All the Things You Can Do With GitHub API and Python

GitHub REST API allows you to manage issues, branches, repos, commits and more, so let’s see how you can do that using Python!

Most of us use GitHub every day either using CLI or its website. Sometimes however, you need to automate these same tasks like, for example creating Gist, querying repository analytics or just pulling, modifying and pushing new file. All these things and more can be done easily using [GitHub API](https://developer.github.com/v3/), and Python is here to help with that and make it even easier.



Original Photo by [BAILEY MAHON](https://unsplash.com/@baileymahon?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/sea?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

**What We Will Need**

Before we start using GitHub API, we first need to generate a personal access token that will allow us to authenticate against the API. We can get one at <https://github.com/settings/tokens> by clicking on *Generate new token*. You will be asked to select scopes for the token. Which scopes you choose will determine what information and actions you will be able to perform against the API. You should be careful with the ones prefixed with write:, delete: and admin: as these might be quite destructive. You can find description of each scope in [docs here](https://developer.github.com/apps/building-oauth-apps/understanding-scopes-for-oauth-apps/).

Now that we have the token, let’s test whether it actually works:

And here is the expected (trimmed) response showing list of my public *Gists*:

**Doing It With Python**

We have the personal token and we tested it with cURL, so now we can switch to doing the same thing in Python. We have two options here though. We can use raw requests or we can use .

*PyGitHub* exposes some of the GitHub API endpoints for most common operations like repository, issue or branch management. It can’t be used for every single feature exposed through the GitHub API, so in the following sections, I will show mixture of *PyGitHub* and *Requests* calls depending on whether it can be done with *PyGitHub* or not.

First things first though — let’s install both libraries ( *PyGitHub* and *Requests*) and see a simple example for both:

Example using *PyGitHub*:

Example using *Requests*:

Both snippets above use the same API endpoint to retrieve all open issues for specified repository.

In both cases we start by taking GitHub token from environment variable. Next, in the example with using *PyGitHub* we use the token to create instance of GitHub class, which is then used to get repository and query its issues in *open state*. The result is paginated list of issues, of which we print the first page.

In the example that uses raw HTTP request, we achieve the same result by building API URL from username and repository name and sending GET request to it containing state as body parameter and token as Authorization header. Only difference is that result is not paginated. Here is the result for both examples:

First one being *PyGitHub* output:

Second, raw Python list of dictionaries (JSON):

**Create an Issue**