INTRO TO GIT

1

- Git is a version control software created by the same guy developed the Linux operating system
  - Enables many people collaborate on large projects
  - Keep eye on modification done by whom and when and what etc.
  - If something goes wrong scroll back where you had stable code
- It is very useful not only for our development but also to collaborate/contribute/clone many large projects

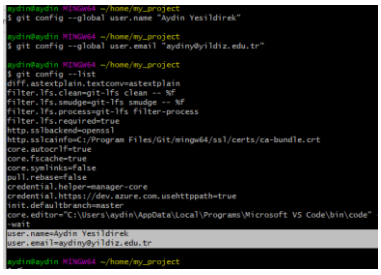
2

Where to Get it?

- The official web site
- Download at https://git-scm.com/downloads
- To earn more about Git https://git-scm.com/book/en/v2

3

Set it up first



- You need to define yourself with your name and email
- Collaborators will know you with the name and contact you by the email you provide
- In a shell window type

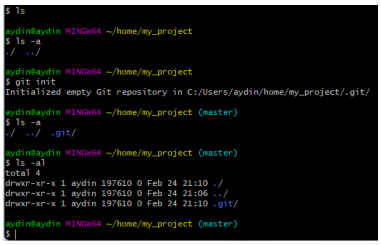
```
$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com
```

- Check it out

```
$ git config --list
```

4

Our First Project

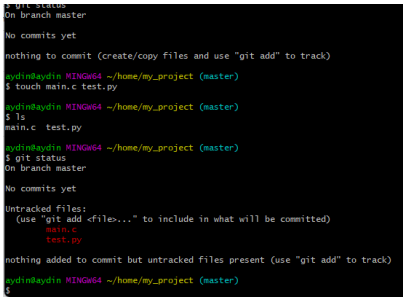


- In terminal type

```
$ mkdir  ~/my_project
$ cd  ~/my_project
$ git init
$ ls
$ ls -a
$ ls -al
```

5

CREATE SOME FILES



- Let's create some (empty) files

```
$ touch main.c
$ touch test.py
$ ls
$ git status
```

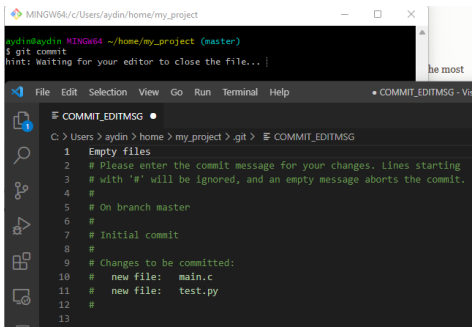
6

STAGING PHASE

```
aydin@aydin MINGW64 ~/home/my_project (master
$ git add main.c test.py

aydin@aydin MINGW64 ~/home/my_project (master
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage
        new file:   main.c
        new file:   test.py


aydin@aydin MINGW64 ~/home/my_project (master
```

- $ **git add** main.c test.py
- $ **git status**

Create a Version



- `$ git commit`
- The edit the file popping to add the message to the log, e.g. `Empty files`
- You could have done so by single line command
- `$ git commit -m "Empty files"`

```
$ git commit
[master (root-commit) 1a111a7] Empty files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.c
 create mode 100644 test.py
```

History Log

- $ git log

```
$ git log
commit 1a111a79ade87122056cc285b2154ccf24efd235 (HEAD -> master)
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:17:17 2022 +0300

    Empty files
```

9

While we continue to work

- Let's assume that we continue to the development and have more files created
- $ touch file1 file2 file3
- $ git status

```
$ touch file1 file2 file3

aydin@aydin MINGW64 ~/home/my_project (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
        file2
        file3

nothing added to commit but untracked files present (use "git add" to track)
```

10

Define a New version

- $ git add .
- $ git commit
- $ git log

```
$ git add .

aydin@aydin MINGW64 ~/home/my_project (master)
$ git commit
[master 635ca49] Created more files
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
 create mode 100644 file2
 create mode 100644 file3

aydin@aydin MINGW64 ~/home/my_project (master)
$ git log
commit 635ca49e6e57bc118e58f8d8166e7c22a502cee9 (HEAD -> master)
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:30:01 2022 +0300

    Created more files

commit 1a111a79ade87122056cc285b2154ccf24efd235
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:17:17 2022 +0300

    Empty files

aydin@aydin MINGW64 ~/home/my_project (master)
```

11

Add more and Modify files

- Add more files and modify the main.c file
- Check the git status
- Stage all files
- Commit to new version

```
$ touch a1 a2 a3

aydin@aydin MINGW64 ~/home/my_project (master)
$ git add .

aydin@aydin MINGW64 ~/home/my_project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   a1
        new file:   a2
        new file:   a3
        modified:   main.c


aydin@aydin MINGW64 ~/home/my_project (master)
$ git commit -m "more files and modifications"
[master f862e2b] more files and modifications
 4 files changed, 4 insertions(+)
 create mode 100644 a1
 create mode 100644 a2
 create mode 100644 a3

aydin@aydin MINGW64 ~/home/my_project (master)
```

12

Going Back to an Older Version

```
$ git log
commit f862e2bd631ac94ebe59f5f2752a0989d249fbca (HEAD -> master)
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:40:21 2022 +0300

    more files and modifications

commit 635ca49e6e57bc118e58f8d8166e7c22a502cee9
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:30:01 2022 +0300

    Created more files

commit 1a111a79ade87122056cc285b2154ccf24efd235
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:17:17 2022 +0300

    Empty files

aydin@aydin MINGW64 ~/home/my_project (master)
$
```

Check the log history and switch to the first version

```
$ git checkout 1a111a79ade87122056cc285b2154ccf24efd235

Note: switching to '1a111a79ade87122056cc285b2154ccf24efd235'.

You are in 'detached HEAD' state. You can look around, make
experimental

changes and commit them, and you can discard any commits you
make in this

state without impacting any branches by switching back to a
branch.

If you want to create a new branch to retain commits you create,
you may

do so (now or later) by using -c with the switch command.
Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable
advice.detachedHead to false

HEAD is now at 1a111a7 Empty files
```

13

Back to the future

- Verify that the modified files are now empty as in the first version. You can change it as you wish
- Going back to the master version type

```
$ git checkout master
Previous HEAD position was 1a111a7 Empty files
Switched to branch 'master'
```

- Check the modified files, they are not empty now

14

Branches

- You may want to have parallel project files separately maintained
  - For development
  - For bug fixes
  - For adding new features
- Use

```
$ git branch bug101
$ git checkout bug101
```

- Edit or fix the bug in the switched branch bug101

15

Work on branch

```
$ git commit -m "Bug101 is fixed"
[bug101 b6eb995] Bug101 is fixed
 1 file changed, 1 insertion(+), 1 deletion(-)

aydin@aydin MINGW64 ~/home/my_project (bug101)
$ git log
commit b6eb9955ff25ba7eb3dc7ea488393ff9b6ebd58c (HEAD -> bug101)
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 22:00:09 2022 +0300

    Bug101 is fixed

commit f862e2bd631ac94ebe59f5f2752a0989d249fbca (master)
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:40:21 2022 +0300

    more files and modifications

commit 635ca49e6e57bc118e58f8d8166e7c22a502cee9
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:30:01 2022 +0300

    Created more files

commit 1a111a79ade87122056cc285b2154ccf24efd235
Author: Aydin Yesildirek <aydiny@yildiz.edu.tr>
Date:   Thu Feb 24 21:17:17 2022 +0300

    Empty files

aydin@aydin MINGW64 ~/home/my_project (bug101)
$
```

16

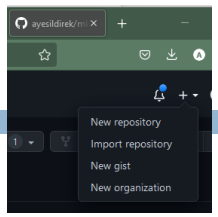
Merging

- Switch to the master branch and compare the log
- It had bug in it
- Now, we will want to apply the changes by merging the bug fix code
- Let's, go back to the master branch

```
$ git checkout master
$ git merge bug101
Updating f862e2b..b6eb995
Fast-forward
 main.c | 2 +-
 1 file changed, 1
insertion(+), 1 deletion(-)
```

17

Repositories

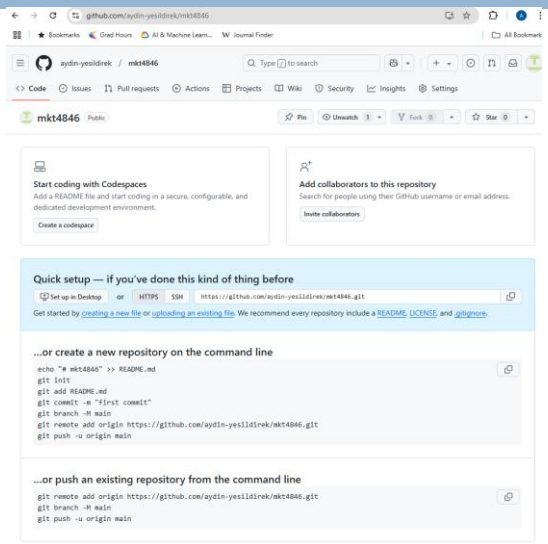


- Some cloud repos supporting git are
  - Github
  - Bitbucket
  - Gitlab
  - Etc.
- To share the project with collaborators over internet
- Some require extra security measures to pair your PC with your cloud account

- Create a user account on github.com if you don't have one
- Create a "New repo.." by clicking the plus sign
- Say mkt4846
- Select Public or Private
- Click the "Create repo" button

18

- Add collaborators
  - Teammates
  - Instructor
- Codespace?
- Copy the url later



19

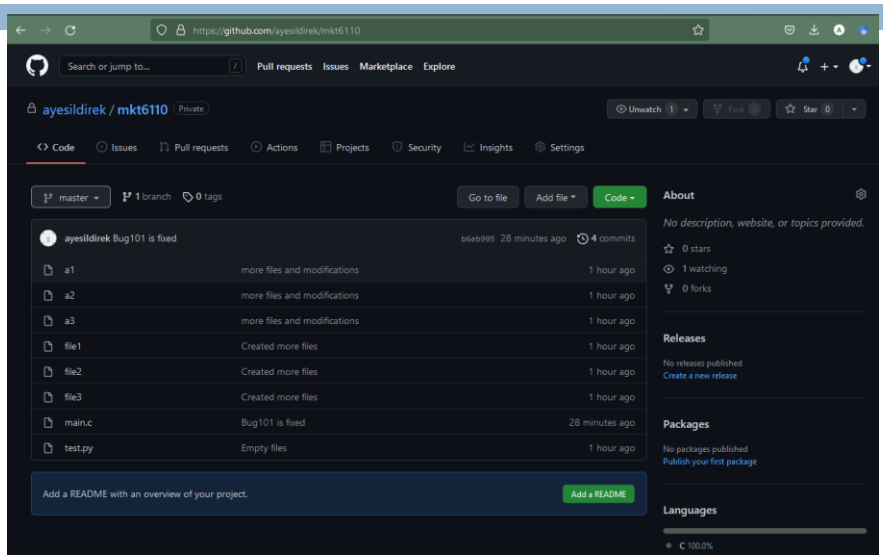
Copy to a Remote Repo

On your PC, paste the url to

```
$ git remote add origin  https://github.com/ayesildirek/mkt4846.git
# Sets the new remote location
$ git remote -v
# Verifies the new remote URL
$ git push origin master
# Pushes the changes in your local master branch to the remote repository at the origin
```

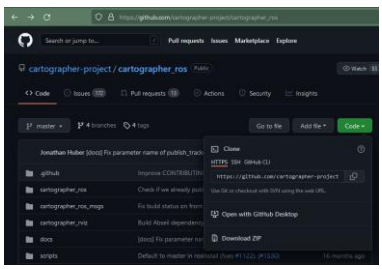
- Make sure you're signed in to github.com

20



21

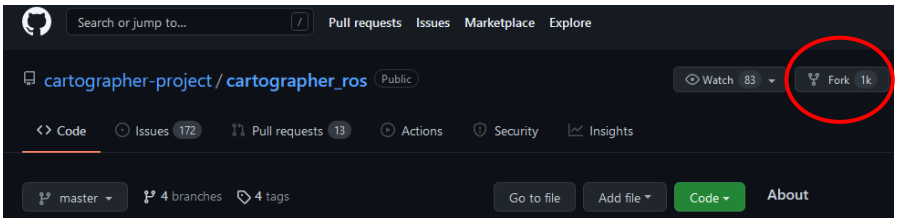
Copy from a Remote Repo



- One can clone the repo of interest to local machine or his own remote repo site
- For instance, a comp efficient slam solution, cartographer integrated with ROS
- https://github.com/cartographer-project/cartographer_ros.git
- You can get a copy of the repo to your own local machine's `catkin_ws/src` directory
- Copy the https address thru the `Code` button

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/cartographer-project/cartographer_ros.git
```

22



23

Pull Requests

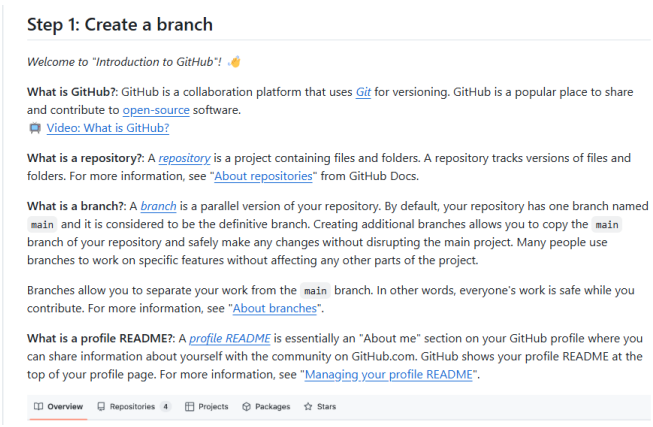
- In git-based collaboration pipeline, you may propose your own contributions to a repo based project to be reviewed, accepted and merged into the main branch by following the steps
  1. Fork or clone a repo
  2. Create your own branch
  3. Make and commit your proposed changes
  4. Open a **pull request**
  5. Others/owners review the proposal and make a discussion whether to include it or not
  6. Merge it if approved

24

Online Practice

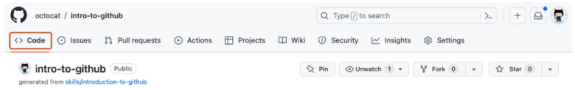
- Clone the intro page https://github.com/skills/introduction-to-github.git
- Go to your cloned page and follow the steps
- Open a new browser tab and navigate to your newly made repository. Then, work on the steps in your second tab while you read the instructions in this tab.

25

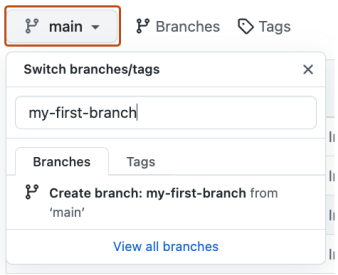


26

2. Navigate to the < > **Code** tab in the header menu of your repository.



3. Click on the **main** branch drop-down.



4. In the field, name your branch `my-first-branch`. In this case, the name must be `my-first-branch` to trigger the course workflow.

5. Click **Create branch: my-first-branch** to create your branch.

27

Step 2: Commit a file

You created a branch! 🎉

Creating a branch allows you to edit your project without changing the `main` branch. Now that you have a branch, it's time to create a file and make your first commit!

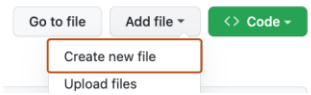
What is a commit?: A *commit* is a set of changes to the files and folders in your project. A commit exists in a branch. For more information, see "About commits".

Activity: Your first commit

The following steps will guide you through the process of committing a change on GitHub. A commit records changes in renaming, changing content within, creating a new file, and any other changes made to your project. For this exercise, committing a change requires first adding a new file to your new branch.

> **Note**
> `.md` is a file extension that creates a Markdown file. You can learn more about Markdown by visiting "Basic writing and formatting syntax" in our docs or by taking the "Communicating using Markdown" Skills course.

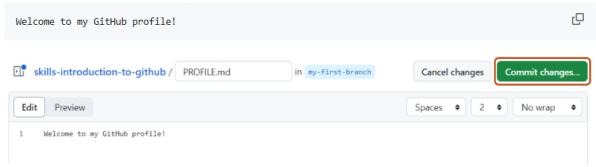
1. On the < > **Code** tab in the header menu of your repository, make sure you're on your new branch `my-first-branch`.
2. Select the **Add file** drop-down and click **Create new file**.



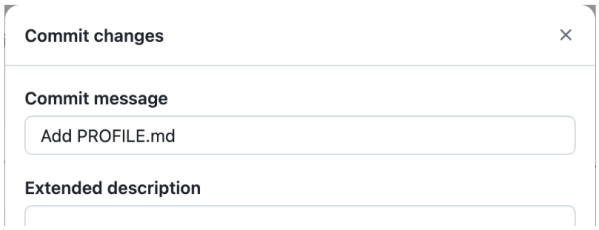
28

3. In the **Name your file...** field, enter `PROFILE.md`.

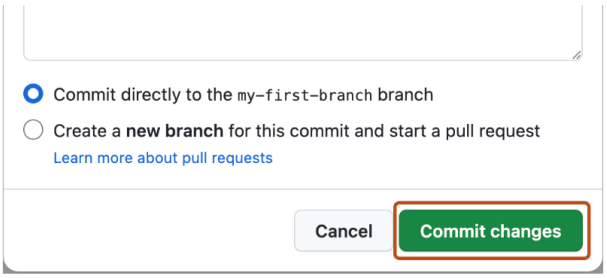
4. In the **Enter file contents here** area, copy the following content to your file:



5. Click **Commit changes...** in the upper right corner above the contents box. For commits, you can enter a short commit message that describes what changes you made. This message helps others know what's included in your commit. GitHub offers a simple default message, but let's change it slightly for practice. First, enter `Add PROFILE.md` in the first text-entry field titled "Commit message".



29



6. In this lesson, we'll ignore the other fields and click **Commit changes**.

7. Wait about 20 seconds then refresh this page (the one you're following instructions from). GitHub Actions will automatically update to the next step.

30

Step 3: Open a pull request

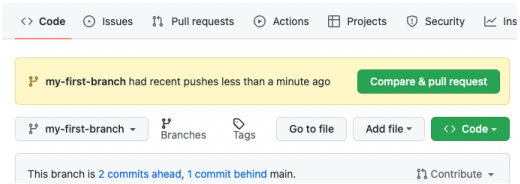
Nice work making that commit! ✨

Now that you have made a change to the project and created a commit, it's time to share your proposed change through a pull request!

What is a pull request?: Collaboration happens on a *pull request*. The pull request shows the changes in your branch to other people and allows people to accept, reject, or suggest additional changes to your branch. In a side by side comparison, this pull request is going to keep the changes you just made on your branch and propose applying them to the `main` project branch. For more information about pull requests, see "About pull requests".

Activity: Create a pull request

You may have noticed after your commit that a message displayed indicating your recent push to your branch and providing a button that says **Compare & pull request**.



31

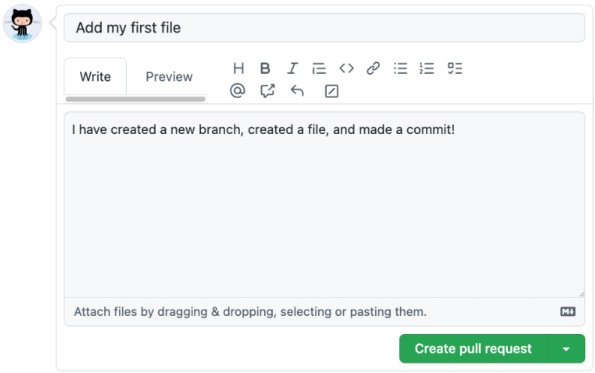
To create a pull request automatically, click **Compare & pull request**, and then skip to step 6 below. If you don't click the button, the instructions below walk you through manually setting up the pull request.

1. Click on the **Pull requests** tab in the header menu of your repository.
2. Click **New pull request**.
3. In the **base:** dropdown, make sure **main** is selected.
4. Select the **compare:** dropdown, and click `my-first-branch`.

5. Click **Create pull request**.
6. Enter a title for your pull request. By default, the title will automatically be the name of your branch. For this exercise, let's edit the field to say `Add my first file`.
7. The next field helps you provide a description of the changes you made. Here, you can add a description of what you've accomplished so far. As a reminder, you have: created a new branch, created a file, and made a commit.

32

7. The next field helps you provide a description of the changes you made. Here, you can add a description of what you've accomplished so far. As a reminder, you have: created a new branch, created a file, and made a commit.



8. Click **Create pull request**. You will automatically be navigated to your new pull request.

9. Wait about 20 seconds then refresh this page (the one you're following instructions from). GitHub Actions will automatically update to the next step.

33

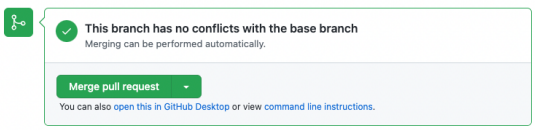
Step 4: Merge your pull request

Nicely done! 😎

You successfully created a pull request. You can now merge your pull request.

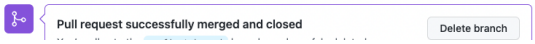
What is a merge?: A *merge* adds the changes in your pull request and branch into the `main` branch. For more information about merges, see "Merging a pull request."

As noted in the previous step, you may have seen evidence of GitHub Actions running which automatically progresses your instructions to the next step. You'll have to wait for it to finish before you can merge your pull request. It will be ready when the merge pull request button is green.



Activity: Merge the pull request

1. Click **Merge pull request**.
2. Click **Confirm merge**.
3. Once your branch has been merged, you don't need it anymore. To delete this branch, click **Delete branch**.



34

Copilot in GitHub

- Exercise
- Work with
  - https://github.com/skills/copilot-codespaces-vscode

35

GitHub Copilot Aided Coding in VS Code

1. Install GitHub Copilot Extensions in VS Code:

• Search for “GitHub Copilot” in “Extensions” and install it.

2. Login in to GitHub (or create a new account if you don’t have one):

• Make sure you have a GitHub account with CoPilot credentials

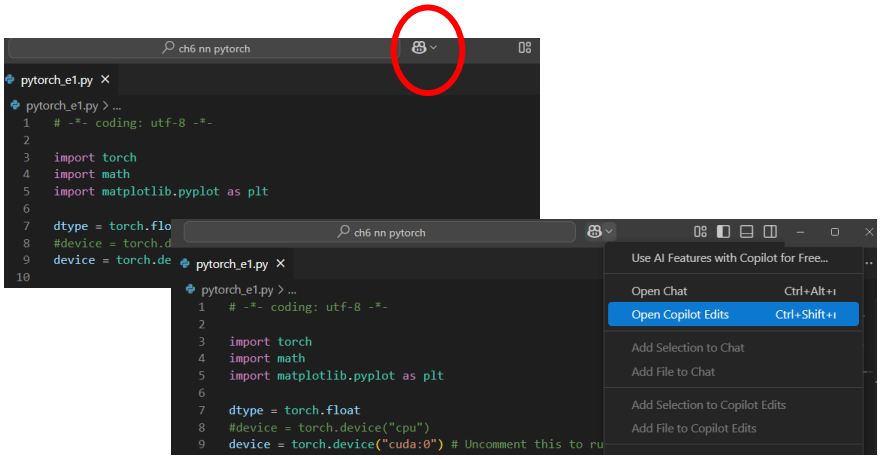
• Sign in to GitHub.

3. Development your code and be advised:

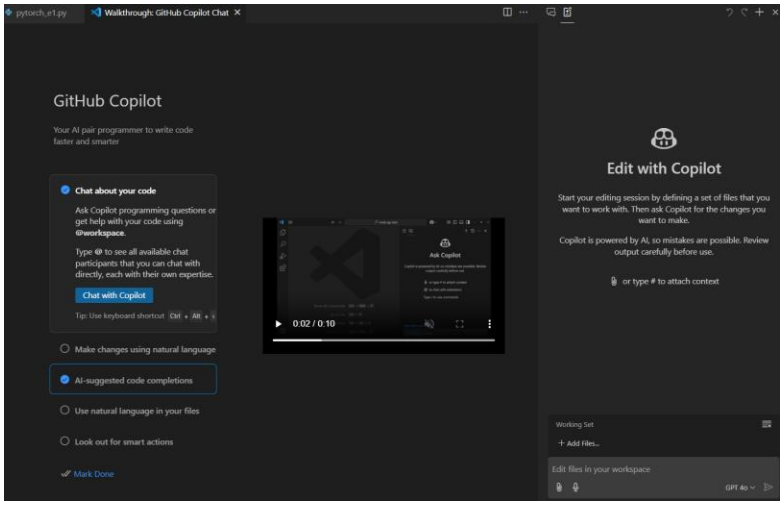
• Start writing a new code or open a file.

• When you type you should see that GitHub Copilot suggestions. Hit Tab Key

36

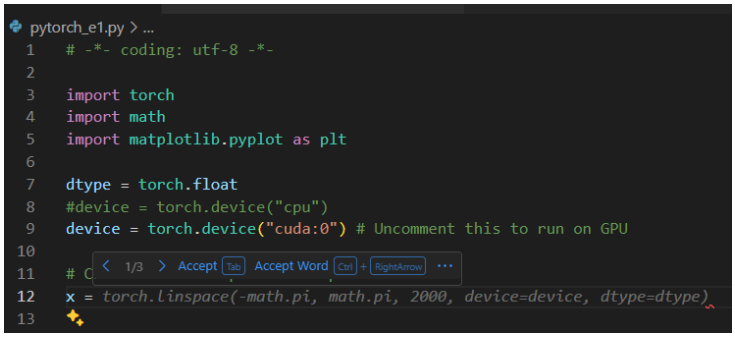

ch6 nn pytorch
pytorch_e1.py
pytorch_e1.py > ...
# -*- coding: utf-8 -*-
import torch
import math
import matplotlib.pyplot as plt
dtype = torch.flo
#device = torch.d
device = torch.de
ch6 nn pytorch
pytorch_e1.py
pytorch_e1.py > ...
# -*- coding: utf-8 -*-
import torch
import math
import matplotlib.pyplot as plt
dtype = torch.float
#device = torch.device("cpu")
device = torch.device("cuda:0") # Uncomment this to ru
Use AI Features with Copilot for Free...
Open Chat Ctrl+Alt+I
Open Copilot Edits Ctrl+Shift+I
Add Selection to Chat
Add File to Chat
Add Selection to Copilot Edits
Add File to Copilot Edits

37


pytorch_e1.py
Walkthrough: GitHub Copilot Chat
GitHub Copilot
Your AI pair programmer to write code faster and smarter
Chat about your code
Ask Copilot programming questions or get help with your code using @workspace.
Type @ to see all available chat participants that you can chat with directly, each with their own expertise.
Chat with Copilot
Tip: Use keyboard shortcut Ctrl + Alt + I
Make changes using natural language
AI-suggested code completions
Use natural language in your files
Look out for smart actions
Mark Done
0:02 / 0:10
Edit with Copilot
Start your editing session by defining a set of files that you want to work with. Then ask Copilot for the changes you want to make.
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.
or type # to attach context
Working Set
+ Add Files...
Edit files in your workspace
GPT 4o

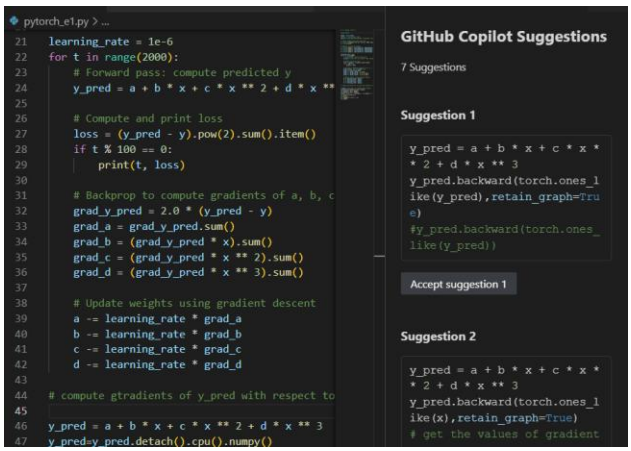
38

- Type '' x = '' at the beginning of your code with some comments
- See the suggestions from CoPilot to complete your code
- You may find similar projects from github repos etc.



39

- Hit Ctrl + Enter to get suggestions under a new function def or comment



40

- There are many other great usage of CoPilot in code devel
  - Add explanation of your code
  - Translate code from a language to another
  - Document your code
  - Clean
  - Fix bugs
  - Etc.
- It will certainly be a great indispenseble devel tool

41