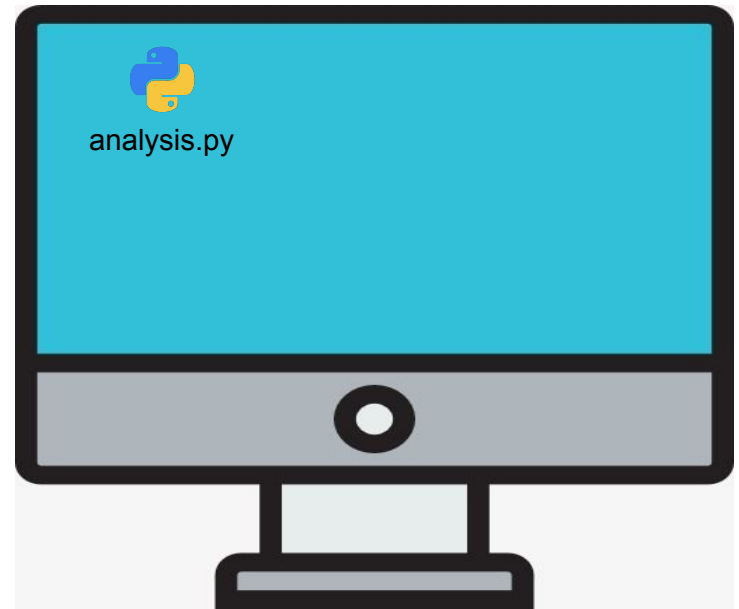


Real Python

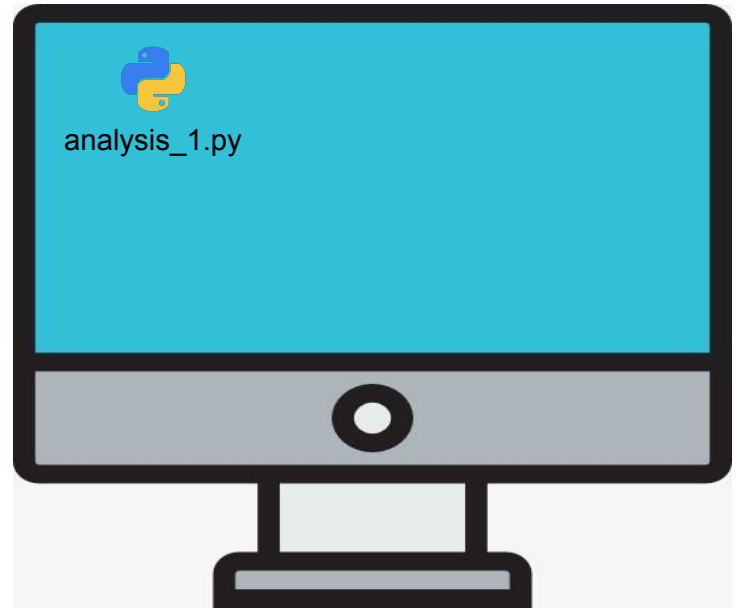


Alice



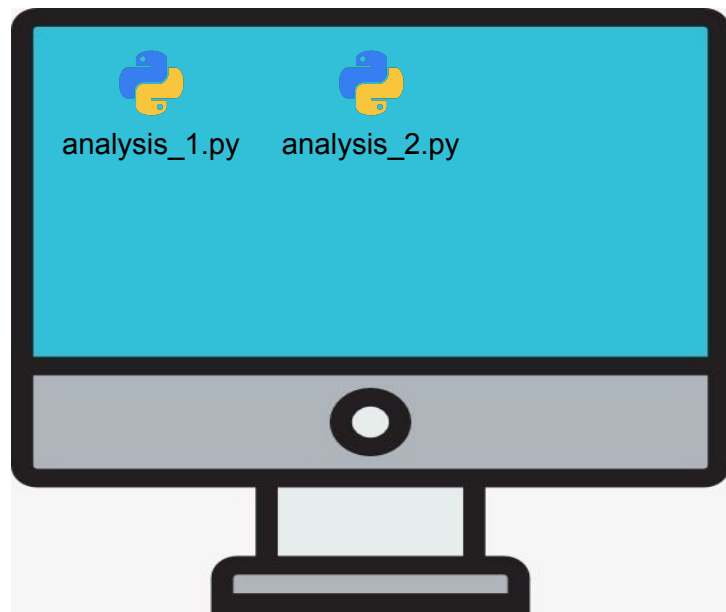


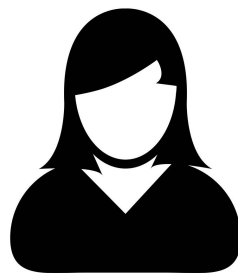
Alice



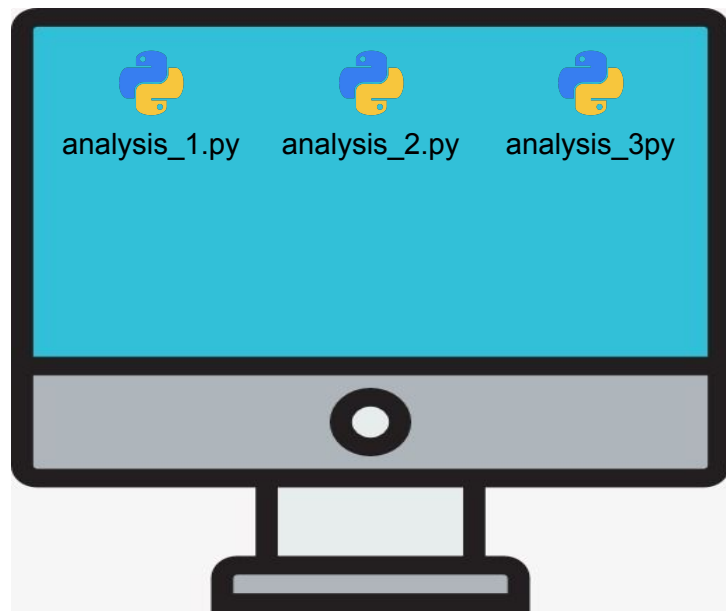


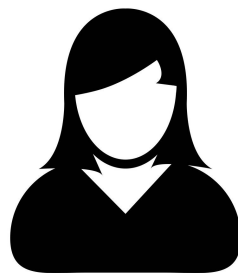
Alice



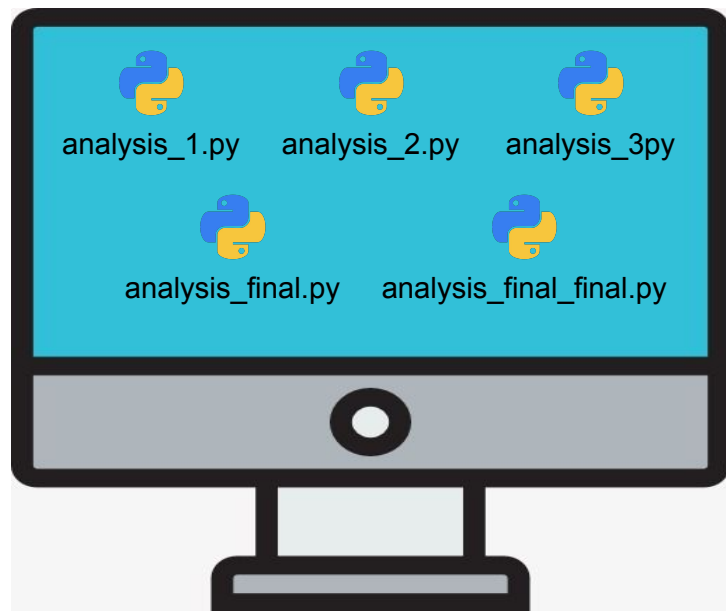


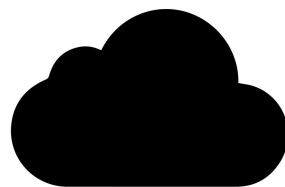
Alice





Alice





Cloud



analysis_1.py



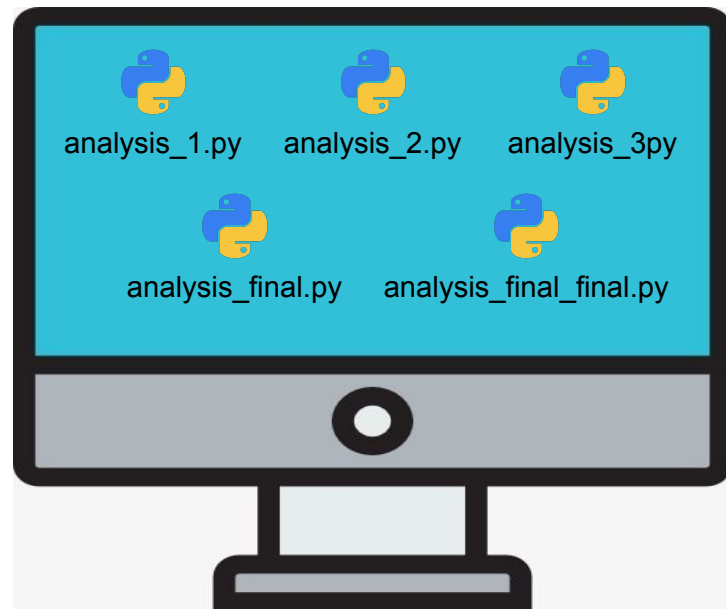
analysis_2.py



analysis_3.py



Alice





Cloud



analysis_1.py



analysis_2.py



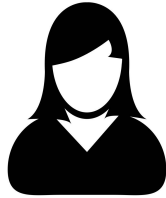
analysis_3.py



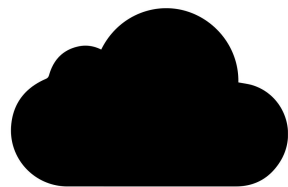
analysis_final.py



analysis_final_final.py



Alice at home



Cloud



analysis_1.py



analysis_2.py



analysis_3.py



analysis_final.py



analysis_final_final.py



Alice at home



Alice in wonderland



analysis_1.py



analysis_2.py



analysis_3.py



analysis_final.py



analysis_final_final.py



Cloud



Alice

analysis_1.py analysis_2.py analysis_3.py

analysis_final.py analysis_final_final.py



Bob

analysis_first.py analysis_sec.py analysis_3rd.py



Cloud




Bob


analysis_1.py


experiment.py


gui.py


data_acq.py


models.py



Alice



Cloud




Bob


analysis_1.py


experiment.py


data_acq.py


models.py


gui.py



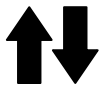
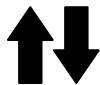
Alice



git



Cloud



Bob

analysis_1.py



experiment.py



data_acq.py



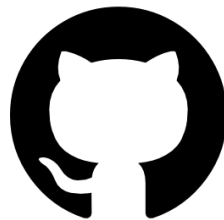
models.py



gui.py



Alice



GitHub



git

How does it work?

Common English Language

- Create a directory which has VC
- Make Changes
- Save your changes
- Implement the changes on the cloud as well
- Implement the changes on the cloud in your local device
(if you are working with multiple devices, or with different people)

How does it work?

Common English Language

- Create a directory which has VC
- Make Changes
- Save your changes
- Implement the changes on the cloud as well
- Implement the changes on the cloud in your local device
(if you are working with multiple devices, or with different people)

Git English Language

- Create a **repository**
- Make Changes
- **commit** your changes
- **Push** the changes on the cloud as well
- **Pull** the changes from the cloud

How does it work?

Common English Language

- Create a directory which has VC
- Make Changes
- Save your changes
- Implement the changes on the cloud as well
- Implement the changes on the cloud in your local device
(if you are working with multiple devices, or with different people)

Git English Language

- Create a **repository**
- Make Changes
- **commit** your changes
- **Push** the changes on the cloud as well
- **Pull** the changes from the cloud

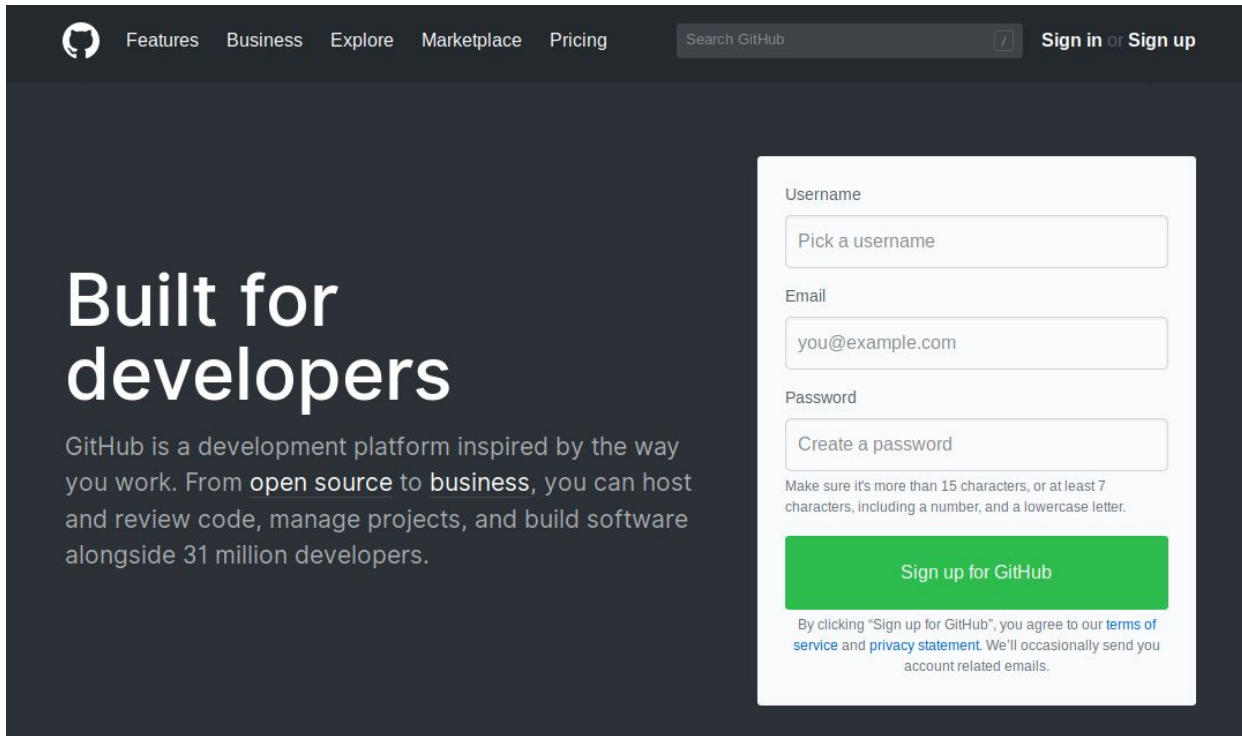


Remote repo
(GitHub - cloud)



Local repo
(your PC)

Let's create our GitHub account



The image shows the GitHub sign-up page. The header includes the GitHub logo, navigation links (Features, Business, Explore, Marketplace, Pricing), a search bar, and links to sign in or sign up. The main content area has a large heading 'Built for developers' and a paragraph describing GitHub as a development platform. On the right, there is a sign-up form with fields for Username, Email, and Password, followed by a green 'Sign up for GitHub' button and a disclaimer about terms of service and privacy statement.

GitHub

Features Business Explore Marketplace Pricing

Search GitHub

Sign in or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From [open source](#) to [business](#), you can host and review code, manage projects, and build software alongside 31 million developers.

Username

Pick a username

Email

you@example.com

Password

Create a password

Make sure it's more than 15 characters, or at least 7 characters, including a number, and a lowercase letter.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.



axosoft

GitKraken

- Install gitkraken
- Sync gitkraken with your GitHub account
- Create a repository on github
- Clone the repository with gitkraken
- And start making changes
-

Minimal usage

- Create a repository on github
- Clone (download) the repository
 - `Git clone <url>`
- Change something in one of the files
- Check what was changed
 - `git status`
- Commit (save) the changes
 - `Git add .`
 - `git commit -m "write a descriptive msg about what was changed"`
- Push (upload) the changes to your remote repo
 - `git push`

Minimal usage

- Create a repository on github
- Clone (download) the repository
 - `Git clone <url>`
- Change something in one of the files
- Check what was changed
 - `git status`
- Commit (save) the changes
 - `Git add .`
 - `git commit -m "write a descriptive msg about what was changed"`
- Push (upload) the changes to your remote repo
 - `git push`
- Change something on the remote repo (GitHub)
- Download (pull) the changes to your local repo
 - `git pull`

Exercise

- Open terminal/cmd
- Navigate to your local repo
- Create a new directory, called notebooks
- Enter the notebooks directory
- Create a jupyter notebook (using Jupyter Lab)
- Solve the exercises ([link](#)) in the notebook
- And update your remote repo, so we can check your answers online ;)

What we should see at the end?

- A notebooks directory, with a Jupyter Notebook which includes the answers to the exercises.

A better minimal usage

- Create a repository on github
- Clone the repository
 - `git clone <url>`
- Display the branches
 - `Git branch`
- Create a branch, called it `new_branch`
 - `git branch new_branch`
- Display the branches
- Go inside the *new_branch*
 - *`git checkout new_branch`*
- Make some changes

Exercise

- Create a new branch in your local repo, called *exercise_2*
- Solve the exercises ([link](#)) in the notebook
- And update your remote repo with the new branch

What we should see at the end is:

- You have two branches on your remote repo (same as your local)
- And the notebook in *exercise_2* branch has more content (the answers to the new exercises) than the notebook in

BREAK TIME



Recap

Can someone explain:

- What is git and Github?
- Why is git useful?
- What is a simple workflow in git?
- What have we done so far?

Keep the number of branches as low as possible

Increasing number of branches is an indicator of number of intended, but unfinished, changes.

That means you are having several mini-projects at the same time

And that means...

Keep the number of branches as low as possible

Increasing number of branches is an indicator of number of intended, but unfinished, changes.

That means you are having several mini-projects at the same time

And that means...

Don't do this!

Instead

Create a branch for a specific change (have descriptive names for branches - reduce your cognitive load). Work on that change until either it's done and you decide to combine it with the “stable” version, or forget about it.

Bottom line is: keep the number of branches low - keep your repo (and your mind) clean!

Instead

Create a branch for a specific change (have descriptive names for branches - reduce your cognitive load). Work on that change until either it's done and you decide to combine it with the “stable” version, or forget about it.

Bottom line is: keep the number of branches low - keep your repo (and your mind) clean!

How to combine two branches?

Combine

⇒

merge

Merge master branch with exercise_2

- Exit from jupyter lab
- Go to your remote repo
- Check the branches. Which branch are you in?
- Have you committed (saved) all the changes?
 - `git status`
- Go to the master branch
 - `git checkout master`
- Check your notebook again (the second exercise solutions should not be there). And exit!
- Merge (combine) the master branch and exercise_2 branch
 - `git merge exercise_2`
- Now check your notebook again and you should have the solutions

What can we do about the course material?

Always be updated with the latest material

- Remove the current directory, and clone it again
- Copy the notebook folder and rename it to “my_notebooks”
- Now, whenever you pull, the course notebook file will be updated
- Then you can copy the stuff from the notebook to my_notebook and play around with them