**PTO-Travel Automation for ICT Department**

**ICT**

**08/08/2023**

Contents

[Purpose: 3](#_Toc142537645)

[Prerequisites: 3](#_Toc142537646)

[Process Workflow 4](#_Toc142537647)

[Auth Workflow 5](#_Toc142537648)

[Script Setup and Logic 10](#_Toc142537649)

[2.1) Data Collection Logic (JERA\_data\_processing.py) 11](#_Toc142537650)

[2.2) Data Processing Logic (JERA\_events\_processing.py) 13](#_Toc142537651)

[2.3) Email Collection and Sending Logic (JERA\_email.py) 14](#_Toc142537652)

[Azure Function App Setup 16](#_Toc142537653)

[Setting Up the Local Environment 17](#_Toc142537654)

[How to Modify the Script to Fit Your Needs 19](#_Toc142537655)

[Change the sender of the email 19](#_Toc142537656)

[Change the recipients of the email 19](#_Toc142537657)

[Change which Teams the script auto collects emails from, but do NOT change which channel and Teams the events data is collected from 20](#_Toc142537658)

[Change where the recipients of the email is being collected from AND change where the events data is being collected from 20](#_Toc142537659)

[Change the subject of the email 20](#_Toc142537660)

[Change the body of the email 21](#_Toc142537661)

[Change the title of the table 22](#_Toc142537662)

[Change the table’s column names 22](#_Toc142537663)

[Change where the data is being collected from (changing the Microsoft Teams team) 22](#_Toc142537664)

[Troubleshooting 24](#_Toc142537665)

[Error Handling 24](#_Toc142537666)

[Maintenance (IMPORTANT) 24](#_Toc142537667)

[Future Steps 26](#_Toc142537668)

[Contact Creator 26](#_Toc142537669)

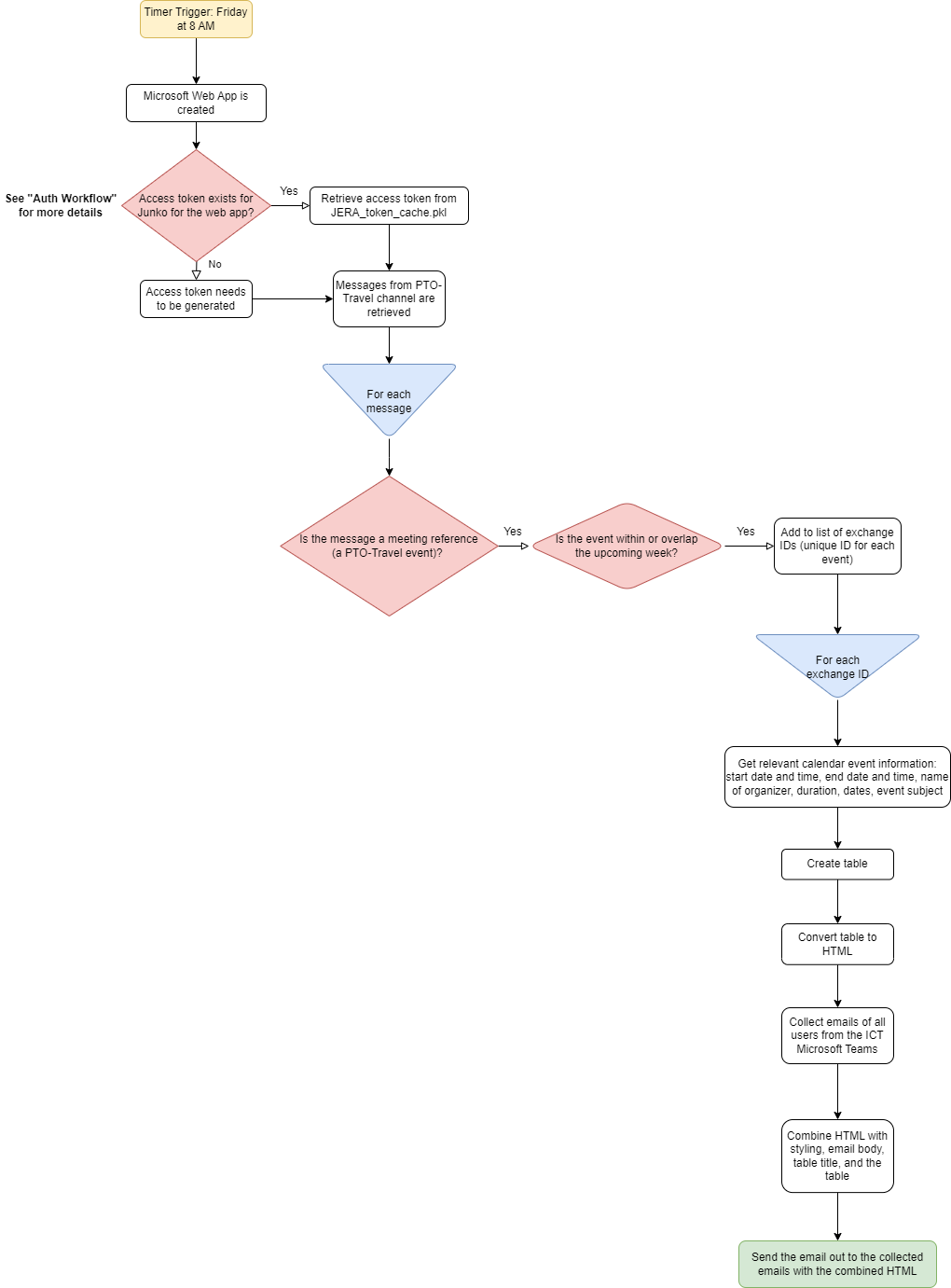
# Purpose:

The document covers the logic and tools behind the PTO-Travel email that is sent to the ICT department every week so that other people can understand and potentially expand on this project.

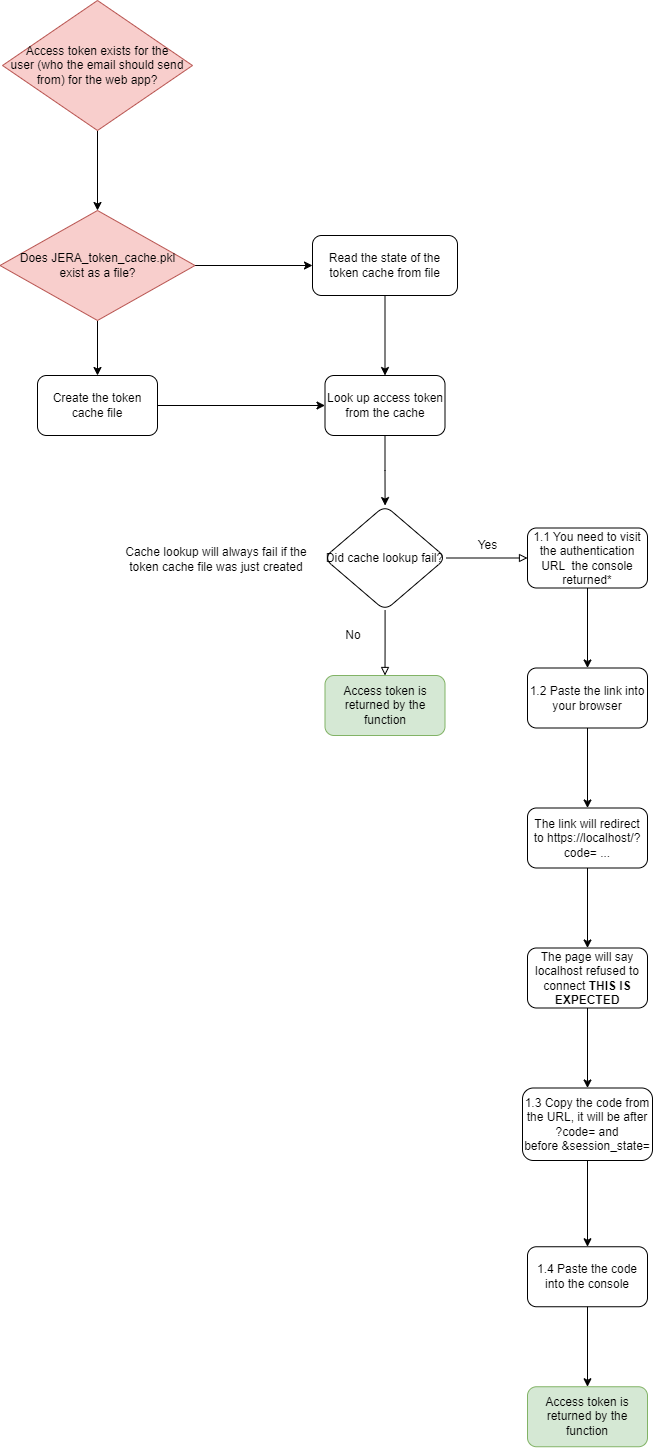
# Prerequisites:

* Understanding of Python
* Understanding of Graph API and Web Apps
* Ability to use Visual Studio Code
* Understanding of Powershell/Terminal
* Understanding of Azure App and Azure Function App

# Process Workflow



# Auth Workflow

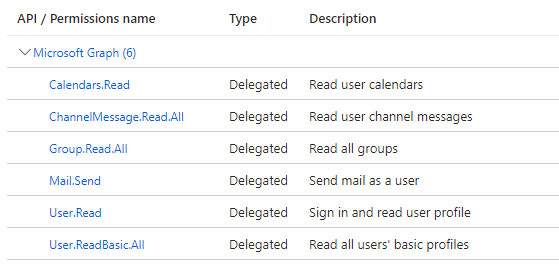


This page details the authentication workflow process. The user who the email should send from should complete this process.

The user should go through this process only if the access token has expired\* or if they want a different user to be the sender of the automated email.

\*The access token will never expire as long as the refresh token in the cache remains valid and is periodically used within its lifespan. However, certain conditions, such as the user changing their password or revoking app permissions, may require reauthentication.

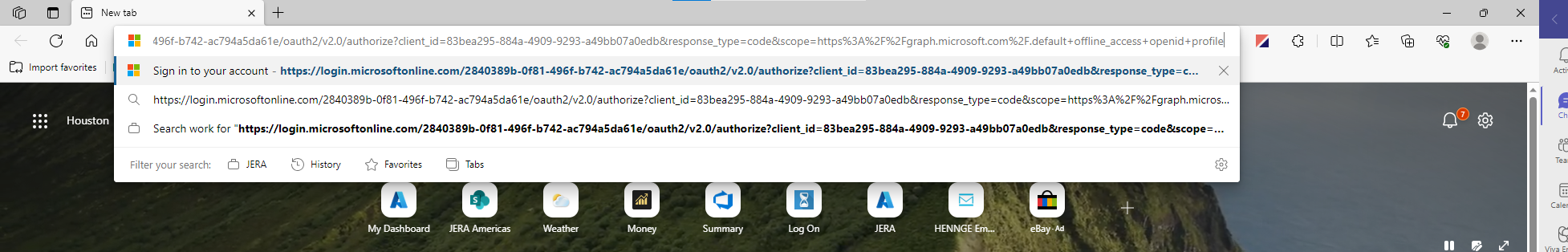
This process grants the web app the permissions to do the following on behalf of the user:

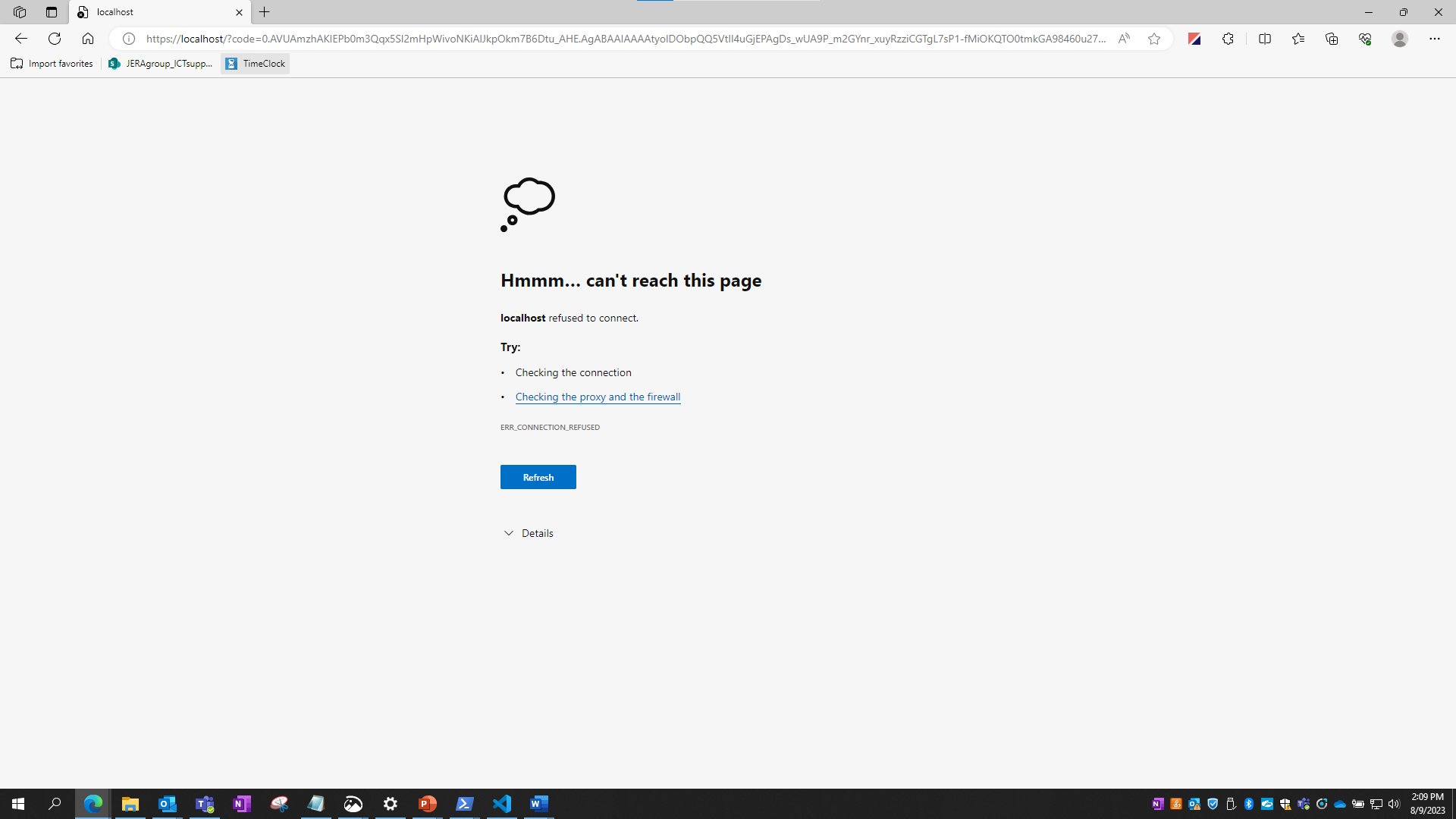


First, run the JERA\_main.py file locally in Visual Studio Code. To setup this local environment, check the Setting Up the Local Environment section.

* 1. The link will look something like:  https://login.microsoftonline.com/2840389b-0f81-496f-b742-ac794a5da61e/oauth2/v2.0/authorize?client\_id=83bea295-884a-4909-9293-a49bb07a0edb&response\_type=code&scope=https%3A%2F%2Fgraph.microsoft.com%2F.default+offline\_access+openid+profile

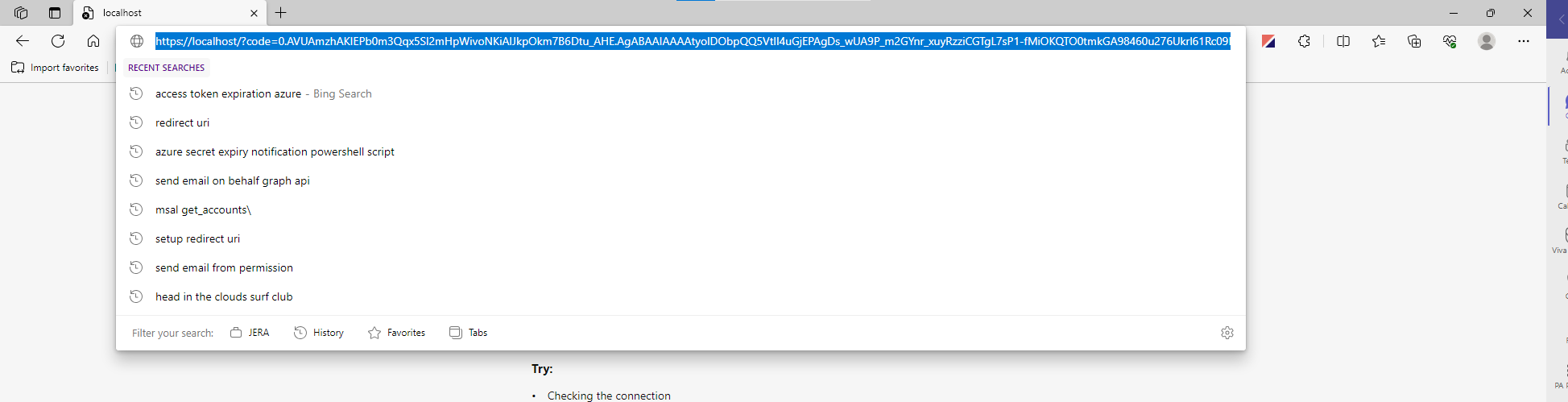


* 1. Paste the link into browser (or click on the link in the console)

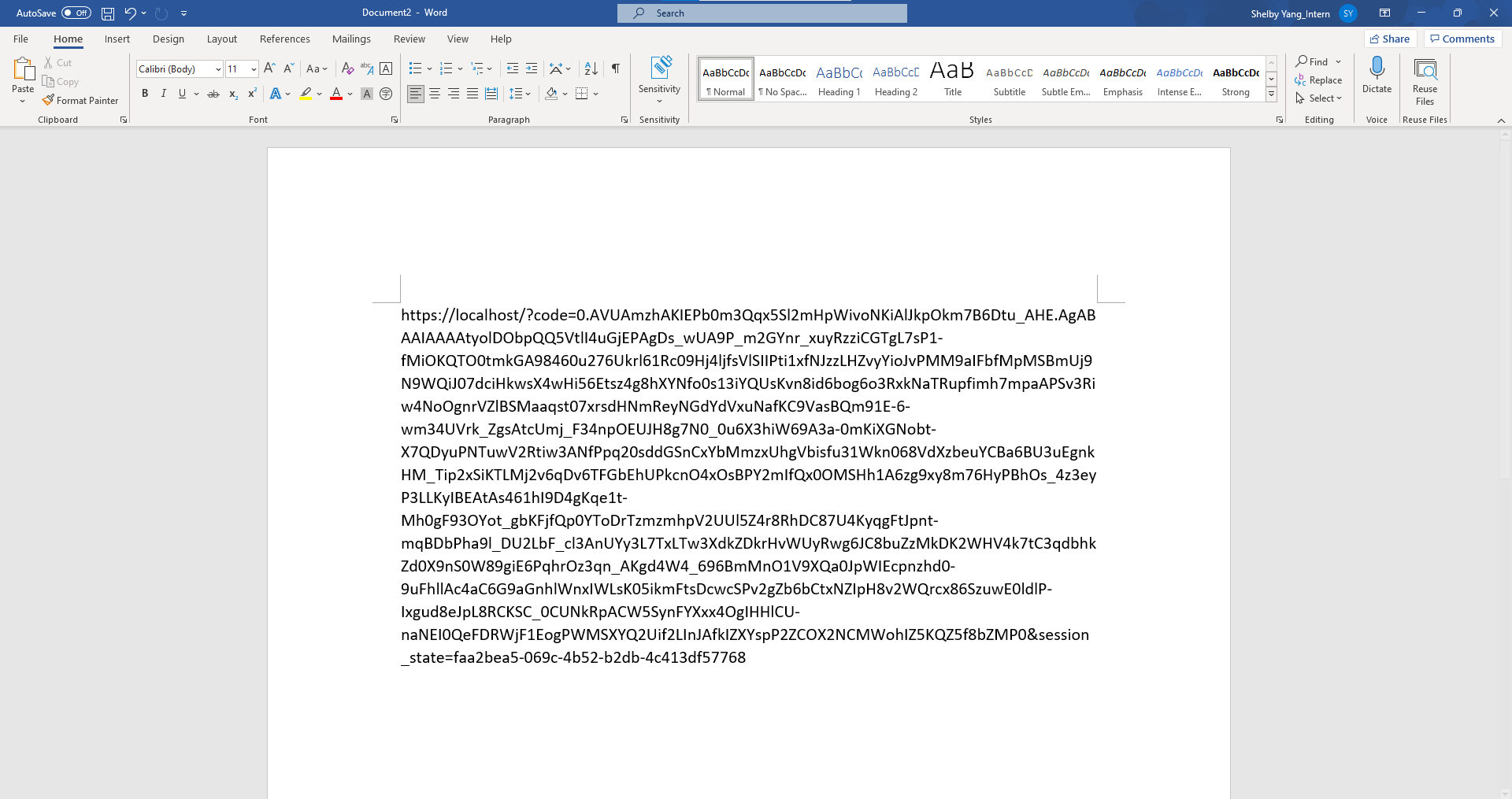


You will see this page. This is expected.

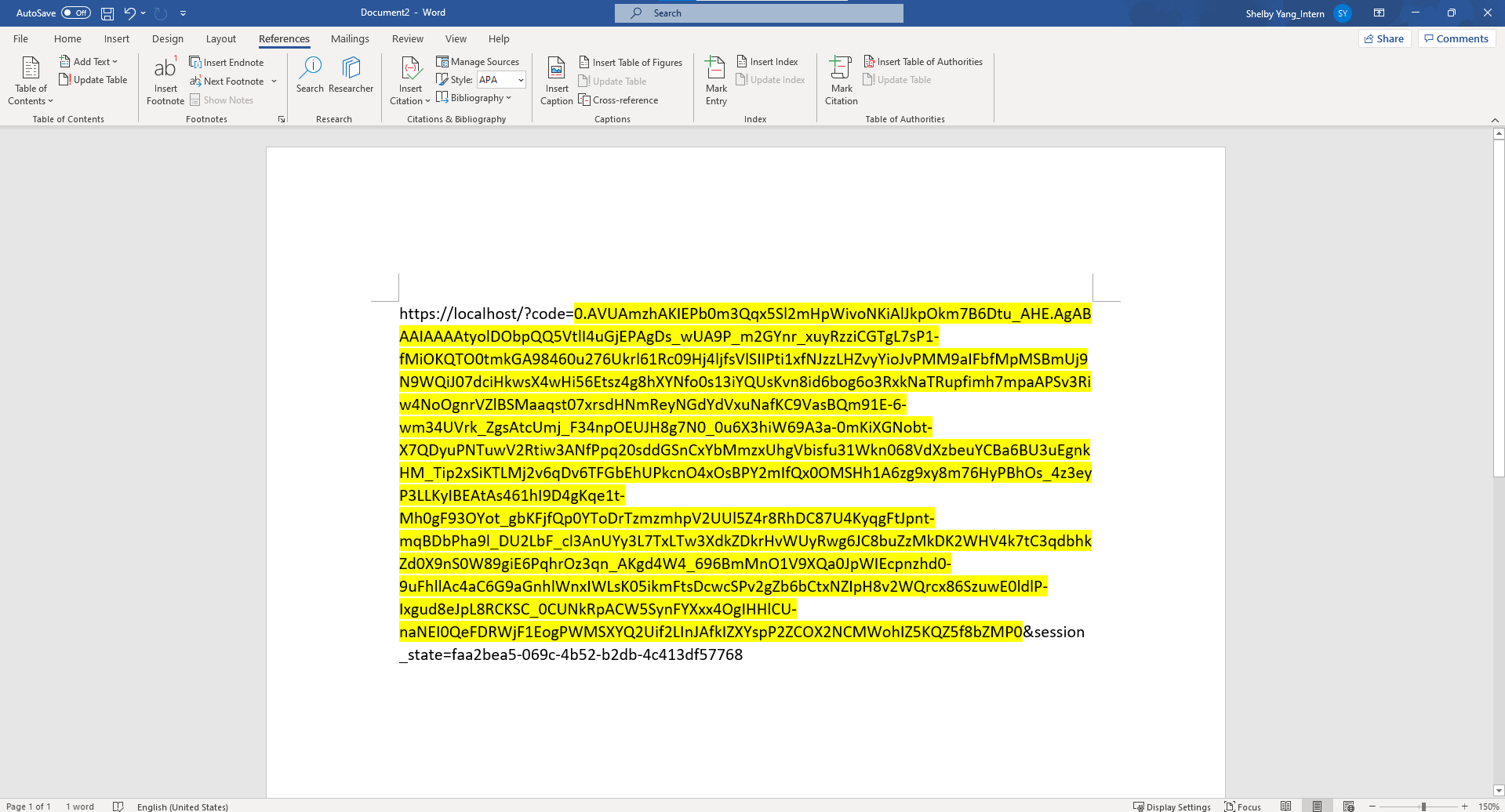
* 1. Highlight and copy the entire <https://localhost/>... URL in the browser



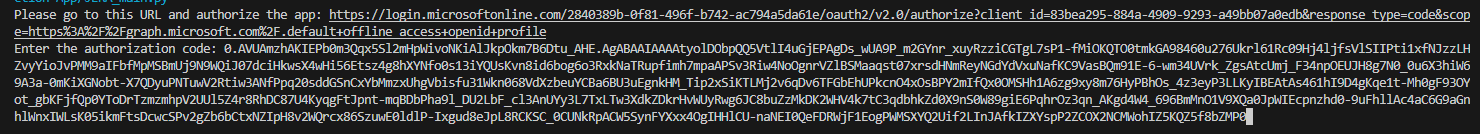
Open a text editor that will make it easier to copy part of the URL for the code and paste the URL as text



Now, copy the part right AFTER ?code= but before &session\_state… as displayed by the image below:



* 1. Now paste the code into the console



And hit enter

The program will now run and send the email as the user you are logged in as.

Notes:

* You can’t do this process from Azure Function App to my knowledge. You’ll have to run the program manually in a local environment. I detail this process in the Setting Up the Local Environment section.
* This process is already setup to send on behalf of Junko’s email, so there is no need to do go through this authentication process unless:

1. You want to send the email as someone else, in which case that person would need to go through this process.
2. The access token and refresh token have expired so you see the “Please go to this URL and authorize the app” in the console of the script.
3. The user's credentials associated with the token (in this case, Junko’s) have changed, such as a password reset or update.
4. The application's permissions in Azure Active Directory were modified or revoked, which would invalidate any previously granted tokens.
5. The token cache file (JERA\_token\_cache.pkl) is accidentally deleted or corrupted.

# Script Setup and Logic

These are the script files:

* JERA\_main.py
* JERA\_auth.py
* JERA\_data\_processing.py
* JERA\_events\_processing.py
* JERA\_email.py

These are helper files for the script:

* JERA\_config.py
* JERA\_utils.py
* JERA\_token\_cache.pkl

JERA\_auth.py contains the logic for creating the web app and creating or getting the access token.

JERA\_data\_processing.py contains the logic for obtaining all the messages in the Teams channel, subsequently obtaining all exchange IDs\*, then subsequently obtaining all relevant calendar information about each exchange ID (start date, end date, organizer, event subject).

\*Exchange ID is a unique ID generated by Microsoft for Teams meetings (events)

JERA\_events\_processing.py contains the logic for filtering the data collected by JERA\_data\_processing.py to be put into a table.

JERA\_email.py contains the email body and styling along with the logic for collecting the emails of all users in the IT Teams and sending the email.

JERA\_main.py is the main driver of the script, it calls methods from JERA\_auth, JERA\_data\_processing, JERA\_events\_processing, and JERA\_email. It invokes all the logic of the script.

JERA\_config.py contains the necessary IDs to authenticate the web app and identify the IT Teams and PTO-Travel channel.

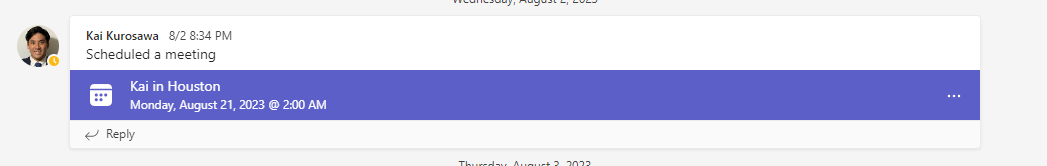
JERA\_utils.py contains logic for calculating the datetime of the start of the next week (the next Monday) and the end of the next week (the next Sunday) and splitting a full name into it’s first and last name.

JERA\_token\_cache.pkl, as detailed by the Auth Workflow section, holds the relevant access token and refresh token information.

So data collection and processing and then formatting into an email are the core functions of my script. The flow process is detailed by the Process Flow section image.

## 2.1) Data Collection Logic (JERA\_data\_processing.py)

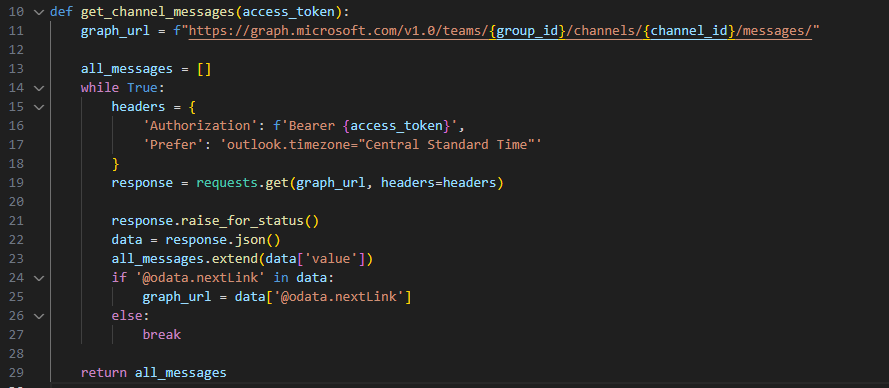
PTO-Travel events are just messages in the Teams channel which are meetings.

e.g. 

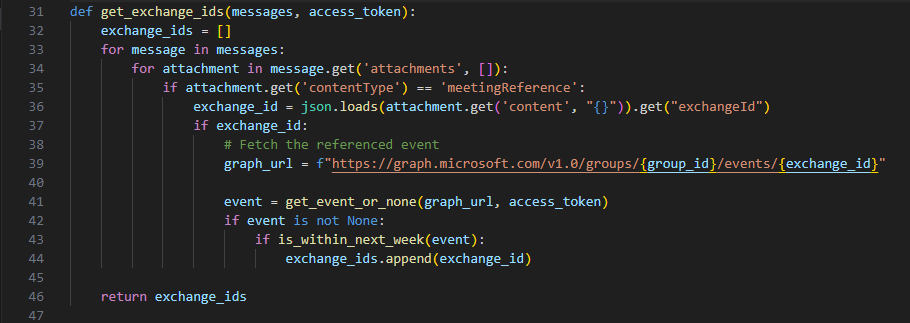
So, by filtering for messages that are meetings, we can collect all PTO-Travel events.

First, JERA\_main() makes a call to get all channel messages

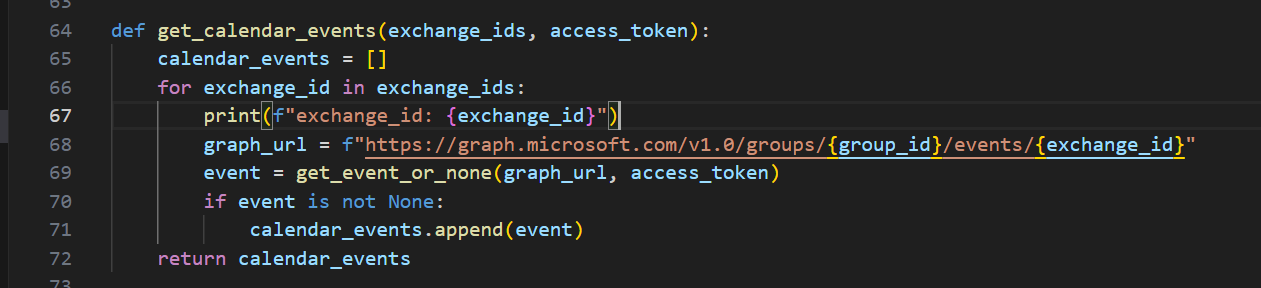
To collect messages, my script makes a GET request to https://graph.microsoft.com/v1.0/teams/{group\_id}/channels/{channel\_id}/messages/ which returns all messages from a specific channel in a specific Microsoft Teams team. Each message is represented in JSON format with information like message ID, messageType, attachments, etc. But, the information we care about the most is messageType and exchangeID.



After collecting all messages, the exchange IDs are collected by checking if the message is a meetingReference, if so, then it has to be a PTO Travel event (because all meetings in the PTO-Travel channel are PTO Travel events). It then checks if the event associated with the exchange ID is within the upcoming week, if so, then it adds it to the list of exchange IDs.

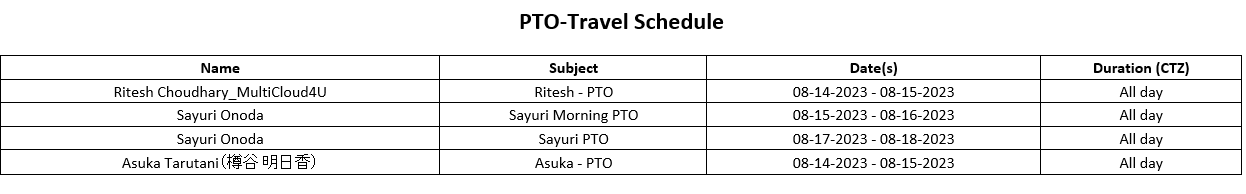


It then takes those list of exchange IDs and make a request to https://graph.microsoft.com/v1.0/groups/{group\_id}/events/{exchange\_id} which returns data about the event in a JSON format. The data includes the event ID (exchange ID), the createdDateTime, lastModifiedDateTime, subject of the event, whether it is all day, the start time and end time, name of organizers and attendees, etc. I add each of these data about each event to a list.

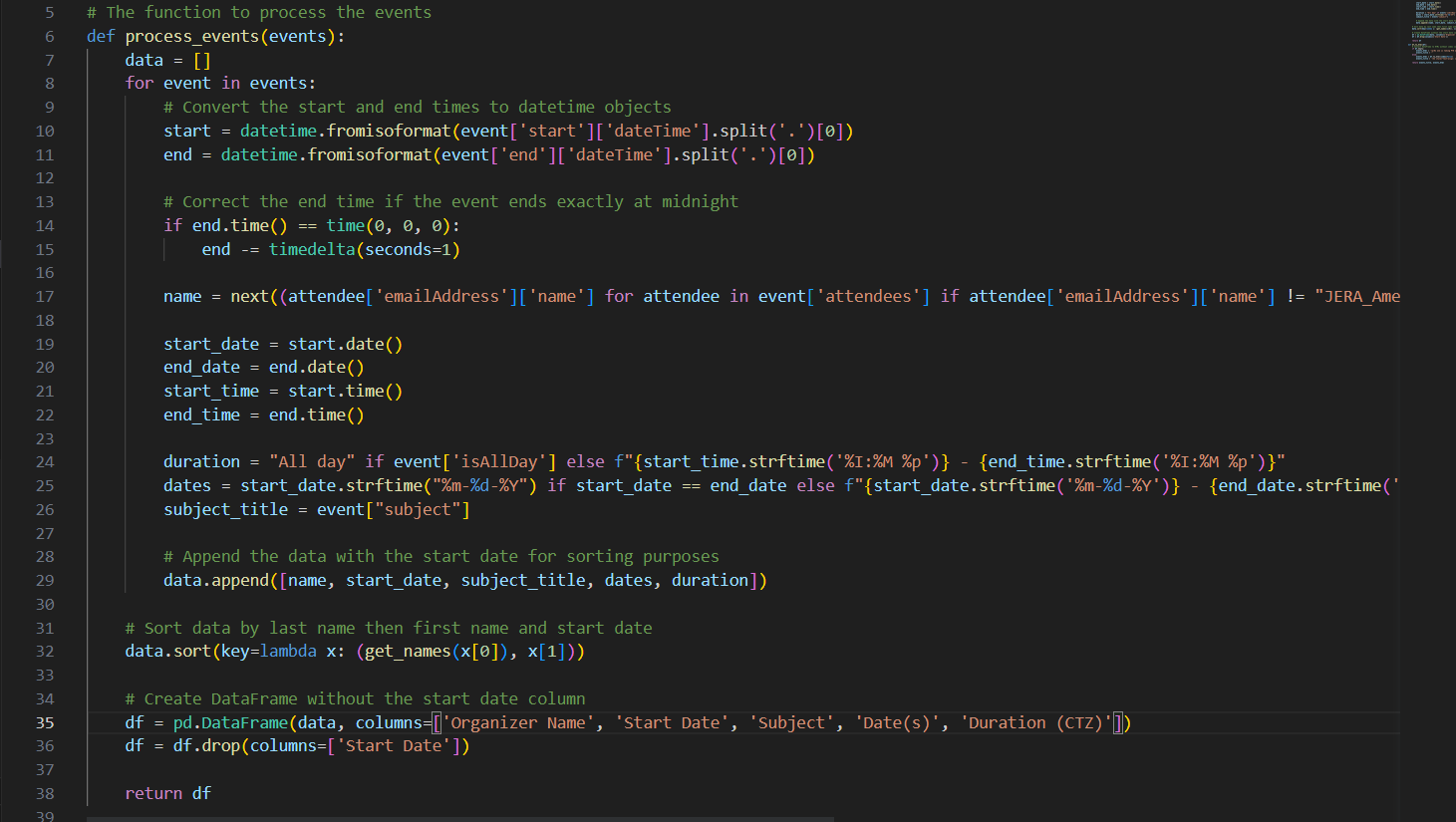


## 2.2) Data Processing Logic (JERA\_events\_processing.py)

Now, with all of the event data, JERA\_main() makes a call to filter the data for these events and create a dataframe (a table). This table is used in the email for the table you see here:



The process\_events() function takes the list of events and filters for the start date, end date, name of the organizer, the duration of the event (all day or a specific time range), and the subject of each event.



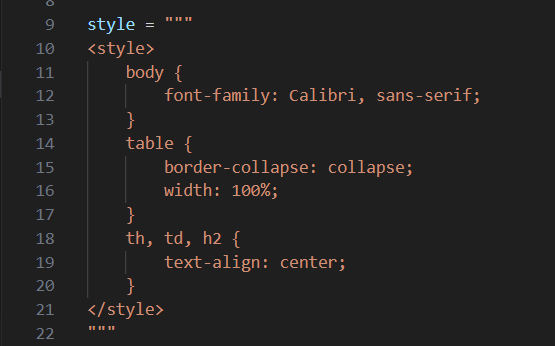
This table is then converted into HTML to be input into the email.

To change the name of the columns, modify the parameters in this line:

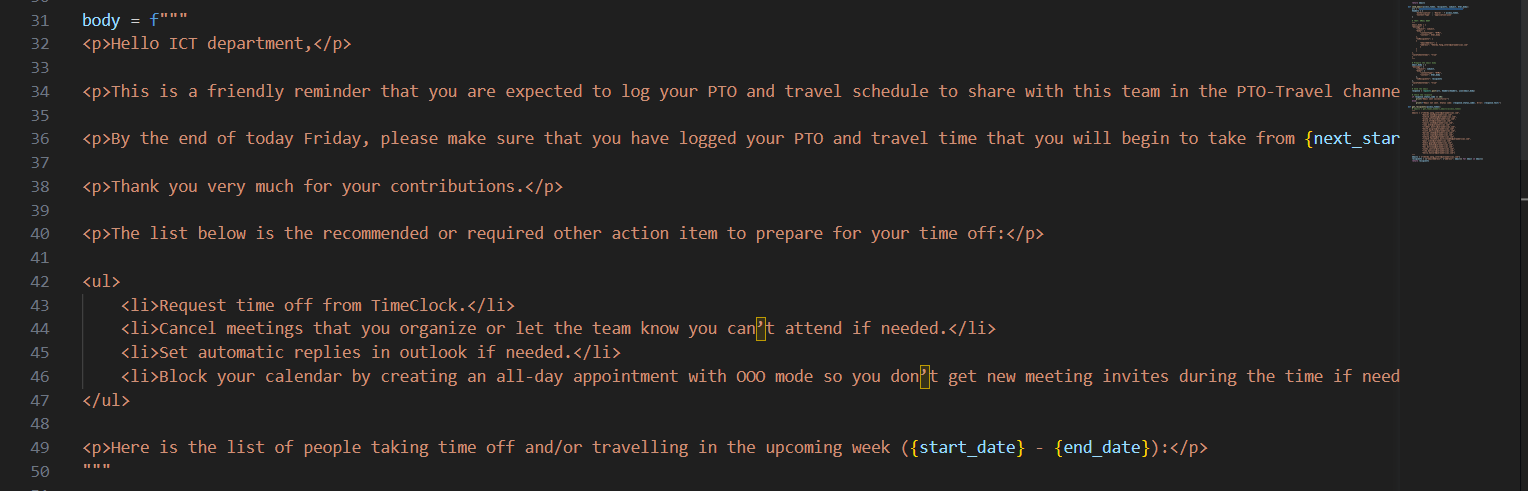


## 2.3) Email Collection and Sending Logic (JERA\_email.py)

JERA\_main lastly collects all the recipient emails from the Microsoft Teams team and sends them a formatted email with styling, the email body, and the events schedule table.

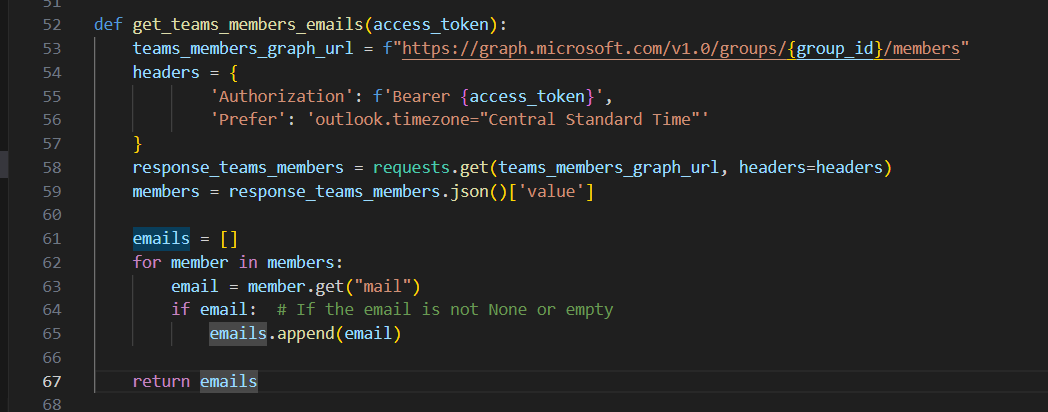


Styling with CSS



HTML email body

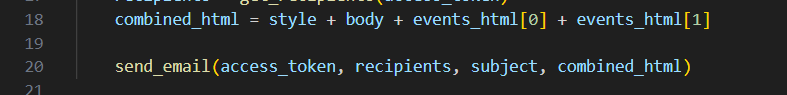
To collect all the emails, a GET request is made to https://graph.microsoft.com/v1.0/groups/{group\_id}/members and for each member returned, I add the email in the “email” field to a list of emails.



Then, a POST request is made to https://graph.microsoft.com/v1.0/me/sendMail send the email to the recipients collected.



The styling + email body + title of the table + the table are then combined and put into the email and sent.



# Azure Function App Setup

These are the all the files in the function app (usprpowermodelfunctionapp01):

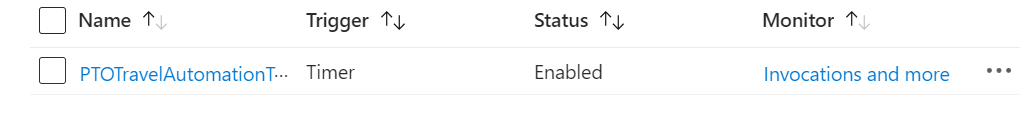
* .funcignore
* .ostype
* JERA\_auth.py
* JERA\_config.py
* JERA\_data\_processing.py
* JERA\_email.py
* JERA\_events\_processing.py
* JERA\_main.py
* JERA\_token\_cache.pkl
* JERA\_utils.py
* function\_app.py
* host.json
* oryx-manifest.toml
* requirements.txt

The JERA\_ … python files and token cache are all files for the script.

function\_app.py contains the timer trigger and will run the script by calling JERA\_main() based on the cron expression "0 0 13 \* \* 5" which runs the script every Friday at 1 PM UTC (or 8 AM central time).

requirements.txt contains the names of the Python libraries needed to be installed for my script to run because my script is dependent on these external libraries. The function app installs all of these libraries at run time.

.funcignore details which files the function app should ignore when deploying (so those files won’t be included in deployment).

My script is ran by the PTOTravelAutomationTrigger function. 

To start the function locally and test the trigger, type func start and hit enter in the terminal.

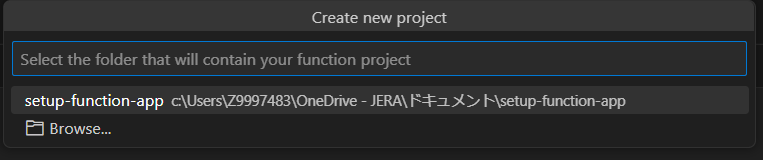
## Setting Up the Local Environment

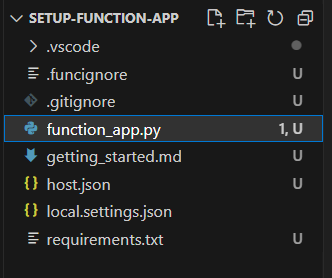
Software you will need:

1. Visual Studio Code
   1. Azure Functions extension (or Azure Tools extension)
2. Azure Command Line Interface
3. Azure Function Core Tools
4. Python

These steps are getting a copy of the code files on your local system:

1. Create a new empty folder
2. Open the folder in Visual Studio Code
3. Click on the Azure icon on the left hand side of Visual Studio Code
4. If you are not logged in, you will be prompted to login with the GUI, or, in Powershell, type in the command “az login” and you will be redirected to login there and then refresh
5. On the “workspace” tab, click on the icon with the yellow electricity and blue accents on the side
6. Click “create function”
7. Click the folder you are working in



1. Click Python (Programming Model V2)
2. Click the Python interpreter (will be showing the path of where you installed Python and the version of Python installed). If no Python interpreter shows up, click skip virtual environment.
3. Click “Timer Trigger”
4. Name the Timer Trigger “PTOTravelAutomationTrigger” to match the current trigger name (this could technically be anything though)
5. Set the Cron expression to be: 0 0 13 \* \* 5
   1. This indicates Friday at 1 PM UTC (8 AM Central)
   2. This can be changed anytime in function\_app.py
6. Click on the icon with two files on the left side bar to get back to your local files
7. You will now see: 
8. Now, visit the Function App and click on App Files
9. Replace the contents of function\_app.py and requirements.txt in your local folder with the contents of the according files in the function app
10. Create files for JERA\_auth.py, JERA\_config.py, JERA\_data\_processing.py, JERA\_email.py, JERA\_events\_processing.py, JERA\_main.py, JERA\_token\_cache.pkl, and JERA\_utils.py in your local folder
11. Copy paste the contents of those files on the function app into their according files you just created locally
12. No other files need to be copied over locally, you can delete getting\_started.md if you want (optional)
13. Now you can edit the code in this local environment!

Now, with your local environment setup, these steps will detail **how you can deploy your changes of the code onto the function app**:

1. On the left hand bar in Visual Studio Code, find the workspaces bar
2. Click on the icon with the yellow electricity and blue accents
3. Then, click on “Deploy to Function App”
4. Select “Standard Applications-US”
5. Select “usprpowermodelfunctionapp01” or whatever the name of the function app you are working on is
6. You will then receive a pop up that asks if you are sure as it will overwrite the files in the current function app
7. Click “deploy”
8. The script will begin deploying and you can check it’s status in the output window of your console
   1. If you face an error like (intermediate value).parsedBody.map is not a function, restart Visual Studio Code and try to deploy again. Typically after a restart, it will deploy successfully, if not try again and again, if the problem persists, then there may be an issue on Microsoft’s side.
   2. Generally, errors with deployment are resolved by deploying again or restarting visual studio code and trying to deploy again
9. Once you see that deployment is successful, the script is now uploaded onto the function app and should run with every trigger

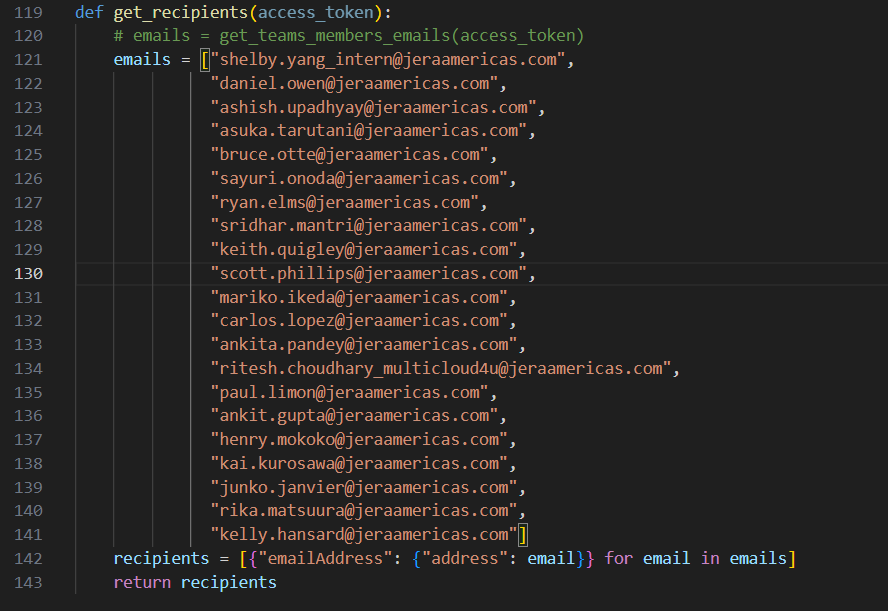
# How to Modify the Script to Fit Your Needs

If you want to change the sender, recipients, body of the email, or where the script gathers data from, then this page will help you.

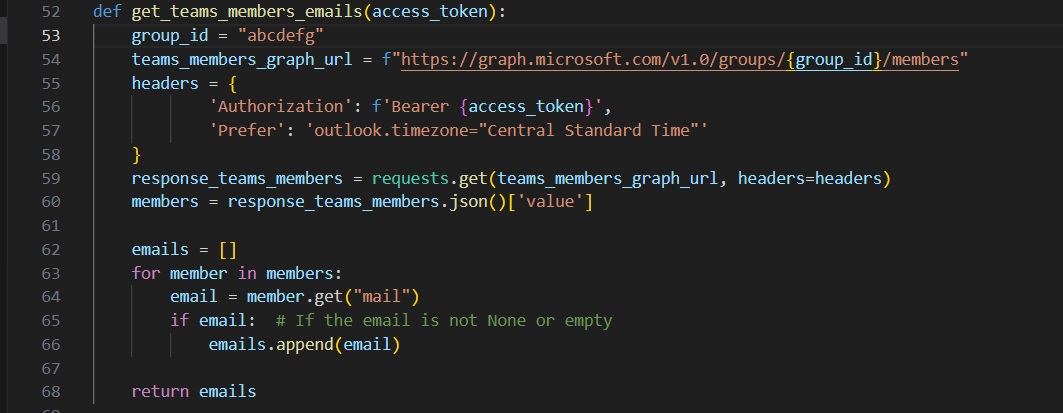
## Change the sender of the email

1. To change the sender of the email, follow the reauthentication steps in Auth Workflow
2. Instead of having to reauthenticate, there may be a workaround with the send on behalf of user permission: [reactjs - How to implement email sending on behalf of a user in Microsoft Azure AD - Stack Overflow](https://stackoverflow.com/questions/75448212/how-to-implement-email-sending-on-behalf-of-a-user-in-microsoft-azure-ad), but I haven’t looked into this much.

## Change the recipients of the email

1. If you want to manually add the recipients, in JERA\_email.py, you can comment out the call to collect the emails from the Microsoft Teams and instead create a list of strings that represent each email in the get\_recipients() method as such: 
2. For testing purposes, you can also use this method of manually adding recipients

## Change which Teams the script auto collects emails from, but do NOT change which channel and Teams the events data is collected from

1. If you want the script to auto collect emails from different users from a different Teams channel but NOT change which channel the events data is being collected from, then in the get\_teams\_members\_emails() function, create a new line before the teams\_members\_graph\_url is createdand create a group\_id string variable in that function with the ID of the channel.
   1. E.g. demonstrated in line 53

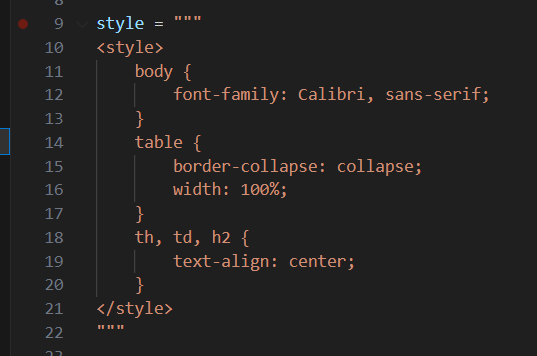
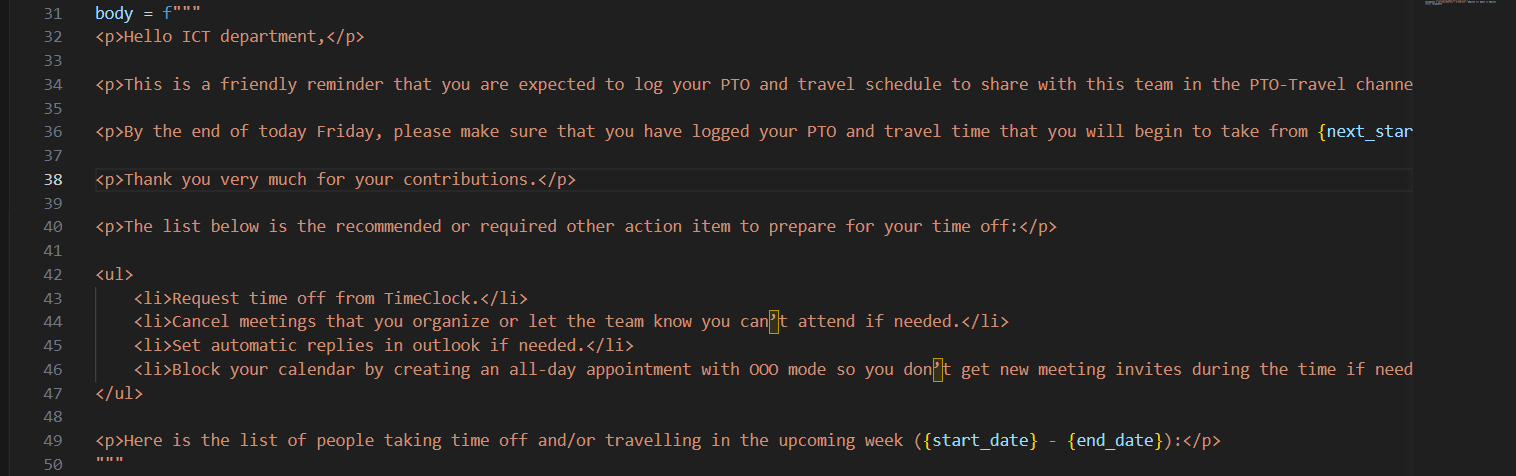
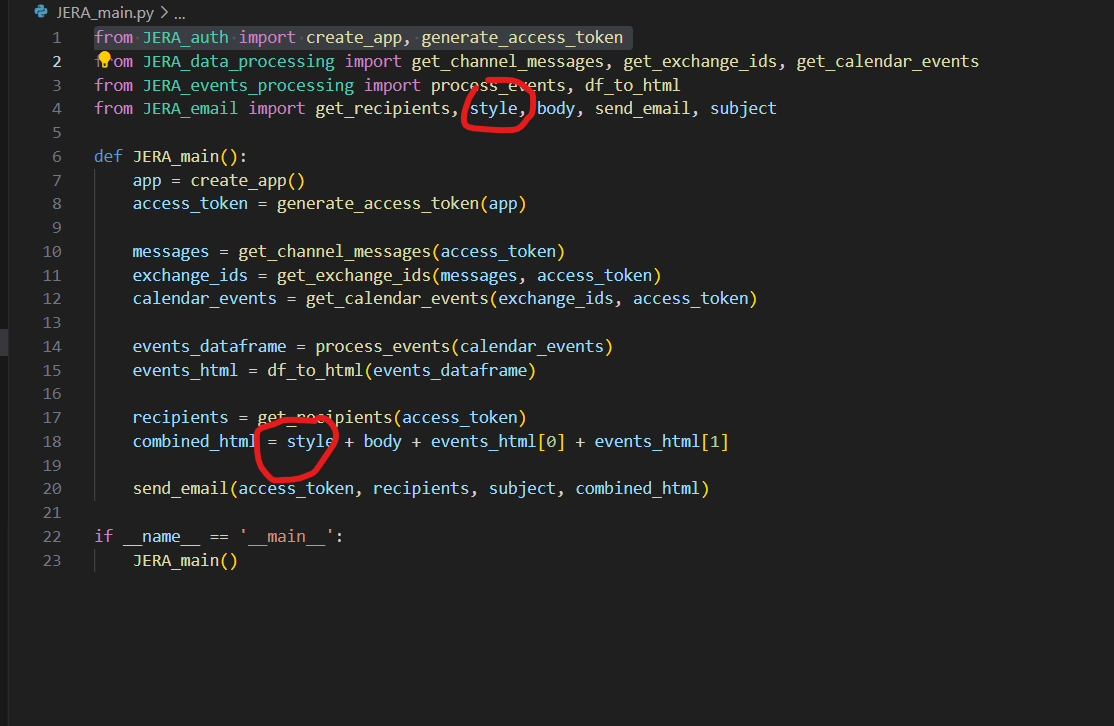
## Change where the recipients of the email is being collected from AND change where the events data is being collected from

1. See section Change where the data is being collected from (changing the Microsoft Teams team)

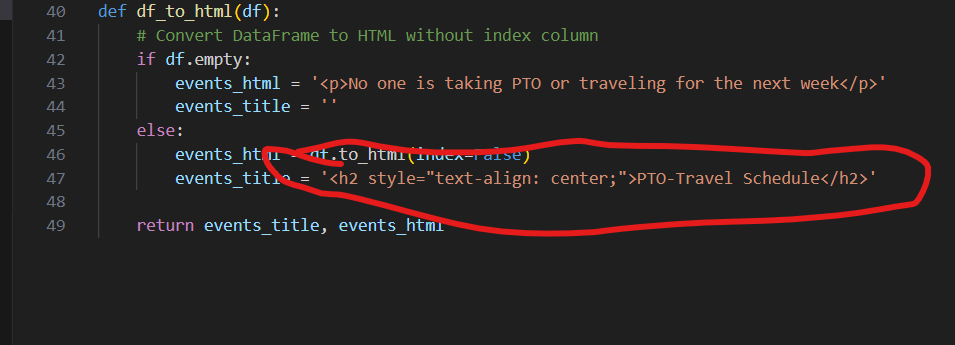
## Change the subject of the email

1. In JERA\_email.py, modify the ‘subject’ variable 

## Change the body of the email

1. To change the style of the email body (the font, margins, alignment, etc.), if you know CSS, you can modify the ‘style’ variable in JERA\_email.py 
2. Likewise, to change the actual email body text, if you know HTML, you can modify the ‘body’ variable in JERA\_email.py 
3. If you do not know CSS or HTML, you can use an online wysiwyg html editor like [Online WYSIWYG HTML Editor - Professional Toolkit - HTMLG.com](https://htmlg.com/html-editor/) to generate the HTML and CSS code
   1. Create how you want your email paragraph content to look on this website (do NOT include the table, the table will always be appended to the end of the email in the script)
   2. Copy the generated HTML
   3. Delete the ‘style’ variable in JERA\_email.py
   4. In JERA\_main.py, delete ‘style, ’ from the line that imports the variable from JERA\_email and delete ‘style’ from the combined\_html line
   5. Paste the generated HTML/CSS in the ‘body’ variable in JERA\_email.py
   6. The email body should now look how you want it to

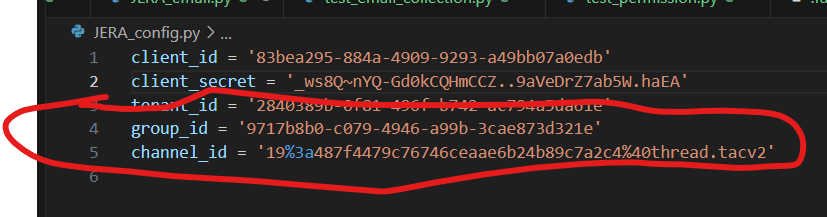
## Change the title of the table

1. If you want to change the “PTO-Travel Schedule” title to something else, you can modify that in the df\_to\_html function in JERA\_events\_processing.py 

## Change the table’s column names

1. See [Section 2.2 Data Processing Logic](#_2.2)_Data_Processing)

## Change where the data is being collected from (changing the Microsoft Teams team)

1. In JERA\_config.py, change the group\_id and the channel\_id where group\_id is the ID of the team you want the data to collect from and channel\_id is the ID of the channel you want to collect data from 
2. To get channel ID: [How To Fetch The Teams ID And Channel ID For Microsoft Teams (c-sharpcorner.com)](https://www.c-sharpcorner.com/blogs/how-to-fetch-the-teams-id-and-channel-id-for-microsoft-teams)
3. To get group ID: [3 Ways to locate a Microsoft Team ID | LinkedIn](https://www.linkedin.com/pulse/3-ways-locate-microsoft-team-id-christopher-barber-/)
4. This will cause the events data to be collected from the Teams channel and the user emails to be collected from all users of the team you inputted

# 

# Troubleshooting

If you see an error message like:

raise HTTPError(http\_error\_msg, response=self)

requests.exceptions.HTTPError: 403 Client Error: Forbidden for url: <https://graph.microsoft.com> …

in the console, then that meansthere’s an API permission error with the web app and that Graph API request.

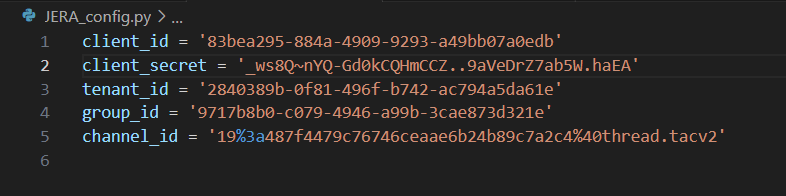
Solution: reauthenticate for the user and try again

# Error Handling

1. Cancelled events: if the event requested is null, then the event must be cancelled and it skips that event in data processing
   1. Potential bug with this method: even though it catches for errors that return a 404 response code, for some reason, invalid graph URLs are also passed
2. Invalid URLs are caught through the raise\_for\_status() method

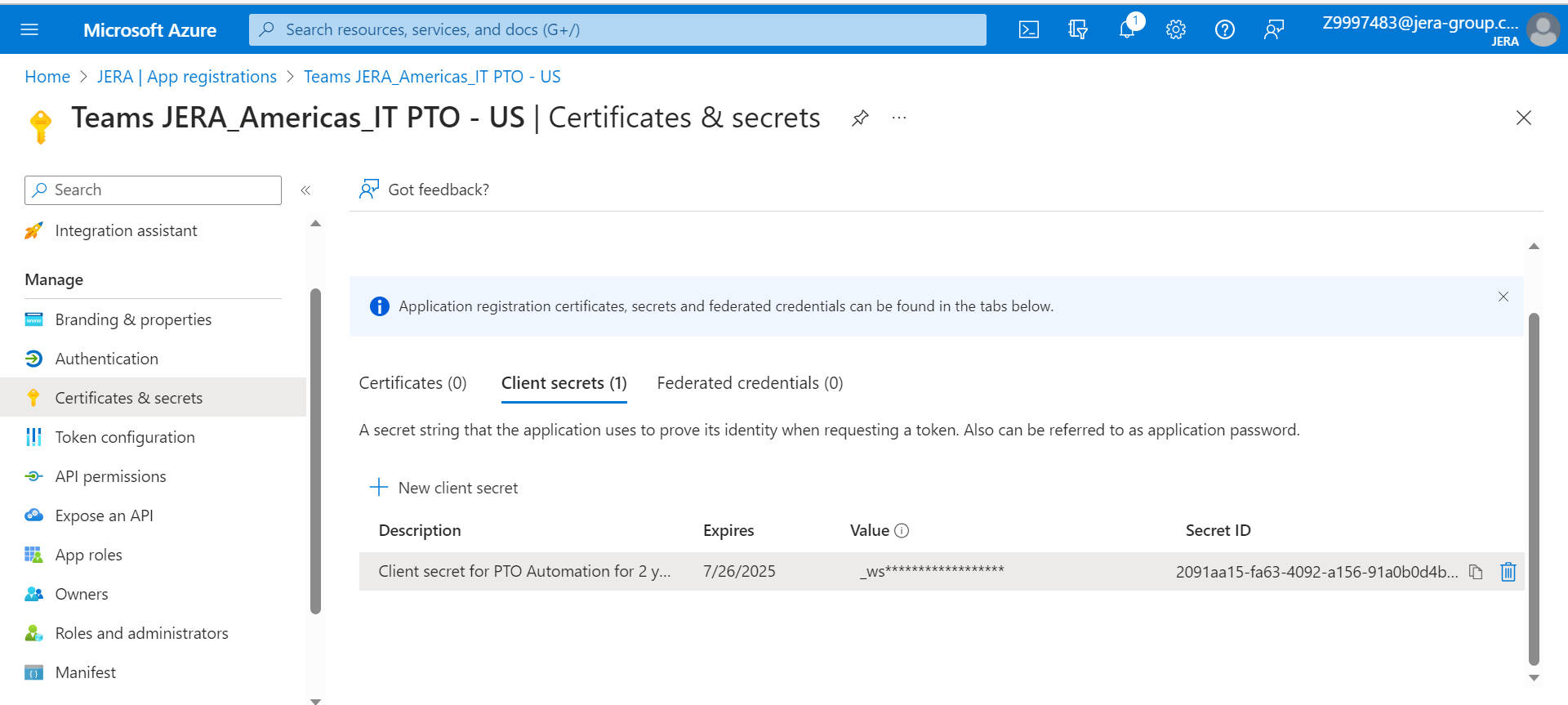
# Maintenance (IMPORTANT)

The one variable that will need to be maintained every TWO years is the client\_secret.



This secret is highly sensitive information so don’t share it with anyone. The secret expires in 2 years since the date it was generated.

In the Azure Active Directory, go to App Registrations, click on Teams JERA\_Americas\_IT PTO – US, then on the left sidebar, click on Certificates & Secrets.



The secret I generated **will expire on 7/26/2025**. Once the secret expires, this script will break. To prevent this from happening, before the secret expires, you need to create a new client secret and replace the ‘client\_secret’ variable in JERA\_config.py with the newly generated client secret value. This value is NOT the Secret ID, it is the secret Value (the one that starts with \_ws in the image and is then censored). Once generated, the secret value cannot be revealed again.

So, at the time of writing this document, please remember to replace the secret with a new one before the date 7/26/2025 and continue this process for as long as this automation flow is needed.

# Future Steps

1. GUI Development to make changing parameters of the script easier
   1. A user interface for changing the email body
   2. A user interface for inputting the channel ID and group ID
   3. A user interface for changing the sender and recipients
   4. A user interface for changing the subject of the email
2. Using an excel file as a database for messages
   1. Purpose: Graph API isn’t fetching all the messages in the Teams message every time and can instead reference the database, will result in faster run time of script
   2. Update database periodically based on lastModifiedDateTime of messages
      1. Pseudo-logic: if the lastModifiedDateTime of the message > the date time of the last database update, then add or update that record in the database
3. Setup a secret expiry notification system
   1. Potential solutions: [Use Power Automate to Notify of Upcoming Azure AD App Client Secrets and Certificate Expirations - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/use-power-automate-to-notify-of-upcoming-azure-ad-app-client/ba-p/2406145)
   2. [How can I tell when a Azure AD client secret expires? - Stack Overflow](https://stackoverflow.com/questions/44075464/how-can-i-tell-when-a-azure-ad-client-secret-expires)
   3. [Using Logic Apps and Microsoft Sentinel to alert on expiring Azure AD Secrets – Microsoft Sentinel 101 (learnsentinel.blog)](https://learnsentinel.blog/2021/12/01/using-logic-apps-and-microsoft-sentinel-to-alert-on-expiring-azure-ad-secrets/)
   4. [Azure AD app registration secrets: a quick script to check who expires soon – Stefano Demiliani](https://demiliani.com/2022/01/28/azure-ad-app-registration-secrets-a-quick-script-to-check-who-expires-soon/)
4. Maintain a SQL database for a database of PTO and Travel events for the accounting department to better calculate payrolls

# Contact Creator

If there are any issues or questions that are not resolved in this document and need attention, feel free to reach out to the creator of this document and script. My contact information is:

Personal Email: [shelbyyang.23@gmail.com](mailto:shelbyyang.23@gmail.com)

School email: [shelbyyang@utexas.edu](mailto:shelbyyang@utexas.edu)

LinkedIn: <https://www.linkedin.com/in/shelbyyang/>