CoGrammar

Version Control with Git

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Progression Criteria

✓ **Criterion 1: Initial Requirements**

- Complete 15 hours of Guided Learning Hours and the first four tasks within two weeks.

✓ **Criterion 2: Mid-Course Progress**

- Software Engineering: Finish 14 tasks by week 8.
- Data Science: Finish 13 tasks by week 8.

✓ **Criterion 3: Post-Course Progress**

- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

✓ **Criterion 4: Employability**

- Record a Final Job Outcome within 12 weeks of graduation, or by 23rd September 2024.

Unlock Prestigious Co-Certification Opportunities

New Partnerships Unveiled!

- **University of Manchester & Imperial College London** join our circle along with The University of Nottingham Online.

Exclusive Opportunity:

- Co-certification spots awarded on a first-come basis.
- Meet the criteria early to gain eligibility for the co-certification.

Key Deadlines:

- **11 March 2024**: 112 Guided Learning Hours & 'Build Your Brand' tasks completion.
- **18 March 2024**: Record interview invitation or self-employment.
- **15 July 2024**: Submit verified job offer or new contract.

CoGrammar

Lecture Objectives

1. Understand what a repository is.
2. Learn the relevance of using Version Control.
3. Learn how to use Git.

What is Version Control?

- **Code base is stored in a central place.**
- **Format used: deltas.**
  - This means that only changes between versions are saved.
  - This means that you can “roll back” your code to a previous version.

Why Version Control?

- **Collaboration**
  - Multiple people working on the same file at the same time.
  - Hard to keep track of what changes happen when.
  - Certain changes can be accidentally overwritten.
- **Storing Versions**
  - Being able to rollback code becomes a great emergency tactic, when bugs become too difficult to handle.
- **Understanding What Happened**
  - Full history of who made what changes.

Some Terminology

- **Version**
  - Code at a particular state.
- **Repository**
  - The collection of all files at all versions.
- **History**
  - The list of all changes made to a set of files.
- **Commit**
  - A wrapper for a set of changes.
- **Staging Area**
  - A file containing changes to be added to the next commit.

Introducing Git

- **Most widely used version control system.**
- **Free and open-source. Designed to handle a large variety of systems.**
- **Distributed architecture:**
  - When you download a repository, you download the full history of changes to your local computer.
- **Everything is run from the command-line using the git application.**

Repositories

- **Two types: local and remote.**
- **All changes stored in a hidden file called “.git”.**
- **Two ways to get a repository:**
  - Create a new one using git init.
  - Get a remote one using git clone <repository-url>.

Committing Code

- **First, you need to add your files to the staging area.**
  - git add <file-name>
- **Once you have added all files to the staging area, then you can commit your code.**
  - git commit -m <commit-message>
  - NB: Each commit has to have a message attached to it.
  - The message just explains what changed.

Viewing the Status of the Commit

- **git status**
- **Shows all new files, changed files, and files added to the current commit.**
- **E.g:**

```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

new file:   newFile.py
```

Viewing the Version History

- **git log**
- **Shows the commit hash (a unique identifier for the commit), Author, Date and the commit message.**
- **E.g:**

```
commit a9ca2c9f4e1e0061075aa47cbb97201a43b0f66f
Author: HyperionDev Student <hyperiondevstudent@gmail.com>
Date: Mon Sep 8 6:49:17 2017 +0200

Initial commit.
```

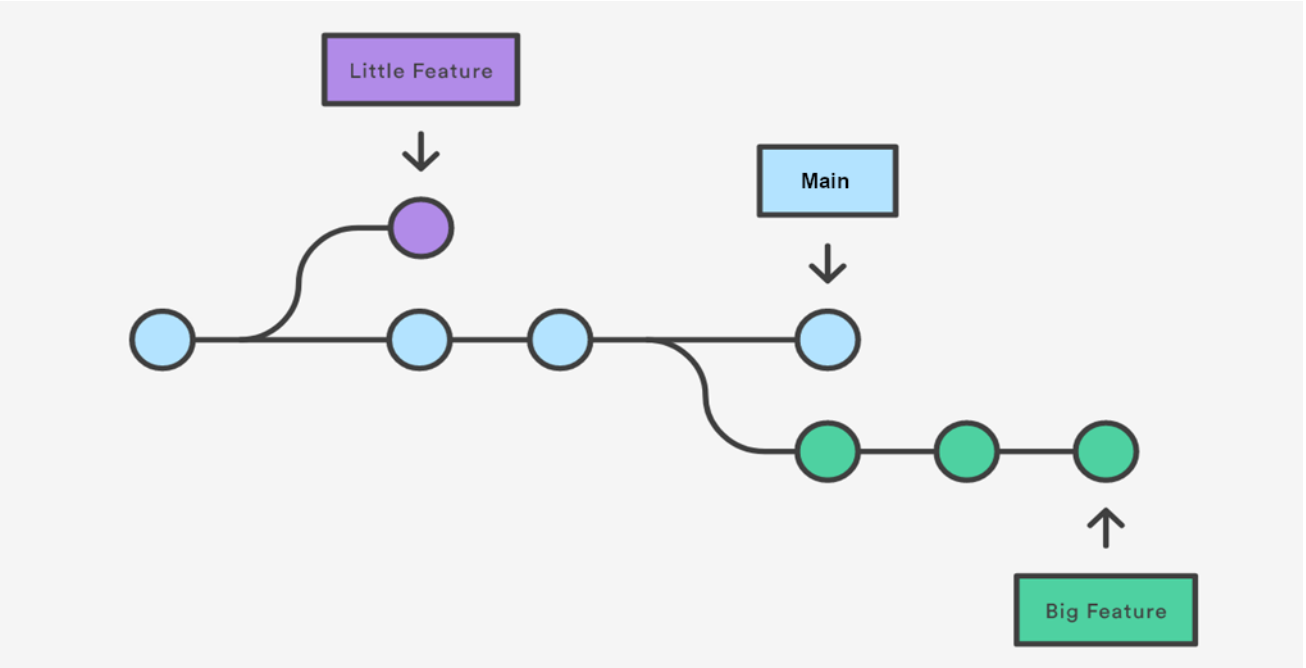
Branching

- Sometimes, a developer needs to work independently on the same code base.
- For example: adding a new feature.
- With other changes constantly being made, this can sometimes be difficult and cause many merge conflicts.
- Solution: branching.

Branching (Continue)

- **git branch <branch-name>**
- **To switch branches:**
  - git checkout <branch-name>
- **By default, Git uses main as the name of the main branch.**
  - This used to be called master, until Git decided that was a bad idea.

Branching Visualisation



Stashing Changes

- When switching branches, Git will throw up a fuss if you have uncommitted changes.
- However, sometimes your changes are not yet ready for a commit.
- You can use git stash to temporarily save your changes to a clipboard without committing.
- To get your changes back, git stash pop will get the latest stash on the clipboard.

Merging

- There is no use in branching code to make a new feature without being able to make it a part of the main branch.
- Merging allows you to take the changes that you have made in your branch and apply them to the main branch (or another branch of your choice).
- To merge bug-fix branch into main branch:
  - git checkout main
  - git merge bug-fix

CoGrammar

Questions around Git

CoGrammar

Thank you for joining