**Git Hub Concepts**

Q) Can you recall what GitHub is?

A) Cloud-based hosting service

GIT = Software, Can be used independently from GitHub

GitHub = Dependent on Git, Hosting Platform

Both = Aids with version control

Q) What is a benefit of GitHub

A) It allows you to store and track projects and files, It has many open-source projects to learn from or contribute to., It allows for easy collaboration with anyone in the world..

Q) You've created your new repository but you want to make some changes to the title. Where can you do so?

A) Settings

**We have 2 ways of copying a repo – Fork vs Cloning**

Cloning a Repo

Cloning a repo is similar to copy-paste; however, it has a link to the original repo. It creates a copy of the repo in its current state on our local computer and allows us to send updates back and forth. We can work on the files locally, push changes back to the repo, or pull any new updates to our local version using Git. Anyone can clone a public repo, but a private repo owner needs to grant access.

* Requires Git
* Creates a linked copy on a local computer - Open a terminal window and use the git clone command.
* Push and pull updates with Git

Forking

We can copy a repo without linking back to it. In GitHub, this is called forking. A fork of a repo creates an independent copy of it on our GitHub, meaning we can run experiments without the risk of anything reaching the original repo. Navigate to fork > create new fork > select the owner > add name to the fork repo > select the branch to the fork > click create fork

* Creates an independent copy on GitHub
* Can all be done within GitHub
* Add changes with a PR (pull request)
* Great for taking risks

PAT (Personal Access Tokens)

A PAT is an alternative to a password when authentication is required for terminal commands. As the output showed us earlier, it is required for interacting with remote repos since August 2021. They are used because they are more secure than entering a password.

Go to your Settings > developer > PAT > gen new token > enter note and expiration time and scope which defines what people can access > gen token > copy our PAT

PAT needed

* Cloning private repo via terminal
* Pulling private repo via terminal
* Cloning public repo via terminal

PAT not needed

* Viewing public repo on GitHub
* Viewing private repo on GitHub

Private repo - people can’t access this repository unless you grant permissions under settings > collaborators section > add people > search by username/fullname/email > add to this repository

Readme file

Purpose = A README is often the first item a visitor will see when visiting your repository. README files typically include information on: What the project does. Why the project is useful. How users can get started with the project.

# Heading 1 (largest heading)

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6 (smallest heading)

\*\*bold\*\*

\*italics\*

Adding links

[Youtube Link]( <https://www.youtube.com/>)

Drop images into the readme file – starts with ! mark

![IMG\_4875-min](https://github.com/Jordanes1871/Introduction\_Importing\_data\_python/assets/107188925/4f6ae4c5-8eec-4022-91eb-b0a527fefeb2)

Pull Requests

Purpose = to let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

* Way to notify others about changes
* Allows repo owner to check changes before they are added
* Best practices to add changes to a branch that is not the main branch then you can merge the separate branch to where changes are added and the main branch

Navigate to pull requests tab > click new pull request > select correct base and compare branches > check for and resolve any conflicts > create pull requests > add title and description for PR > assign the PR to someone > click create pull request

Comment

* You have some general feedback for the contributor to consider first, but are not ready to approve
* You don’t have any changes but want to discuss some things about PR

Approve

* You want to send them kuddos and are happy with changes
* You have no comments and are ready to merge the branches

Request Changes

* You don’t think one of the changes works well and want to suggest something else
* You have feedback that needs to be incorporated first.

Actions

Purpose = allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

Issues

Purpose = let you track your work on GitHub, where development happens. When you mention an issue in another issue or pull request, the issue's timeline reflects the cross-reference so that you can keep track of related work. To indicate that work is in progress, you can link an issue to a pull request.

Navigate to repo that you want to create issue for > click issue tab > click new issue > write issue and assign to teammate > submit issue > provide feedback through comments > send thank you note and close issue.

Assigning a user clarifies who should be working on the issue, while tagging a user is more of a communication tool to ensure the right people read the message. Tag the users GitHub handle in the issue using the @ symbol. Symbol that represents quote in the comments = >

Branches

Purpose = branches are used for concurrent work on different parts of a project. They can also reduce the risk of conflicting versions of files. For example, we may analyze Spanish soccer data in a branch called la liga, while a colleague analyzes English Premier League data in a branch called premier league. Main is the default branch and is created when we make the repo.

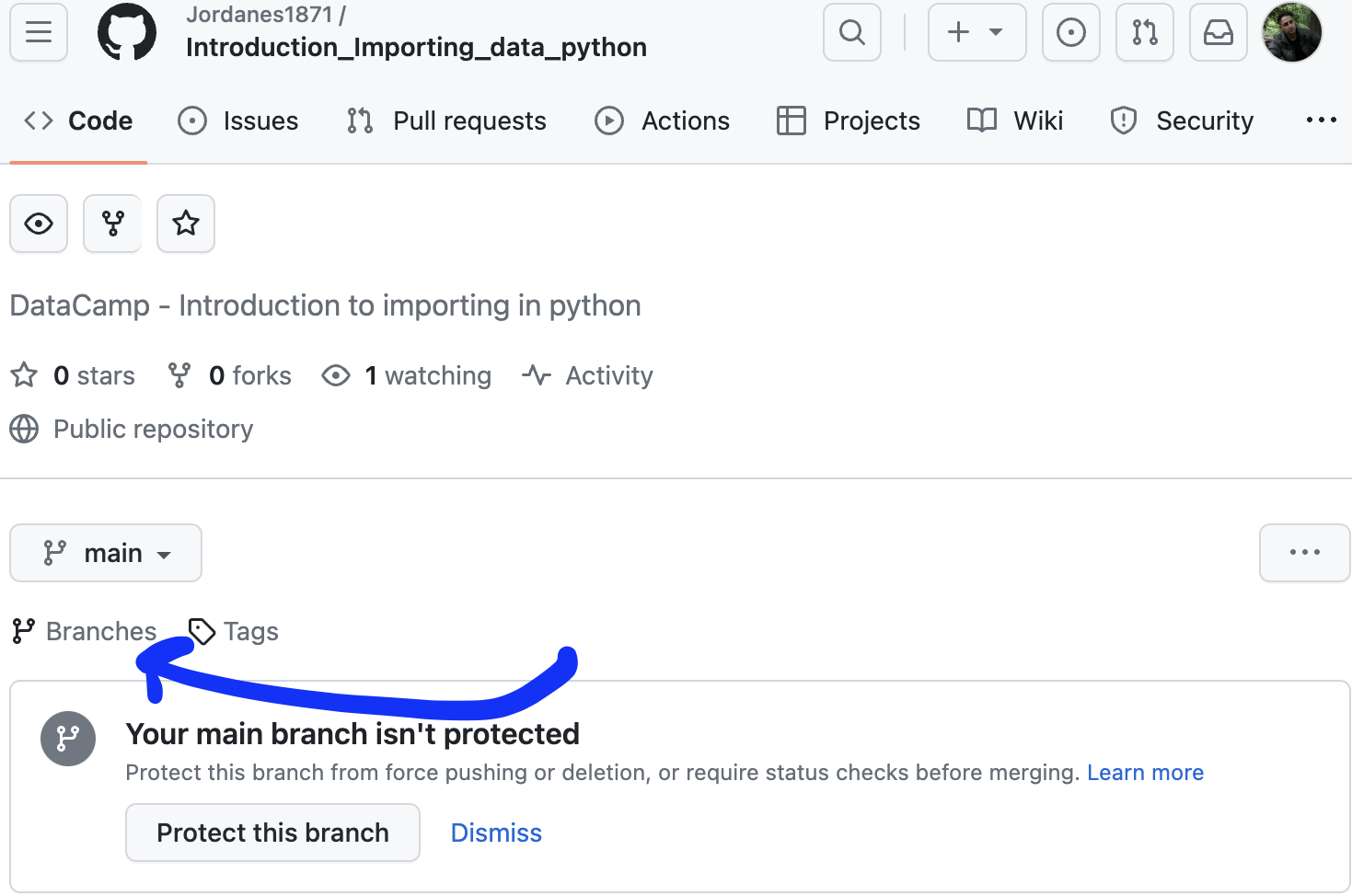
Go to branch icon > new branch > branch name > branch source

Branch protection rule benefit

* Requiring reviews can help improve code quality
* Adds layer protection against introducing code errors in the main branch
* Restricts who can delete protected branches

GitHub benefits

* Allow concurrent code development by colleagues
* Access repos form anywhere with internet connection
* Cloud storage of code and documents



**Example**:

Your goal is to add this image to the existing EDA directory.

1. Click directory
2. Add files
3. Upload files
4. Upload file to repo
5. Add commit message
6. Commit changes

Q) You want to add a new directory to your repo where you will store references for your project. The directory will be called References.

After clicking "Add file" and "Create new file", what is the next step to make this new directory?

1. Write References/README.md.

Q) Now you have created your References directory, you decide that you don't need the README.md file for this project section.

You navigate to README.md in the References directory and press the trash icon:



What else do you need to do to remove this file?

1. Add a commit message and make a commit.

Q) Let's say you are working on the soccer\_analysis repo and have been working on the laliga\_eda branch to do some exploratory data analysis for La Liga. You're ready to add your work to the project in the main branch.

The "Comparing changes" page comes up after clicking on "New pull request."

What is the correct setup for the PR you'd like to do to add your changes?

1. Base: main; Compare: laliga\_eda

Q) Many projects are collaborative, meaning we want to make sure everyone is happy with the changes before they are implemented. One way to do this is to assign a reviewer to a PR in GitHub.

What is the correct way to assign a reviewer to a PR?

1. Select the correct user in the Reviewer section.

Q) How do you bring a branch back after deleting it following a PR?

A) click restore