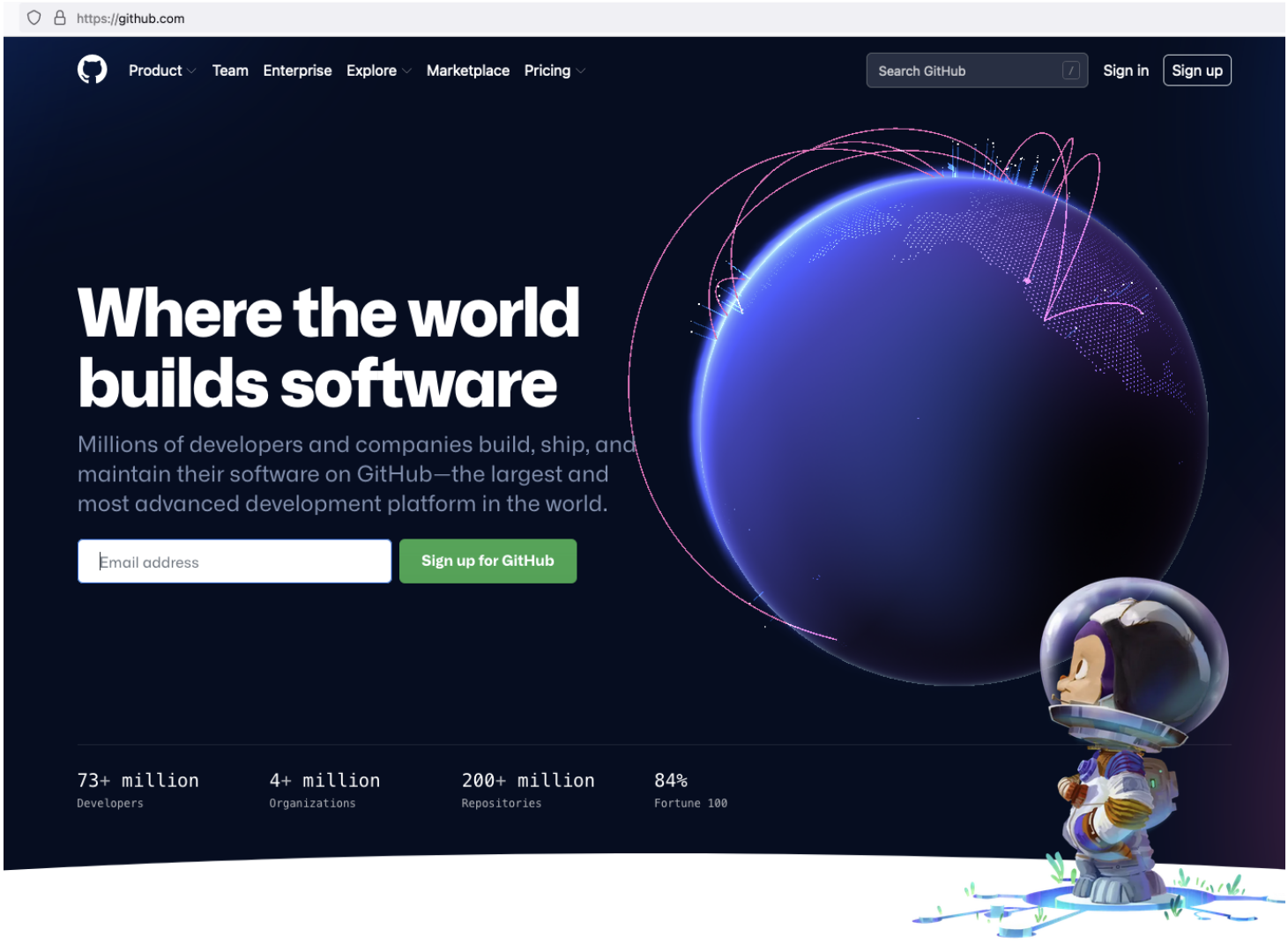
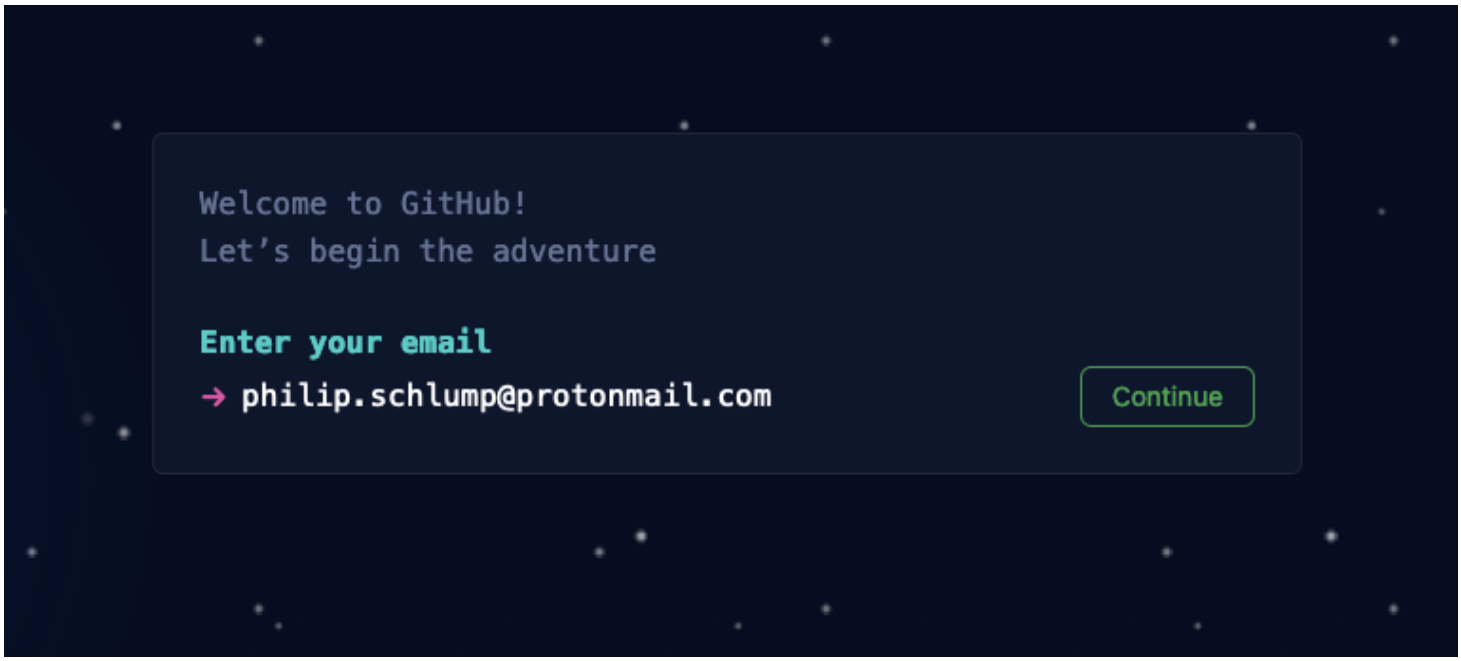
Lecture 27 - github.com with some code

Setup Github.com

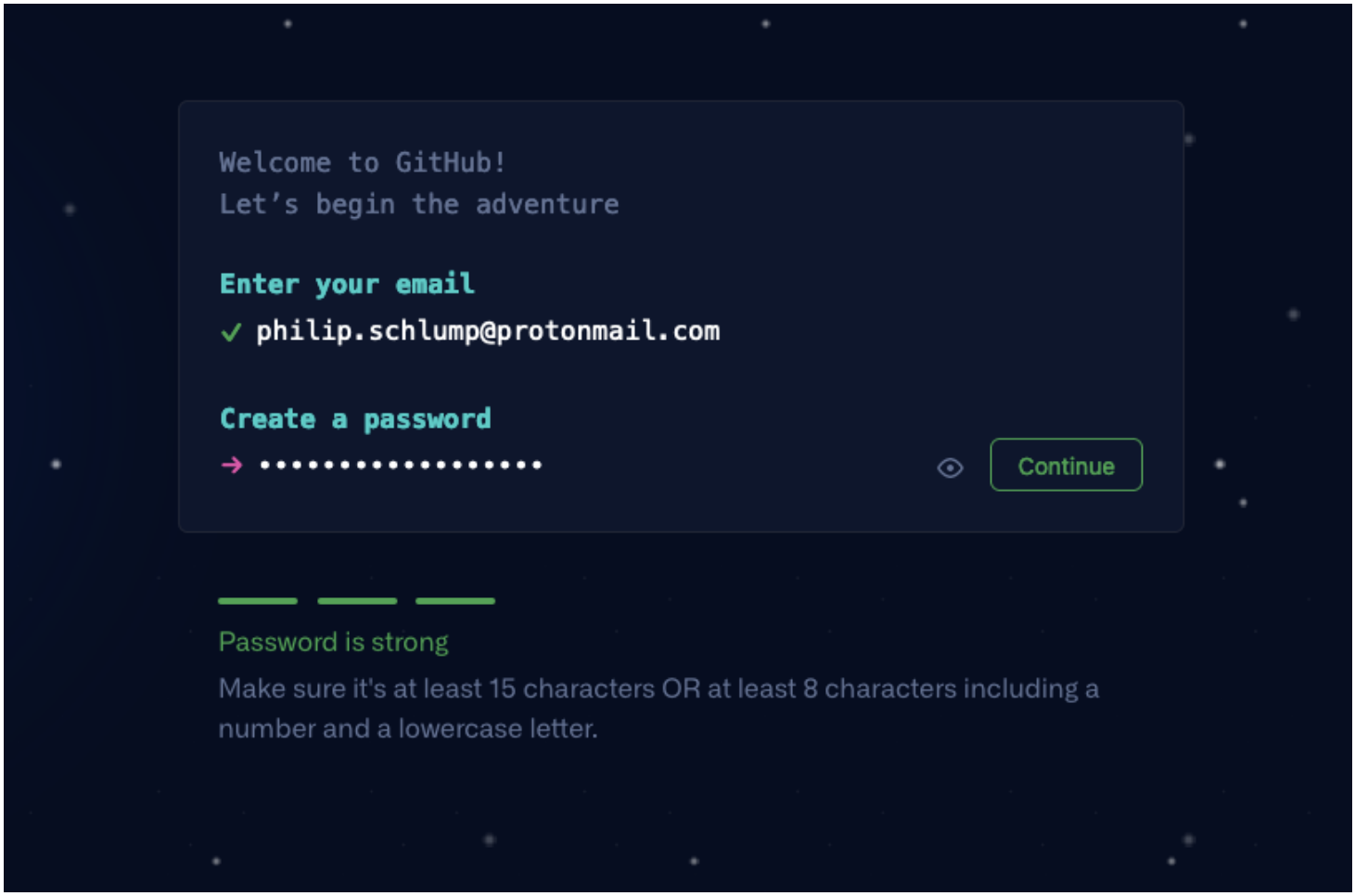
Go to https://github.com



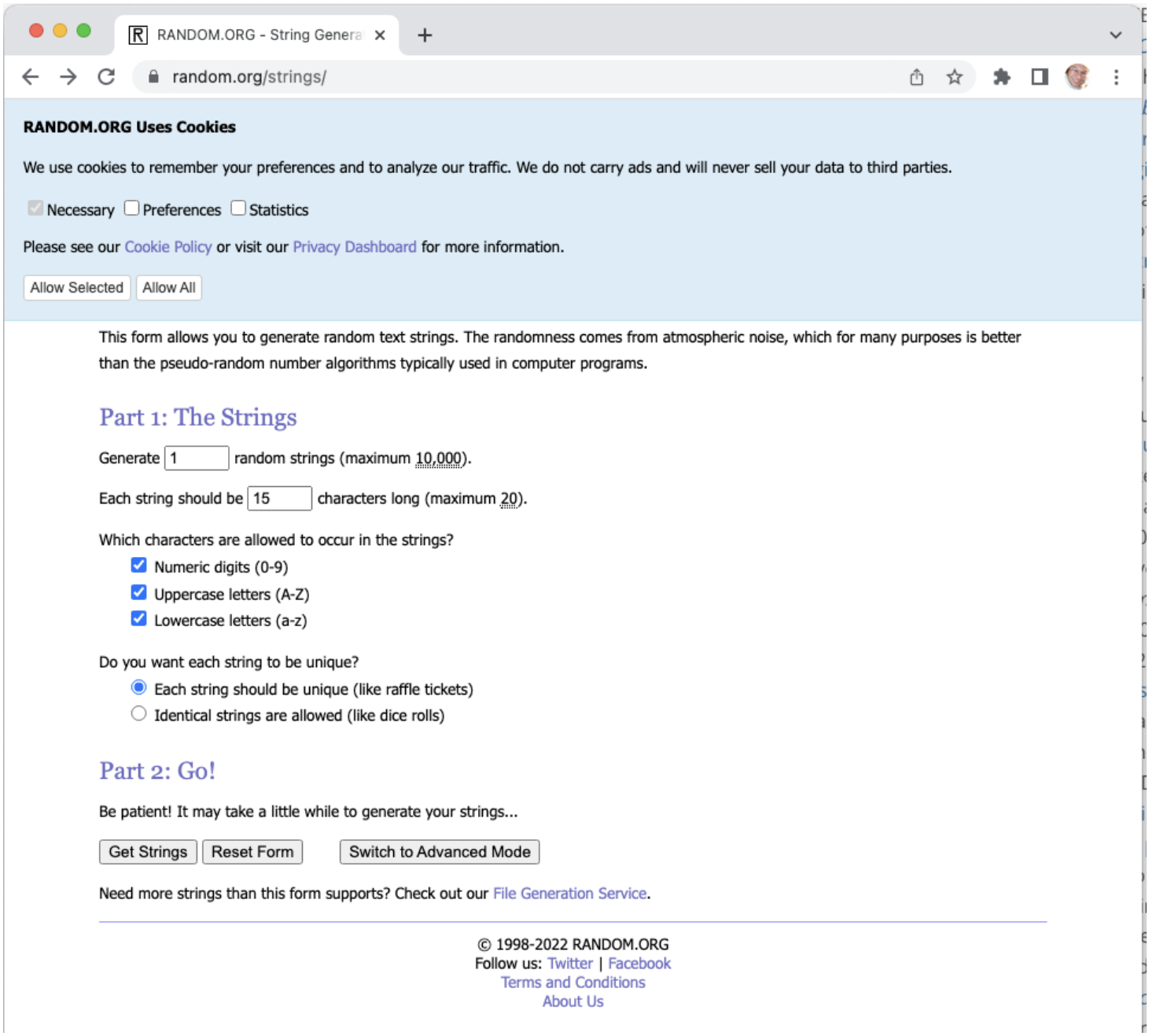
Step 000. Click on the button to create an account. The Green One that says `Sign up for GitHub`



Step 001. You should get a prompt that looks kind of like a terminal prompt. That is the little red arrow. Enter your email address. Click `Continue` on the bottom right.

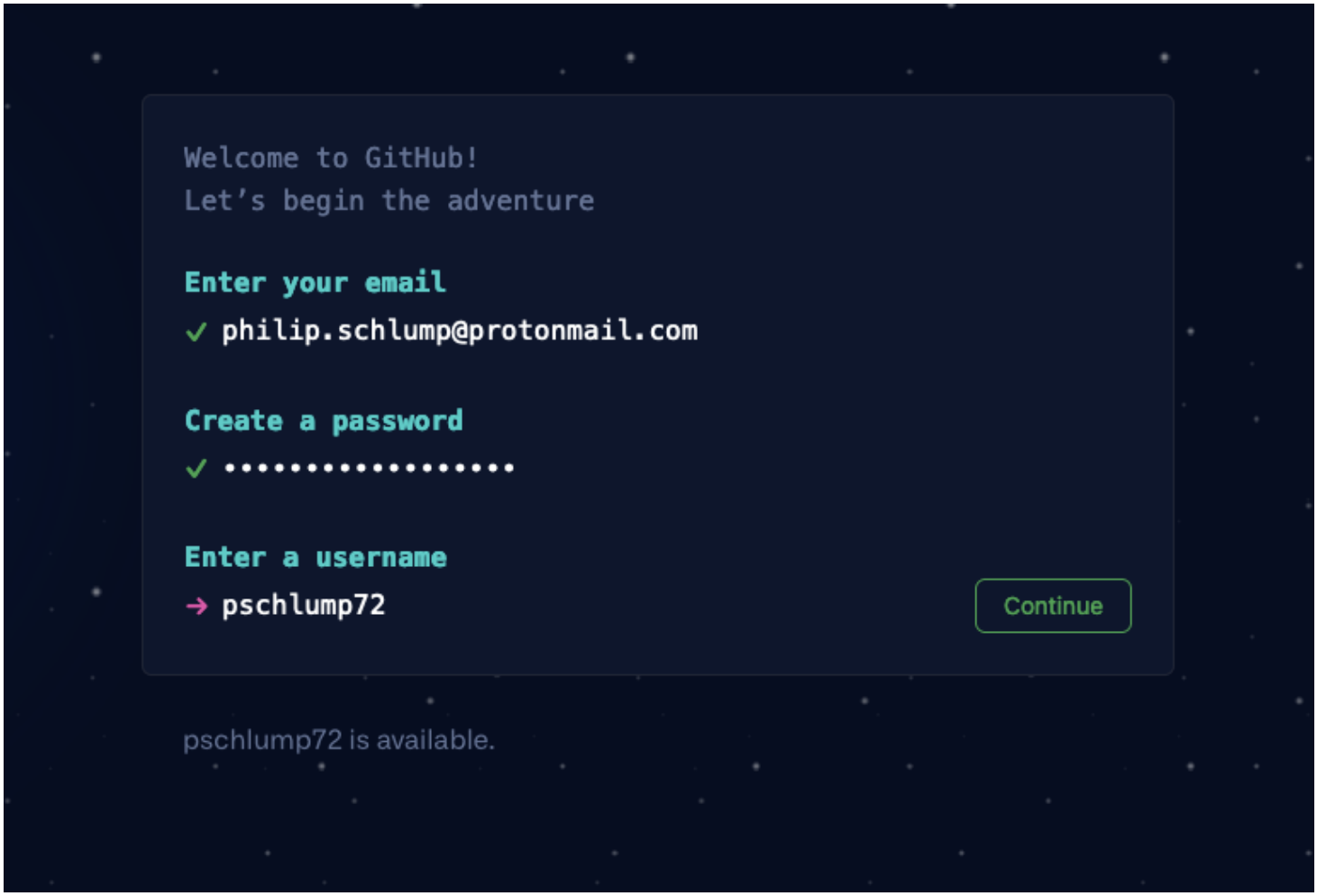


Step 002. A password. If you are using a password manager then generate a random password and use that. If not then use random.org, https://www.random.org/strings/ to generate a password. You will need to save the generated password in a file - you won't remember a random password. When you use it later use cut/paste it from the file. **Make backups of the file! Print it out. Save it in your sock drawer! ALL computers eventually crash - files get deleted. Save it because if you loose it you will not get it back.** Change the number, check the boxes for Numeric, Upper lower case and click the button at the bottom. It should look like:

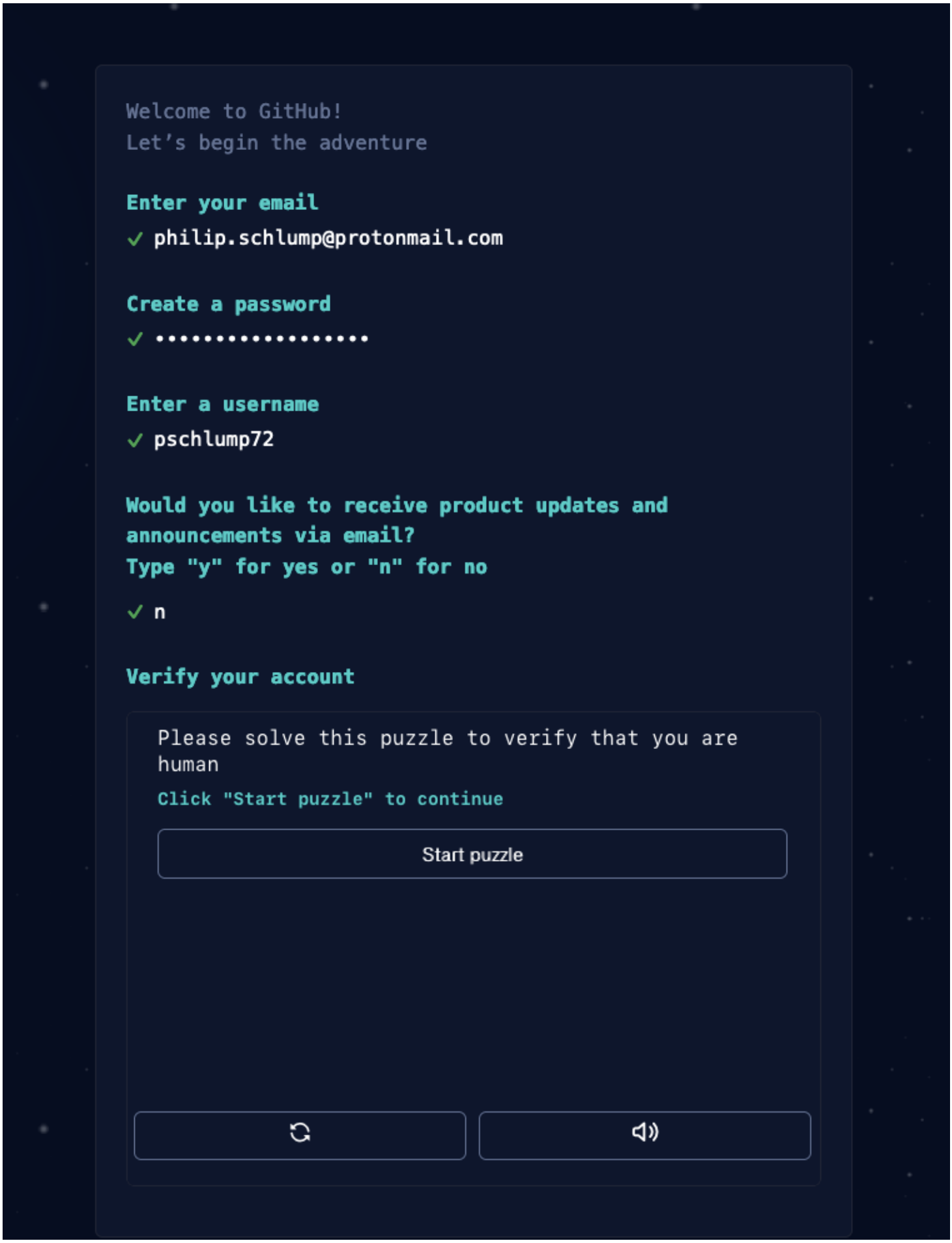


Now you should have a password to enter like, HVo9BcyMIoHD04J

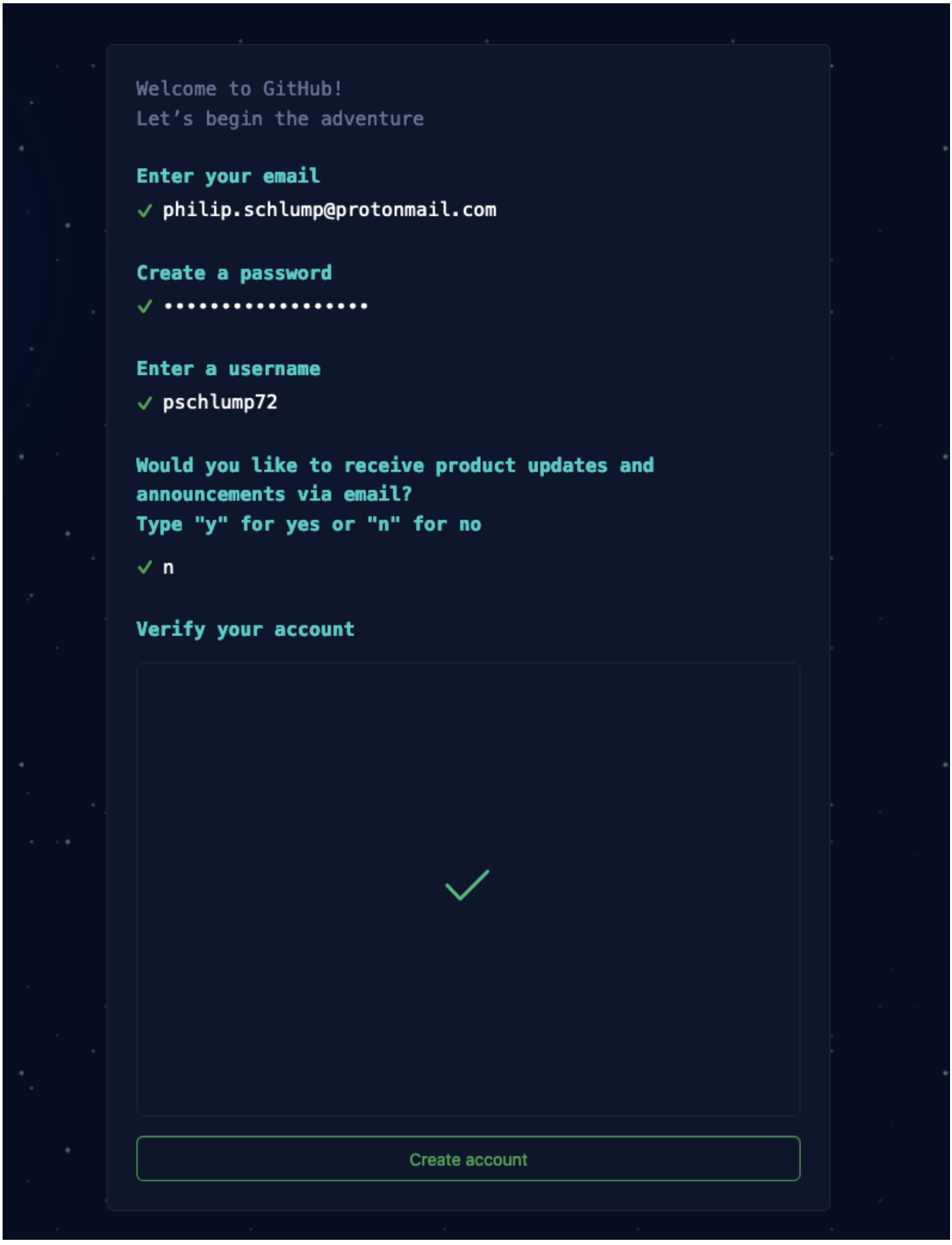
Click `Continue` and give yourself a username that you like.



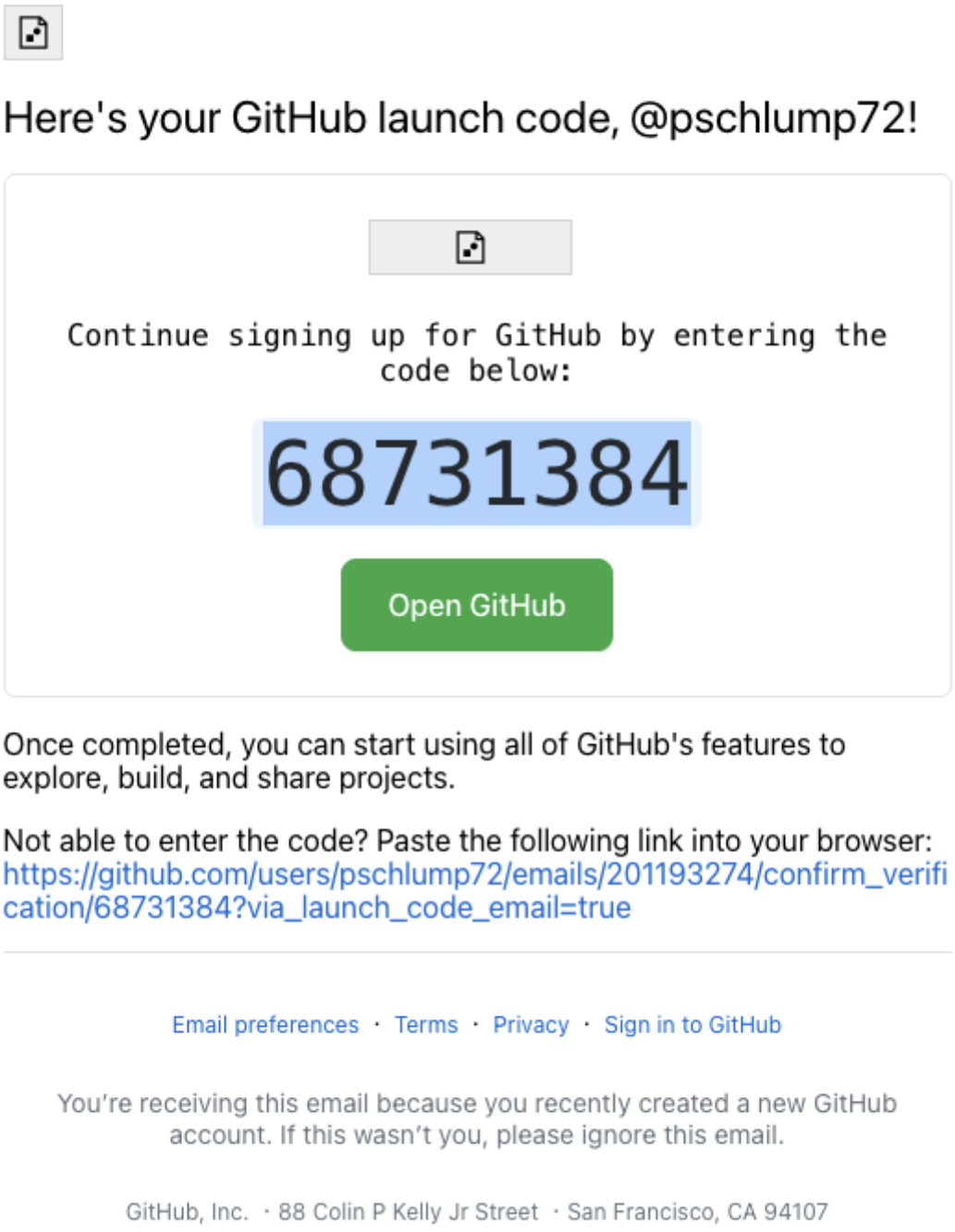
Step 003. Solve a puzzle - so that only humans and machine learning programs in python can login.



Step 004. Get a checkbox and ... Yes click the big button at the bottom, `Create account` .



Step 005. Now off to your email.... Have to confirm the email.



Here's your GitHub launch code, @pschlump72!

Continue signing up for GitHub by entering the code below:

68731384

Open GitHub

Once completed, you can start using all of GitHub's features to explore, build, and share projects.

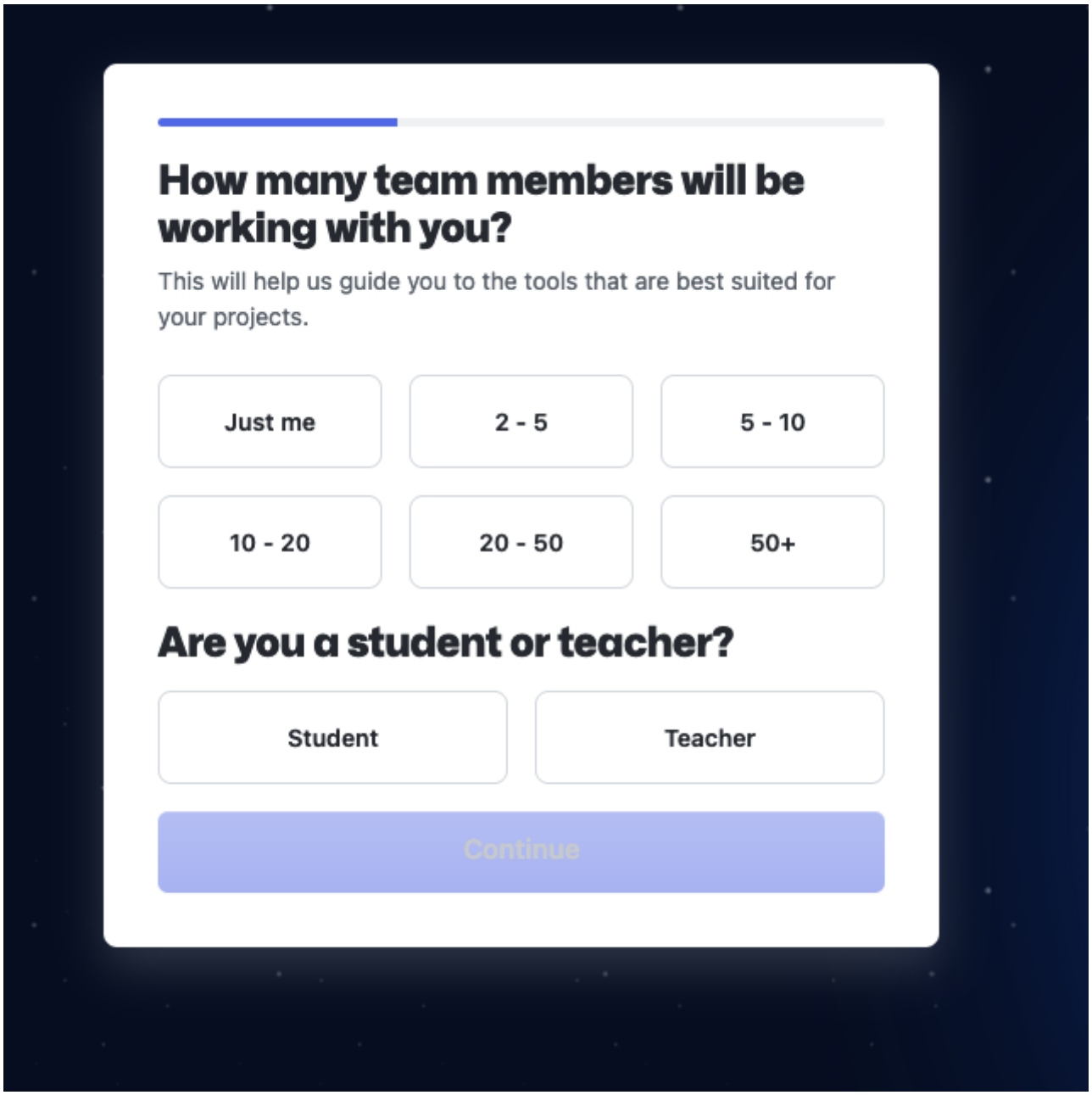
Not able to enter the code? Paste the following link into your browser: https://github.com/users/pschlump72/emails/201193274/confirm_verification/68731384?via_launch_code_email=true

Email preferences · Terms · Privacy · Sign in to GitHub

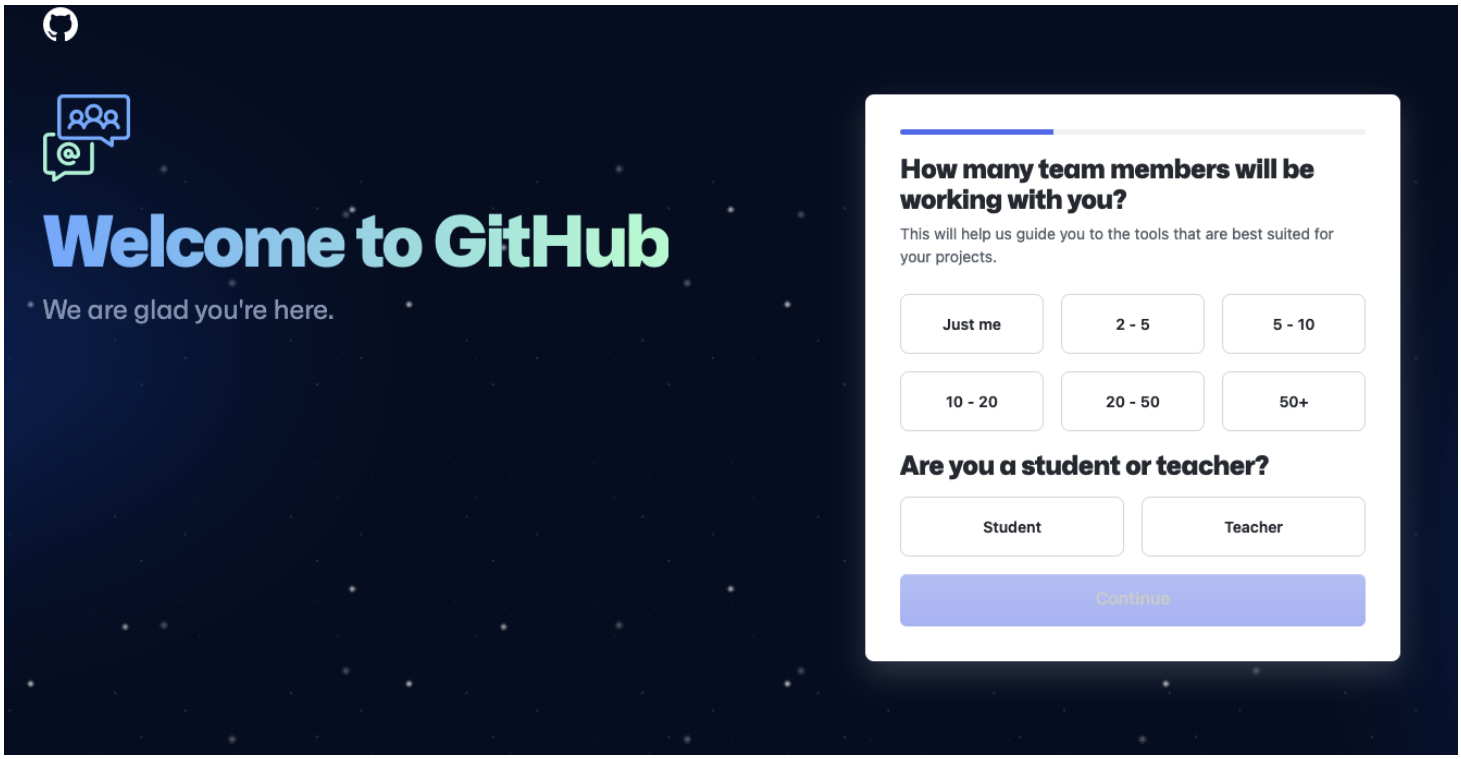
You're receiving this email because you recently created a new GitHub account. If this wasn't you, please ignore this email.

GitHub, Inc. · 88 Colin P Kelly Jr Street · San Francisco, CA 94107

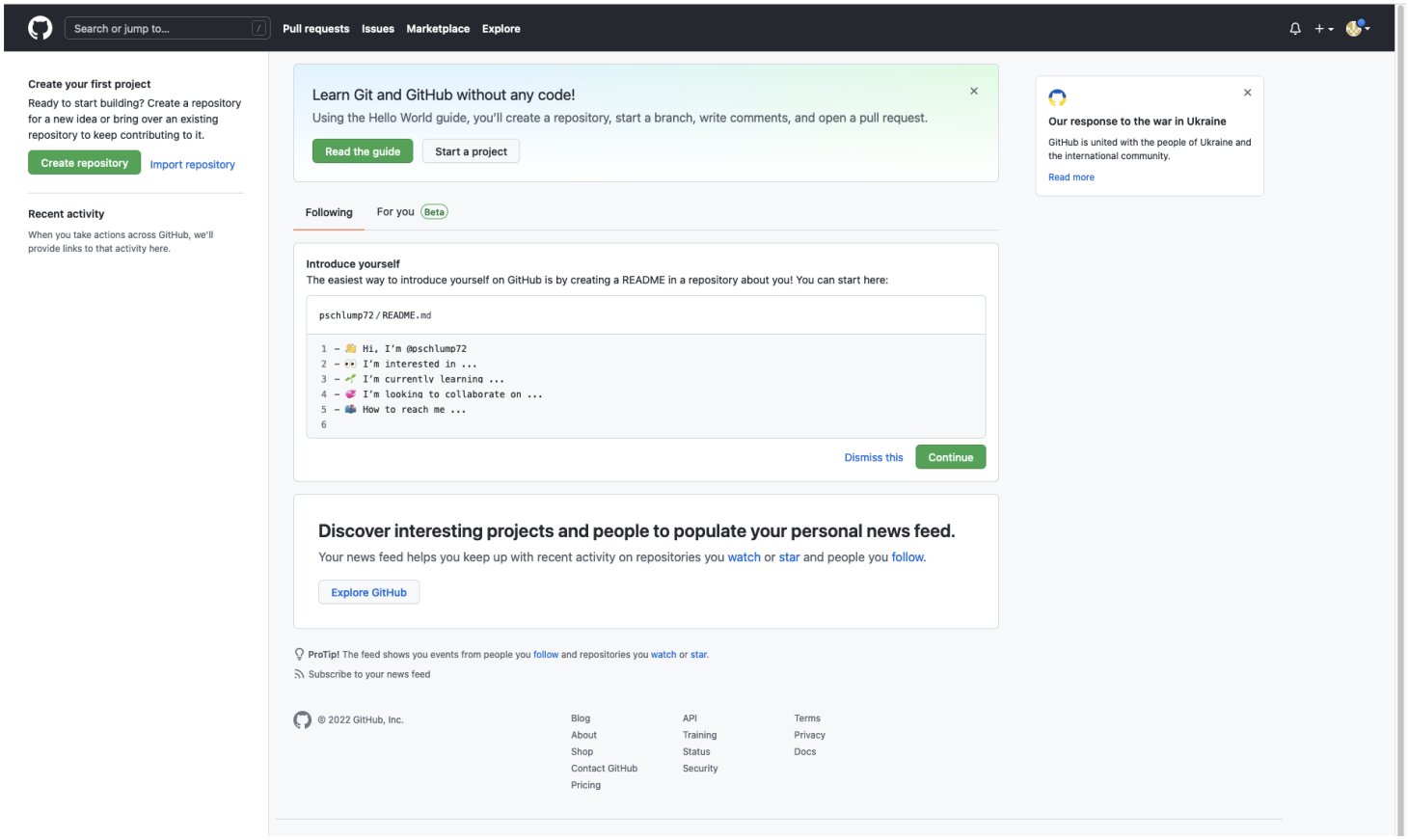
Step 006. You should get an email that looks like this. The number is a one time password. The way that these work is it generates a secret, takes the secret - then uses the secret to generate a password that lasts for a little while and sends that to you. You enter the number and that confirms to the system that the email address was real and you are can login to it.



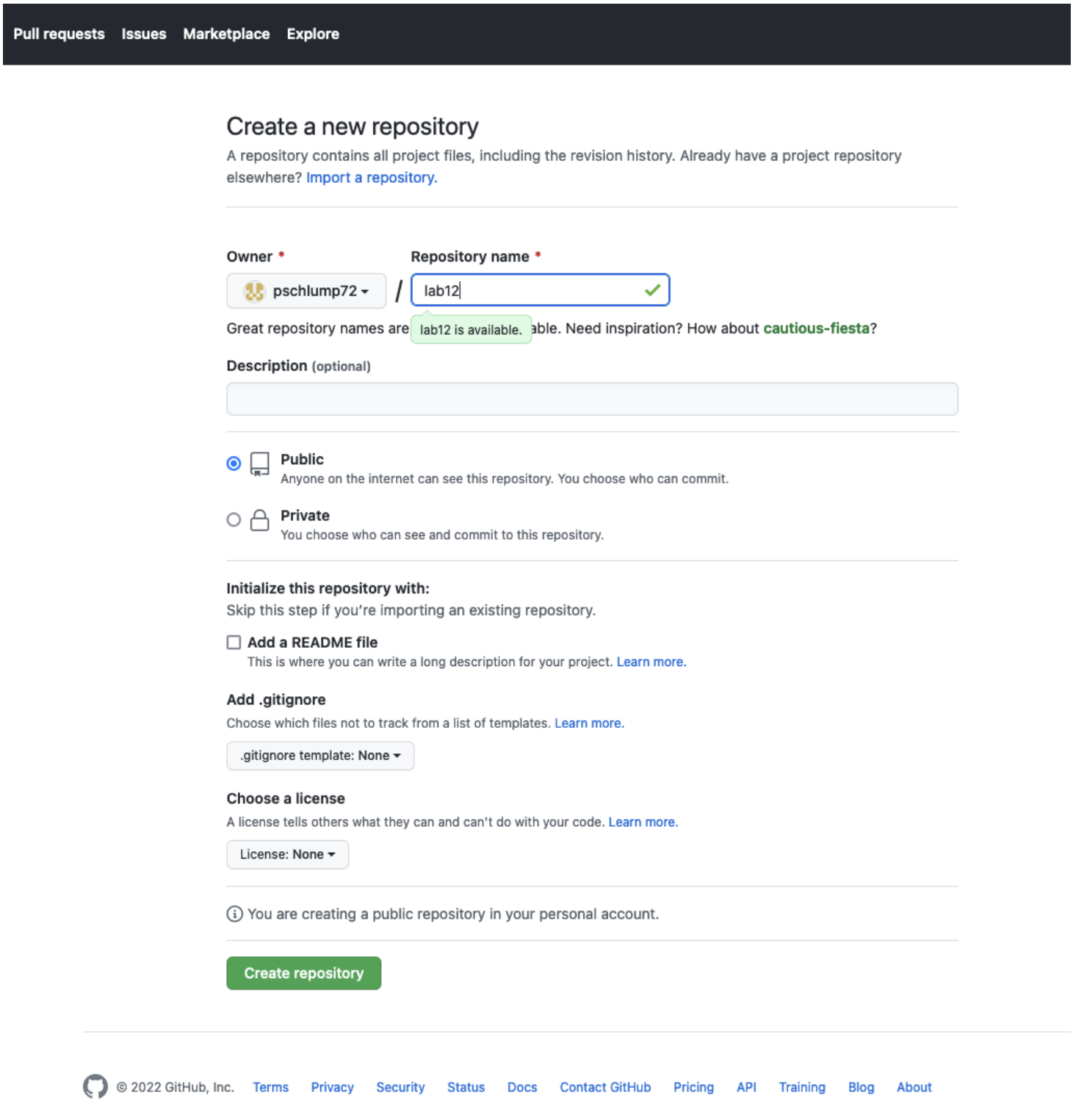
Step 007. Answer some questions.



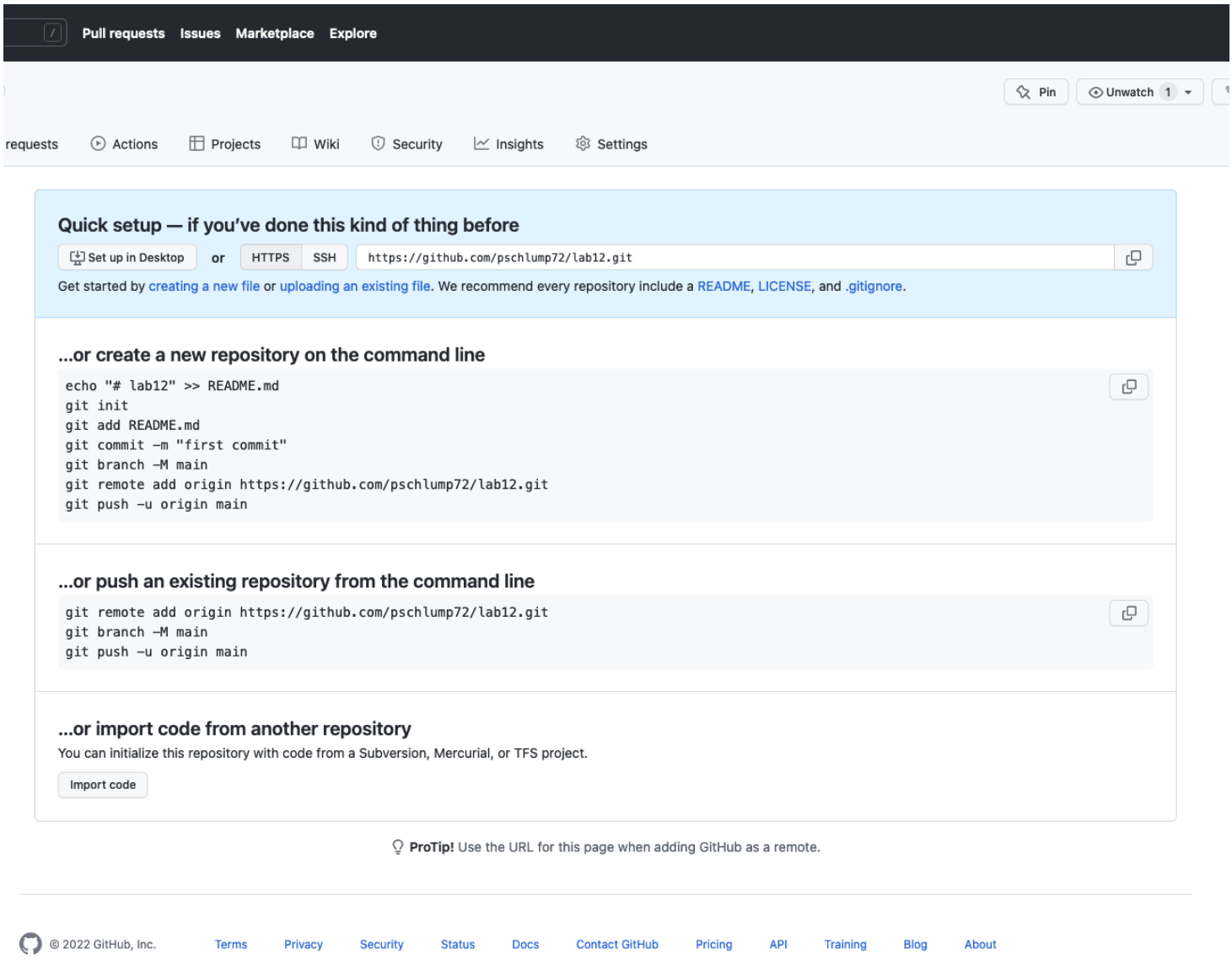
Step 008. Click on the green button on the left, `Create repository`.



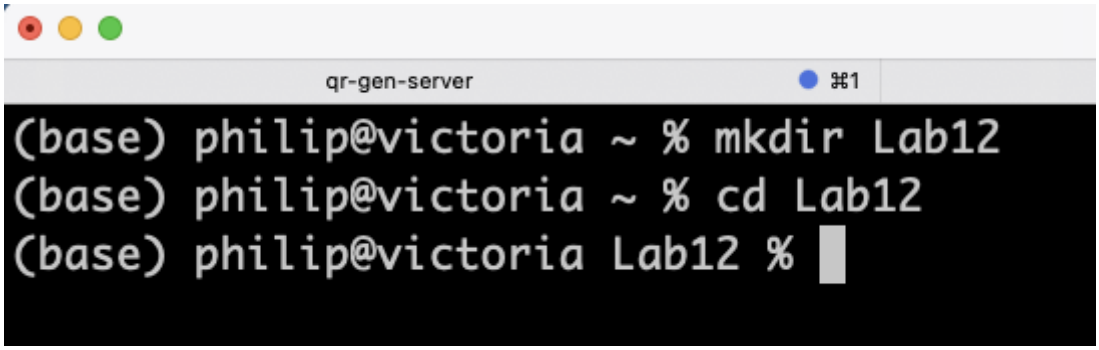
Step 009. Give it a name. You can make it public for this class. Click on the `Create repository` button at the bottom of the page.



Step 010. You should get a screen that looks like this. Copy the commands in the first block that says, `... or create a new repository on the command line .`



Step 011. Go and create a directory on your system. This is using PowerShell on Windows or Terminal on Mac.

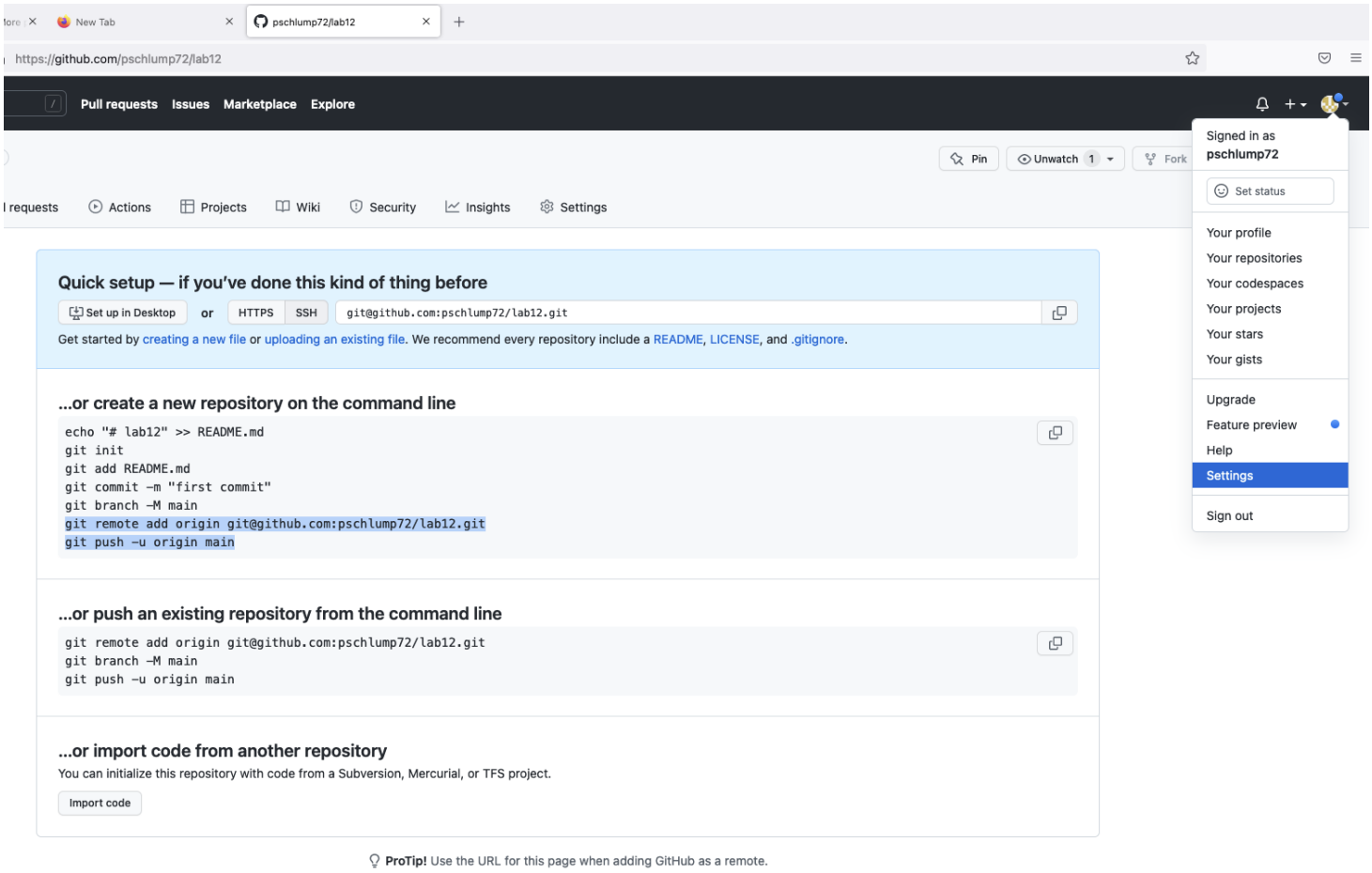


Step 012. Run the commands from Step 010. Some of you may get an error on the last command. If it prompts you for a username and the password to login this is good. If you just get an error then - Step 013 to 018 are for you. They are on how to create a personal access token to login with. Generally if you have not used github.com before you will not need to create a personal access token. If you are already a user and you are creating a new account - then you will need to configure github.com to know which account and what security to use.

You will want to configure your system to know your username and account information for github.com. To do this:

```
$ git config --global user.name "Philip Schlump"
$ git config --global user.email "philip.schlump@protonmail.com"
```

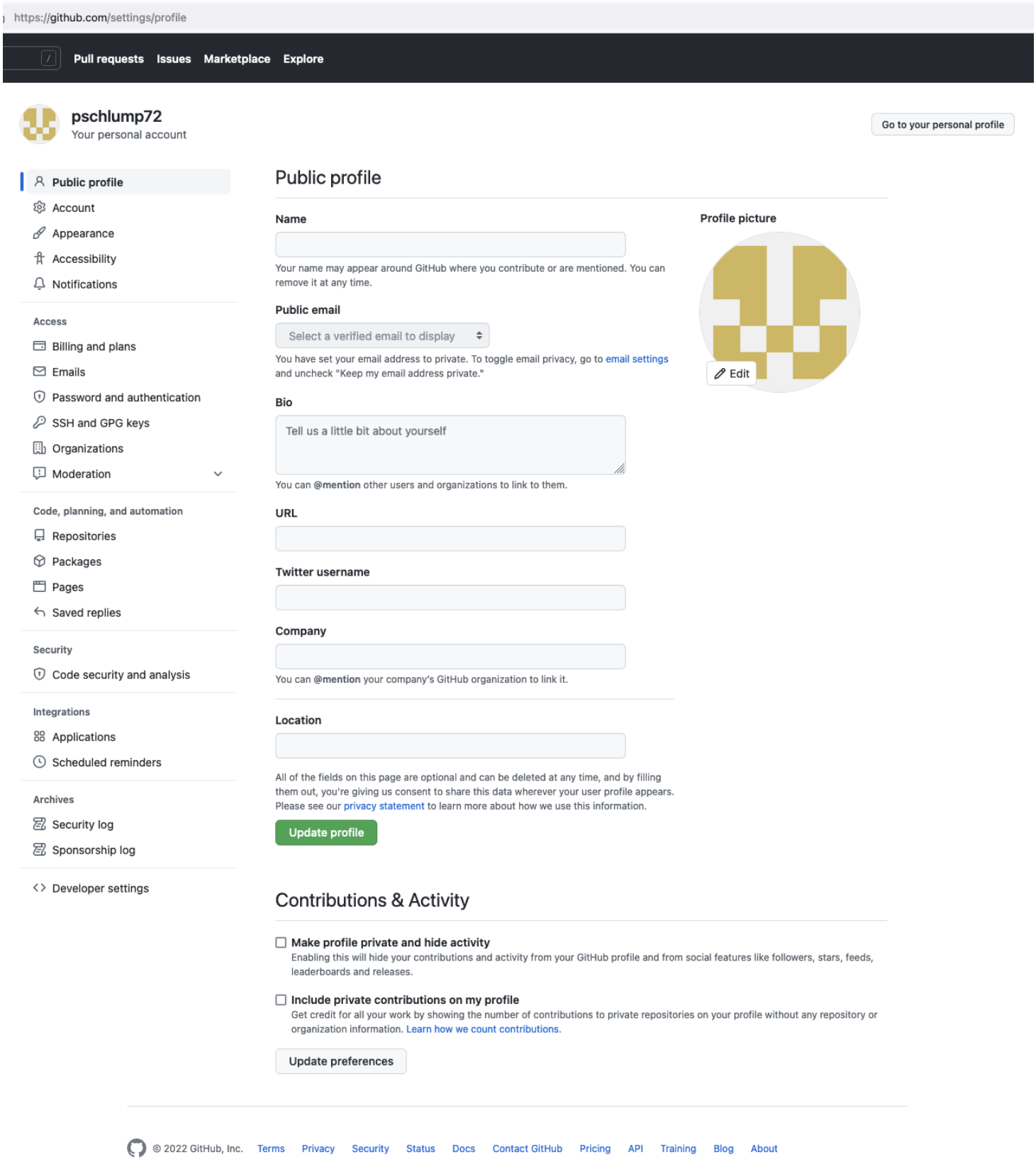
You will want to use your name and email for this.



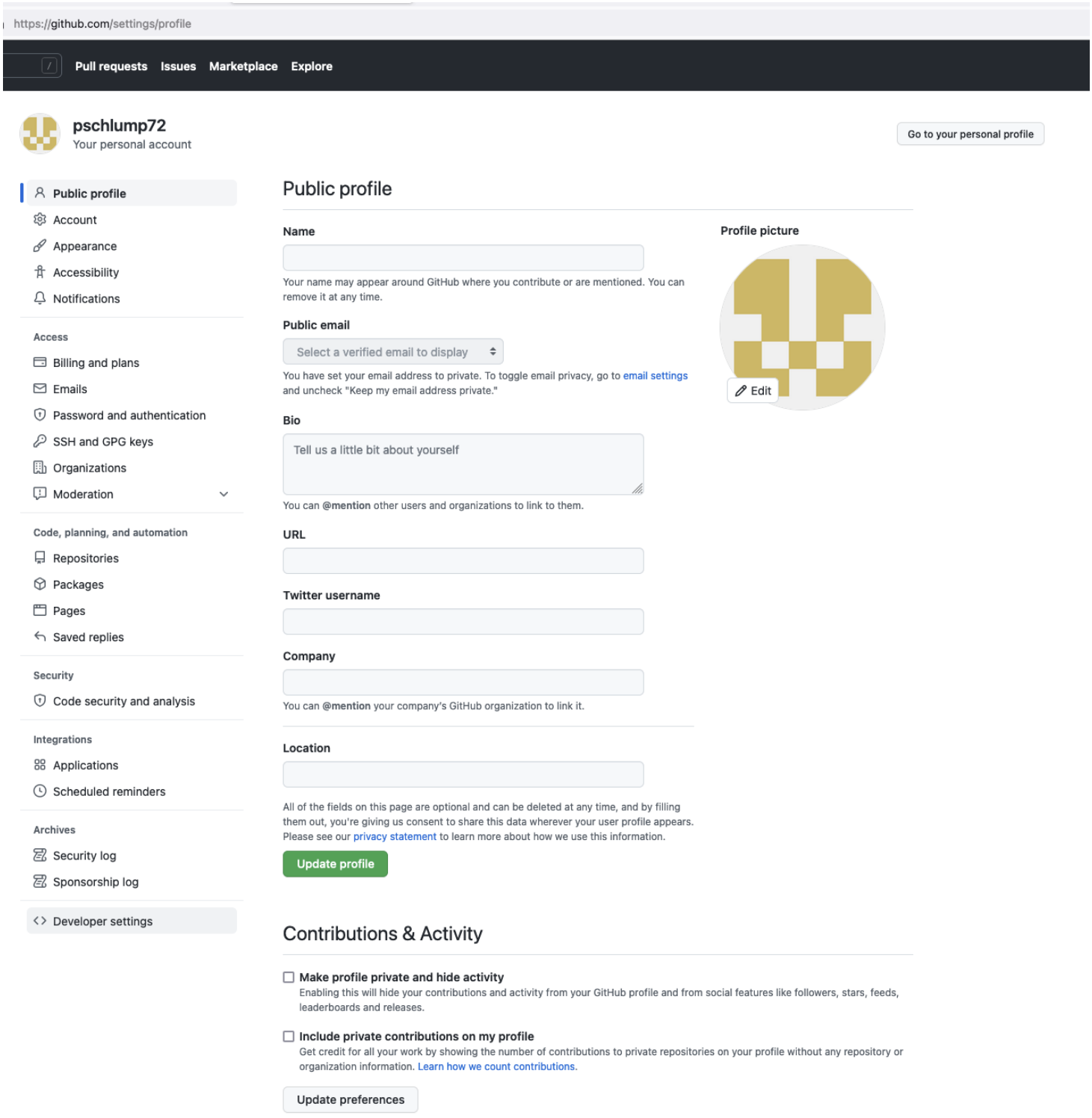
Step 013. Click on the little round image on the top right, there is a dropdown that should show

Your gists
Upgrade
Feature preview
Settings
Sign Out

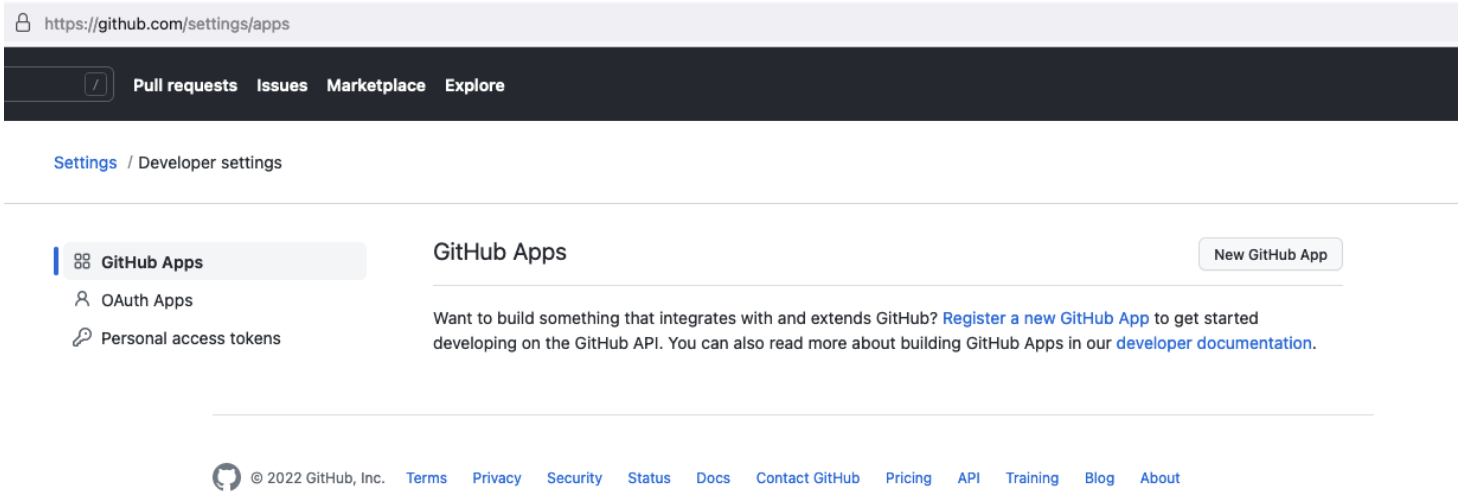
Pick on Settings. In the previous screen capture I have the drop down open.


https://github.com/settings/profile
Pull requests
Issues
Marketplace
Explore
pschlump72
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
SSH and GPG keys
Organizations
Moderation
Code, planning, and automation
Repositories
Packages
Pages
Saved replies
Security
Code security and analysis
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings
Public profile
Name
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.
Public email
Select a verified email to display
You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."
Bio
Tell us a little bit about yourself
You can @mention other users and organizations to link to them.
URL
Twitter username
Company
You can @mention your company's GitHub organization to link it.
Location
All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.
Update profile
Profile picture
Edit
Contributions & Activity
Make profile private and hide activity
Enabling this will hide your contributions and activity from your GitHub profile and from social features like followers, stars, feeds, leaderboards and releases.
Include private contributions on my profile
Get credit for all your work by showing the number of contributions to private repositories on your profile without any repository or organization information. Learn how we count contributions.
Update preferences
© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

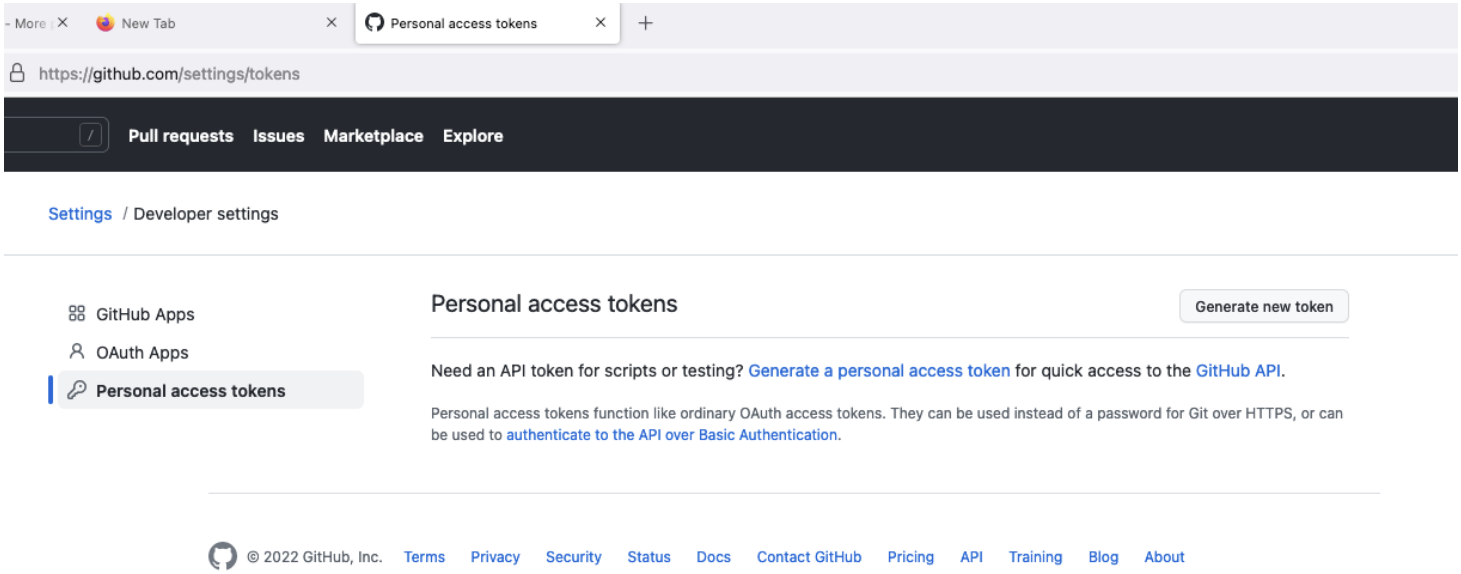
Step 014. This brings up your Profile page. Click on `<> Developer setting` on the menu on the left at the very bottom.



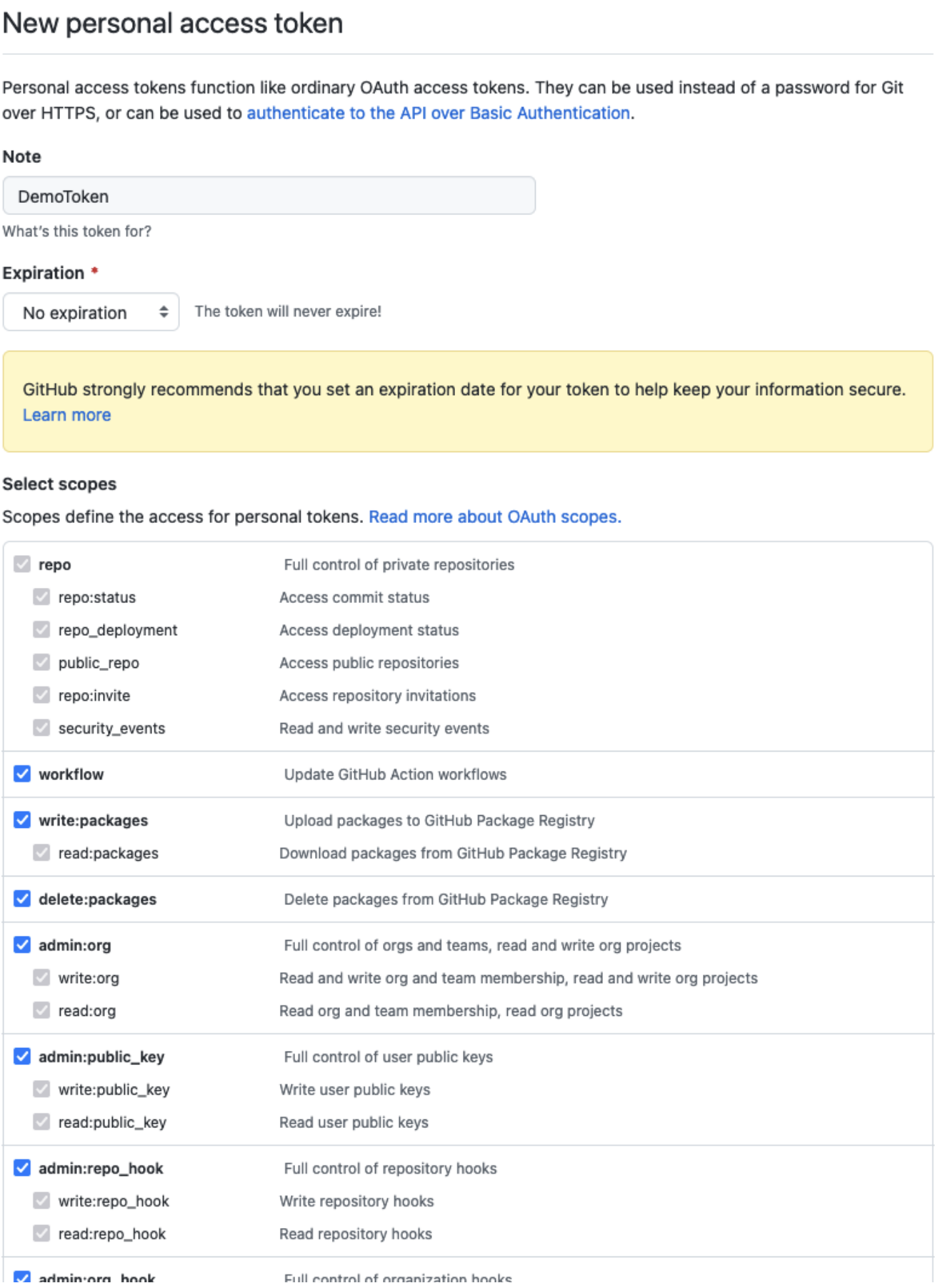
Step 015. You should get a menu with 3 items, “Github Apps”, “OAuth Apps”, “Personal Access Tokens” on the left. You want the last one, “Personal access tokens” click on that.



Step 016. You should see this page - click on the white button on the right, `Generate new token` .



Step 017. You should see a form that has a huge number of checkboxes - and a “Note” field. Fill in the note with a title for this token and pick an expiration date. I usually make my tokens last for a year. I check all the boxes that are not indented as shown.



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

DemoToken

What's this token for?

Expiration *

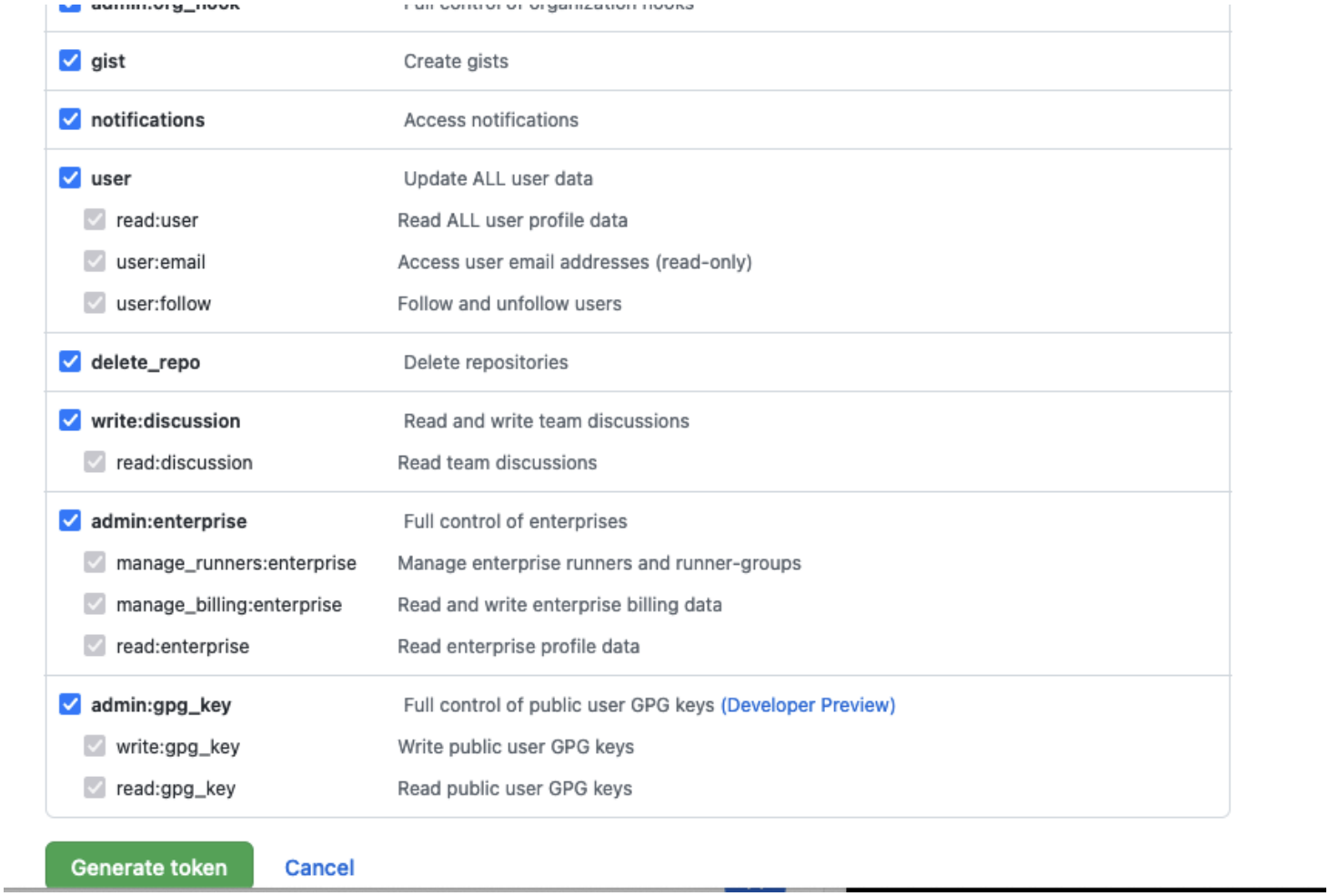
No expiration ⇕ The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. Learn more

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☑ security_events	Read and write security events
☑ **workflow**	Update GitHub Action workflows
☑ **write:packages**	Upload packages to GitHub Package Registry
☑ read:packages	Download packages from GitHub Package Registry
☑ **delete:packages**	Delete packages from GitHub Package Registry
☑ **admin:org**	Full control of orgs and teams, read and write org projects
☑ write:org	Read and write org and team membership, read and write org projects
☑ read:org	Read org and team membership, read org projects
☑ **admin:public_key**	Full control of user public keys
☑ write:public_key	Write user public keys
☑ read:public_key	Read user public keys
☑ **admin:repo_hook**	Full control of repository hooks
☑ write:repo_hook	Write repository hooks
☑ read:repo_hook	Read repository hooks
☑ **admin:org_hook**	Full control of organization hooks


admin:org_hook
Full control of organization hooks
gist
Create gists
notifications
Access notifications
user
Update ALL user data
read:user
Read ALL user profile data
user:email
Access user email addresses (read-only)
user:follow
Follow and unfollow users
delete_repo
Delete repositories
write:discussion
Read and write team discussions
read:discussion
Read team discussions
admin:enterprise
Full control of enterprises
manage_runners:enterprise
Manage enterprise runners and runner-groups
manage_billing:enterprise
Read and write enterprise billing data
read:enterprise
Read enterprise profile data
admin:gpg_key
Full control of public user GPG keys (Developer Preview)
write:gpg_key
Write public user GPG keys
read:gpg_key
Read public user GPG keys
Generate token
Cancel

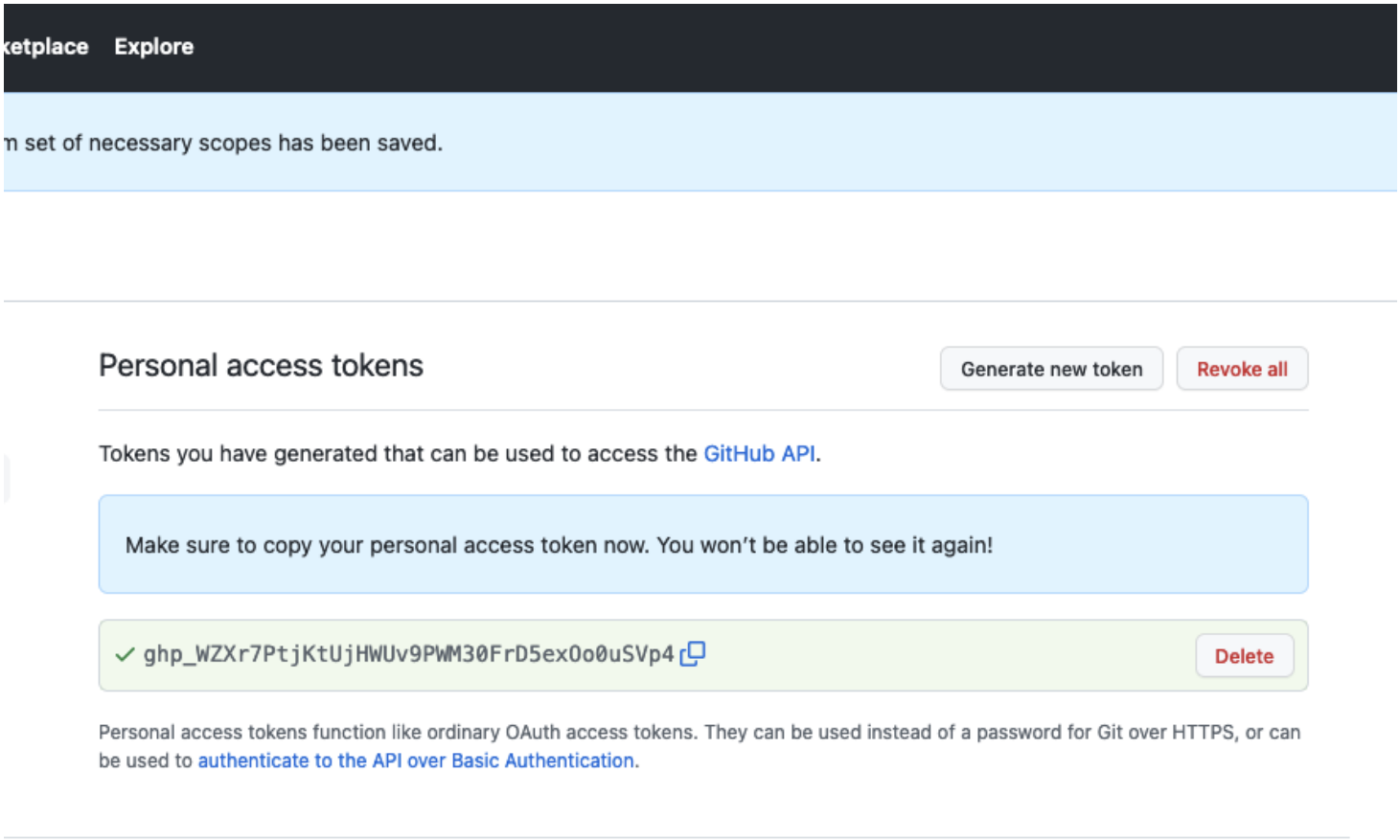
Step 018. You get a token in the light green box. Copy the token. Github will not have a copy of the token and it can not show it to you again. If you loose it you will need to create a new token. (This is not the token that I use - I deleted this one and created a different one for this demo) Put the token in a file that is *not* in your repository. Usually it is a good idea to put the token with it's expiration date into the same file that you saved your random login password in.

To use the token you have to save it in the configuration for this repository. I did the folowing

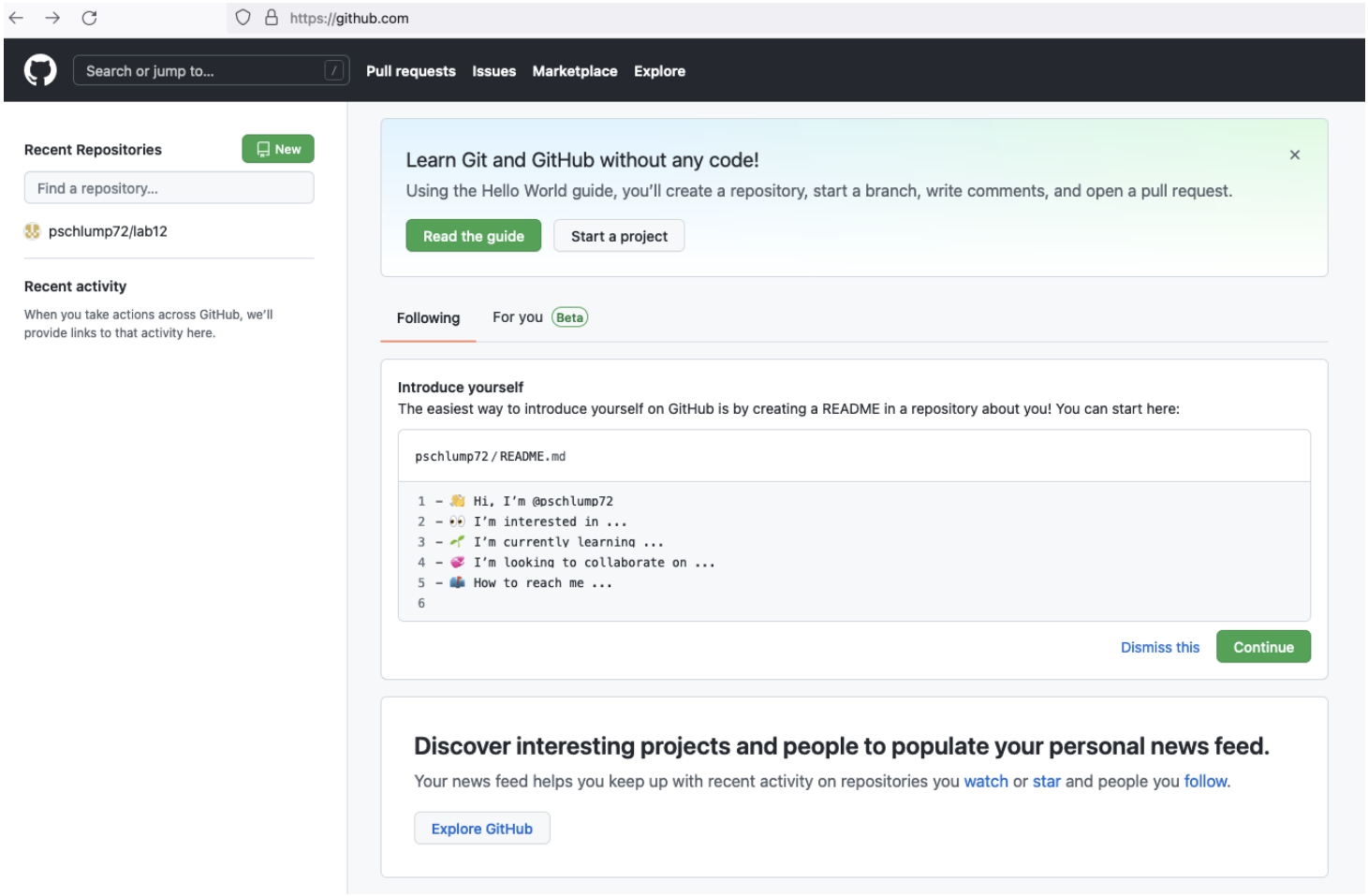
```
$ git remote remove origin
$ git remote add origin https://ghp_B6900zDLykPMv3VlJYbF3Mz2jBPSoa4gwhHY@github.com/pschlump72/Lab12
$ git push -u origin main
```

The command with the token is `git remote add origin https://[token]@github.com/[user]/[repository]` with the appropriate stuff substituted in.

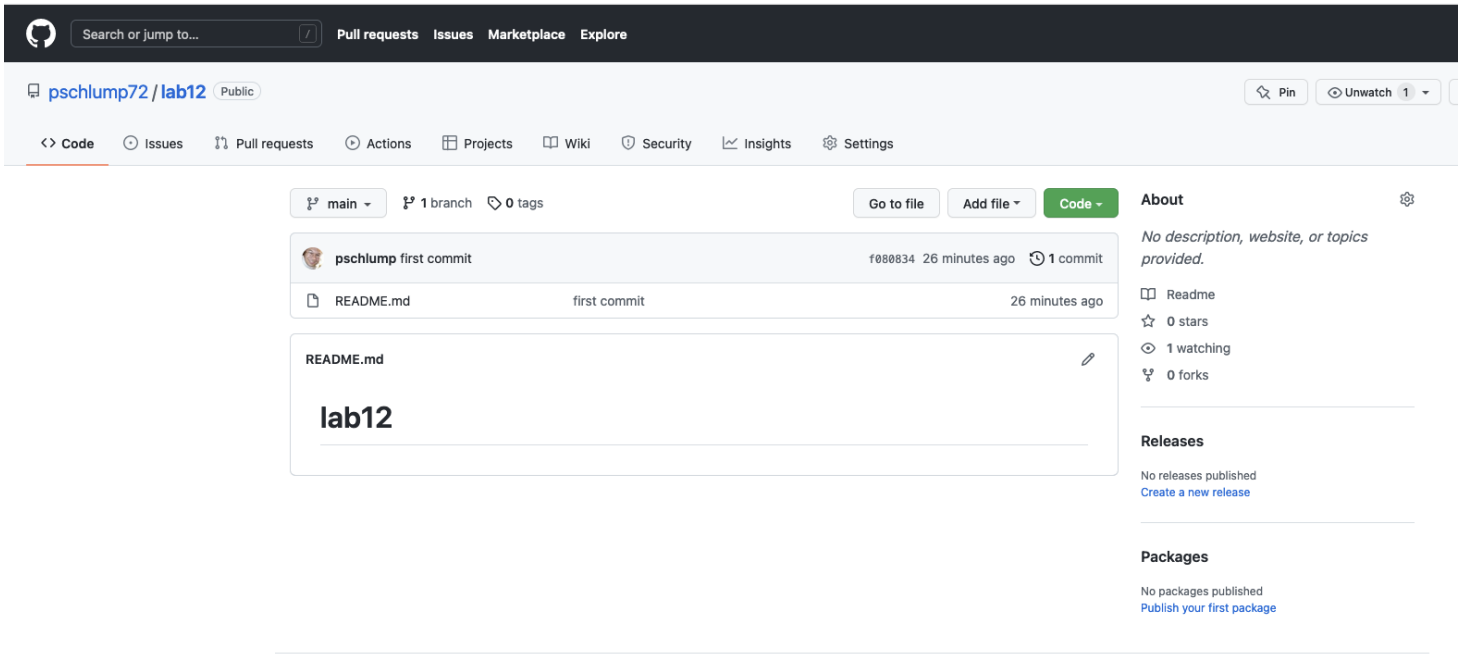
This is saved in a configuration file in `.git/config` . You can edit the file and see what is in it. Careful to not change it!



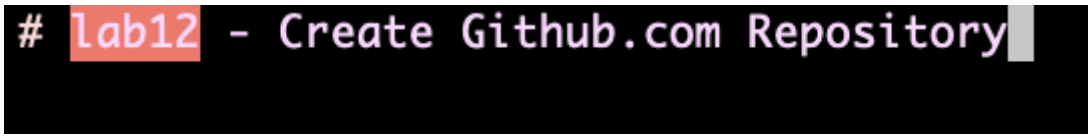
Step 019. Now click on the “octa-cat” on the top left side to get back to the main github.com page. On the left you should see a list of your new repository - I named my Lab12, click on that.



Step 020. You should see the nearly empty readme.md file.



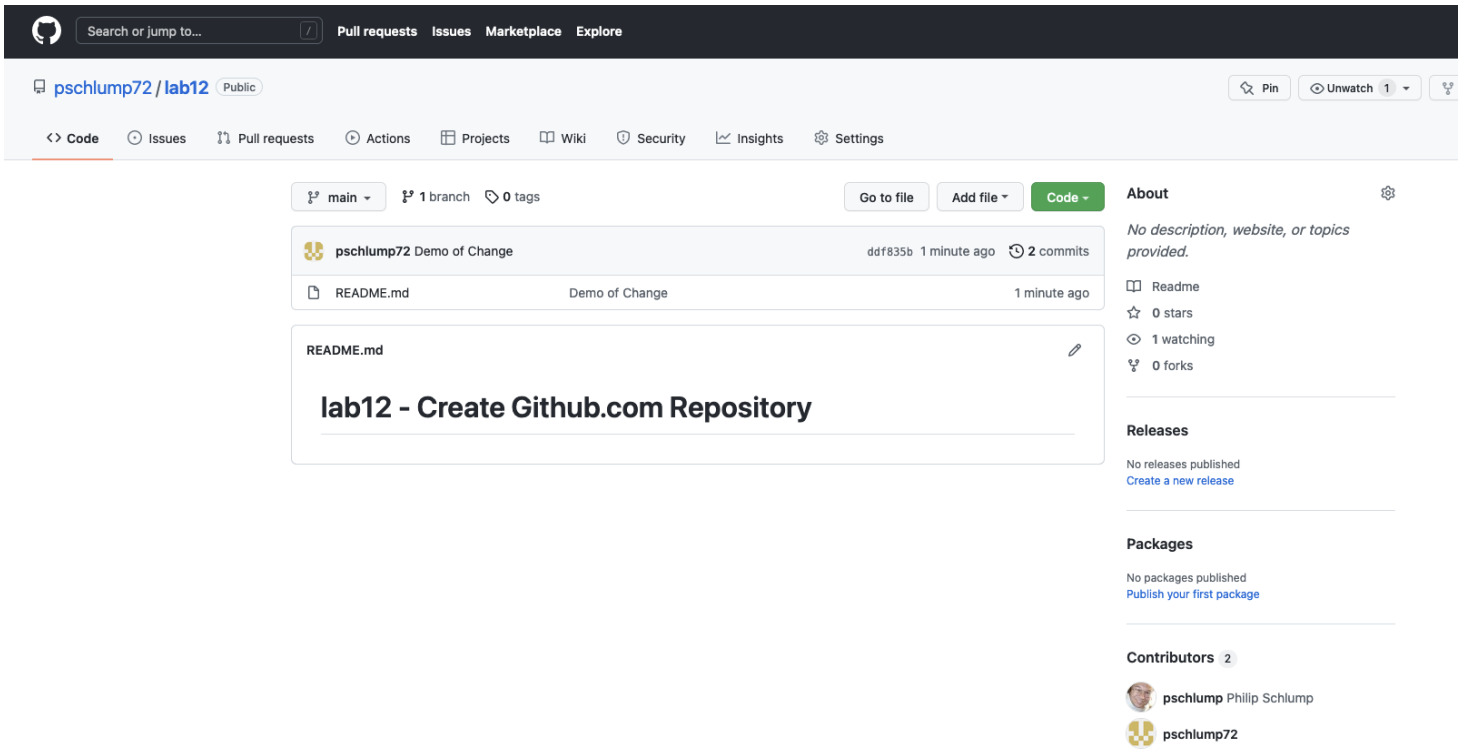
Step 021. Go and edit the file.



Step 022. Add the modifications

```
(base) philip@victoria Lab12 %
(base) philip@victoria Lab12 % git add -A README.md
(base) philip@victoria Lab12 % git commit -m "Demo of Change"
[main ddf835b] Demo of Change
 1 file changed, 2 insertions(+), 1 deletion(-)
(base) philip@victoria Lab12 % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: This repository moved. Please use the new location:
remote:   https://github.com/pschlump72/lab12.git
To https://github.com/pschlump72/Lab12
   f080834..ddf835b  main -> main
(base) philip@victoria Lab12 %
```

Step 023. Save them to github.com



Yea! You are now a git user that can modify stuff and share it and save it.

Create our Python Project Code

First let's get the display class working. The code is: https://github.com/Univ-Wyo-Education/S22-1010/blob/main/class/lect/Lect-27/ins_hm/display/display.py

Create a directory in our project called 'display' and one called 'words'.

```
$ cd Lab12              # if you are not already there
$ mkdir display
$ mkdir words
$ cd display
$ pwd
```

Use VSCode - navigate to the "display" directory and cut/paste the code for it into a file called display.py. Save it and then we can test it.

Run the code. Enter some numbers between 0 and 7 and see that it shows the correct output.

At the command line add your file to the repository.

```
$ git add display.py
$ git commit -m "Shows the ASCII art image"
$ git push
```

Go and edit the code and put a comment in it - explain what it is doing.

Now we will push the changes up.

```
$ git add display.py
$ git commit -m "Added Comment"
$ git push
```

You should be able to see the log on the changes that you have made.

```
$ git log
```

and go to https://github.com and look at your directory and changes online.

Add the "words" code.

Get to the correct directory.

```
$ cd ../words
$ pwd
```

Do a similar process to create the `pick_word.py` file, it is on github.com at: https://github.com/Univ-Wyo-Education/S22-1010/blob/main/class/lect/Lect-27/ins_hm/words/pick_word.py

It is missing some code in the ‘guessLetter’ method/function a line 647. The code is to record a picked letter. We will add the code in a second.

First create and push up the file. Navigate to the correct directory in VSCode. Create the file so that it is in the ./words directory. The file needs to be named `pick_word.py` . If you do a `ls` command you should see

```
$ ls
pick_word.py
$ ls ../display
display.py
$ ls ..
README.md
display
words
```

Let’s save the file to github.com, the we will add the missing code and push it up a 2nd time.

```
$ git add pick_word.py
$ git commit -m "initial code with some missing"
$ git push
```

Let’s add the missing code to

In VSCode change the function

```
    def guessLetter ( self, letter ) :
        return False
```

To look like this. The function is near line 647

```
    def guessLetter ( self, letter ) :
        if letter not in self.letters_picked:
            self.letters_picked = self.letters_picked + letter
        if letter in self.word:
            self.n_success += 1
        else:
            self.n_incorrect_letters += 1
        if self.n_incorrect_letters > 8:
            return True
        return False
```

Checkin the result

Test the code interactively by running it.

Now save the changes

```
$ git add pick_word.py
$ git commit -m "Fixed gessLetter to save a guess"
$ git push
```

You should now see both sets of code on you https://github.com site.

The main program.

A similar process. https://github.com/Univ-Wyo-Education/S22-1010/blob/main/class/lect/Lect-27/ins_hm/hangman.py

Copy the code.

Go up 1 level and edit in VSCode to create the file.

```
$ cd ..
$ ls
README.md
display
hangman.py
words
```

You should be able to run the hangman.py code now.

Give it a test.

now add it to the repository.

```
$ git add hangman.py
$ git commit -m "Main program added"
$ git push
```

Capture Screen - to turn in for Lab

Yes just do a screen capture of your `https://github.com/[User]/Lab12` and turn that in. (Use your username in the URL)