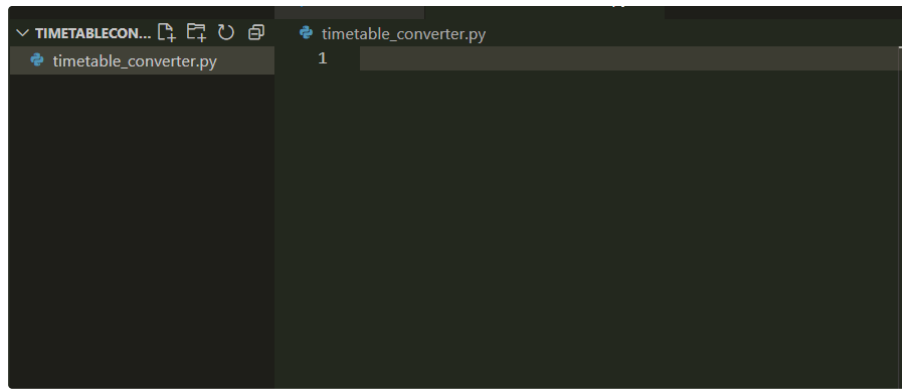
3. If you noticed in the example url, the parameters have week, intake, and intake group, which you can specify timetable by your own.
4. In the web page, click F12 to inspect the web page. You will see the HTML text of current web page in the "Element" tab.
5. Click Ctrl + Shift + C and hover around the information of the classes, the element in HTML form will be shown.

The screenshot shows a web browser displaying a timetable for 'APD3F2402CS(AI)' for Week 29-Apr-2024. The timetable table lists classes with columns for Date, Time, Classroom, Location, Subject/Module, and Lecturer. The developer tools (F12) are open, showing the HTML structure of the page. The 'Elements' tab highlights the table element, and the 'Styles' tab shows the CSS rules applied to it, including background colors and text alignment.

6. That's the information that we wanted to scrape from this website.
7. In your desktop, create a folder with name TimetableConverter.
8. Open VSCode, open the folder you have created.



9. Click on the "New File" button on your folder tab, and create a python file named "timetable\_converter.py".



10. First, define a main function and `get_week_start` function for the program, this function will cover the parameters (intake, intake group, elective module to be filtered, week of timetable) for the url. The `get_week_start` function is to retrieve the date for the first day of next week when then program run.

```
1 from datetime import date, timedelta, datetime # import libraries
2
3 def get_week_start():
4     today = date.today()
5     days_until_next_week = 7 - today.weekday()
6     next_week_start = today + timedelta(days=days_until_next_week) # next week
7     return next_week_start
8
9 def main():
10     intake = "APU3F2402CS(AI)"
11     intake_group = "G1"
12     remove_list = ['ALG']
13     week_start = get_week_start()
```

11. Now we have gathered the necessary parameters to navigate to the url, next we will start fetching the timetable from the website. This `fetch_timetable` function will retrieve the whole timetable from the website

```
1 import requests # import library
2 from bs4 import BeautifulSoup # import library
3
4 def fetch_timetable(week, intake, intake_group):
5     response = requests.get(f'https://api.apiit.edu.my/timetable-print/index.php?Week={week}&Intake={intake}&IntakeGroup={intake_group}')
6     soup = BeautifulSoup(response.text, 'html.parser')
7     return soup.find('table', class_='table')
```

12. In your main function, assign the timetable into a variable "timetable\_table".

```
1 timetable_table = fetch_timetable(week_start, intake, intake_group)
```

13. Next, we will get information for all the classes in this week as we have seen in the inspection page in the browser, and assign it into `timetable_data`. In main function, insert:

```
1 if timetable_table:
2     timetable_data = []
3     rows = timetable_table.find_all('tr')[2:]
4
5     if rows:
6         for row in rows:
7             cells = row.find_all('td')
8
9             # date, time, classroom, location, subject, lecturer
10            date = cells[0].text.strip()
```

```

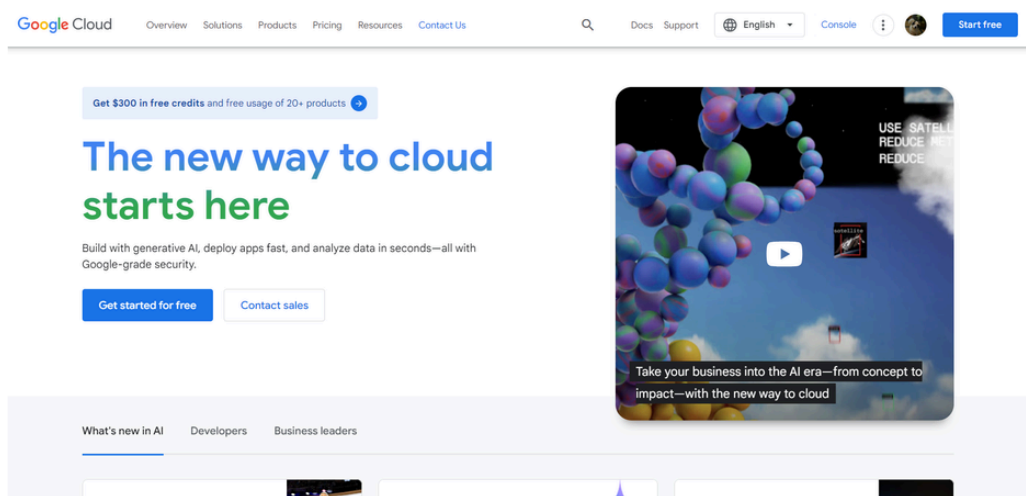
11     time = cells[1].text.strip()
12     classroom = cells[2].text.strip()
13     location = cells[3].text.strip()
14     subject = cells[4].text.strip()
15     lecturer = cells[5].text.strip()
16
17
18     module_name = subject.split('-')[3]
19     if module_name not in remove_list:
20         timetable_data.append({
21             'Date' : date,
22             'Time' : time,
23             'Classroom' : classroom,
24             'Location' : location,
25             'Subject/Module' : subject,
26             'Lecturer' : lecturer,
27         })

```

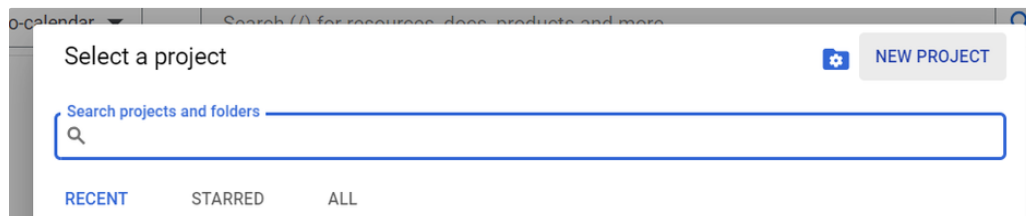
14. We have finished the web scraping part as the information we needed have been retrieved from the website.

## Google Calendar API Integration

1. Now, we will need to enable Google Calendar API for our Google Account.
2. Navigate to Google Cloud Console: <https://console.cloud.google.com/> and sign in with your Google Account.




3. Click on New Project.




4. Create a new project named Timetable Converter.


## New Project

 You have 11 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name \*  
Timetable Converter 

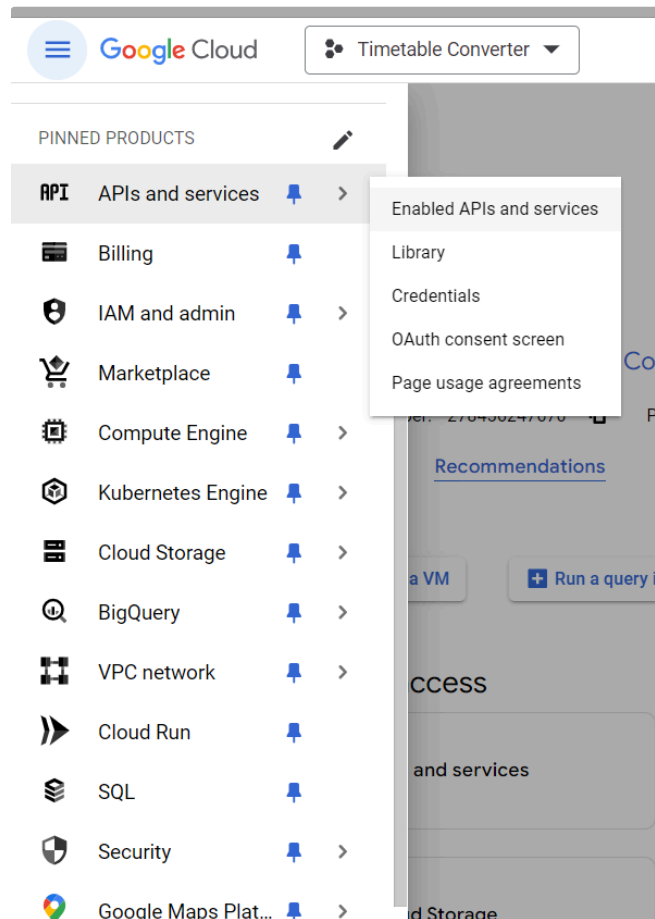
Project ID: timetable-converter-421207. It cannot be changed later. [EDIT](#)

Location \*  
 No organisation [BROWSE](#)

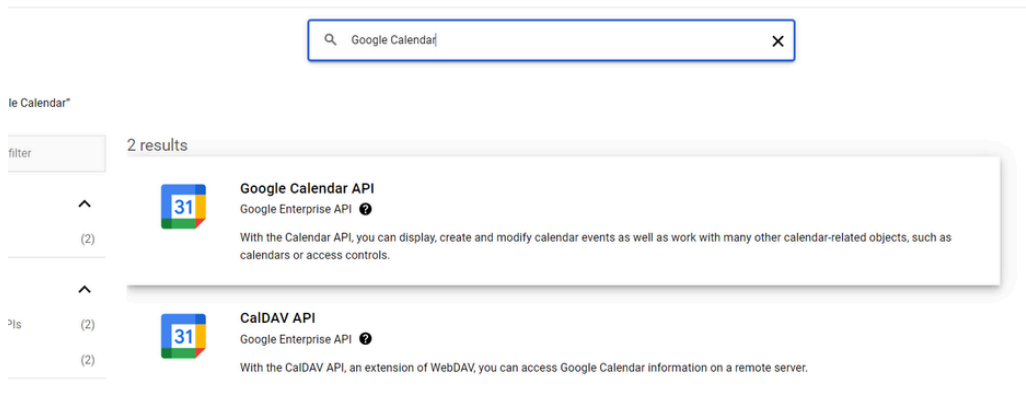
Parent organisation or folder

[CREATE](#) [CANCEL](#)

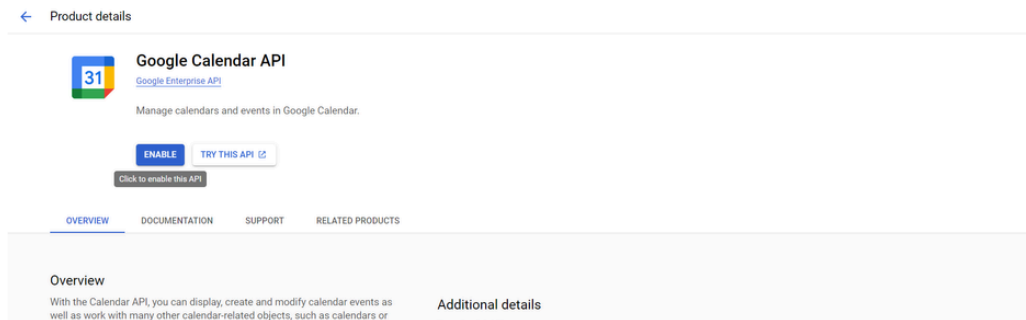
5. Click on the Navigation menu on the left side, hover to APIs and service, Click on "Library".



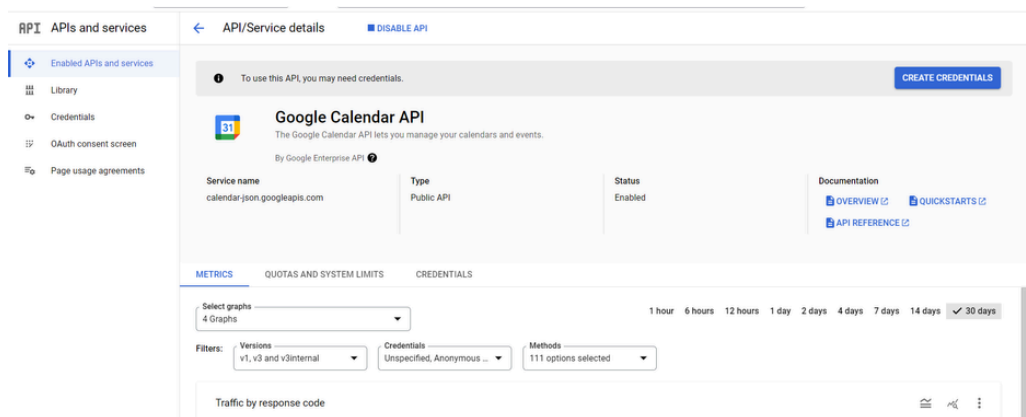
6. Search for Google Calendar API and click on it.



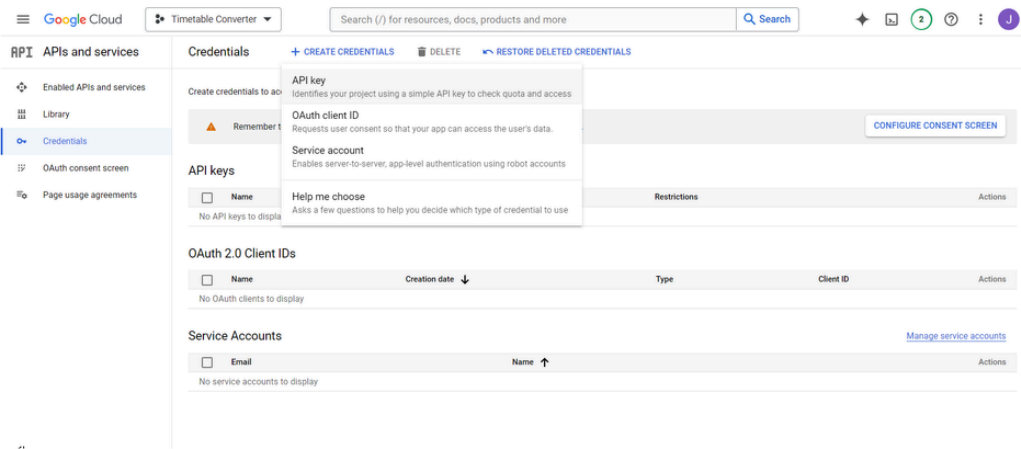
7. Click on Enable.



8. After enabled the API, you will be redirected to the API/service detail page, it will ask you to create a credential to use the service. Click on “Credentials” on the left panel.



10. Click Create Credentials, followed by Service account.



11. Type the service account name and service account ID. You may type in any name related, click Done.

1

Service account details

Service account name

timetable\_converter

Display name for this service account

Service account ID \*

timetable-converter

Email address: timetable-converter@timetable-converter-

421207.iam.gserviceaccount.com

Service account description

Describe what this service account will do

CREATE AND CONTINUE

2

Grant this service account access to the project (optional)

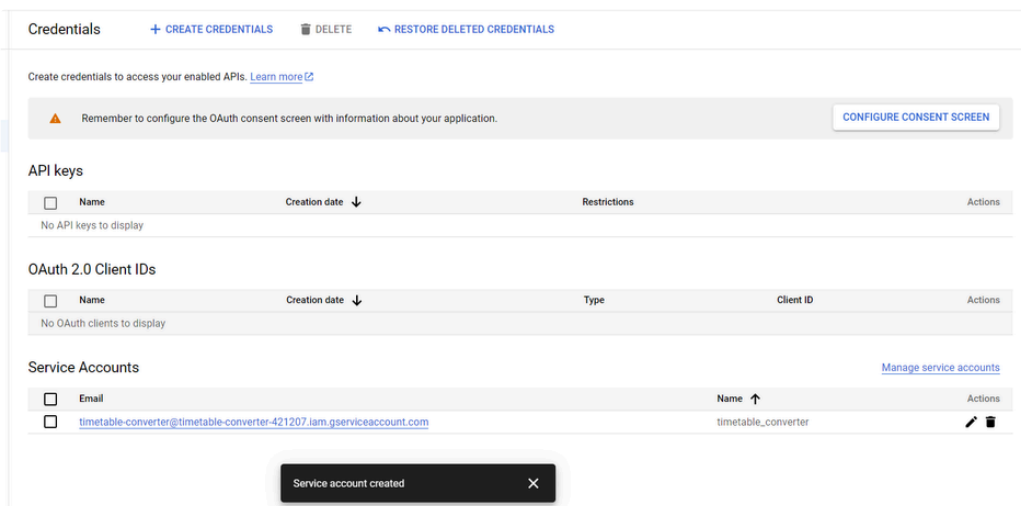
3

Grant users access to this service account (optional)

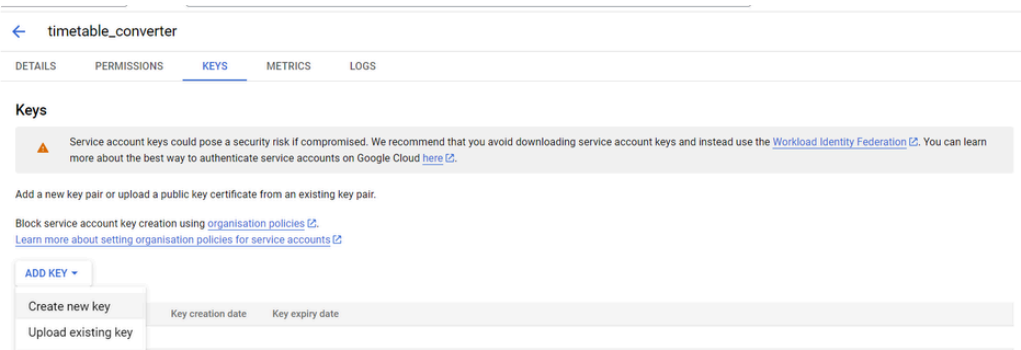
DONE

CANCEL

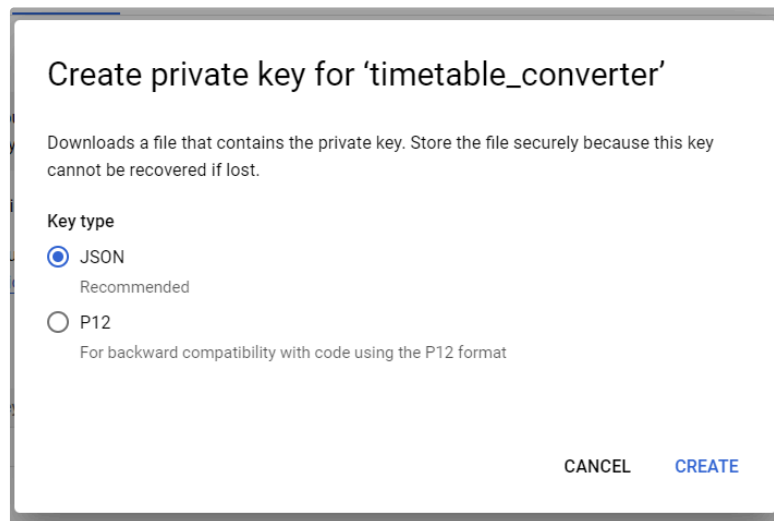
12. After the service account has been created, click on the email address of that service account.



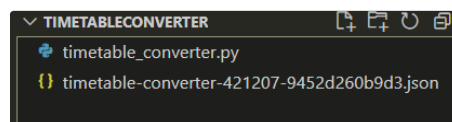
13. Click on Keys in the detail page, click Add Key, followed by Create new key.



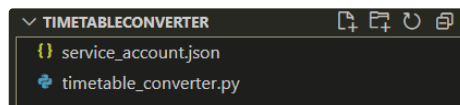
14. Select JSON and create.



15. After you create the private key, it will automatically download a json file that contains of your service account credential, move it to the same directory as your python file.

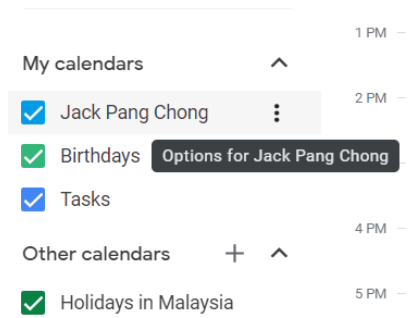


16. Rename the credential file to service\_account.json.

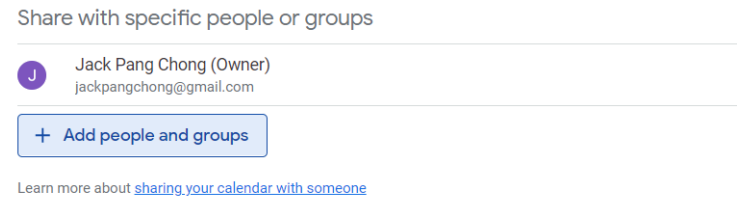


17. In your browser, go to Google Calendar: <https://calendar.google.com/calendar/>. On the left panel, under My calendars, hover to the calendar with your username and click on the three-dots button, click Settings and sharing, you will go to the setting page of your calendar.

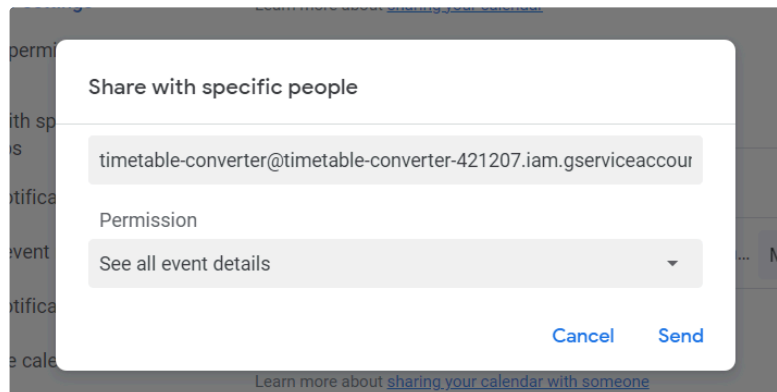




18. Under the Share with specific people or groups, Click Add people and groups.



19. Paste the service account email address you have created just now and change permission to Make changes to events, then click Send.



20. Back to your python script. Add a create\_event function to create the events (class) from the timetable\_data you have assigned just now.

We will do some transformation on the datetime in the timetable\_data as event creation required a specific format of date time. Replace the "email\_address@gmail.com" to email that you signed in to the Google Cloud Console.

```

1 def create_event(service, entry):
2
3     date_str = entry['Date']
4     time_str = entry['Time']
5
6     date = datetime.strptime(date_str, '%a, %d-%b-%Y')
7     start_time_str, end_time_str = time_str.split(' - ')
8     start_time = datetime.strptime(start_time_str, '%H:%M')
9     end_time = datetime.strptime(end_time_str, '%H:%M')
10    start_datetime = date.replace(hour = start_time.hour, minute= start_time.minute)
11    end_datetime = date.replace(hour = end_time.hour, minute= end_time.minute)
12
13    start_isoformat = start_datetime.isoformat()
14    end_isoformat = end_datetime.isoformat()
15
16    time_zone = 'Asia/Kuala_Lumpur'
  
```

```

17
18     event = {
19         'summary': f"{entry['Subject/Module']}",
20         'location': f"{entry['Classroom']}",
21         # 'description': 'A chance to hear more about Google\'s developer products.',
22         'start': {
23             'dateTime': f'{start_isoformat}',
24             'timeZone': f'{time_zone}',
25         },
26         'end': {
27             'dateTime': f'{end_isoformat}',
28             'timeZone': f'{time_zone}',
29         },
30         # 'recurrence': [
31         #     'RRULE:FREQ=DAILY;COUNT=2'
32         # ],
33         # 'attendees': [
34         #     {'email': 'lpage@example.com'},
35         #     {'email': 'sbrin@example.com'},
36         # ],
37         'reminders': {
38             'useDefault': False,
39             'overrides': [
40                 # {'method': 'email', 'minutes': 24 * 60},
41                 {'method': 'popup', 'minutes': 10},
42             ],
43         },
44     }
45
46     return service.events().insert(calendarId='email_address@gmail.com', body=event).execute()

```

18. Then, add a `get_credentials` function to allow the API calls with your service account.

```

1 from googleapiclient.discovery import build
2 from google.oauth2 import service_account
3
4 import os.path
5
6 def get_credentials(SCOPES):
7     SERVICE_ACCOUNT_FILE = os.path.join(os.path.dirname(__file__), 'service_account.json')
8     creds = service_account.Credentials.from_service_account_file(
9         SERVICE_ACCOUNT_FILE, scopes = SCOPES
10    )
11    return creds

```

19. In the main function, define the scope to use calendar with the service account, assign the credential by calling the `get_credentials` function. Then, build the service and call `create_event` function for each entry in the `timetable_data`.

```

1     SCOPES = ['https://www.googleapis.com/auth/calendar']
2     creds = get_credentials(SCOPES)
3
4     service = build('calendar', 'v3', credentials= creds)
5     for entry in timetable_data:
6         create_event(service, entry)

```

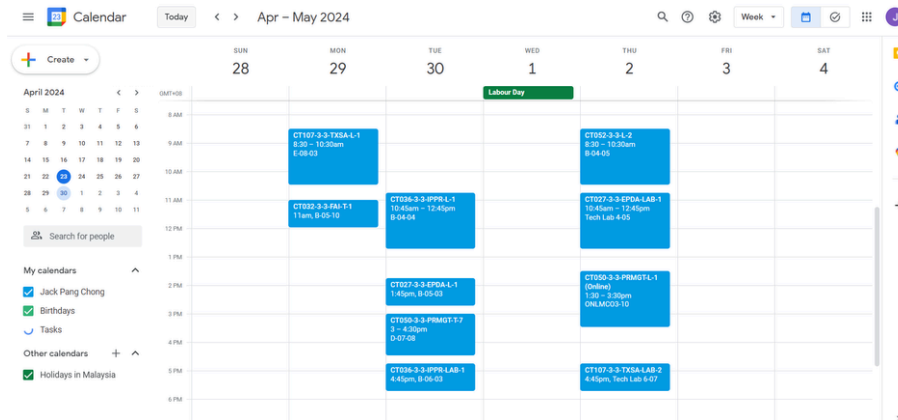
19. Now it should be able to create event in your calendar. You may try it by adding the following lines at the end of the script and run.

```

1 if __name__ == "__main__":
2     main()

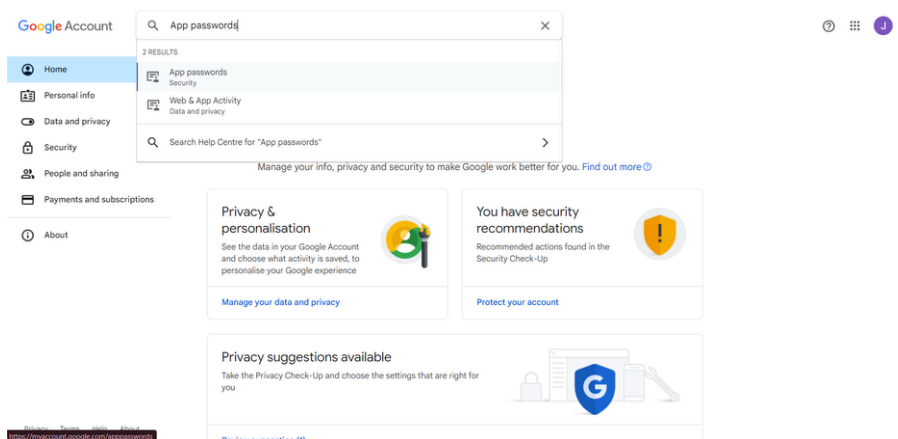
```

20. After the script finished running, there are new events created in the next week in your calendar.



## Email Automation

1. Go to your Google Account in browser: <http://myaccount.google.com>, search App passwords.



2. Enter the App name and click Create.


## ← App passwords

App passwords help you sign in to your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

### Your app passwords

Timetable-to-CalendarCreated at 31 Jan, last used at 19 Apr

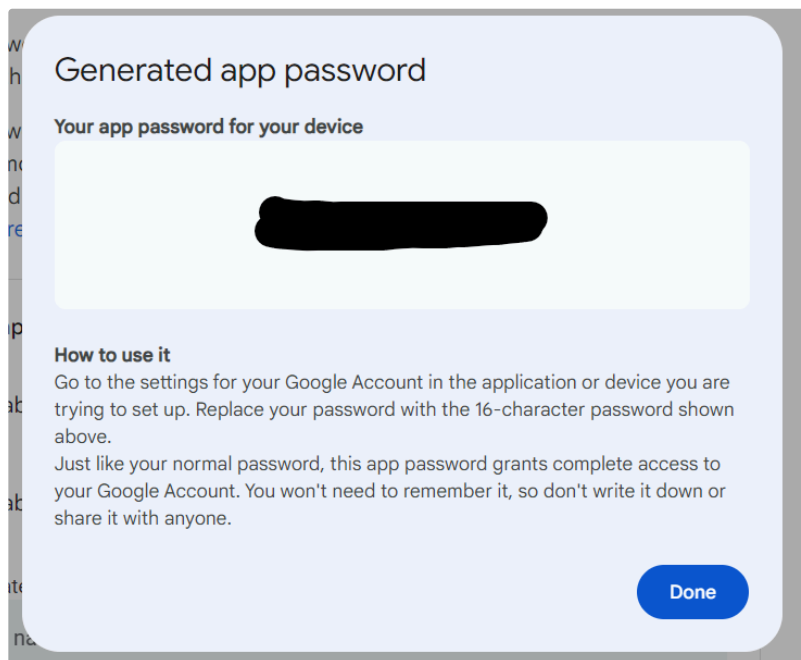
To create a new app-specific password, type a name for it below...

App name

Timetable Converter

Create

3. After it generated an 16-character password that will allow you to use Google services in the script with your Google Account. Please keep the password in somewhere secured.



4. In VSCode, create a new file named cred.env, enter the following:
  - a. Sender email: Your Google Account email address
  - b. Password: App password generated without blank space
  - c. Receiver email: Receiver email address

```

1 SENDER=<sender>@gmail.com
2 PASSWORD=<apppassword>
3 RECEIVER=<receiver>@gmail.com

```

5. Back to the python script, add lines below before the functions creation. These lines will locate and read your env file.

```

1 from dotenv import load_dotenv
2
3 current_directory = os.path.dirname(__file__)
4 env_file_path = os.path.join(current_directory, 'cred.env')
5
6 load_dotenv(env_file_path)

```

6. Create email class

```

1 import smtplib
2 import ssl
3 from email.message import EmailMessage
4
5 class Email:
6     def __init__(self):
7         self.sender = os.getenv('SENDER')
8         self.password = os.getenv('PASSWORD')
9         self.receiver = os.getenv('RECEIVER')
10        self.subject = 'Your Weekly APU Timetable is Ready!'
11
12    def send_email(self, html_content):
13        context = ssl.create_default_context()
14        em = EmailMessage()
15        em['From'] = self.sender
16        em['To'] = self.receiver
17        em['Subject'] = self.subject
18        em.set_content(html_content, subtype = 'html')
19
20        with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
21            smtp.login(self.sender, self.password)
22            smtp.sendmail(self.sender, self.receiver, em.as_string())
23

```

7. In the main function, after the for-loop for event creation, add the line to send email to your receiver.

```

1         calendar_link = f'https://calendar.google.com/calendar/u/0/r/week/{week_start.year}/{week_start.month}/{
2         html_content = f"""
3             <html>
4                 <body>
5                     <p>Hi, your APU timetable for the week of {week_start} is ready</p>
6                     <p>Please find calendar with the link below :)</p>
7                     <p>Link: {calendar_link}</p>
8                 </body>
9             </html>
10        """
11
12        email_client = Email()
13        email_client.send_email(html_content)
14        print("Export Completed!")
15    else:
16        html_content = f"""
17            <html>
18                <body>

```

```

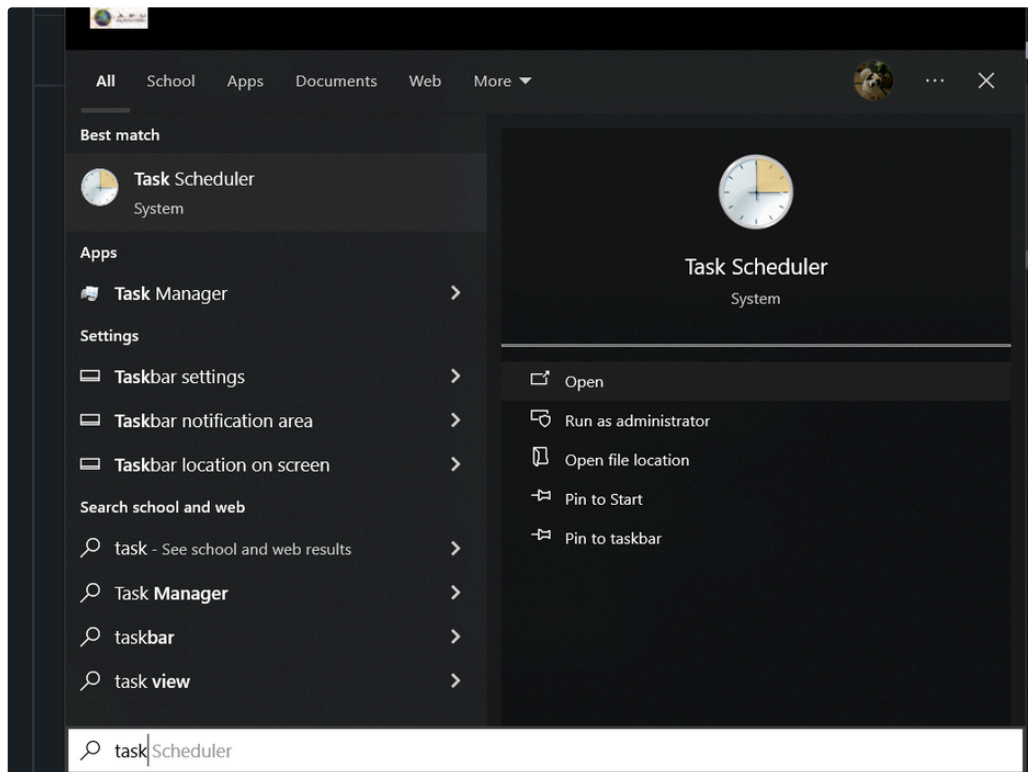
19         <p>Hi, there is no class scheduled for week {week_start}</p>
20         <p>Please enjoy your holiday :)</p>
21     </body>
22 </html>
23 """
24
25     email_client = Email()
26     email_client.send_email(html_content)
27     print("No class this week!")

```

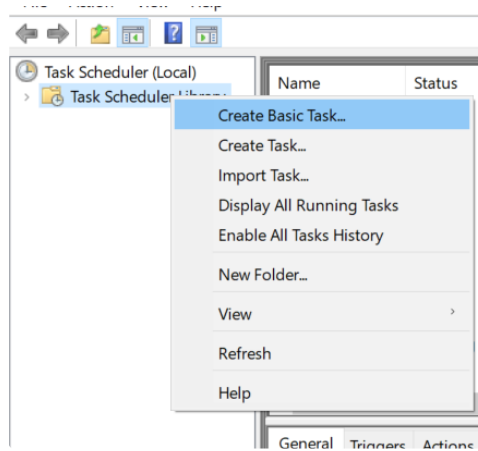
## Pipeline Automation

### Windows User

1. In Windows, search "Task Scheduler".



2. Under Task Scheduler Library, right click and select Create Basic Task.




3. Insert the name of this task.

A screenshot of the 'Create Basic Task Wizard' dialog box. The title bar says 'Create Basic Task Wizard'. The main area is titled 'Create a Basic Task'. On the left, there are four steps: 'Create a Basic Task' (selected), 'Trigger', 'Action', and 'Finish'. The 'Create a Basic Task' step is highlighted. The main area contains a text box labeled 'Name:' with the text 'Timetable Converter' entered. Below it is a larger text box labeled 'Description:'. At the bottom right, there are three buttons: '< Back', 'Next >' (highlighted), and 'Cancel'. A small icon of a clock is visible next to the 'Create a Basic Task' title.

4. Select trigger.

Create Basic Task Wizard

 Task Trigger

Create a Basic Task

Trigger

Weekly

Action

Finish

When do you want the task to start?

☐ Daily

☒ Weekly

☐ Monthly

☐ One time

☐ When the computer starts


☐ When I log on

☐ When a specific event is logged

< Back Next > Cancel

5. Select occurrence.

Create Basic Task Wizard

 Weekly

Create a Basic Task

Trigger

Weekly

Action

Finish

Start: 24/04/2024 01:27:51 ☐ Synchronize across time zones

Recur every: 1 weeks on:

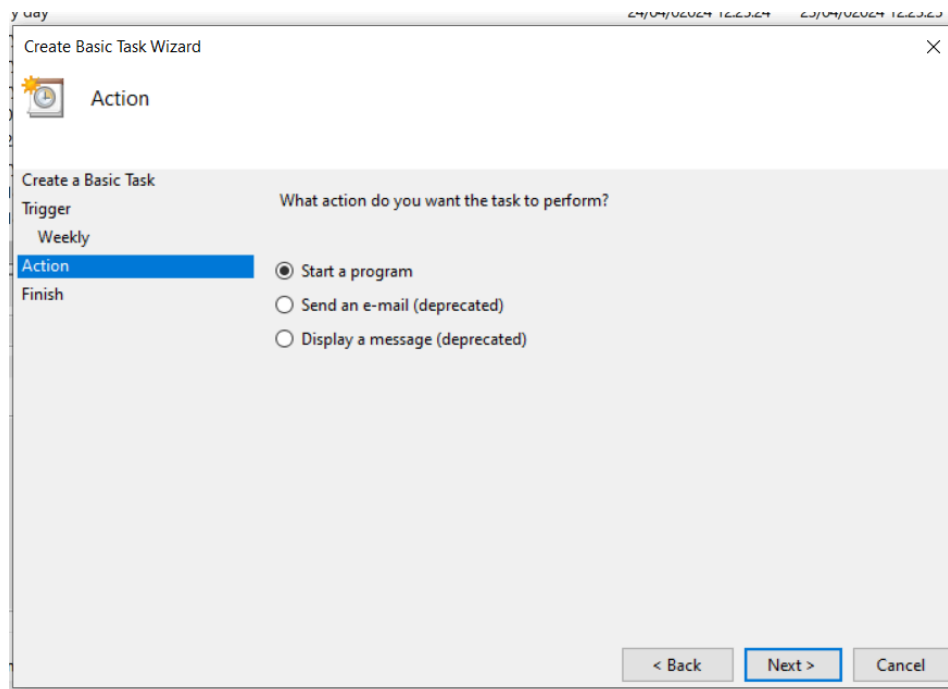
☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday

☐ Thursday ☒ Friday ☐ Saturday

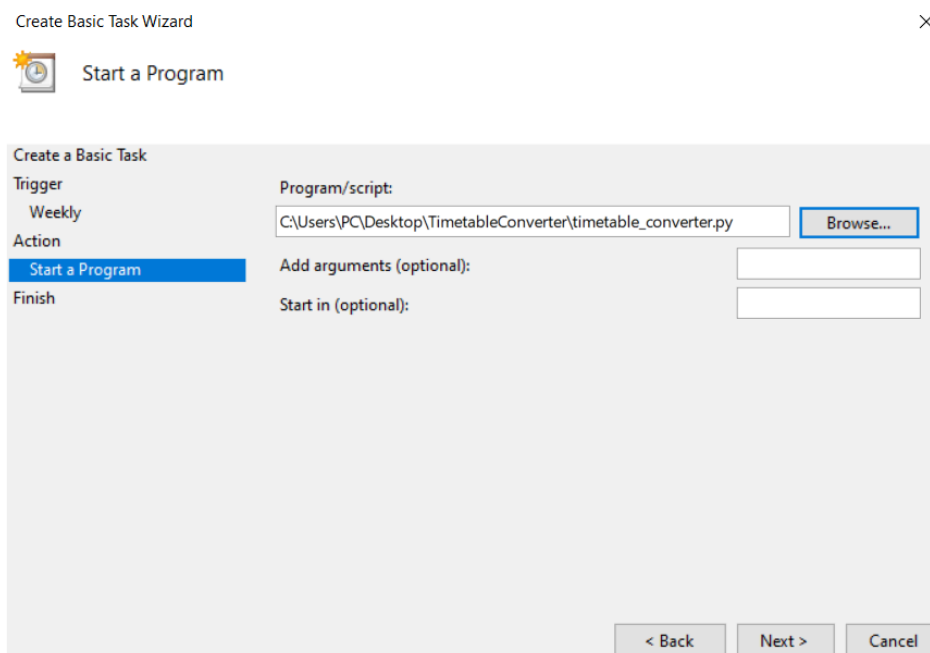
< Back Next > Cancel

6. Select Start a program.





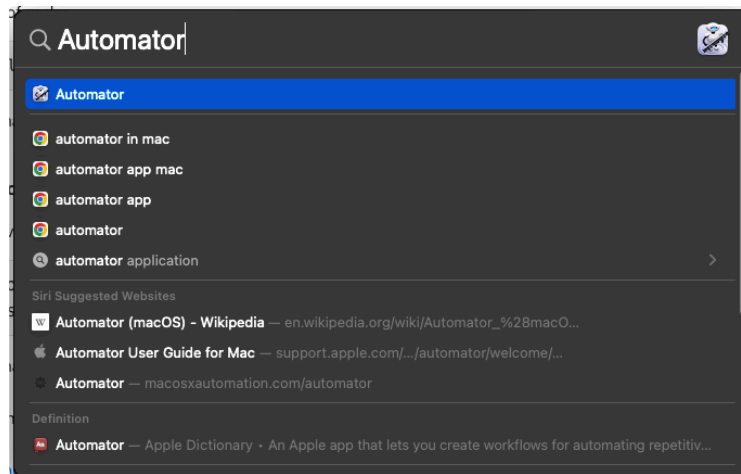
7. Browse on the location of your python file.



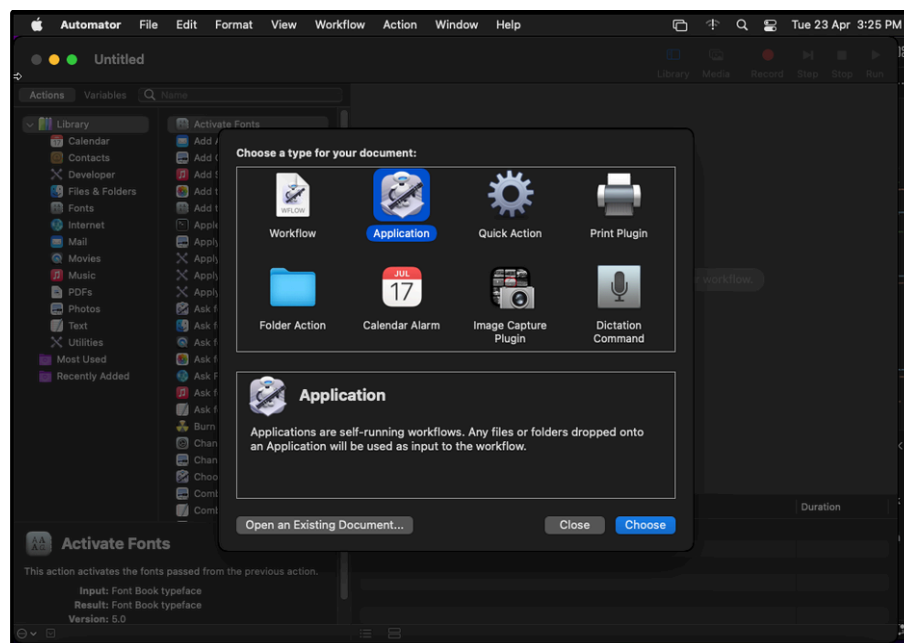
8. Click on Finish.

## MacOS User

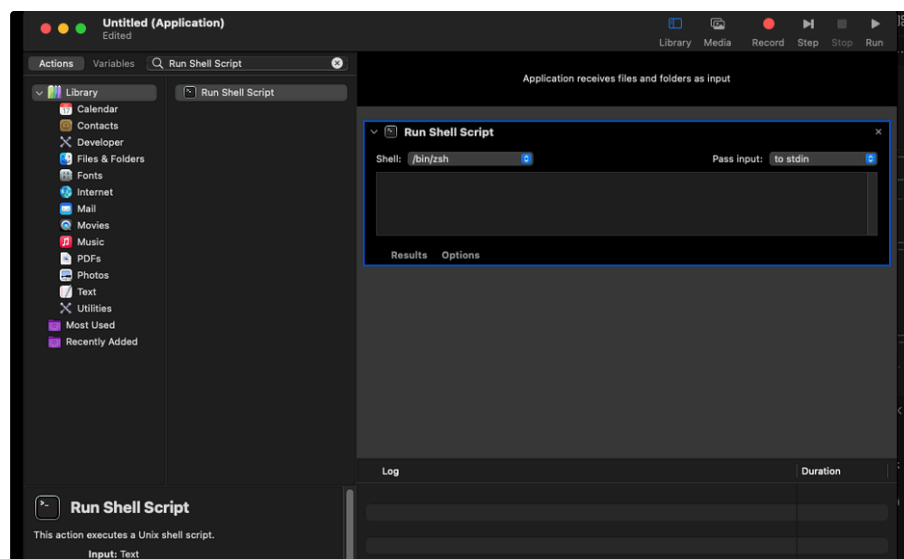
1. Spotlight search and launch Automator.



2. Select Application.



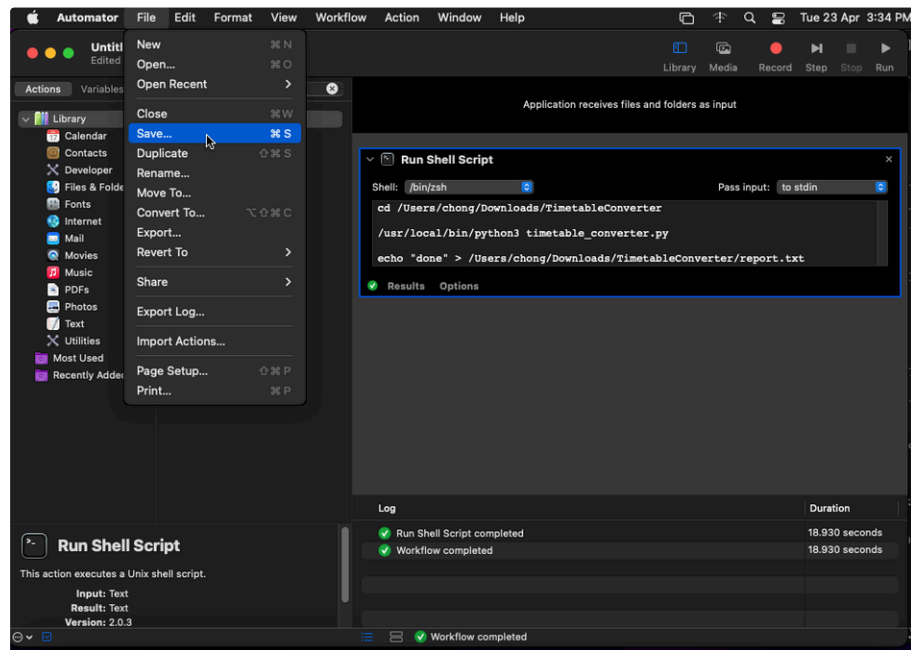
3. In the search bar, search for Run Shell Script and drag to the workflow on the right side.



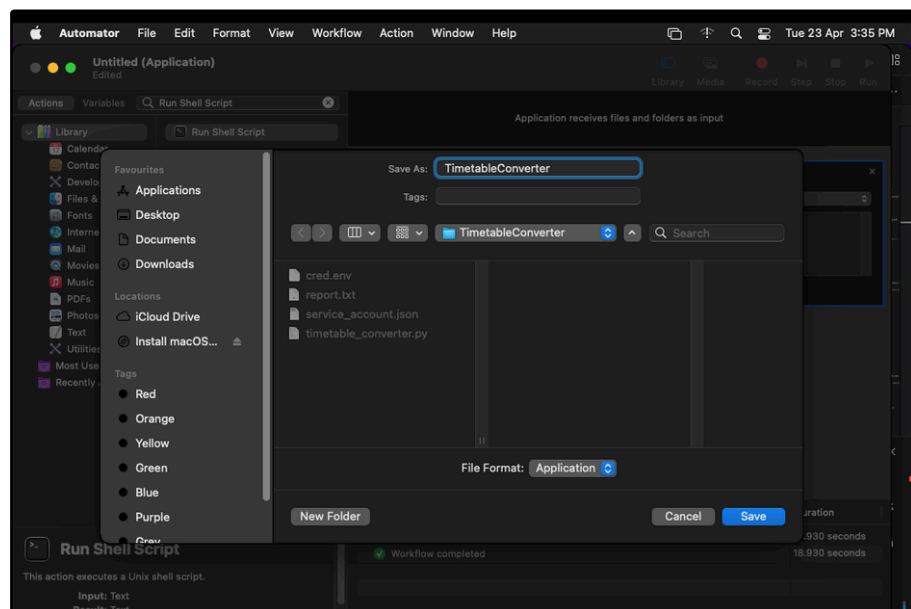
4. Insert the shell script below:

```
1 cd /<your>/<directory>/<to-project-folder>/TimetableConverter
2
3 /usr/local/bin/python3 timetable_converter.py
```

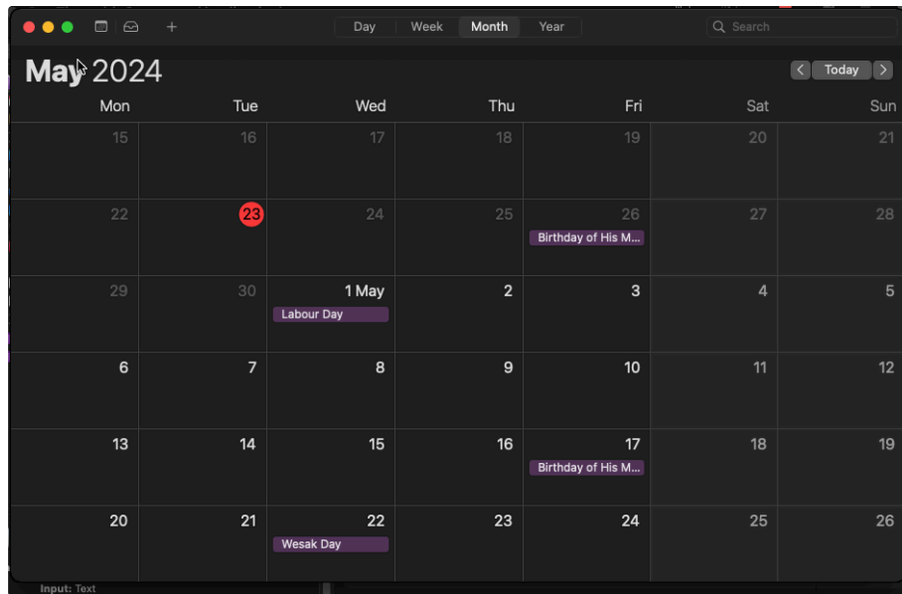
5. Click “File” on the top, followed by Save.



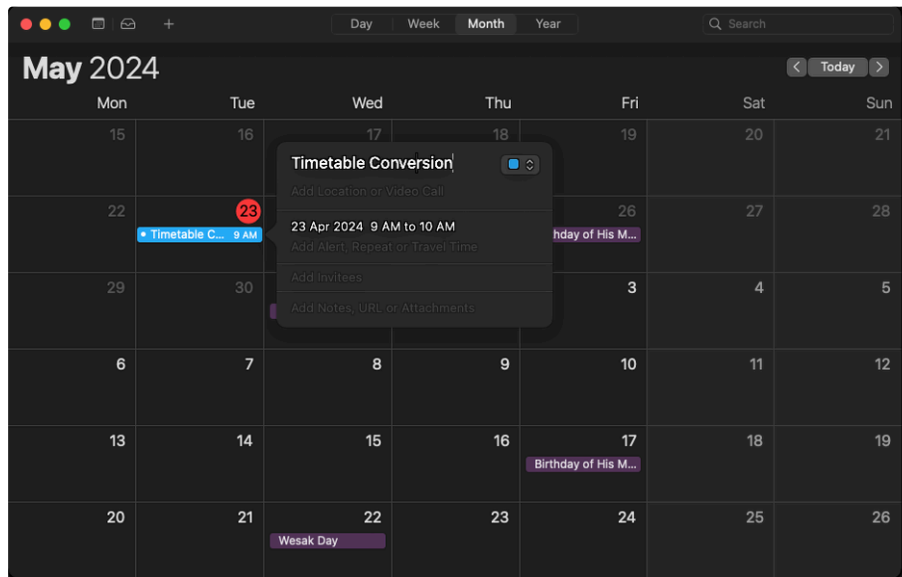
6. Name it as “TimetableConverter” and save it in the project folder.



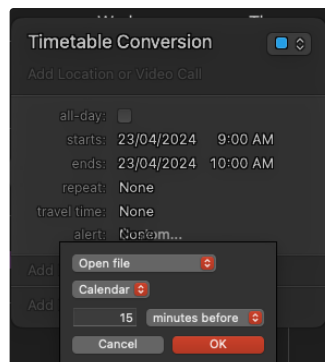
7. Open Calendar in your Mac.



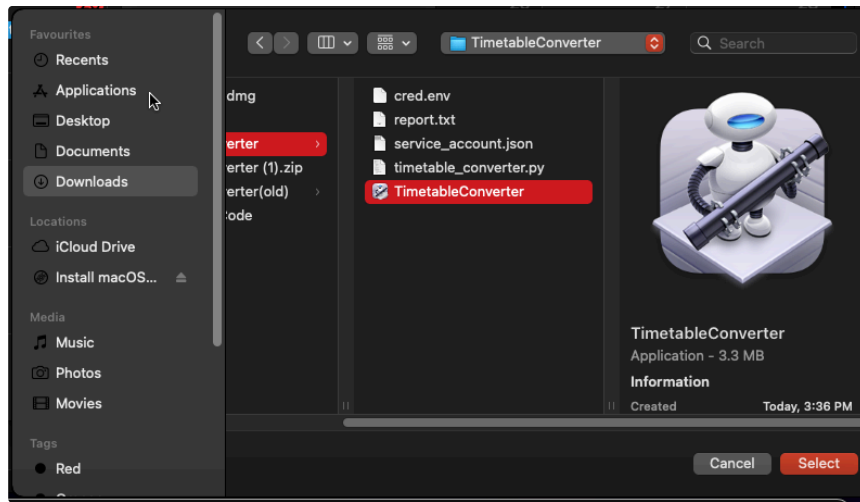
8. On the day of week you wanted to set the timetable conversion, right click and add new event, name it “Timetable Conversion”.



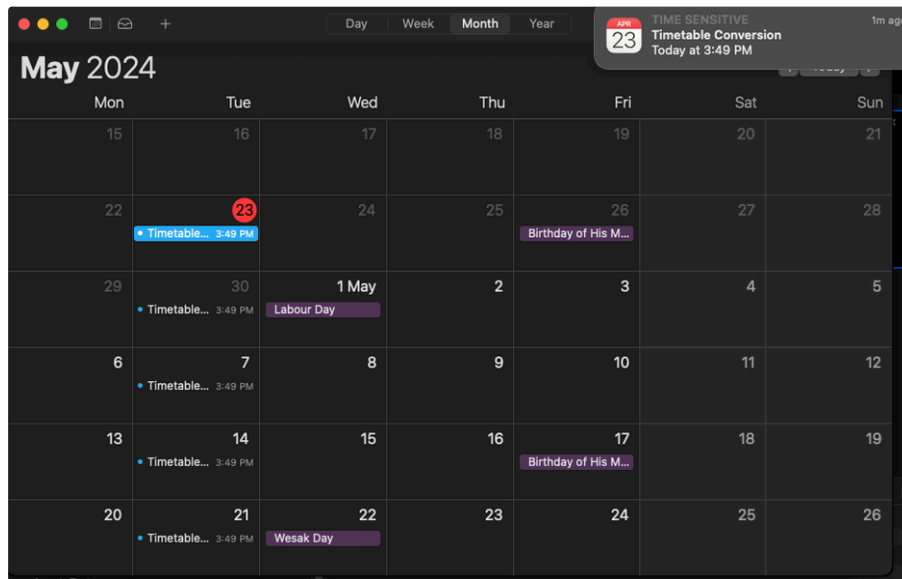
9. Click Add Alert, Repeat or Travel Time. Set repeat to every week. In the Alert, click custom and change to Open file.



10. Change Calendar to Other, and choose the application save in Automator just now.



11. Then, the application will be launched and timetable conversion will be done when the alert triggered. (For the first time triggered alert, you will need to manually click on confirm on the message to give Calendar access to launch the application)



NOTE: Please keep your computer on and with internet connection when the scheduled task is triggered, otherwise the trigger for the schedule will be missed.