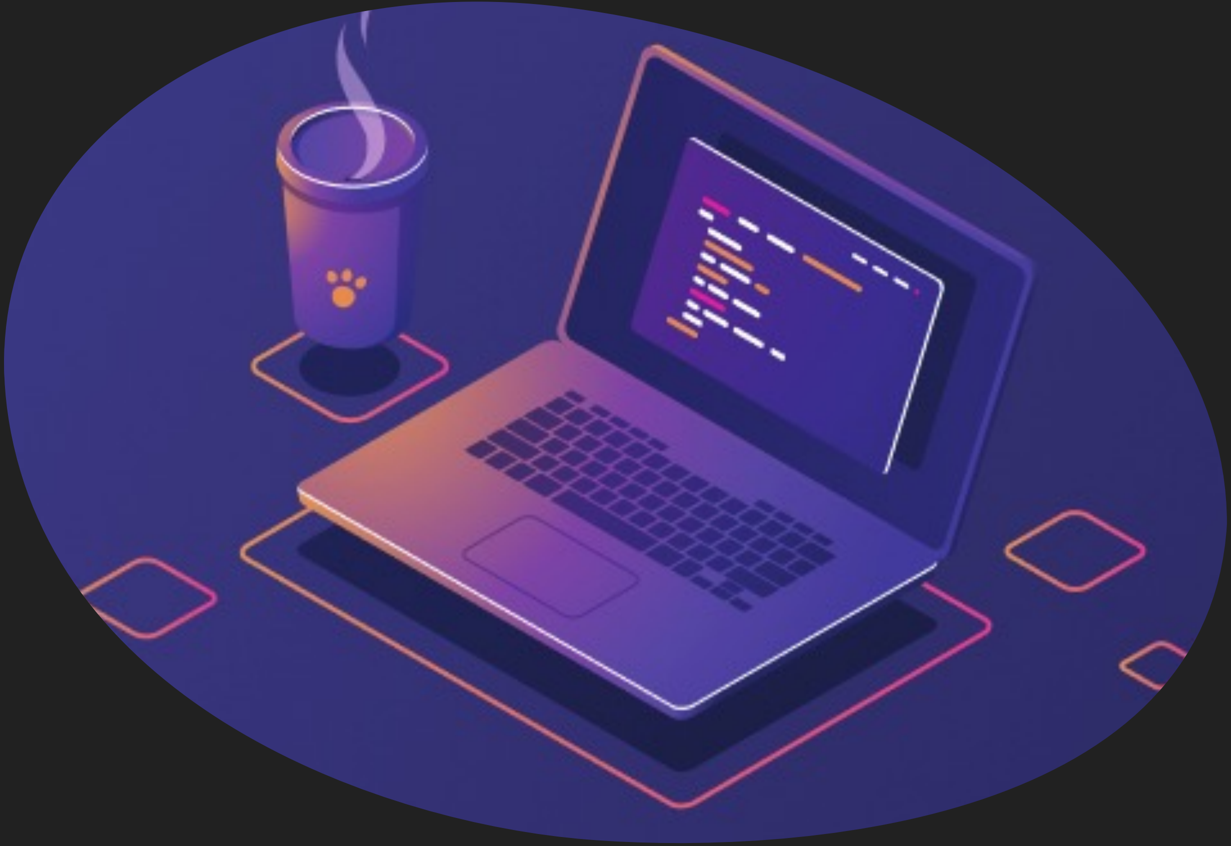


Week 2 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Week2 Getting Started with Git

- **Week2 Commands:**
  - Changing code in a Repository
    - **git add**
  - Committing these changes
    - **git commit**
  - Pushing or Pulling Changes
    - **git push** and **git pull**
  - Checking Status, Logs, and Changes
    - **git status**, **git log**, **git diff**

Week 2
Basic Git Usage

Week 2 Getting Started with Git

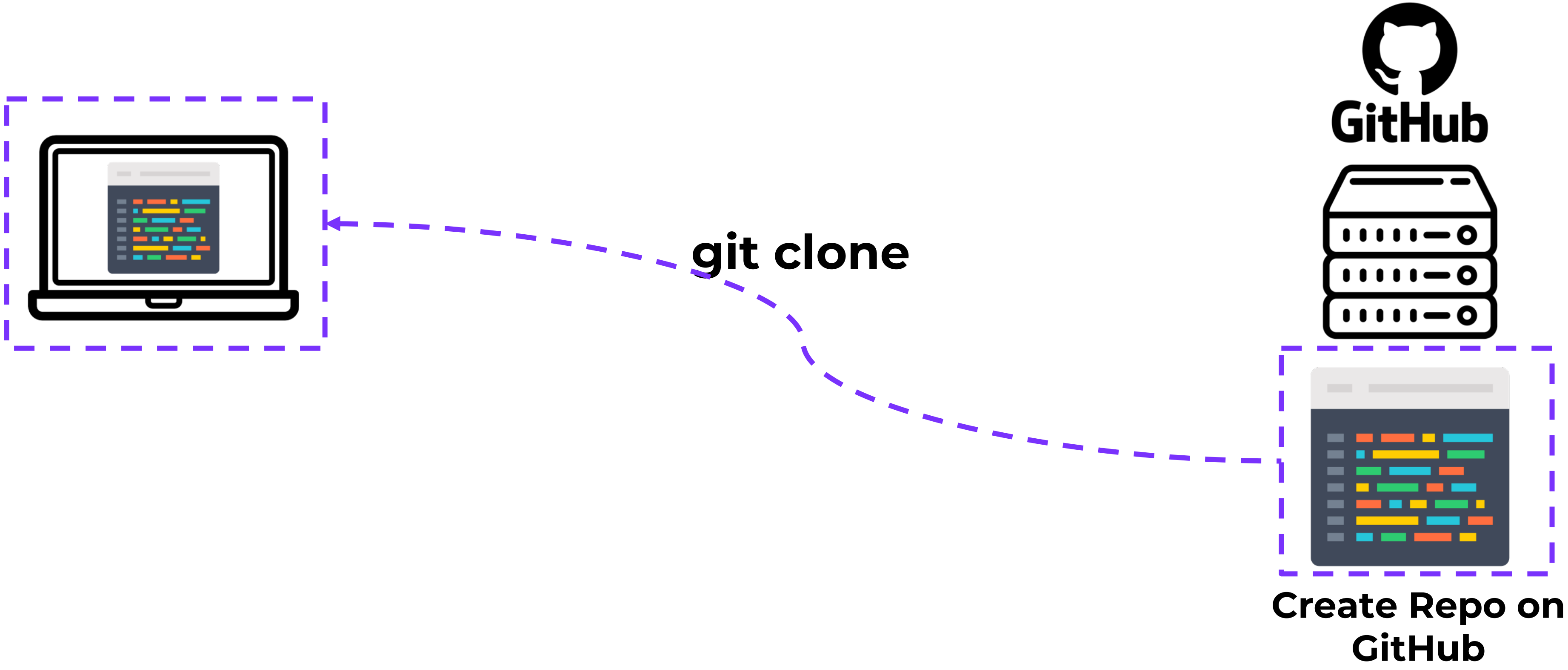
- **Basic Git Usage**
  - Let’s cover the basic cycle of a workflow of using Git and GitHub.
  - This particular basic example will assume just a solo developer and everything working on the same branch.
  - We’ll cover branches and working with others on Week 3.

Week 2 Getting Started with Git

Week 2 Getting Started with Git

GitHub
Create Repo on GitHub

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Week 2 Getting Started with Git

git init
GitHub

Week 2 Getting Started with Git

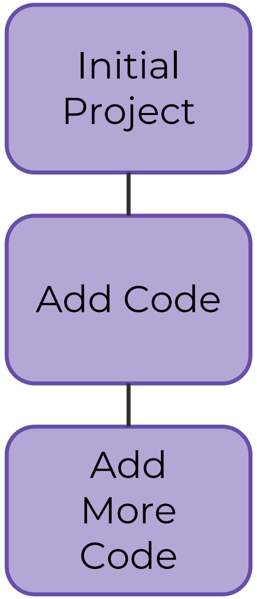
git init
GitHub

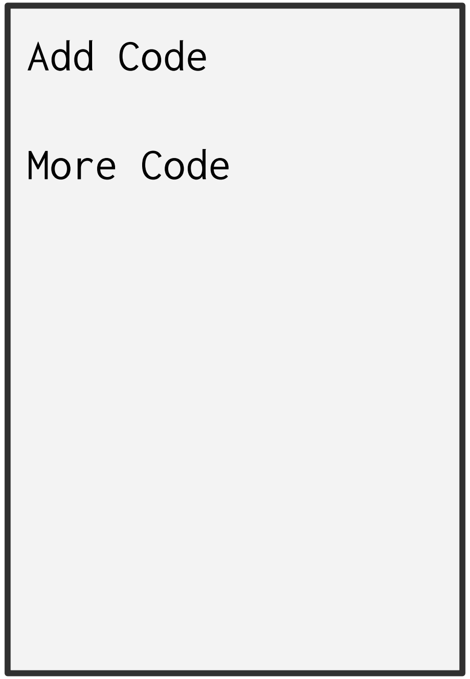
Week 2 Getting Started with Git

What we need to learn toWeek:

- Git Workflow
- How to tell Git about changes to our code
- How to push changes to GitHub
- How to pull changes from GitHub

Week 2 Getting Started with Git

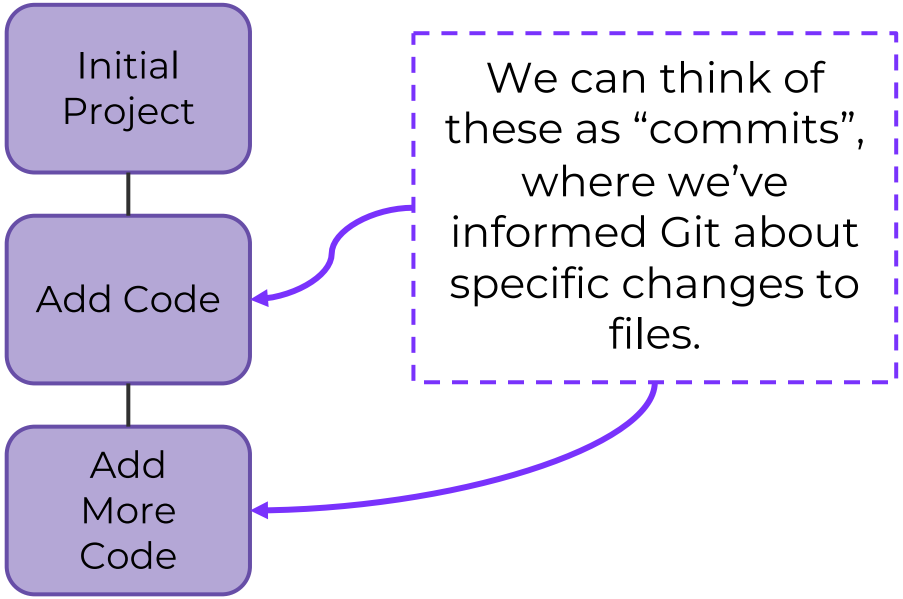




```
Add Code

More Code
```

Week 2 Getting Started with Git

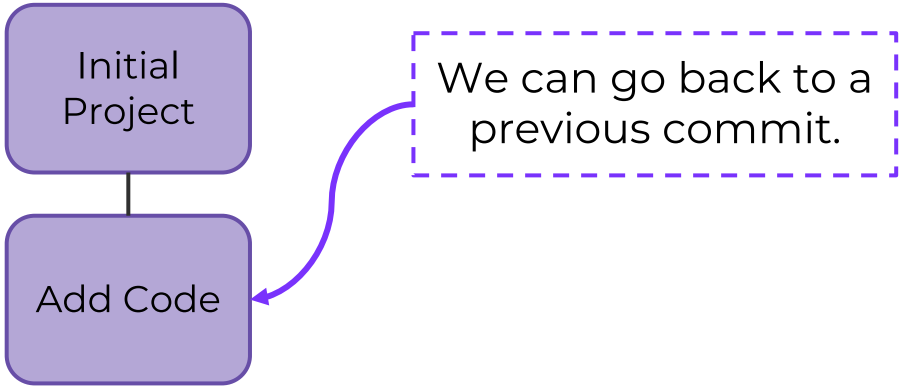


We can think of these as “commits”, where we’ve informed Git about specific changes to files.

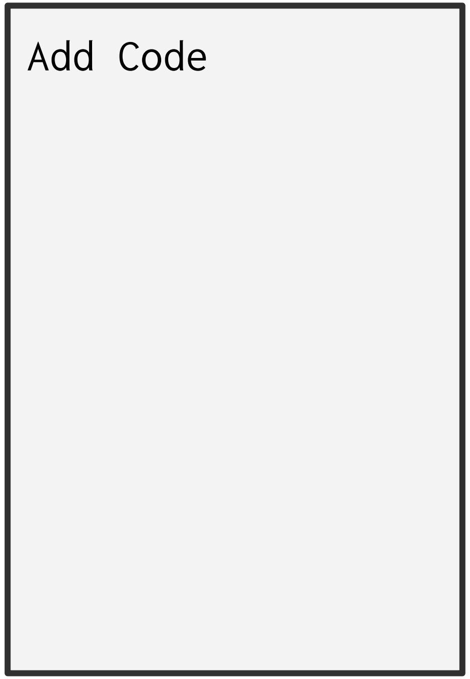
```
Add Code

More Code
```

Week 2 Getting Started with Git

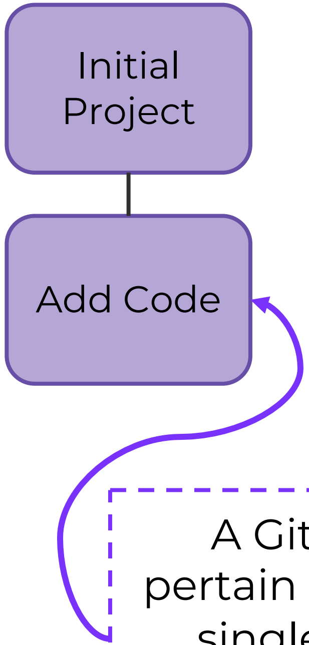


We can go back to a previous commit.

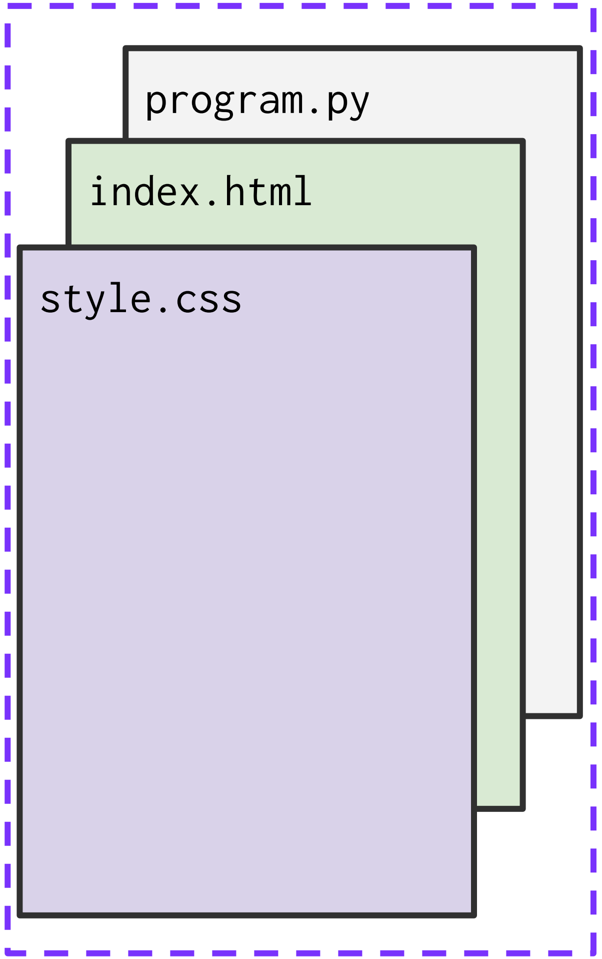


```
Add Code
```

Week 2 Getting Started with Git



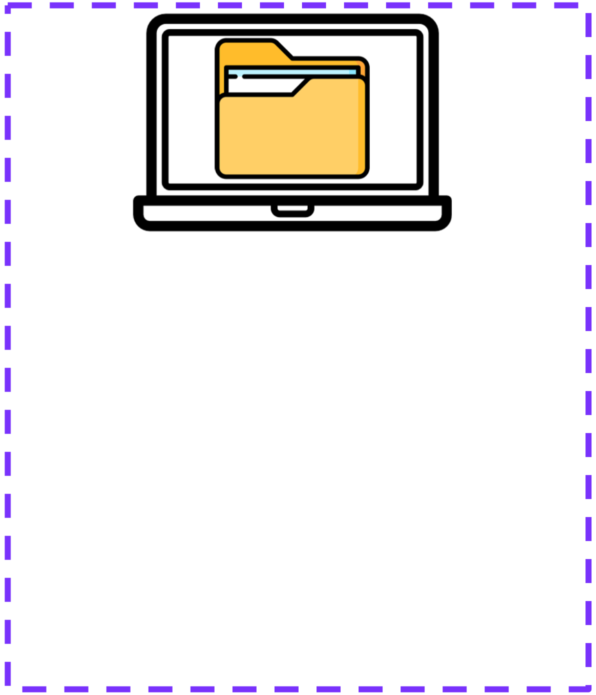
A Git commit doesn't just pertain to a saving changes in a single file. It can constitute specific changes across an entire **working directory**.



Week 2 Getting Started with Git

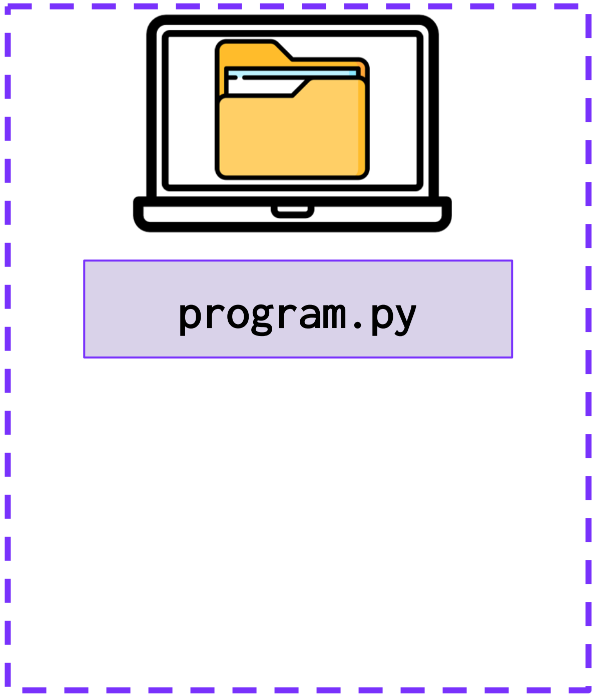
Week 2 Getting Started with Git

Working Directory



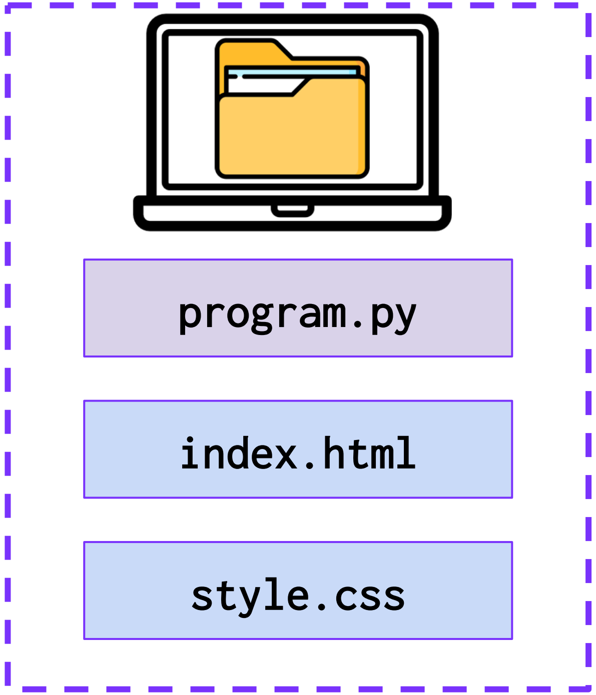
Week 2 Getting Started with Git

Working Directory

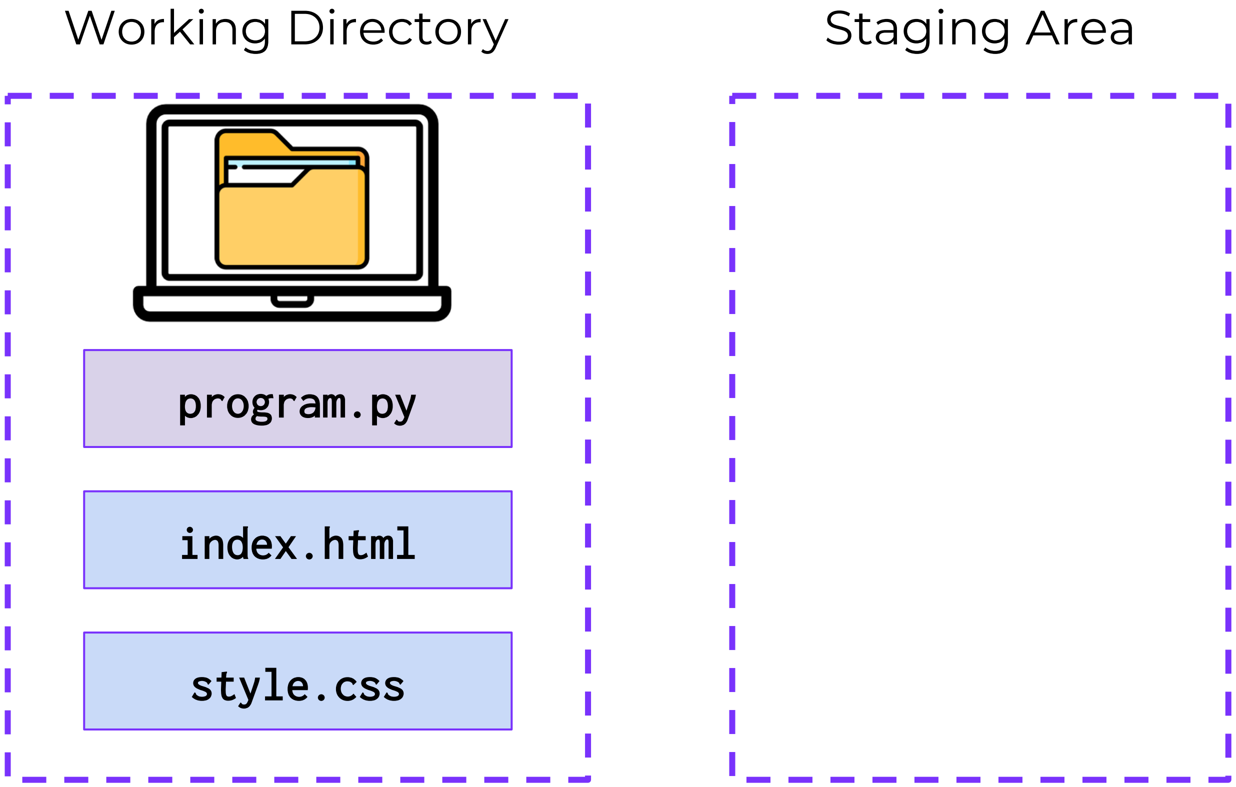


Week 2 Getting Started with Git

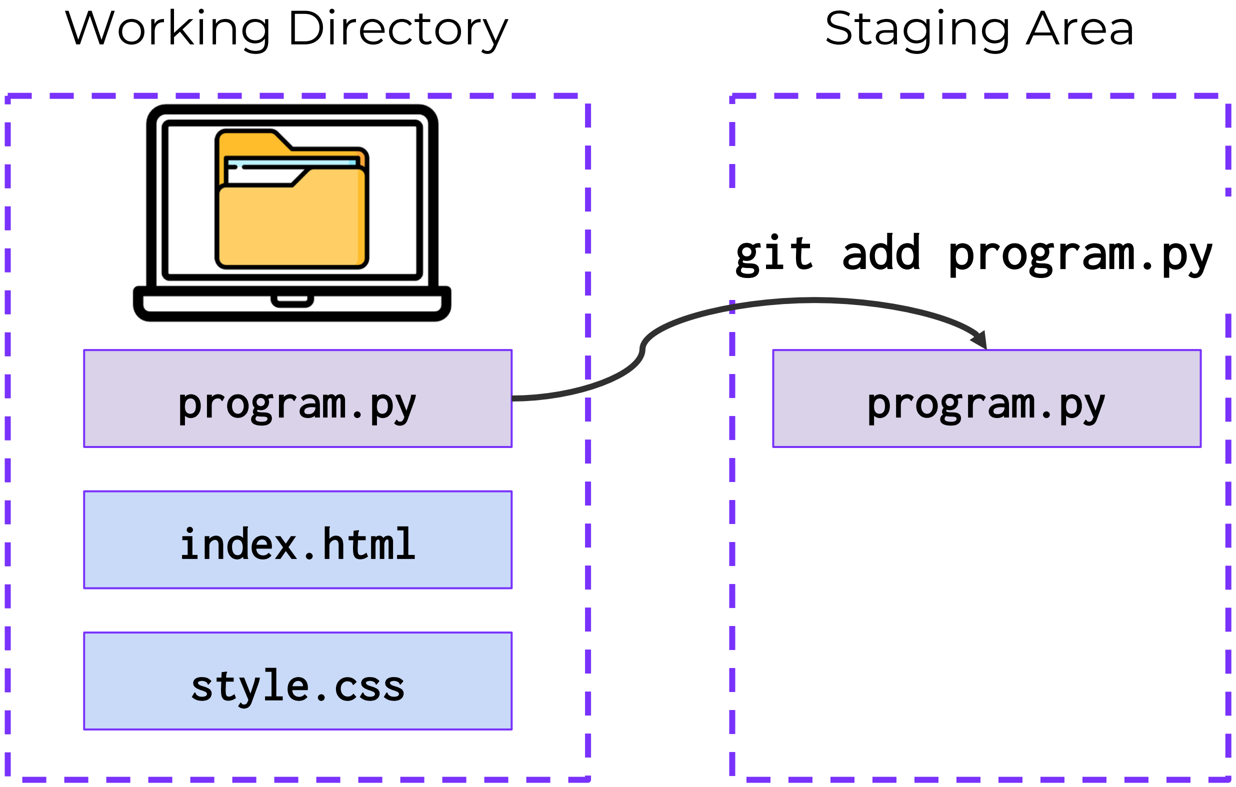
Working Directory



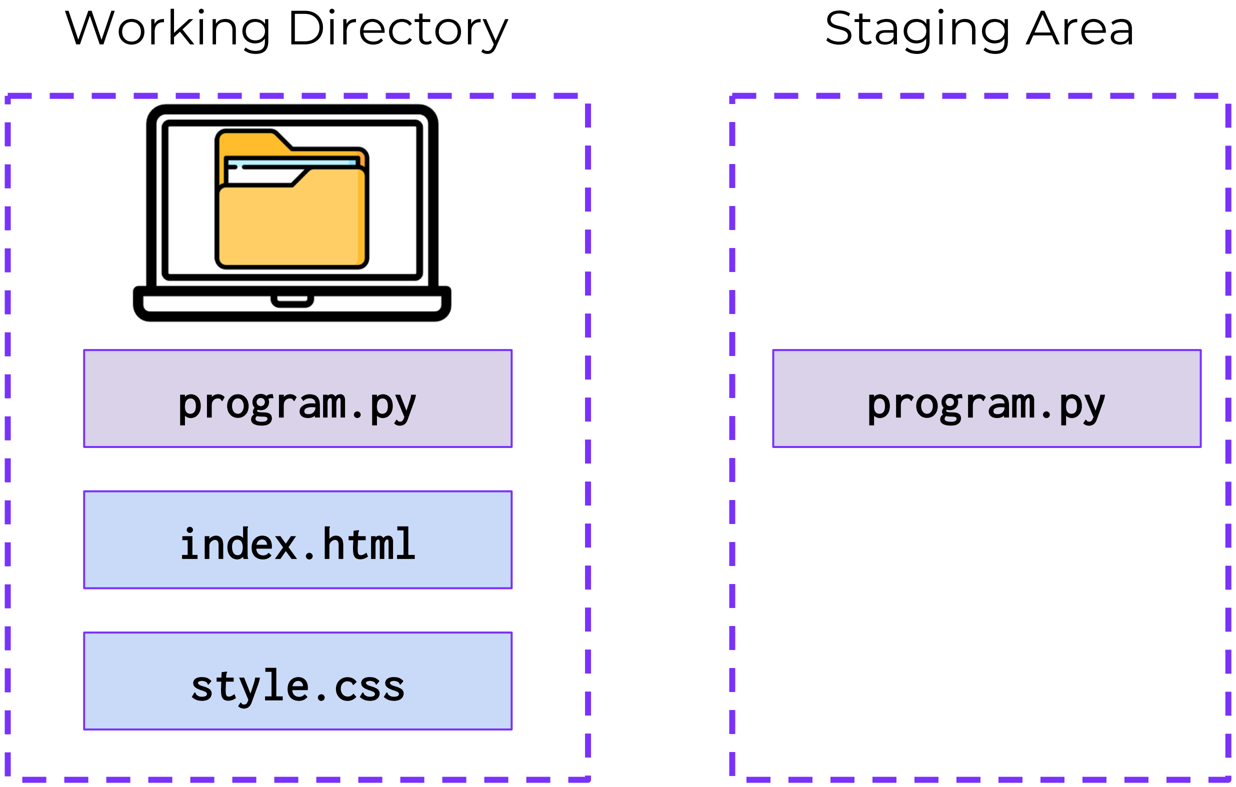
Week 2 Getting Started with Git



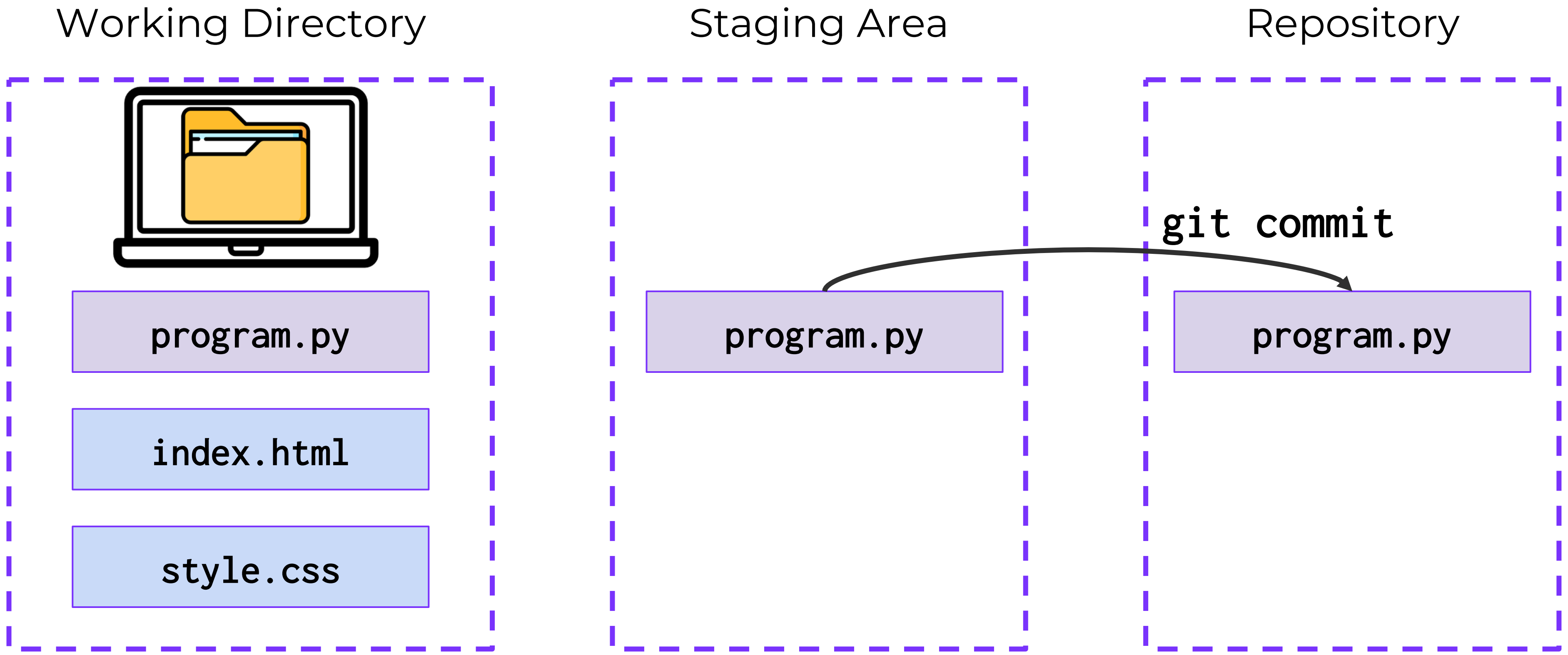
Week 2 Getting Started with Git



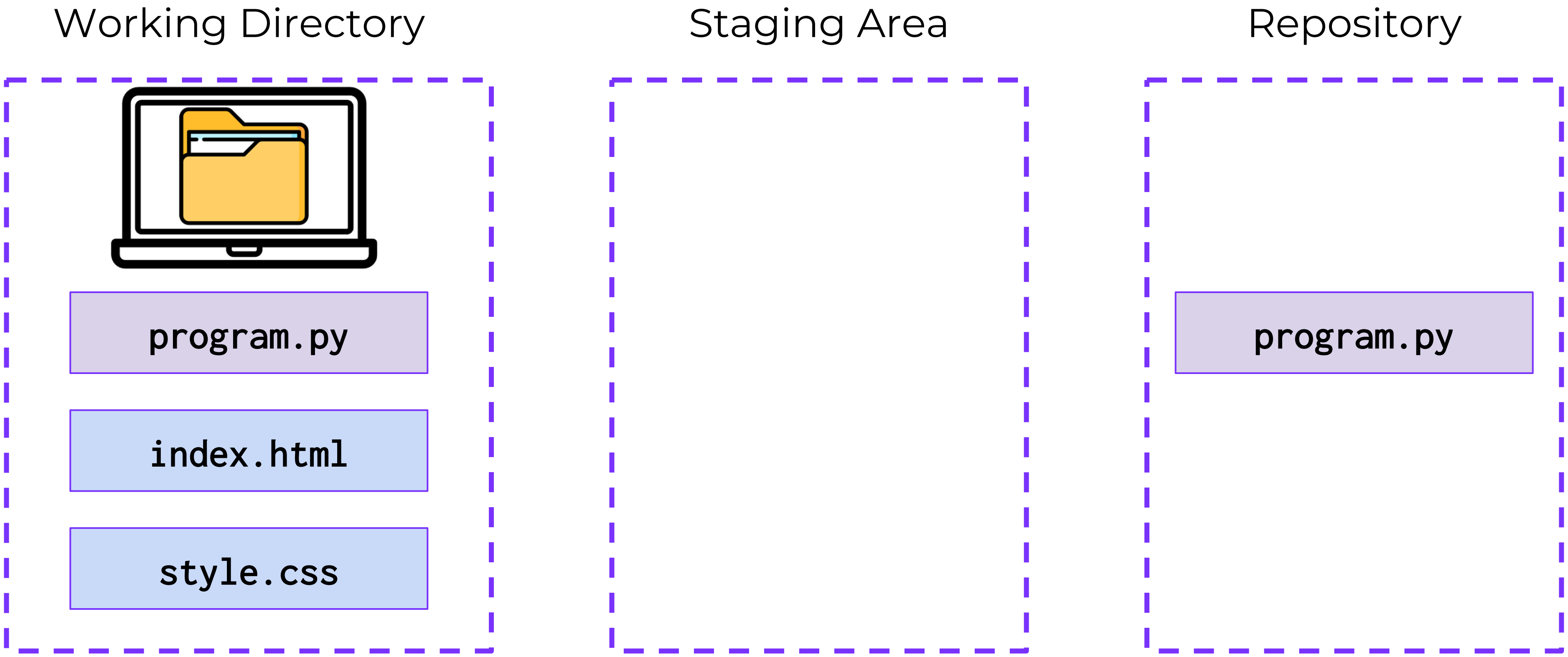
Week 2 Getting Started with Git



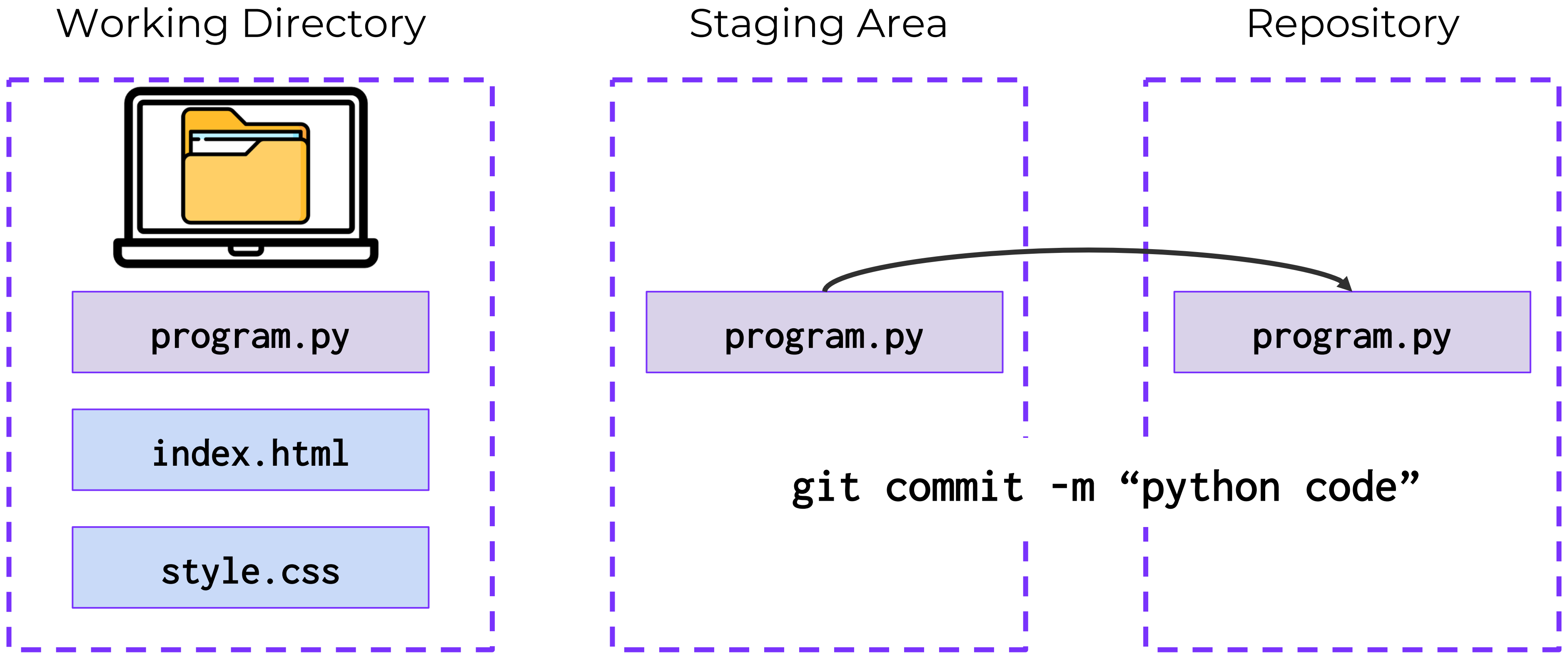
Week 2 Getting Started with Git



Week 2 Getting Started with Git

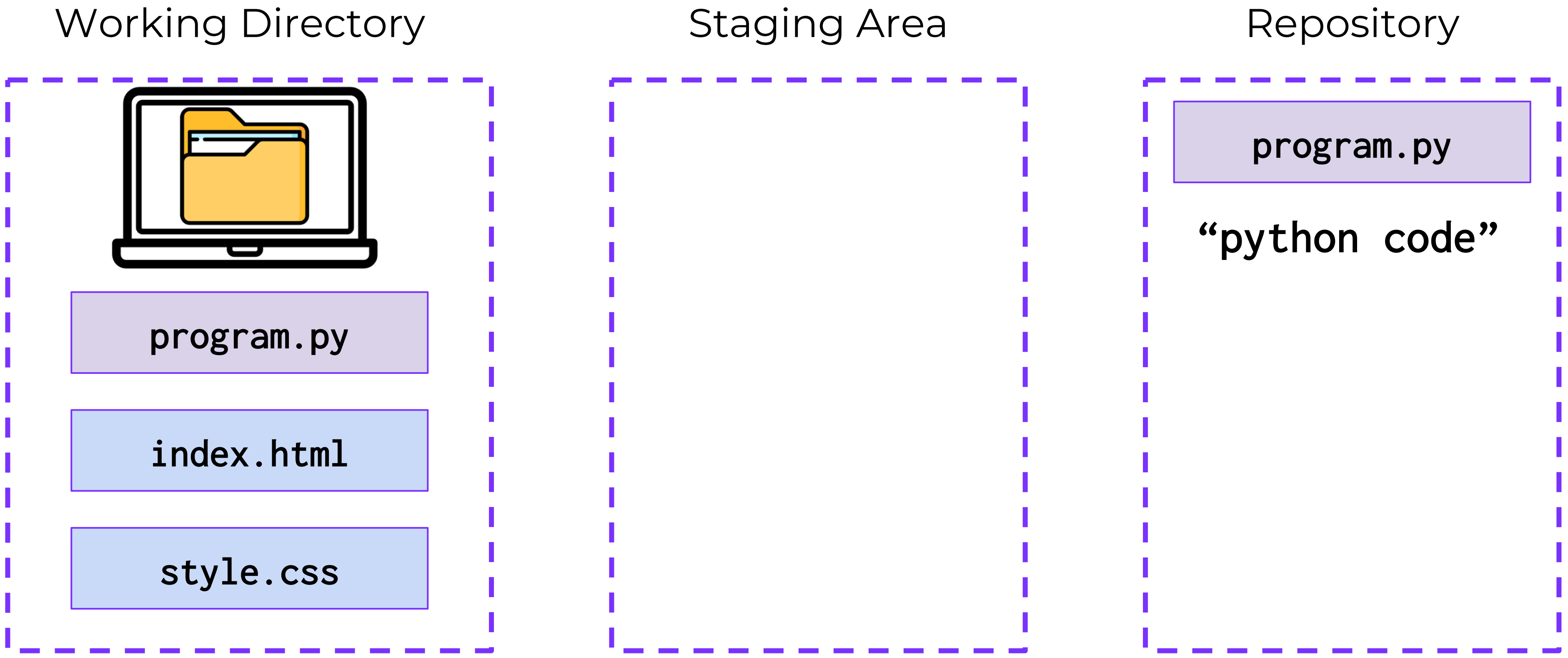


Week 2 Getting Started with Git

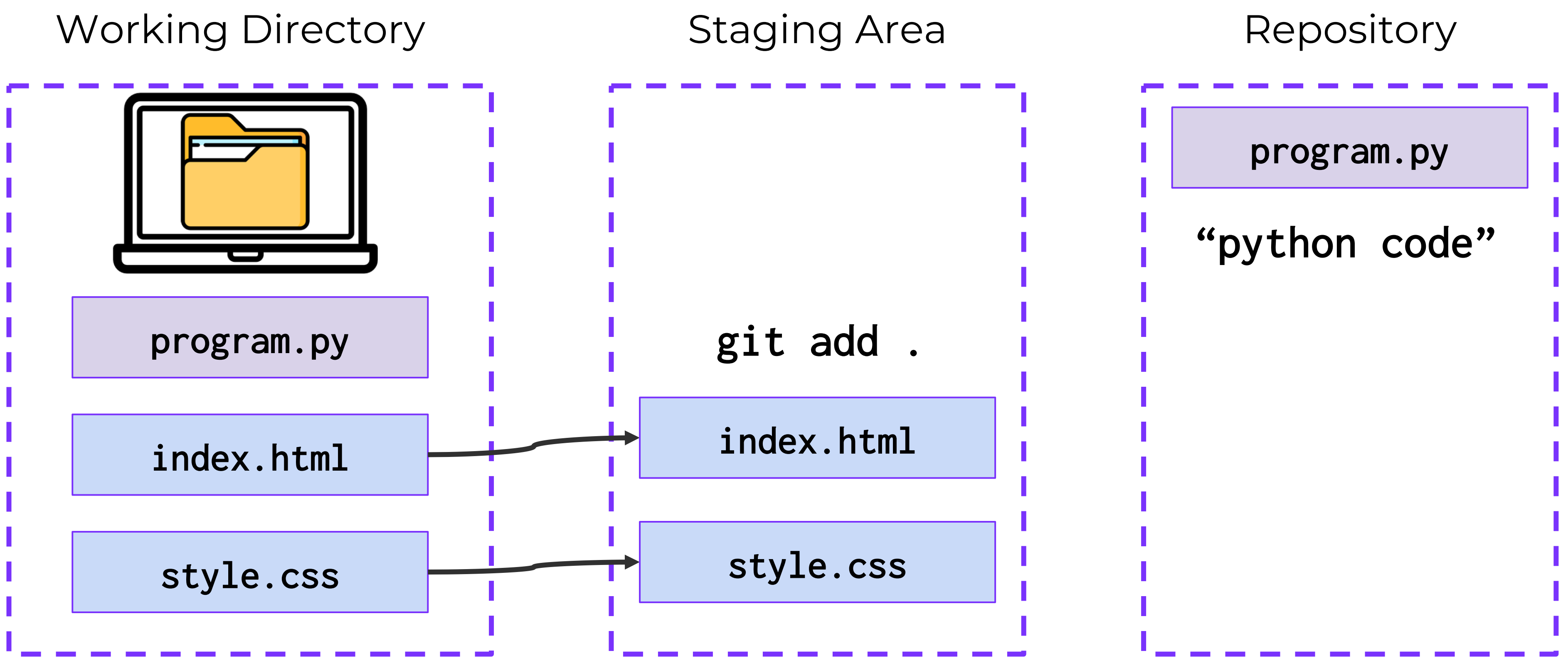


```
git commit -m “python code”
```

Week 2 Getting Started with Git

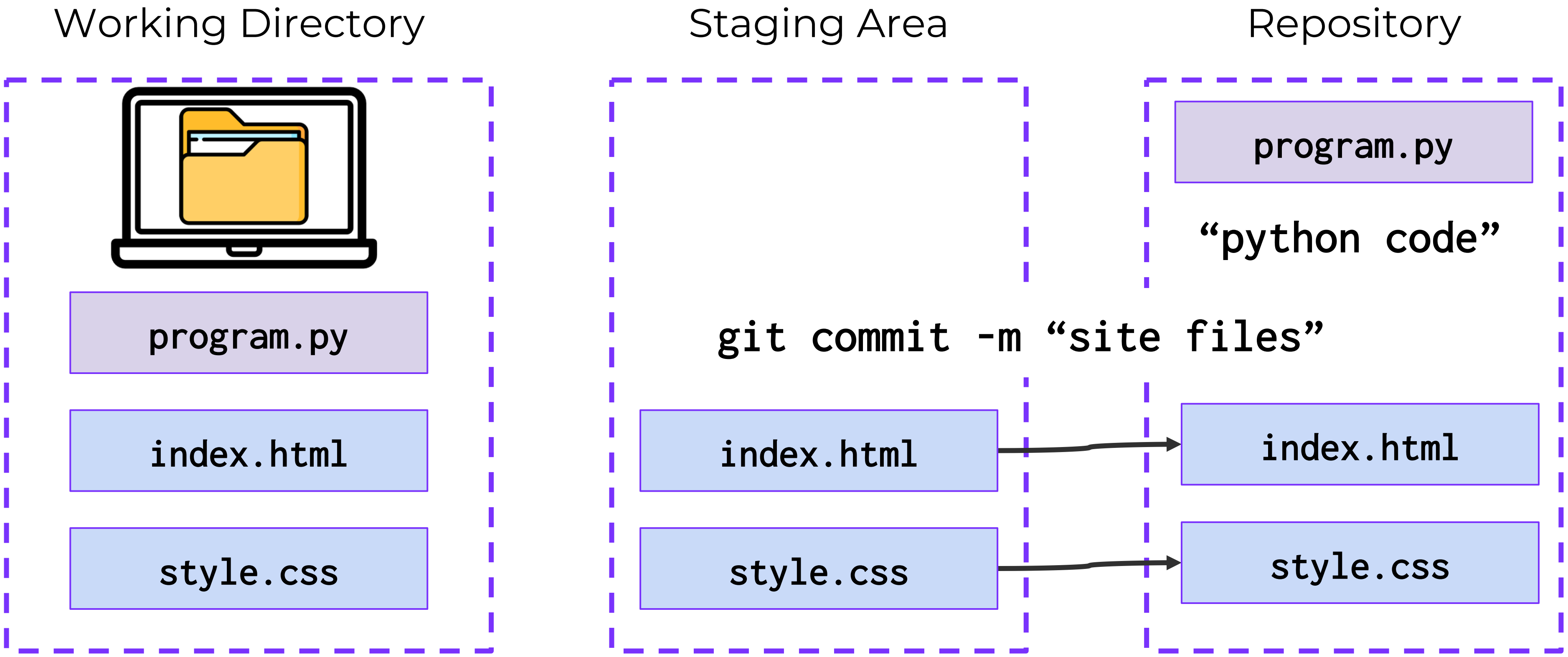


Week 2 Getting Started with Git



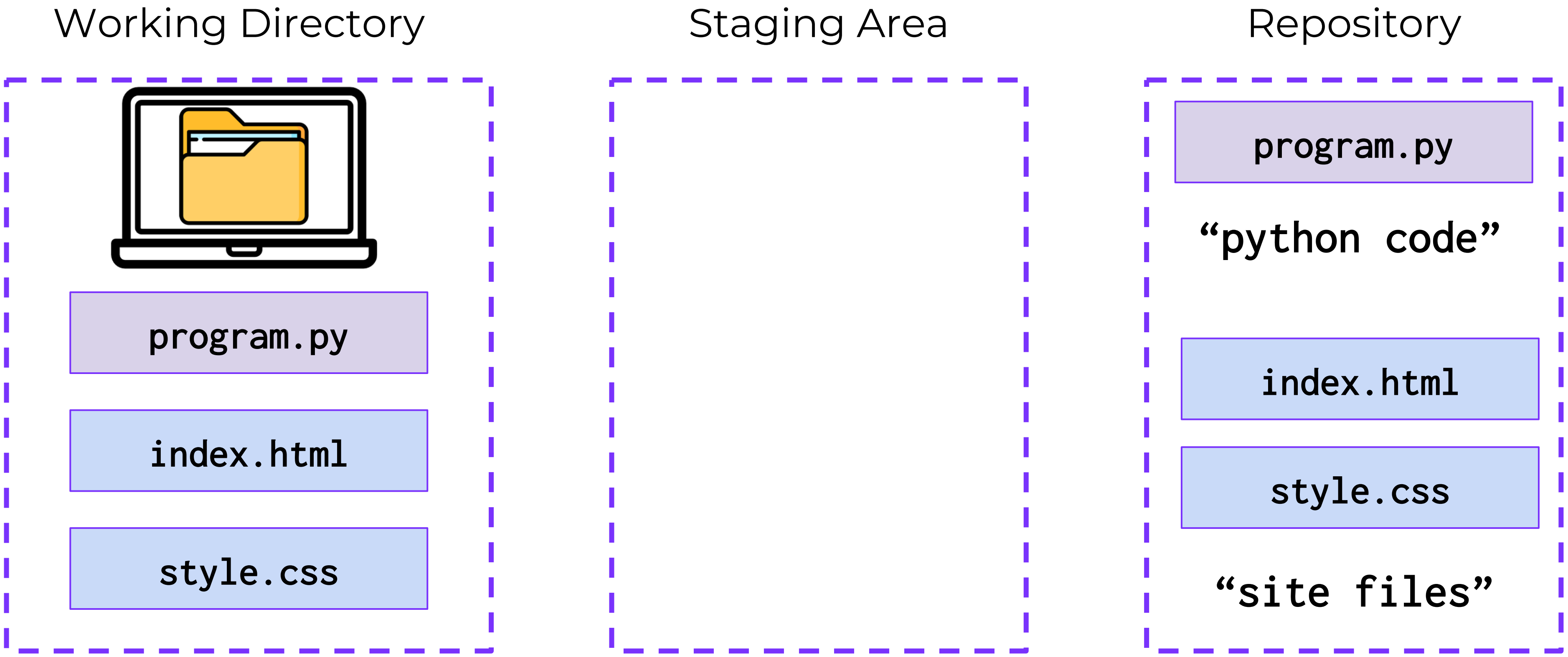
```
git add .
```

Week 2 Getting Started with Git



```
git commit -m “site files”
```

Week 2 Getting Started with Git



Week 2 Getting Started with Git

Working Directory

program.py

index.html

style.css

Repository

program.py

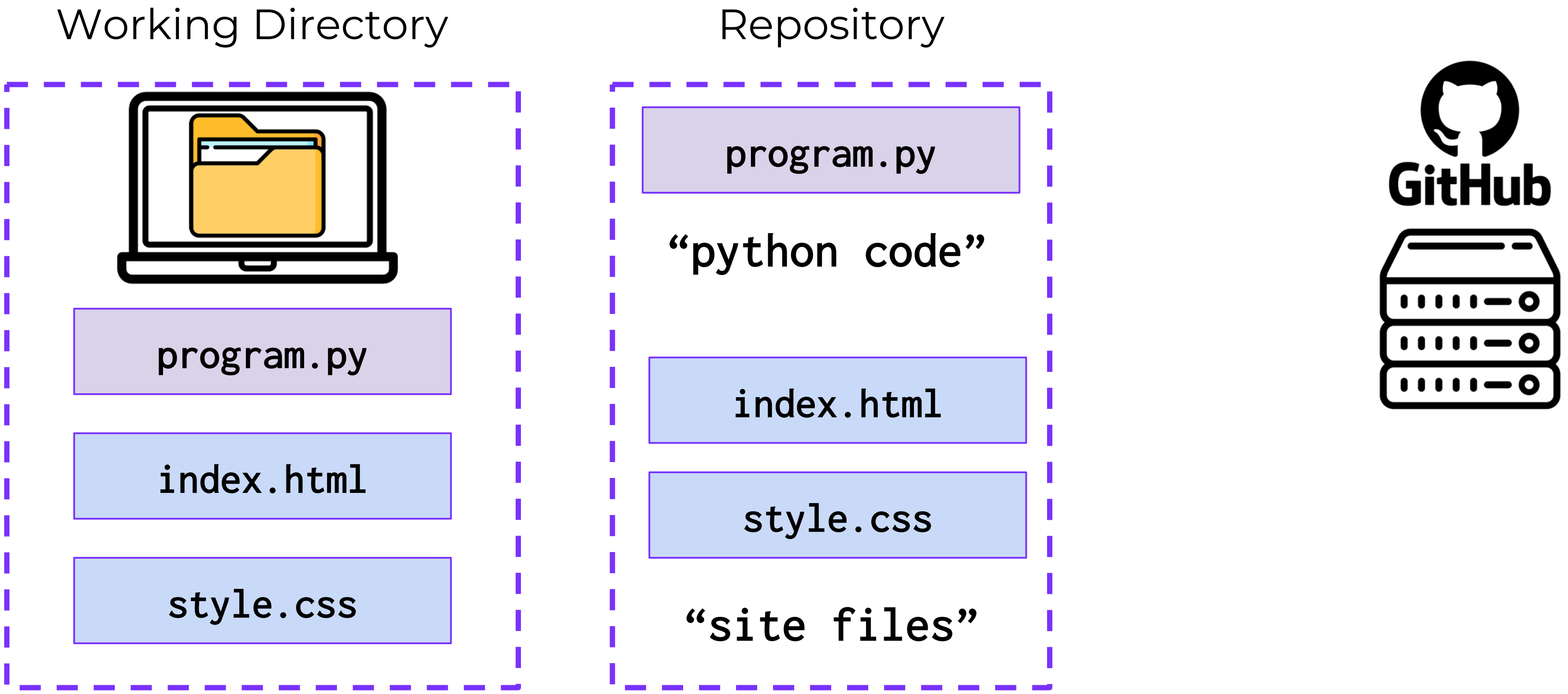
“python code”

index.html

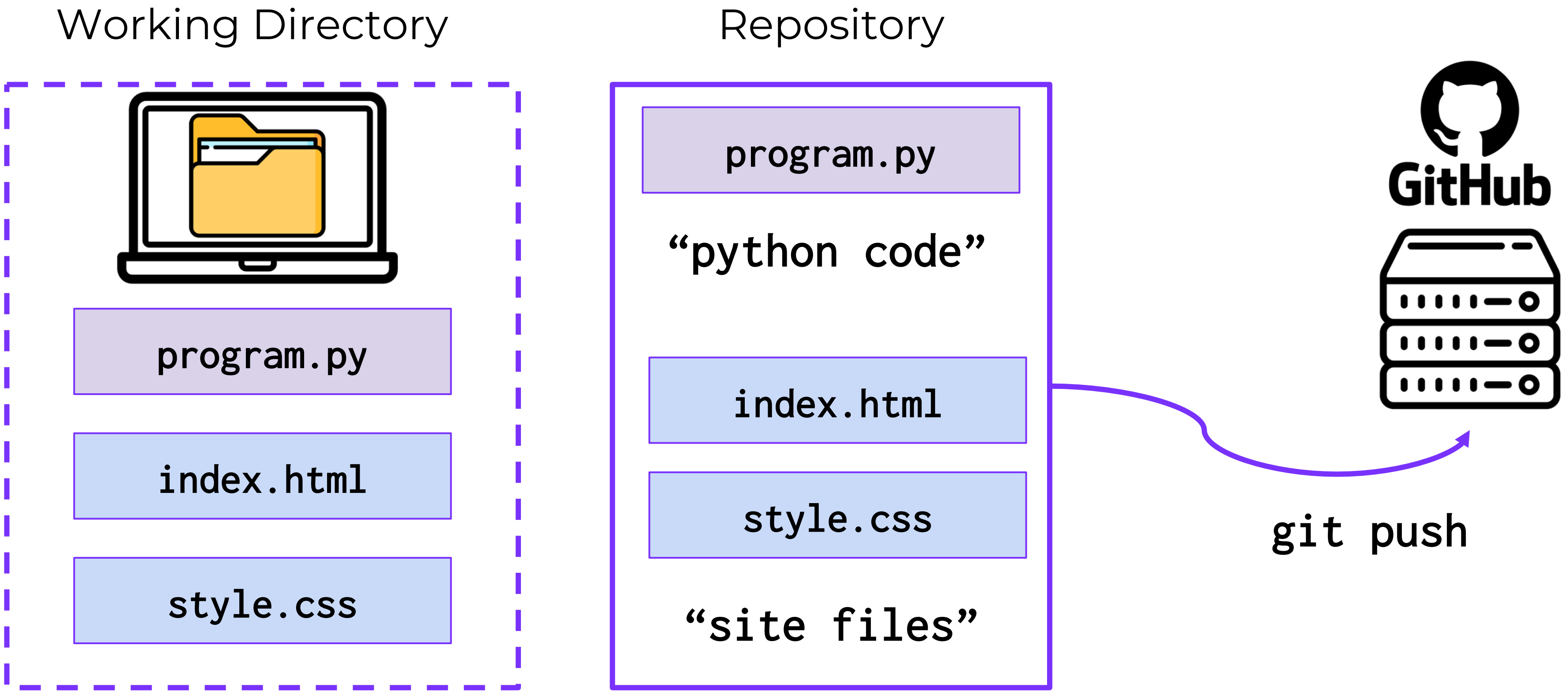
style.css

“site files”

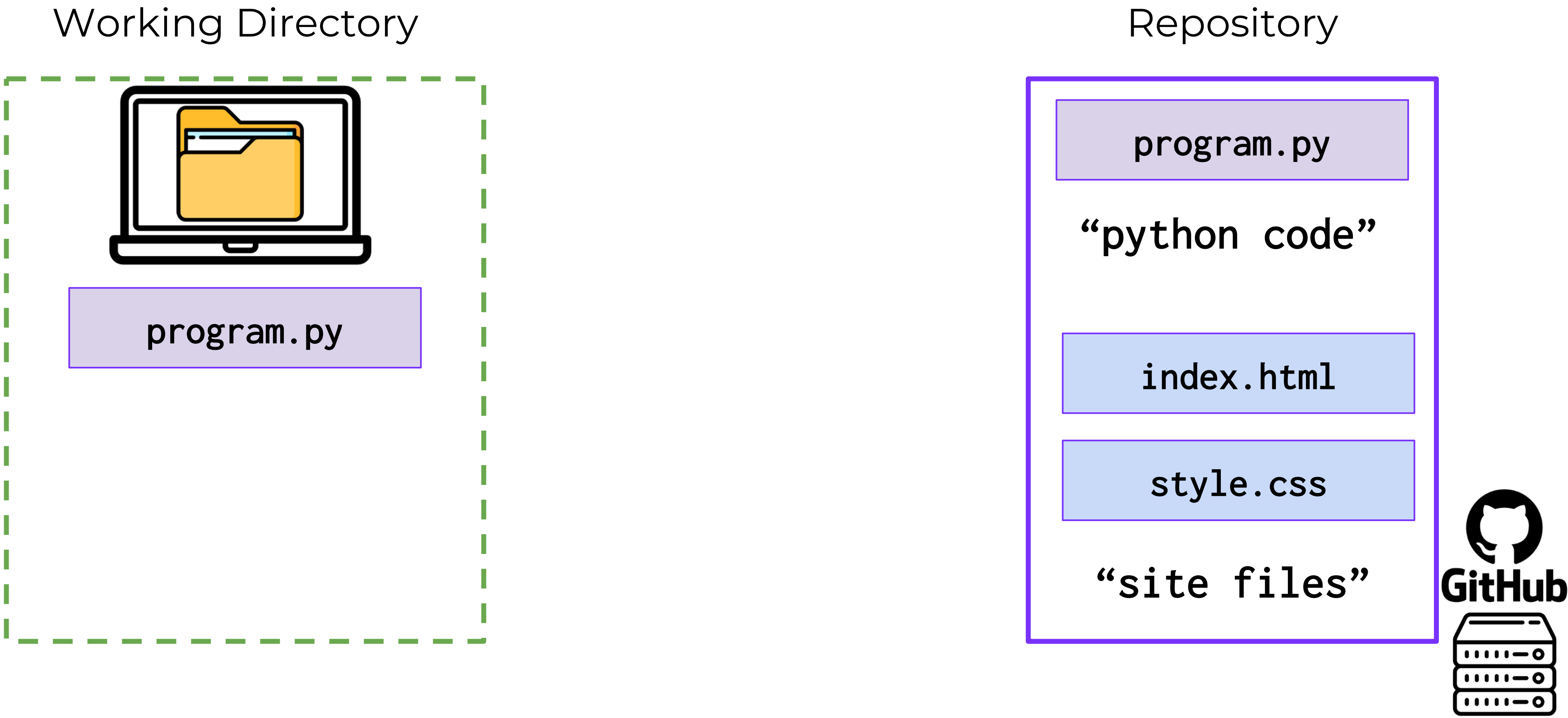
Week 2 Getting Started with Git



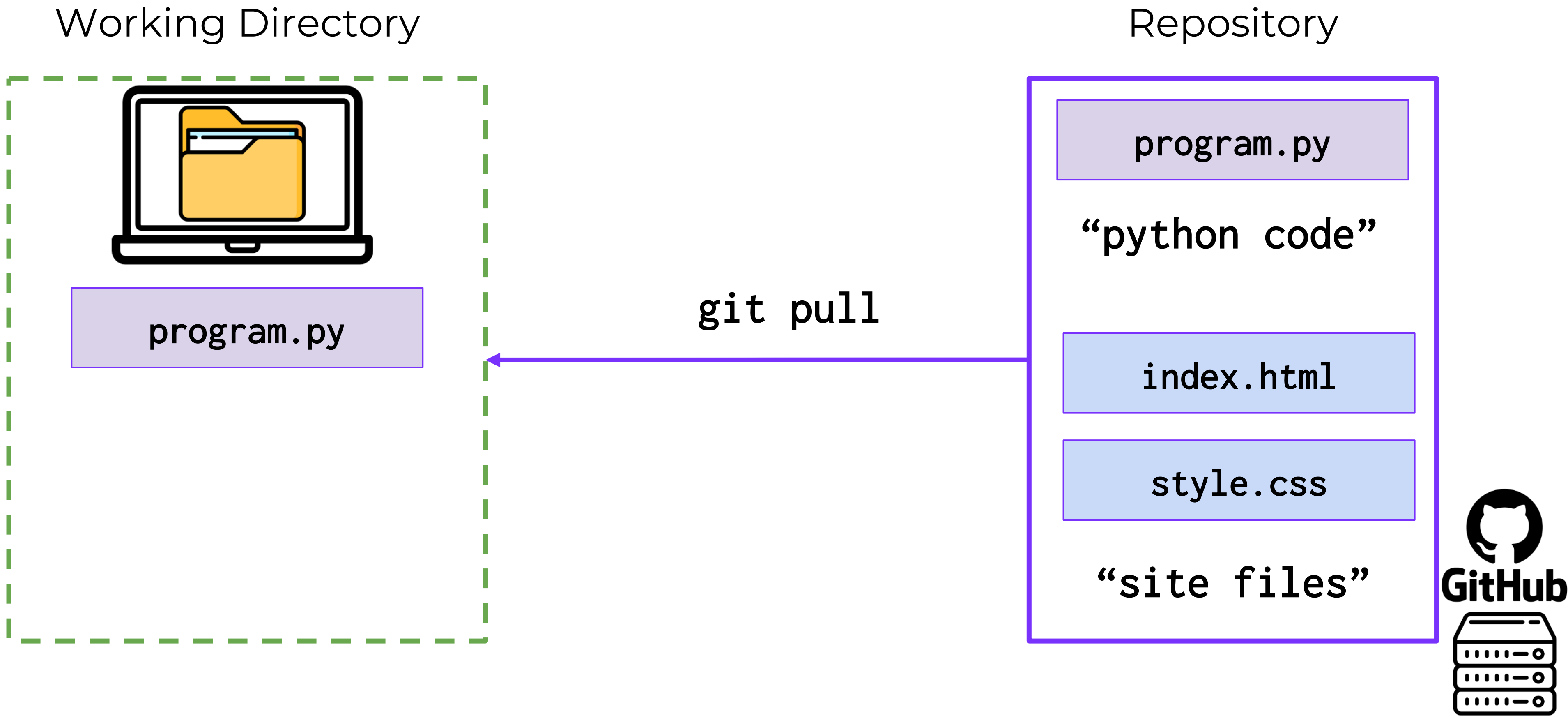
Week 2 Getting Started with Git



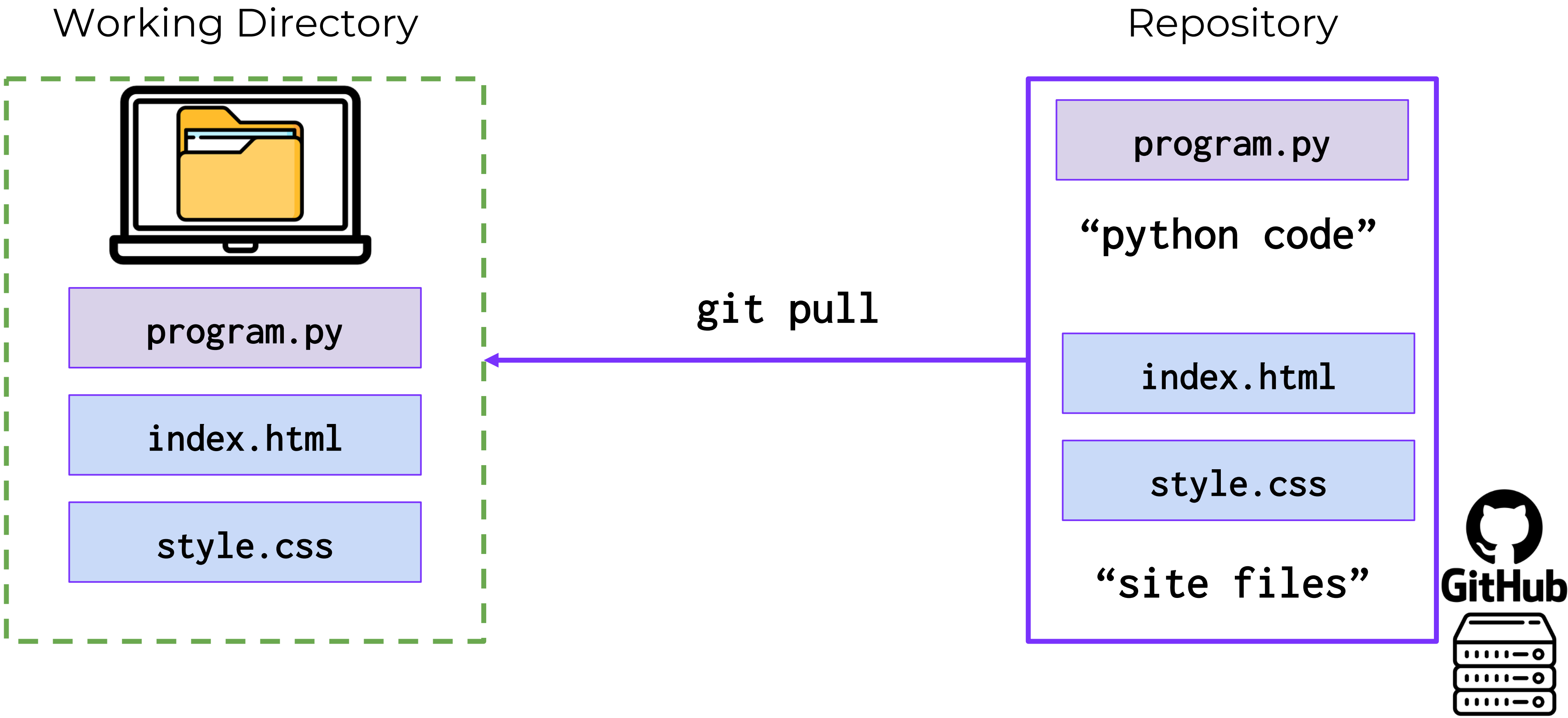
Week 2 Getting Started with Git



Week 2 Getting Started with Git



Week 2 Getting Started with Git



Week 2 Getting Started with Git

- This is the main workflow we're covering toWeek.
- However as we continue, try to keep in mind what would happen if we had multiple people working on different parts of the code at different times.
  - *How would we deal with differences or conflicting code? What about branches?*

Week 2
Add and Commit

Week 2 Getting Started with Git

- Let's now go through the process of creating a repository, creating new files inside the working directory, adding them to the staging area, and then committing them to the repository.
- You will need a text editor, we recommend VS Code, download it here:
    - **https://code.visualstudio.com/**

Week 2
Push and Remote Branches

Week 2 Getting Started with Git

- We’ve learned how to create repositories locally and add changes to the staging area and then commit them to the main (master) branch we have locally.
- If you are also using GitHub as a hosting service, we can think of this as a **remote branch**.

Week 2 Getting Started with Git

- We’re still operating under the assumption that the context is just a single solo developer operating on just a single branch, but later on we’ll talk about branches in more detail.
- Let’s learn how to **push** local code to a **remote branch** on GitHub.

Week 2 Getting Started with Git

- We can check for remote branches with the command:
    - **git remote -v**
- If you run this command on a cloned repo, you will **view** the URL of the remote branch, like the GitHub URL.
- If there is no connection to a remote branch, then you won’t see a URL.

Week 2 Getting Started with Git

- After we've created a repository locally, we need to create the repository on GitHub.
- Once you've created the repository on GitHub, you will actually see the instructions under: ***"...or push an existing repository from the command line"***

Week 2 Getting Started with Git

- We tell git we want to add a remote branch using the git remote command syntax:
  - **git remote add name https://url.git**
- By convention, we call this remote branch the **origin** branch.
  - **git remote add origin https://url.git**
- You then replace the .git URL with the .git URL from the repository you created.

Week 2 Getting Started with Git

- **Important Note:**
  - *When watching along with our video, you'll need to create your own repository, you won't be able to just push to the repository shown in the video (which makes sense that not just anybody could push to any repo with just the URL). Also you will need a PAT.*

Week 2 Getting Started with Git

- **Important Note:**
    - We won’t use these commands in the video, but just in case you need them in the future:
        - **git remote rename <old> <new>**
        - **git remote remove <name>**

Week 2 Getting Started with Git

- Once we’ve connected to our remote branch on GitHub, we can **push** our code to the remote branch.
- We tell git to push to the remote main/master branch called origin with the command:
  - **git push -u origin main/master**

Week 2 Getting Started with Git

- **Important Note:**
  - GitHub has officially changed the naming convention of its **master** branch to **main** branch.
  - You'll see this reflected in the instructions that GitHub provides:
    - **git branch -M main**

Week 2 Getting Started with Git

- Let’s explore all of this:
  - Create a new repo on GitHub
  - Connect our repo from the previous lecture to this remote branch on GitHub
  - Push our commits to GitHub

Week 2
Git Log

Week 2 Getting Started with Git

- Before we jump into using git fetch and git pull, let’s quickly show you how to use **git log**.
- The **git log** command will show a list of all the commits made to a repository, including the hash, message, and metadata.
- Think of it as the history of a repo.

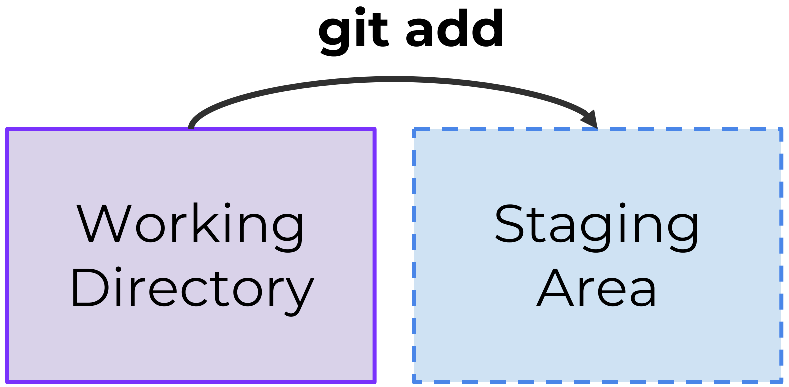
Week 2
Fetch and Pull

Week 2 Getting Started with Git

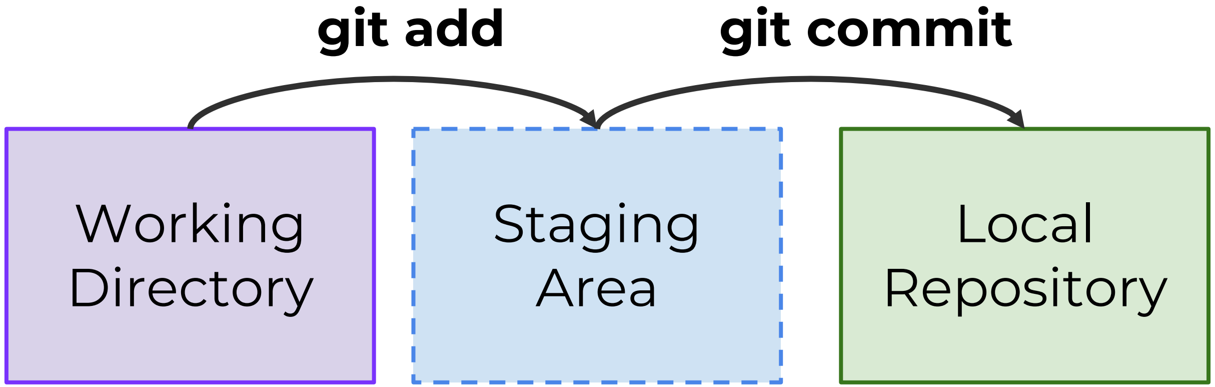
- There are two options of getting repository changes from a remote branch (like the remote branch on GitHub).
  - **git pull**
  - **git fetch**

Week 2 Getting Started with Git

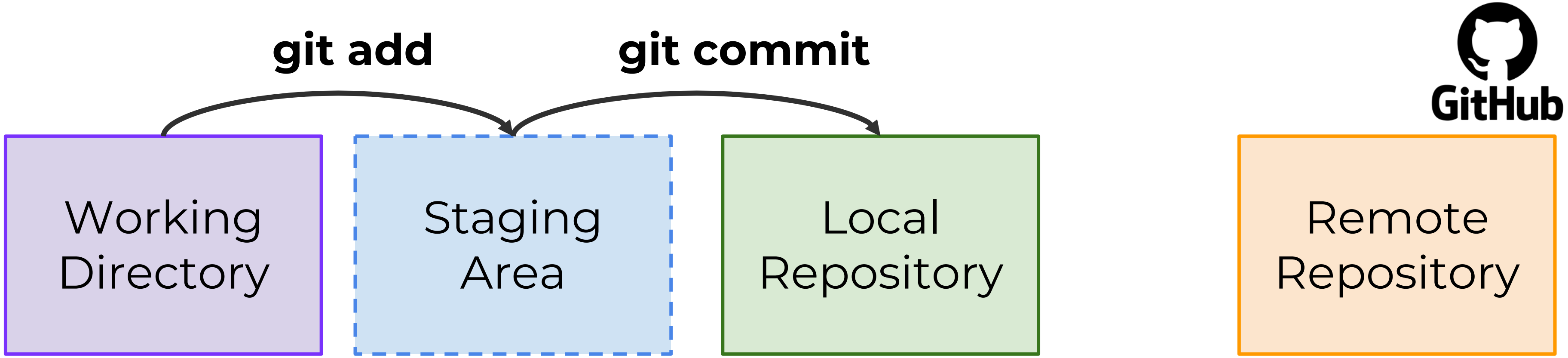
Week 2 Getting Started with Git



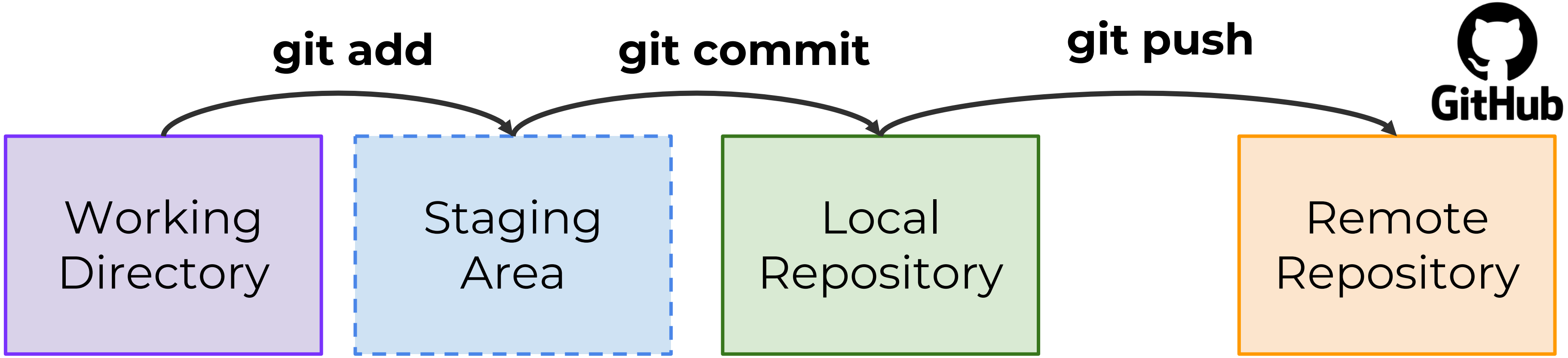
Week 2 Getting Started with Git



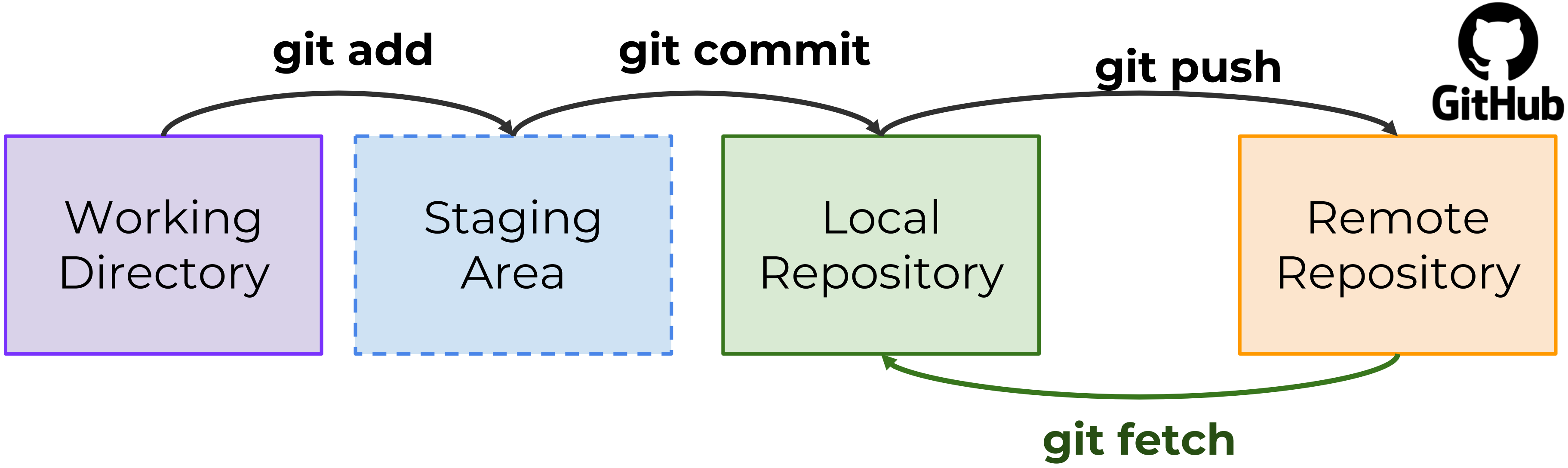
Week 2 Getting Started with Git



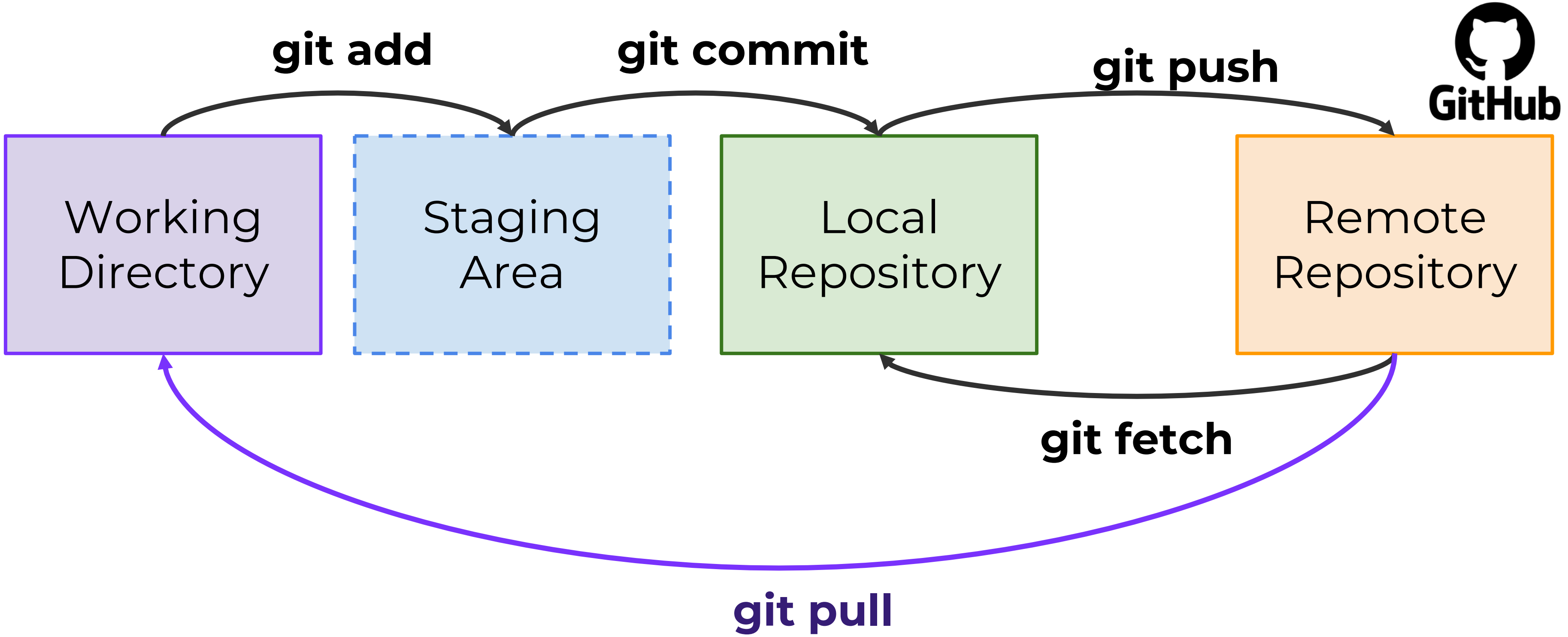
Week 2 Getting Started with Git



Week 2 Getting Started with Git



Week 2 Getting Started with Git



Week 2 Getting Started with Git

- Using **git fetch** will download changes from the GitHub remote repository, however you will not see those changes be part of the files you have in the working directory.
- Fetch allows you to grab additional work done on the remote master branch, without needing to merge it in your working directory files.

Week 2 Getting Started with Git

- Using **git fetch** makes sense when you're working with others and want to see what changes they've made but aren't ready to overwrite your own files yet.
- Also if you are simply working by yourself, you may want to just fetch remote changes without overwriting your latest work (later we'll discover branches are a better way of doing this).

Week 2 Getting Started with Git

- When using fetch, often you’ll just use:
  - **git fetch**
- But you can specify to fetch specific branches using:
  - **git fetch <remote> <branch>**
  - **git fetch origin <branch>**

Week 2 Getting Started with Git

- Using **git pull** makes sense when you want to directly grab changes from the remote branch and directly merge them into your current branch.
- This means you will literally update the files in your working directory to match up and merge with the remote branch.

Week 2 Getting Started with Git

- If you're a solo developer working on a single master branch, you often skip using a combination of **git fetch** and **git merge** and go straight for a **git pull**.
- We're not going to condone this as the "best practice", but we also want to be realistic of the workflow of a solo developer on a single branch.

Week 2 Getting Started with Git

- In the next Weeks when we learn about branches and merging changes more carefully or stashing current changes, we'll have a more nuanced understanding of using **git fetch** vs. **git pull**.
- In general you should pull before pushing to a remote branch, to make sure you're in sync.

Week 2 Getting Started with Git

- Let’s explore these git methods by making changes directly on GitHub on the remote origin and then bring those changes to our local repository and working directory.

Week 2
Exercise and Solution

Week 2 Getting Started with Git

- Let’s test your understanding of what you’ve learned so far with a quick exercise.
- We’ll give you a list of tasks to complete.

Week 2 Getting Started with Git

- **Tasks:**
  - Create a new local repo
  - Add a text file locally
  - Create the remote repo on GitHub
  - Push the Local repo changes to GitHub
  - Create a new file on GitHub
  - Pull these changes from GitHub to the local repo

Week 2
Solution