Welcome to this CoGrammar session:

Git Revision

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- 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 Academic Sessions. You can submit these questions here: Questions

Software Engineering Session 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

Software Engineering Session Housekeeping cont.

- "Please check your spam folders for any important communication from us. If you have accidentally unsubscribed, please reach out to your support team.“
- Rationale here: Career Services, Support, etc will send emails that contain NB information as we gear up towards the end of the programme. Students may miss job interview opportunities, etc.

Skills Bootcamp 8-Week Progression Overview

✓ Criterion 3: Course Progress

- ***Completion:*** All mandatory tasks, including Build Your Brand and resubmissions by study period end
- *Interview Invitation:* Within 4 weeks post-course
- *Guided Learning Hours:* Minimum of 112 hours by support end date (10.5 hours average, each week)

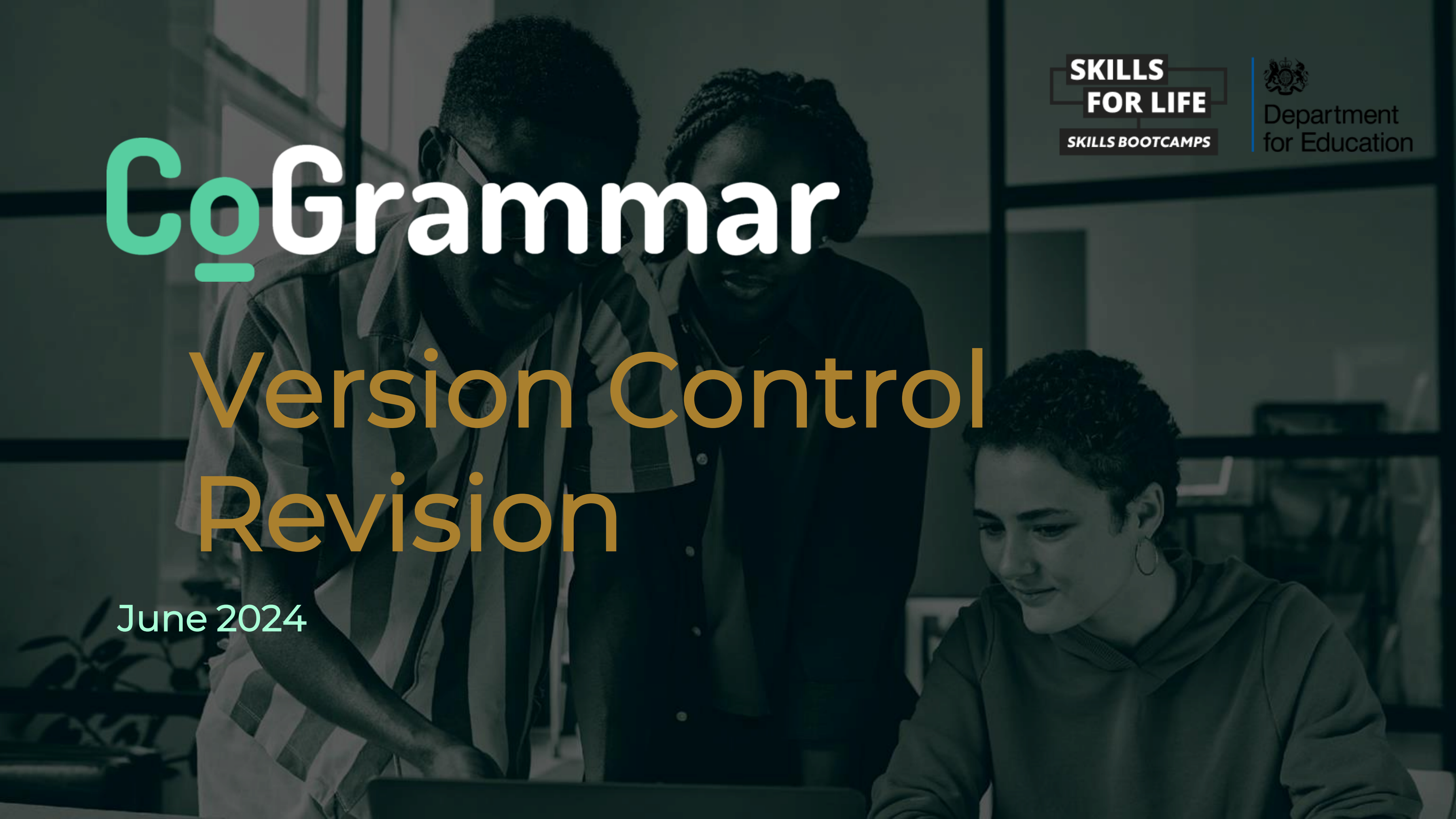
✓ Criterion 4: Demonstrating Employability

- *Final Job or Apprenticeship Outcome:* Document within 12 weeks post-graduation
- *Relevance:* Progression to employment or related opportunity

CoGrammar

Learning Outcomes

- Identify the basic concepts of version control and Git.
- Explain the purpose and benefits of version control systems.
- Describe the basic commands and operations in Git.
- Initialise a Git repository.
- Stage and commit changes to a repository.
- Resolve merge conflicts effectively.
- Assess the impact of version control on collaboration.
- Collaborate on a shared project using remote repositories and platforms like GitHub.



SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education

CoGrammar

Version Control Revision

June 2024

What is Version Control?

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- The Code base is stored in a central place.
- Format used: deltas.
- This means that only changes between versions are saved.
- You can therefor “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.
- Understanding What Happened
  - Full history of who made what changes.

CoGrammar

Why Version Control?

- Storing Versions
  - Being able to rollback code becomes a great emergency tactic, when bugs become too difficult to handle.
  - Multiple versions and branches of a project can be managed.

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.

Git

CoGrammar

What is Git?

- Git is a distributed version control system for tracking changes in source code during software development.

CoGrammar

Why 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.

Git Installation and Setup

- Download and install Git from git-scm.com.
- Configure user name and email:
  - git config --global user.name "Your Name"
  - git config --global user.email "your.email@example.com"

CoGrammar

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>.

CoGrammar

Initialising a New Local Repository

- mkdir my-project
- cd my-project
- git init
  - Create a .git directory that contains all the repository's metadata and object database.

Viewing the Commit Status

- 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
```

CoGrammar

Staging Changes

- First, you need to add your files in the working directory to the staging area.
  - git add <file-name>
- The file is now being tracked and staged for commit.

CoGrammar

Committing Changes

- 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 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

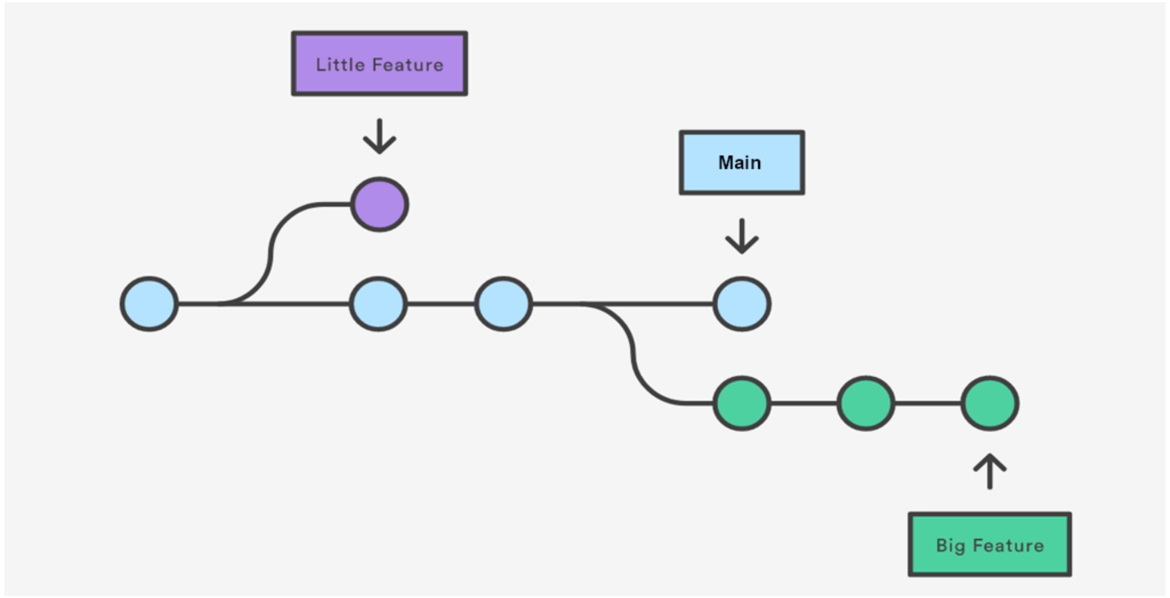
CoGrammar

Branching (Continue)

- To create a new branch:
    - 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.

CoGrammar

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

Handling Merge Conflicts

- Merge conflicts occur when changes in two branches conflict.
- Resolution Steps:
    - Identify conflict files using git status.
    - Manually resolve conflicts in the files.
    - Stage the resolved files using git add.
    - Complete the merge with git commit.

Working with Remote Repositories

- **Commands:**
  - git remote add origin <remote-url> : Add a remote repository.
  - git push -u origin main : Push changes to the remote repository.
  - git pull origin main : Pull changes from the remote repository.
- This will synchronise the local repository with the remote repository.

CoGrammar

Let’s take a short break

CoGrammar

Demo time!

Questions and Answers

CoGrammar

Summary

- Why OOP is Essential in Programming
- Implementing a Class
- Usage of Access Control
- Principles of Encapsulation and Abstraction
    - Encapsulation bundles data and methods that operate on the data within a single unit (class), hiding details.
    - Abstraction focuses on representing the essential features of an object while hiding unnecessary details, improving code readability and maintenance.
- Demonstration of Inheritance and Polymorphism

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar