

GIT AND GITHUB

CHETAN CHAWLA

TECH CONSULTANT @ZS, EX-ASTROPHYSICS RESEARCHER @ASIAA, TAIWAN

SOURCE CONTROL

- ▶ Git is a **source or version control system** - to make versions of our project, structuring them and making it easy for us to revert/refer to the old versions and see for changes
 - ▶ The most traditional way to do source control is by **making different versions of your whole folder** - v1, v2, v3, etc. This is also done by cloud storages like Google Drive and Dropbox, which store different versions of the same file/folder
 - ▶ Git allows us to do this in a more flexible manner and in a more useful way for the code
- ▶ It **maintains history of all the changes in our code and tracks the changes as deltas**. It also allows collaborative work

WHY IS SOURCE CONTROL (GIT) USEFUL FOR US?

- ▶ We can revert back to older versions of our code
- ▶ We have a maintained history of all the changes in our code
- ▶ We can collaborate with multiple people!
- ▶ It takes less space than using several versions of the whole folder
- ▶ It keeps track of changes between increments

**USING
AIRDROP**



**USING
GOOGLE DRIVE**



**USING A
PRIVATE DISCORD**



**USING
GITHUB**



GIT MECHANICS

For the sake of simplicity, we will not be going into very deep details of Git, but will try to understand things on a higher level of abstractions which is useful (and often enough) in Astronomy



Disclaimer: This presentation is heavily motivated by Dr. Jason Wang's slides in Code/Astro and takes several references from them. The original slides can be found at [this link](#)

GIT REPOSITORY

- ▶ A git repository is a folder with tracking enabled
- ▶ It is a collection of codes and other files in a folder which keeps track of all the changes in these files
- ▶ All version changes are stored as chains of deltas in this .git subfolder. Think of it as a metadata that stores all information about your data and the changes
- ▶ To initialize any folder as a git repo, we use the command `git init`



git init



GIT



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/chetanchawla/Astrophysics/Mentoring/Astrosprint_Git/Example_Tutorial/.git/
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```


ADDING A NEW FILE

- ▶ Let us create a new code for Python
- ▶ We save the .py file in the folder we created and initialized git into
- ▶ Note: We are using the folder name as Example_Tutorial

Name	Date Modified	Size
▼  Example_Tutorial	Today at 5:45 PM	
 python_example.py	Today at 6:09 PM	

Tutorial

└─python_example.py

```
1 print ('Welcome to Intro2Astro 2025')
2 print ('We will use this for the Git and Github Tutorial')
3
4 ball_1='blue'
5 ball_2='red'
6
7 print ('Color of ball 1 is ', ball_1)
8 print ('Color of ball 2 is ', ball_2)
```

GIT STATUS

- ▶ `git status` command tells us the current status of all the files in the git folder (both tracked and untracked)
- ▶ As we added a new file, git status will tell us that the file is currently untracked by git

```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
python_example.py


nothing added to commit but untracked files present (use "git add" to track)
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```


GIT STAGING


- ▶ Git staging is used to add a file for tracking its changes
- ▶ Once a file is added, every change that is made in the file is tracked by git
- ▶ In case we have a large number of files, we can also use `git add *` to stage all the files at once
- ▶ A word of precaution: When using `git add *` to stage all files together, make sure you do not include unnecessary files. This can also be achieved by setting up a `.gitignore` file which is used to specify intentionally untracked files to ignore



```
git add example_python.py
```



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git add python_example.py
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git status
On branch master

No commits yet

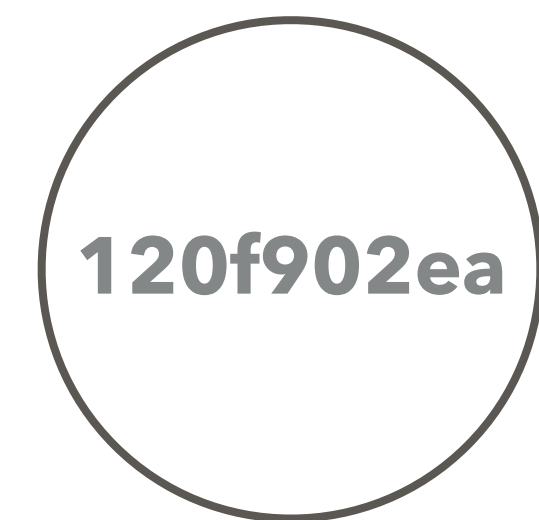
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   python_example.py

(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```

GIT COMMIT

- ▶ A git commit is the most fundamental unit of our model. It commits (or logs and saves) our changes upto a certain point as a checkpoint in git
- ▶ We can revert back to the previous git commit in case we want to go back to some checkpoint in our code history
- ▶ We can think of git commit as an object. It holds a specific ID, a parent ID, the changes, a user specified message, and metadata.

An example commit



ID: 120f902ea

Parent ID: 0012182ad

Changes = {create abc.py: add <text>
line 0
MODIFIED pqrs.py: add <text> line 12
delete [TEXT] line 36
...}

Message: Made xyz changes to abc.py
and pqrs.py

Author: Chetan Chawla

Email: chetan@abc.com

Time: 23/08/2021 6:15pm

GIT COMMIT

- ▶ So, a git commit holds all the changes compared to a previous git commit
- ▶ In case it is the first commit, it will track the changes from git init, or all the changes as compared to a blank repository
- ▶ A git repository typically contains chains of such git commits

```
git commit -m "Initial commit,  
added file example_python.py"
```



ID: e62dfea

Parent ID: None

Changes = {CREATED example_python.py
ADD <text> line 0}

Message: Initial commit, added
file example_python.py

GIT



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git commit -m "Initial commit, added file example_python.py"
[master (root-commit) e62dfea] Initial commit, added file example_python.py
 1 file changed, 8 insertions(+)
 create mode 100644 python_example.py
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git status
On branch master
nothing to commit, working tree clean
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```

GIT COMMIT TREE

- ▶ When we make a change (here adding a new ball to the python code), we have to stage the changes and then commit them
- ▶ The new commit saves these changes as deltas (and not entire copy of the folder)
- ▶ This commit is connected to its parent commit using the parent ID. Each commit knows of its parent commit but not vice versa



```
1 print ('Welcome to Intro2Astro 2025')
2 print ('We will use this for the Git and Github Tutorial')
3
4 ball_1='blue'
5 ball_2='red'
6 ball_3='green'
7
8 print ('Color of ball 1 is ', ball_1)
9 print ('Color of ball 2 is ', ball_2)
10 print ('Color of ball 3 is ', ball_3)
```



```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git status
```

On branch master

Changes **not** staged **for** commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes **in** working directory)

modified: python_example.py

no changes added to commit (use "git add" **and/or** "git commit -a")

```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git add python_example.py
```

```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git commit -m "Added new ball in example_python.py"
```

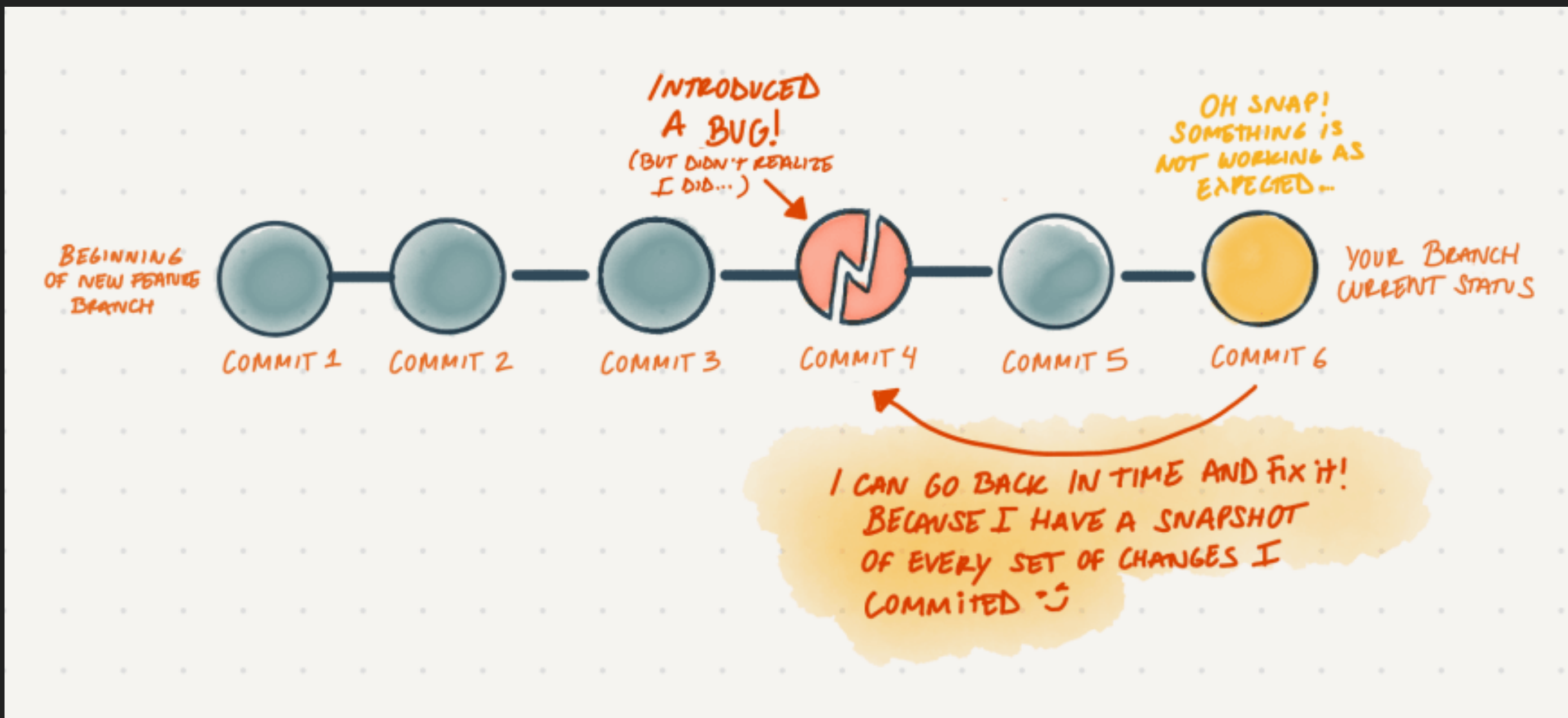
[master f8e58c8] Added new ball **in** example_python.py

1 file changed, **3** **insertions**(+), **1** **deletion**(-)

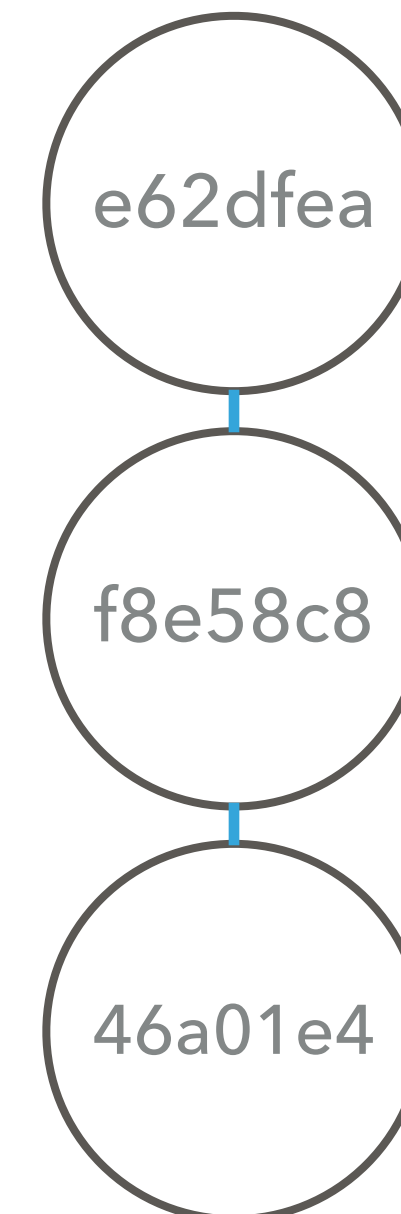
```
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```

GIT COMMIT TREE

- ▶ All the commits work like this and form a tree which can be used to track back the history of all the tracked files in the git repository
- ▶ We can also revert to specific commits based on their IDs



```
git commit -m "Change in  
example_python.py"
```



ID: 46a01e4

Parent ID: f8e58c8

```
Changes = {MODIFIED python_example.py  
DELETE <text> line 6  
ADD <text> line 6  
}
```

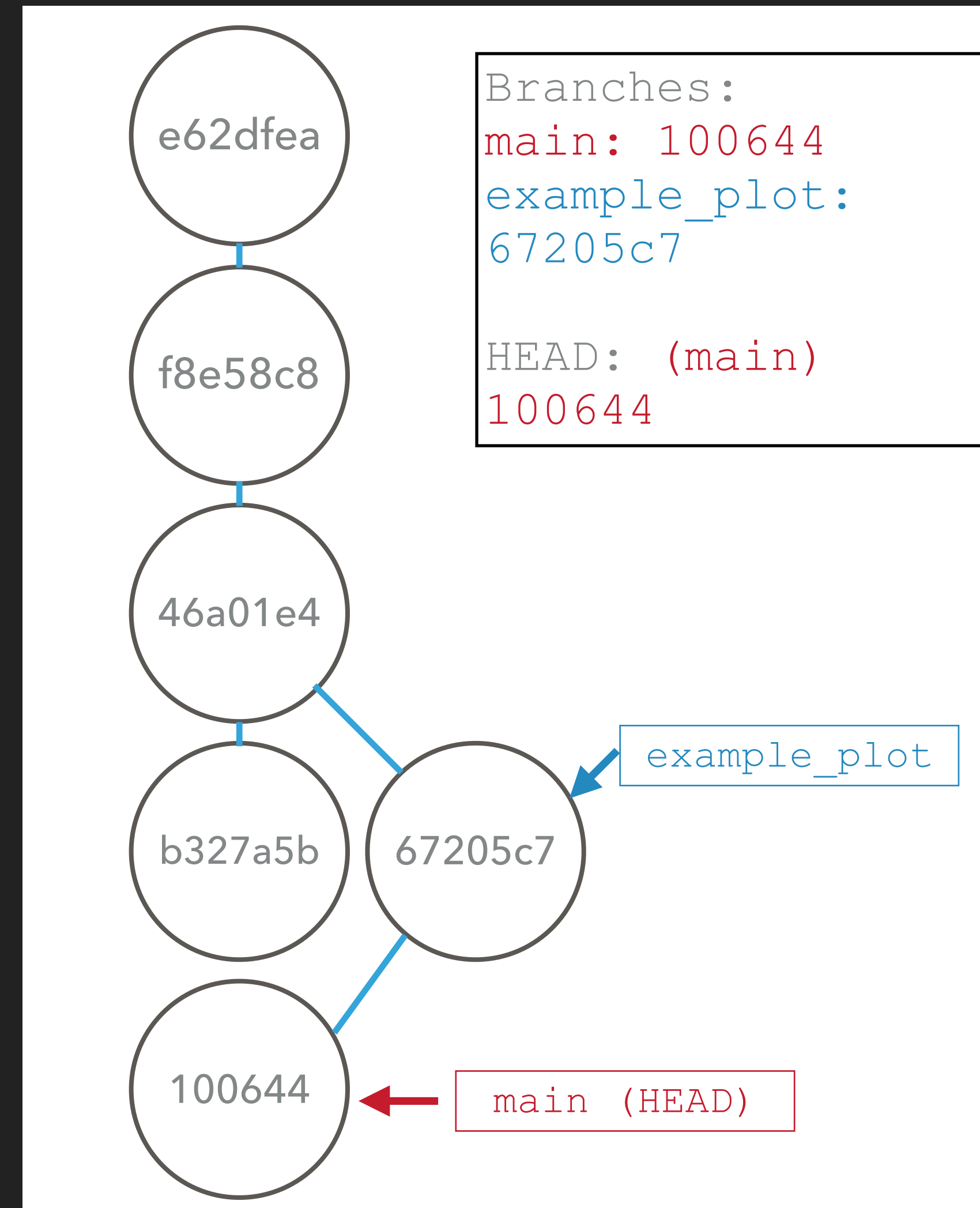
Message: Change in example_python.py,
input ball_2 color

GIT BRANCHES

- ▶ Git allows us to work on different branches of the repository. This can be used for having a main or base branch for the final features, and having separate branches for new features, etc.
- ▶ The base branch (with which we have been working so far) is called main or master

GIT MERGE

- ▶ Branches are also used by different users to effectively isolate their code bases and merge the code whenever required



MERGE CONFLICTS

- ▶ However there are cases where two different branches have different text on the same line in the same file. This is a conflict
- ▶ To merge the two branches, git asks us to manually resolve these conflicts by choosing the correct code or making a hybrid (as we see fit)

```
15 <<<<<<< HEAD
16 print('1')
17 =====
18 print('2')
19 >>>>>>> example_plot
```

INSTALLING GIT

For mac users, it is already installed

For windows/linux users, head to the site - <https://git-scm.com/downloads> and follow the instructions

TRYING ON YOUR OWN

Let us try what we discussed and summarize it!



```
#To create a repository  
git init
```

```
#After adding files, writing code, etc  
#Staging files for tracking  
git add file_name
```

```
#Checking the status of the files, branches and commits  
git status
```

```
#Committing the changes as a savepoint with a message  
git commit -m "This commit contains that change"
```

GITHUB

Github is an online hosting website for hosting your repositories, either publicly or privately. Github also hosts a set of functionalities to make open source development easier.

GitHub enables us to collaborate with people around the world!

USING GITHUB

- ▶ Create your account on GitHub : github.com
- ▶ After you make your account, create a new repository
- ▶ If you are working on a completely new project (without any files already present on your local machine), we use `git clone` to download this repository into your local machine. If we do this, we can skip the `git init` part. We can also check the README, .gitignore and LICENSE checkboxes in this case



```
git clone https://github.com/chetanchawla/astrosprint_git.git
```

USING GITHUB

- For our case, we will connect our local repository to this github repository using

```
● ● ●
#This lone is used to connect the repository
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git remote add origin
https://github.com/chetanchawla/astrosprint_gittut.git

#This is used to specify that the Master branch is called the main branch
#(A better nomenclature than the traditional master-slave terminology)
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git branch -M main

#Git push uploads our local changes to the remote repository on GitHub
#-u origin main tells that we have to upload things by setting the upstream to origin
#repository that we connected above in the main branch (Only needed for the first time
#and when we are changing branches)
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial % git push -u origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (17/17), 1.67 KiB | 1.67 MiB/s, done.
Total 17 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/chetanchawla/astrosprint_gittut.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
(base) chetanchawla@Chetans-MacBook-Air Example_Tutorial %
```

When you're dead but remember you forgot to git commit git push your last code iterations



USING GITHUB

- ▶ If there are multiple machines that we want to setup this repository on, we use `git clone` to clone the repository on the second machine
- ▶ We use `git pull` to download any changes that are present in the GitHub repository but not on your local git repo (can be due to multiple users, or multiple machines)

USING GITHUB WITH COLLABORATORS

- ▶ In case we need to work with several users to make a modification/change/addition/deletion to someone else's repository, we first need to make a copy of the repo into our own GitHub (`howardisaacson/intro-2-astro2025` -> `chetanchawla/intro-2-astro2025`). This is done using the **fork** button
- ▶ We then need to clone this repository to our local system using `git clone`.
- ▶ After we are done making changes in the local repository, we `push` these changes. These would be uploaded in our version of the repository (`chetanchawla/intro-2-astro2025`)
- ▶ To ask the main repository owner to incorporate these changes in their own repository, we need to make a **Pull Request** using GitHub along with explaining the changes we made. The main user can then accept the changes which will be automatically incorporated into the main repository as well (`howardisaacson/intro-2-astro2025`)

USING GITHUB WITH COLLABORATORS

- ▶ Word of caution: Always check if there are any changes in the main repository before you are making changes. if there are, you can pull the latest changes in your version of GitHub repo from the main repo (`howardisaacson/intro-2-astro2025` -> `chetanchawla/intro-2-astro2025`)
- ▶ In case the main author as well as you worked on the same part of the code, conflict resolution will be required

TRYING ON YOUR OWN

Let us try what we discussed and summarize it!



SUMMARY

For a very basic workflow

IN CASE OF FIRE 🔥

1. git commit

2. git push

3. git out!

```
● ● ●  
#Create a new repository on GitHub for your project/folder  
#Clone the repository into a folder in your local machine  
git clone https://github.com/chetanchawla/astrosprint_gittut.git  
  
#Work on code, stage the changes, commit the changes and push the changes  
git add filename  
git commit -m "Initial commit"  
git push  
  
#Pull changes from repo  
git pull
```

ASSIGNMENT

- ▶ Login to GitHub and fork [Intro-2-Astro 2025 repository](#) to your own GitHub
- ▶ Clone the forked Repository (your_username/Intro-2-Astro2025) on your local
- ▶ Paste your assignments from week 1 and 2 in the respective folder
- ▶ Add and stage these files and commit the changes
- ▶ Push these changes to GitHub and share the links to your [GitHub Repos here](#)

THANK YOU FOR LISTENING 🎉

Feel free to send questions to
chetanchawlacc4@gmail.com

Website: sites.google.com/view/chetanchawla

SOME REFERENCES

- ▶ An intro to Git Mechanics and Git Flow: <https://github.com/semaphoreP/codeastro/tree/main/Day2>
- ▶ Git Cheatsheet: <https://education.github.com/git-cheat-sheet-education.pdf>
- ▶ A nice blogpost (memes were also courtesy of this blogpost): <https://medium.com/@lulu.ilmaknun.q/kompilasi-meme-git-e2fe49c6e33e>
- ▶ Official Git Book: <https://git-scm.com/book/en/v2>
- ▶ Like most other programming tools, you can find most things using Google!