Introduction to the Mastodon API using CircuitPython

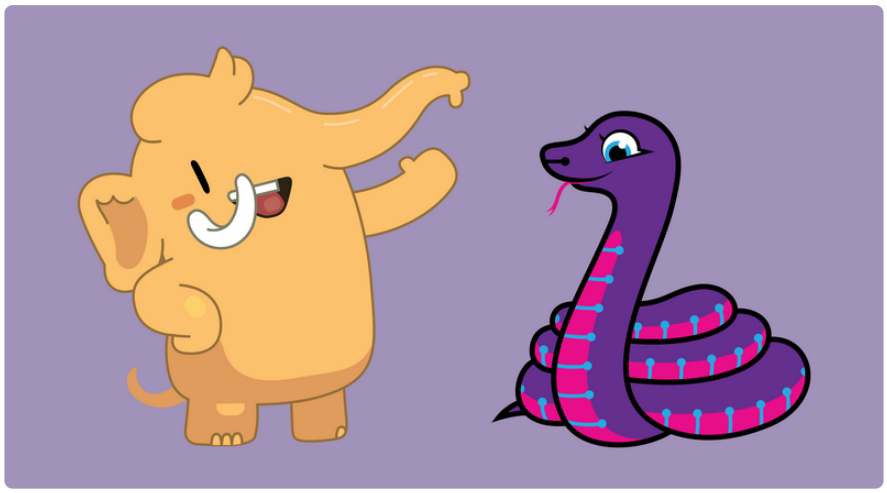
Created by Kattni Rembor

https://learn.adafruit.com/intro-to-mastodon-api-circuitpython

Last updated on 2023-04-03 10:59:17 AM EDT

Table of Contents

Overview



Mastodon is an open source, decentralised social media platform. Similar to other social media platforms, once you create an account you can create posts, communicate with other users, follow their feeds, and so on. You can read up on it further () for more information.

Mastodon has a simple API that allows you to write code that accesses Mastodon actions in various ways. An API (application programming interface) is a way for two computer programs to "talk" to each other. Put simply, you can write a CircuitPython program that interacts with Mastodon to do things like send a post, or read all posts under a certain hashtag, and much more!

This guide is an introduction to the Mastodon API. It does not refer to the ActivityPub API.

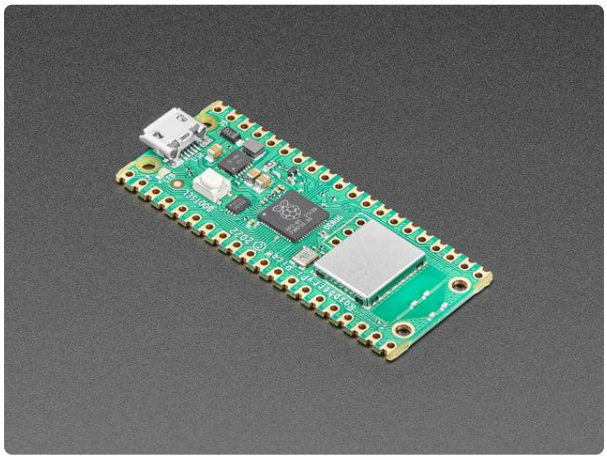
The first thing you need is your Mastodon API access token, which is necessary to communicate with the Mastodon API. The next page covers how to get your token. Then, there are two CircuitPython examples: posting to Mastodon, and tracking all messages using a particular hashtag on Mastodon. All of this using your Raspberry Pi Pico W!

Prerequisites

This guide assumes that you have a Mastodon account. If you do not, you can visit the Mastodon site () to sign up. If you would like more information before doing so, check out the Mastodon documentation ().

Hardware

For this guide, you'll need the following.



Raspberry Pi Pico W
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/5526

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

Create Your settings.toml File

If you've worked on WiFi projects with CircuitPython before, you're probably familiar with the secrets.py file. This file is a Python file that is stored on your CIRCUITPY drive that contains all of your secret WiFi information, such as your SSID, SSID password and any API keys for IoT services.

As of CircuitPython 8 (), there is support for a settings.toml file. Similar to secrets.py, the settings.toml file separates your sensitive information from your main code.py file.

Your settings.toml file should be stored in the main directory of your CIRCUITPY drive. It should not be in a folder.

settings.toml File Example

Here is an example on how to format your settings.toml file.

```
# Comments are supported
CIRCUITPY_WIFI_SSID="guest wifi"
CIRCUITPY_WIFI_PASSWORD="guessable"
CIRCUITPY_WEB_API_PORT=80
CIRCUITPY_WEB_API_PASSWORD="passw0rd"
test_variable="this is a test"
thumbs_up="\U0001f44d"
```

In a settings.toml file, it's important to keep these factors in mind:

- Strings are wrapped in double quotes; ex: `"your-string-here"`
- Integers are not quoted and may be written in decimal with optional sign (`+1`, `-1`, `1000`) or hexadecimal (`0xabcd`).
    - Floats, octal (`0o567`) and binary (`0b11011`) are not supported.
- Use `\u` escapes for weird characters, `\x` and `\ooo` escapes are not available in .toml files
    - Example: `\U0001f44d` for (thumbs up emoji) and `\u20ac` for € (EUR sign)
- Unicode emoji, and non-ASCII characters, stand for themselves as long as you're careful to save in "UTF-8 without BOM" format

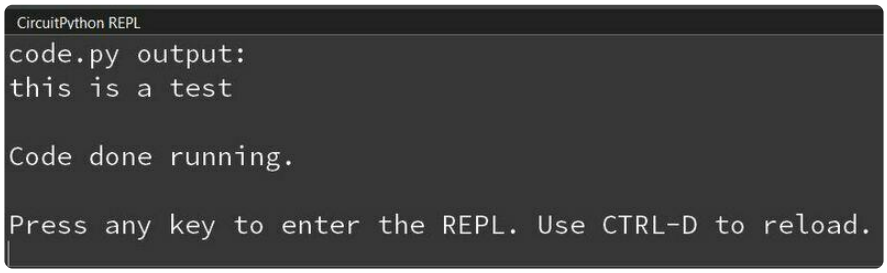
When your settings.toml file is ready, you can save it in your text editor with the .toml extension.

Accessing Your settings.toml Information in code.py

In your code.py file, you'll need to `import` the `os` library to access the settings.toml file. Your settings are accessed with the `os.getenv()` function. You'll pass your settings entry to the function to import it into the code.py file.

```
import os

print(os.getenv("test_variable"))
```



In the upcoming CircuitPython WiFi examples, you'll see how the settings.toml file is used for connecting to your SSID and accessing your API keys.

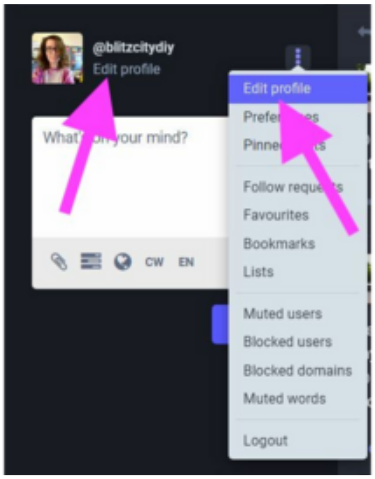
Environment Variables Docs

Environment Variables Docs ()

Generate Your Mastodon Token

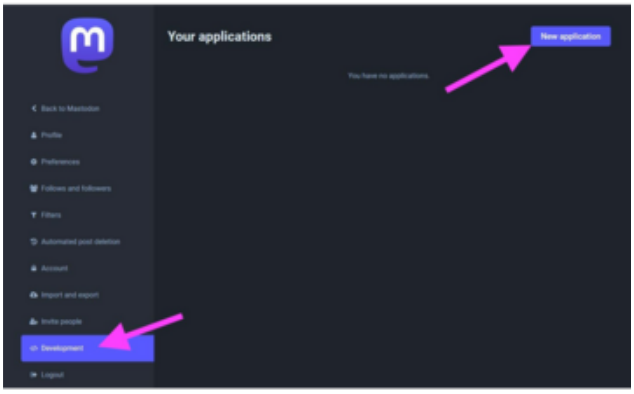
The Mastodon API requires you to have an access token to make requests. There are a few steps in this process. You'll want to follow the steps below to generate your access token.

Edit Your Profile

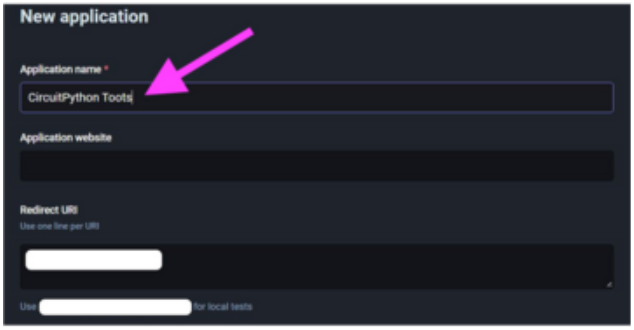


Visit your profile page on your Mastodon instance. Click Edit profile, either from the link shown under your Mastodon username, or by clicking the three dots to the right near your username and choosing Edit profile from the menu.

Create an Application

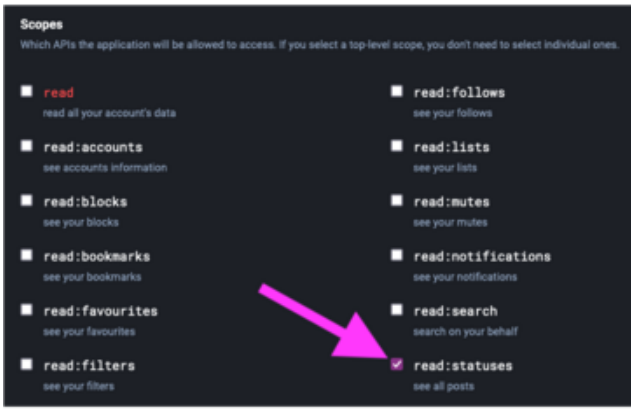


From the list of options on the left, choose Development. Then click New Application.



Give your application a name. It can be whatever you want it to be. Have fun with it!

Application Scope and Creation



Choose the scope of your application. It is good practice to limit the scope to only the specific things you need. The first section contains the read scopes. You only need read:statuses. Check the box next to it.

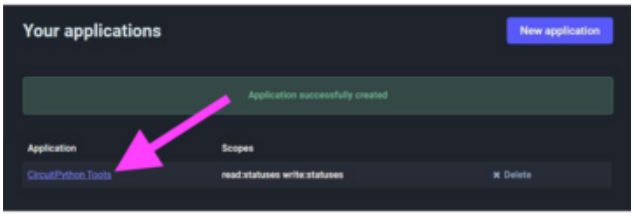
Scroll to the next section.

The second section contains the write scopes. You only need write:statuses. Check the box next to it.

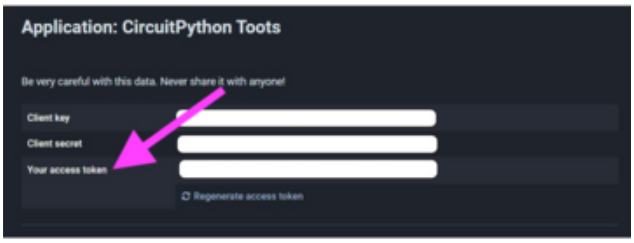
Scroll to the bottom of the page.

Those two are the only scopes you need to allow. You're ready to submit your application. Click SAVE CHANGES.

Find your Token



You'll see a green rectangle letting you know your application was successfully created. Below it, you'll find your new application listed with name and scopes. Click on your application name.



There you'll find Your access token followed by the token itself.

You've successfully generated your Mastodon access token!

Keep the current webpage up as you're going to want your access token for the next step.

Post to Mastodon

You can send a toot to Mastodon using CircuitPython and your Raspberry Pi Pico W! This example provides a prompt through the serial console where you can type in your message, and then works with the Mastodon API to send your content as a toot.

This example will also run on any CircuitPython-compatible microcontroller board with native WiFi. No changes necessary!

Load the Example and Library

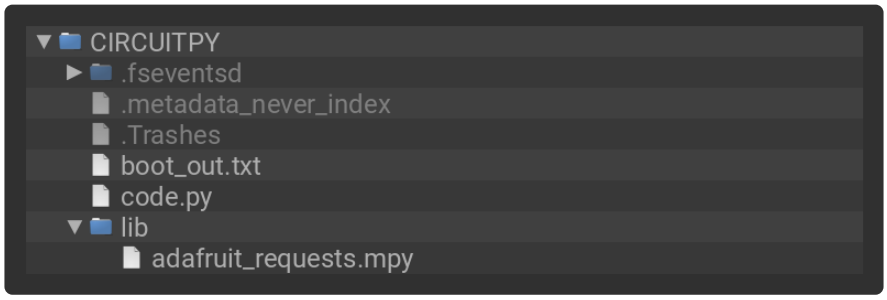
This example requires one external library. Luckily you can load the code and the necessary library together. Click the blue Download Project Bundle button above the code below to download the necessary library and the code.py file in a zip file.

Connect the Raspberry Pi Pico W to your computer via a known good data+power USB A to micro B cable. The Pico W should show up in your File Explorer or Finder (depending on Operating System) as a thumb drive named CIRCUITPY. Extract the contents of the zip file, and copy the entire lib folder, and the code.py file to your CIR CUITPY drive.

Your CIRCUITPY/lib folder should contain the following file:

- adafruit_requests.mpy

Your contents of your CIRCUITPY drive should resemble the following:



Example Code

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import os
import ssl
import wifi
import socketpool
import adafruit_requests

# add your mastodon token as 'mastodon_token' to your settings.toml file
headers = {'Authorization': 'Bearer ' + os.getenv('mastodon_token')}

# add your mastodon instance to your settings.toml file as mastodon_host
url = 'https://' + os.getenv('mastodon_host') + '/api/v1/statuses'

# connect to SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

pool = socketpool.SocketPool(wifi.radio)
```

```
requests = adafruit_requests.Session(pool, ssl.create_default_context())

# you'll be prompted in the REPL to enter your post text
post = input("Please enter your Mastodon post text: ")
# pack the post for sending with the API
post_text = {"status": post}

# confirm in the REPL that you want to post by entering y for yes or n for no
send_check = input("Send post to Mastodon (y/n)?")

# if you type y
if send_check == "y":
    # send to mastodon with a POST request
    r = requests.post(url, data=post_text, headers=headers)
    print()
    print("You posted '%s' to Mastodon. Goodbye." % post)
# if you type n
else:
    print("You did not post to Mastodon. Goodbye.")
```

Add Your settings.toml File

Remember to add your settings.toml file as described in the Create Your settings.toml File page () earlier in the guide. You'll need to include your `CIRCUITPY_WIFI_SSID`, `CIRCUITPY_WIFI_PASSWORD`, `mastodon_host` and `mastodon_token` in the file.

The `mastodon_host` variable should be assigned only the text part of the URL to your Mastodon instance. For example, `'mastodon.social'` or `'fosstodon.org'` without the https:// or a trailing /.

```
CIRCUITPY_WIFI_SSID = "your-wifi-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-wifi-password-here"

# Your host should be only the text part of the URL to your Mastodon instance.
# For example, 'mastodon.social' or 'fosstodon.org' without the https://.
mastodon_host = "your-mastodon-instance-here"
mastodon_token = "your-mastodon-token-here"
```

Sending a Toot

Once everything is saved to CIRCUITPY, you'll want to connect to the serial console. You should see the following prompt:

Type in the content of your message.

When you're happy with your message, press enter. You'll be greeted with the following question:

This is included as another step between writing up your message and posting it. It gives you a chance to make sure your message is what you want it to be.

If you're happy with your message, and wish to post it, type `y`.

If you're not happy with your message, and you do not wish to post it, type `n`.

If you choose `y`, the code follows through with interacting with the API and sends your message. Take a look at your Mastodon profile to see your new post! You'll also see the following in the serial console, where your message content will be the content of the message you sent.

If you choose `n`, it skips sending anything, and simply prints the following to the serial console:

That's all there is to sending a toot to Mastodon using CircuitPython and the Pico W!

Code Walkthrough

Here are some details of what's happening behind the scenes in this example.

Headers and URL

At the top of the code, your authentication token is packed into `headers` via your set tings.toml file. This is included in your API POST request to gain access to your Mastodon account. `url` is your API request URL. It passes your Mastodon instance via your settings.toml file.

```
# add your mastodon token as 'mastodon_token' to your settings.toml file
headers = {'Authorization': 'Bearer ' + os.getenv('mastodon_token')}

# add your mastodon instance to your settings.toml file as mastodon_host
url = 'https://' + os.getenv('mastodon_host') + '/api/v1/statuses'
```

`input()` Your Post

The `input()` function is used to prompt you to enter your post text in the REPL. `input()` is blocking, so the code is paused while it awaits your text. After entering your post text, it is packed into the variable `post_text` which will be sent via the API.

Before sending the toot though, an additional `input()` is requested to confirm. If you type `y`, then the post will be tooted. If you type `n`, then the post will not be tooted. An `if` statement checks this logic. With `y`, a `requests.post()` request is sent to Mastodon, which includes the API URL, your post text and your authentication headers.

```
# you'll be prompted in the REPL to enter your post text
post = input("Please enter your Mastodon post text: ")
# pack the post for sending with the API
post_text = {"status": post}

# confirm in the REPL that you want to post by entering y for yes or n for no
send_check = input("Send post to Mastodon (y/n)?")

# if you type y
if send_check == "y":
    # send to mastodon with a POST request
    r = requests.post(url, data=post_text, headers=headers)
    print()
    print("You posted '%s' to Mastodon. Goodbye." % post)
# if you type n
else:
    print("You did not post to Mastodon. Goodbye.")
```

Track a Hashtag

You can send track all Mastodon posts with a particular hashtag using CircuitPython and your Raspberry Pi Pico W! This example allows you to customise which hashtag you'd like to follow, and then works with the Mastodon API to retrieve toots with that hashtag.

This example will also run on any CircuitPython-compatible microcontroller board with native WiFi. No changes necessary!

Load the Example and Library

This example requires one external library. Luckily you can load the code and the necessary library together. Click the blue Download Project Bundle button above the code below to download the necessary library and the code.py file in a zip file.

Connect the Raspberry Pi Pico W to your computer via a known good data+power USB A to micro B cable. The Pico W should show up in your File Explorer or Finder (depending on Operating System) as a thumb drive named CIRCUITPY. Extract the contents of the zip file, and copy the entire lib folder, and the code.py file to your CIR CUITPY drive.

Your CIRCUITPY/lib folder should contain the following file:

- adafruit_requests.mpy

Your contents of your CIRCUITPY drive should resemble the following:

Example Code

```
# SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

import os
import re
import time
import ssl
import wifi
import socketpool
import microcontroller
import adafruit_requests

#  enter the hashtag that you want to follow
hashtag = "CircuitPython"

#  connect to SSID
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'),
os.getenv('CIRCUITPY_WIFI_PASSWORD'))

#  add your mastodon token as 'mastodon_token' to your settings.toml file
headers = {'Authorization': 'Bearer ' + os.getenv('mastodon_token') +
'read:statuses'}

pool = socketpool.SocketPool(wifi.radio)
requests = adafruit_requests.Session(pool, ssl.create_default_context())

#  initial request, gets most recent matching hashtag post
#  add your mastodon instance (mastodon.social, tech.lgbt, etc)
#  to your settings.toml file as mastodon_host
r = requests.get("https://%s/api/v1/timelines/tag/%s?limit=1" %
(os.getenv('mastodon_host'), hashtag), headers=headers) # pylint: disable=line-too-
```

```
long
json_data = r.json()
post_id = str(json_data[0]['id'])
print("A new #%s post from @%s:" % (hashtag, str(json_data[0]['account']['acct'])))
print(re.sub('<[^>]+>', '', json_data[0]['content']))
print()

while True:
    try:
        time.sleep(360)
        # compares post_id to see if a new post to the hashtag has been found
        r = requests.get("https://%s/api/v1/timelines/tag/%s?since_id=%s" %
(os.getenv('mastodon_host'), hashtag, post_id), headers=headers) # pylint:
disable=line-too-long
        json_data = r.json()
        json_length = len(json_data)
        # if the id's match, then the json array is empty (length of 0)
        # otherwise there is a new post
        if json_length > 0:
            post_id = str(json_data[0]['id'])
            print("A new #%s post from @%s:" % (hashtag, str(json_data[0]['account']
['acct'])))
            print(re.sub('<[^>]+>', '', json_data[0]['content']))
            print()
        else:
            print("no new #%s posts" % hashtag)
            print(json_length)
            print()

    except Exception as e:  # pylint: disable=broad-except
        print("Error:\n", str(e))
        print("Resetting microcontroller in 10 seconds")
        time.sleep(10)
        microcontroller.reset()
```

Add Your settings.toml File

Remember to add your settings.toml file as described in the Create Your settings.toml File page () earlier in the guide. You'll need to include your `CIRCUITPY_WIFI_SSID`, `CIRCUITPY_WIFI_PASSWORD`, `mastodon_host` and `mastodon_token` in the file.

The `mastodon_host` variable should be assigned only the text part of the URL to your Mastodon instance. For example, `"mastodon.social"` or `"fosstodon.org"` without the https:// or a trailing /.

```
CIRCUITPY_WIFI_SSID = "your-wifi-ssid-here"
CIRCUITPY_WIFI_PASSWORD = "your-wifi-password-here"

# Your host should be only the text part of the URL to your Mastodon instance.
# For example, "mastodon.social" or "fosstodon.org" without the https://.
mastodon_host = "your-mastodon-instance-here"
mastodon_token = "your-mastodon-token-here"
```

Tracking a Hashtag

Once everything is saved to CIRCUITPY, you'll want to connect to the serial console. After a moment, you'll see the latest post that included the #CircuitPython hashtag.

```
code.py output:
A new #CircuitPython post from @todbot@mastodon.social:
First light on PicoTouchSynth and the capsense pads work better than my PicoTou
ch board! And the reverse-mount #NeoPixel LEDs are totally awesome. The test co
de is in #CircuitPython of course
```

Once the latest post is grabbed, the code will continue to check the hashtag for new posts. If there are no new posts, you'll see the following in the serial console.

```
no new #CircuitPython posts
0
```

When it finds a new post, it will print it to the serial console as it did with the first one.

That's all there is to tracking a hashtag on Mastodon using CircuitPython and the Raspberry Pi Pico W!

Code Walkthrough

Here are some details of what's happening behind the scenes in this example.

Hashtag

At the top of code, you can enter the hashtag that you want to follow as a string. This will be included in your API request URL.

```
#  enter the hashtag that you want to follow
hashtag = "CircuitPython"
```

Headers

Your authentication token is packed into `headers` via your settings.toml file. This is included in your API POST request to gain access to your Mastodon account. `url` is your API request URL. It passes your Mastodon instance via your settings.toml file.

```
#  add your mastodon token as 'mastodon_token' to your settings.toml file
headers = {'Authorization': 'Bearer ' + os.getenv('mastodon_token') +
'read:statuses'}
```

Initial Request

Before the loop, an initial request is made to the API to access the most recent post with the hashtag that you are tracking. In the URL, your Mastodon instance is passed via your settings.toml file.

The response from the request, is packed into a JSON. From the JSON, the post ID number is tracked as `post_id`. This will be used in the loop to see if a new hashtag post has been tooted.

In the REPL, you'll see the hashtag post printed with the account name and the post content. `re.sub()` is used to remove extraneous HTML (`<` and `>`) tags that are included in the raw post text.

```
#  initial request, gets most recent matching hashtag post
#  add your mastodon instance (mastodon.social, tech.lgbt, etc) to your
settings.toml file as mastodon_host
r = requests.get("https://%s/api/v1/timelines/tag/%s?limit=1" %
(os.getenv('mastodon_host'), hashtag), headers=headers) # pylint: disable=line-too-
long
json_data = r.json()
post_id = str(json_data[0]['id'])
print("A new #%s post from @%s:" % (hashtag, str(json_data[0]['account']['acct'])))
print(re.sub('&lt;[^&gt;]+&gt;', '', json_data[0]['content']))
print()
```

The Loop

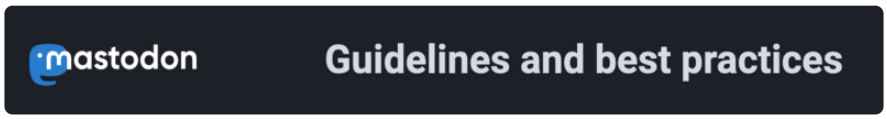
In the loop, a request is made via the API approximately every 5 minutes to see if a new post with the hashtag has been tooted. `post_id` is passed to the `?since_id=` object in the URL. This allows you to see if there is a new post via the API rather than using logic in CircuitPython.

If there a new post has not been tooted, then the request returns an empty JSON. This is checked using the `len()` function. If the JSON isn't empty, then the newly tooted post with the hashtag is printed to the REPL and `post_id` is updated with the new post ID number.

```
time.sleep(360)
# compares post_id to see if a new post to the hashtag has been found
r = requests.get("https://%s/api/v1/timelines/tag/%s?since_id=%s" %
(os.getenv('mastodon_host'), hashtag, post_id), headers=headers) # pylint:
disable=line-too-long
json_data = r.json()
json_length = len(json_data)
# if the id's match, then the json array is empty (length of 0)
# otherwise there is a new post
if json_length &gt; 0:
```

```
  post_id = str(json_data[0]['id'])
  print("A new #%s post from @%s:" % (hashtag, str(json_data[0]['account']
['acct'])))
  print(re.sub('&lt;[^&gt;]+&gt;', '', json_data[0]['content']))
  print()
else:
  print("no new #%s posts" % hashtag)
  print(json_length)
  print()
```

Going Further



If tooting a post or tracking a hashtag aren't quite what you're looking for, have no fear: the Mastodon API is very well documented (). You can reference the documentation to build your own API request URL to suit your project needs.

Mastodon API Documentation

With any social media usage, great power comes great responsibility. Be mindful when you're using the API for projects that you are respecting your instance's policies and community guidelines, especially around automated posting. This keeps it fun and wholesome for everyone.