Welcome to this CoGrammar Task Walkthrough: Task 11

The session will start shortly...

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

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** throughout this session, should you wish to ask any follow-up questions.
- 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**

Enhancing Accessibility: Activate Browser Captions

Why Enable Browser Captions?

- Captions provide **real-time text for spoken content**, ensuring inclusivity.
- Ideal for individuals in noisy or quiet environments or for those with **hearing impairments**.

How to Activate Captions:

1. **YouTube or Video Players:**
   - Look for the CC (Closed Captions) icon and click to enable.
2. **Browser Settings:**
   - Google Chrome: Go to *Settings* > *Accessibility* > *Live Captions* and toggle ON.
   - Edge: Enable captions in *Settings* > *Accessibility*.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Progression Overview

✅ Criterion 1 - Initial Requirements

Specific achievements within the first two weeks of the program.

To meet this criterion, students need to, by no later than **01 December 2024:**

- **Guided Learning Hours** (GLH)**:** Attend a minimum of 7-8 GLH per week (lectures, workshops, or mentor calls) for a total minimum of **15 GLH**.
- **Task Completion:** Successfully complete the **first 4 of the assigned tasks**.

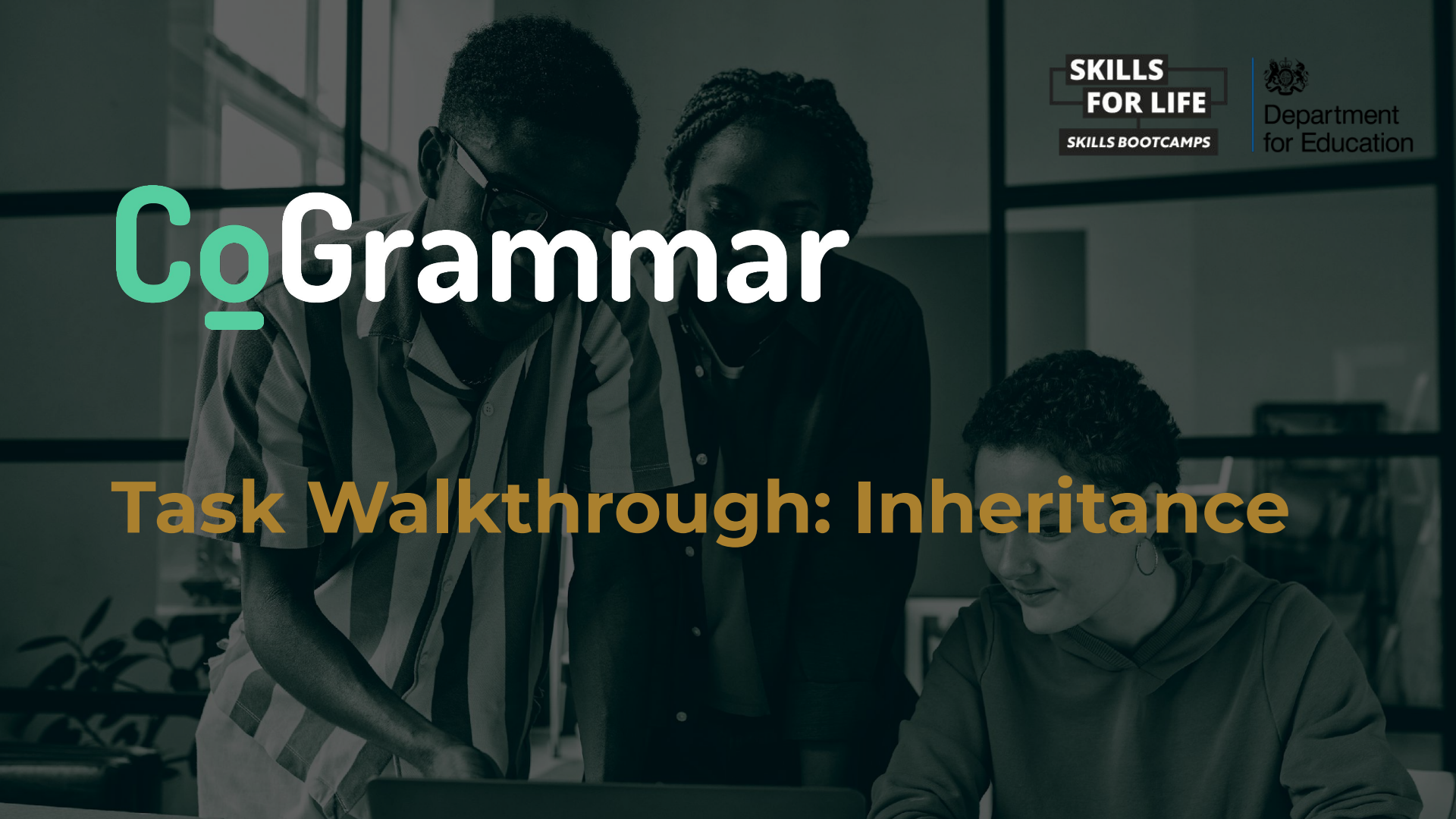
✅ Criterion 2 - Mid-Course Progress

Progress through the successful completion of tasks within the first half of the program.

To meet this criterion, students should, by no later than **12 January 2025:**

- **Guided Learning Hours** (GLH)**:** Complete at least **60 GLH**.
- **Task Completion :** Successfully complete the **first 13 of the assigned tasks**.

CoGrammar



Task Walkthrough: Inheritance

Learning Outcomes

- Understand and implement Object-Oriented Programming concepts: Classes, objects, attributes, methods, inheritance and method overriding.
- Implement conditional logic for object instantiation.
- Understand the default role of a class data structure.

OOP Recap

- A **class** is a blueprint for creating objects, defining attributes and methods that objects from this class can use.
- An **object** is an instance of a class that contains actual values for the attributes defined by the class.
- **Attributes** are variables that belong to an object defined in the class's constructor using the __init__ method.
- **Methods** are functions defined inside a class that operate on instances of that class.
- A **constructor** is a special method that is automatically called when an object is created.

Advanced Concepts

- **Inheritance** is a mechanism in object-oriented programming where a subclass inherits attributes and methods from a parent class.
- **Method overriding** occurs when a subclass provides a specific implementation of a method that is already defined in its parent class, allowing the subclass to customise or extend the behavior of that method.

Inheritance Task Walkthrough: Task 1

CoGrammar

Practical task 1

In this task, you will demonstrate your understanding of inheritance. Make a copy of the **task1_instructions.py** file and name it **practical_task_1.py.** Then, follow the instructions below.

- Add another method in the Course class that prints the head office location: Cape Town.
- Create a subclass of the Course `class` named `OOPCourse.`
- Create a constructor that initialises the following attributes with default values:
    - `description = “OOP Fundamentals”`
    - `trainer = "Mr Anon A. Mouse"`
- Create a method in the `OOPCourse` subclass named `trainer_details` that prints what the course is about and the name of the trainer by using the description and trainer attributes.
- Create a method in the `OOPCourse` subclass named `show_course_id` that prints the ID number of the course: `#12345`
- Create an object of the OOPCourse subclass called `course_1` and call the following methods
    - `contact_details()`
    - `trainer_details()`
    - `show_course_id()`
- These methods should all print out the correct information to the terminal.

Inheritance Task Walkthrough: Task 2

CoGrammar

Practical task 2

Create a file named **method_override.py** and follow the instructions below:

- Take user inputs that ask for the name, age, hair colour, and eye colour of a person.
- Create an Adult class with the following attributes and method:
    - Attributes: `name, age, eye_color,` and `hair_color`
    - A method called `can_drive()` which prints the name of the person and that they are old enough to drive.
- Create a subclass of the `Adult` class named `Child` that has the same attributes, but overrides the `can_drive()` method to print the person's name and that they are too young to drive.
- Create some logic that determines if the person is 18 or older and create an instance of the `Adult` class if this is true. Otherwise, create an instance of the `Child` class. Once the object has been created, call the `can_drive()` method to print out whether the person is old enough to drive or not.

Be sure to place files for submission inside your **task folder** and click "**Request review**" on your dashboard.

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar