Welcome to this CoGrammar Task Walkthrough: Task 9

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** 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

<u>Why Enable Browser Captions</u>?

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

<u>How to Activate Captions</u>:

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

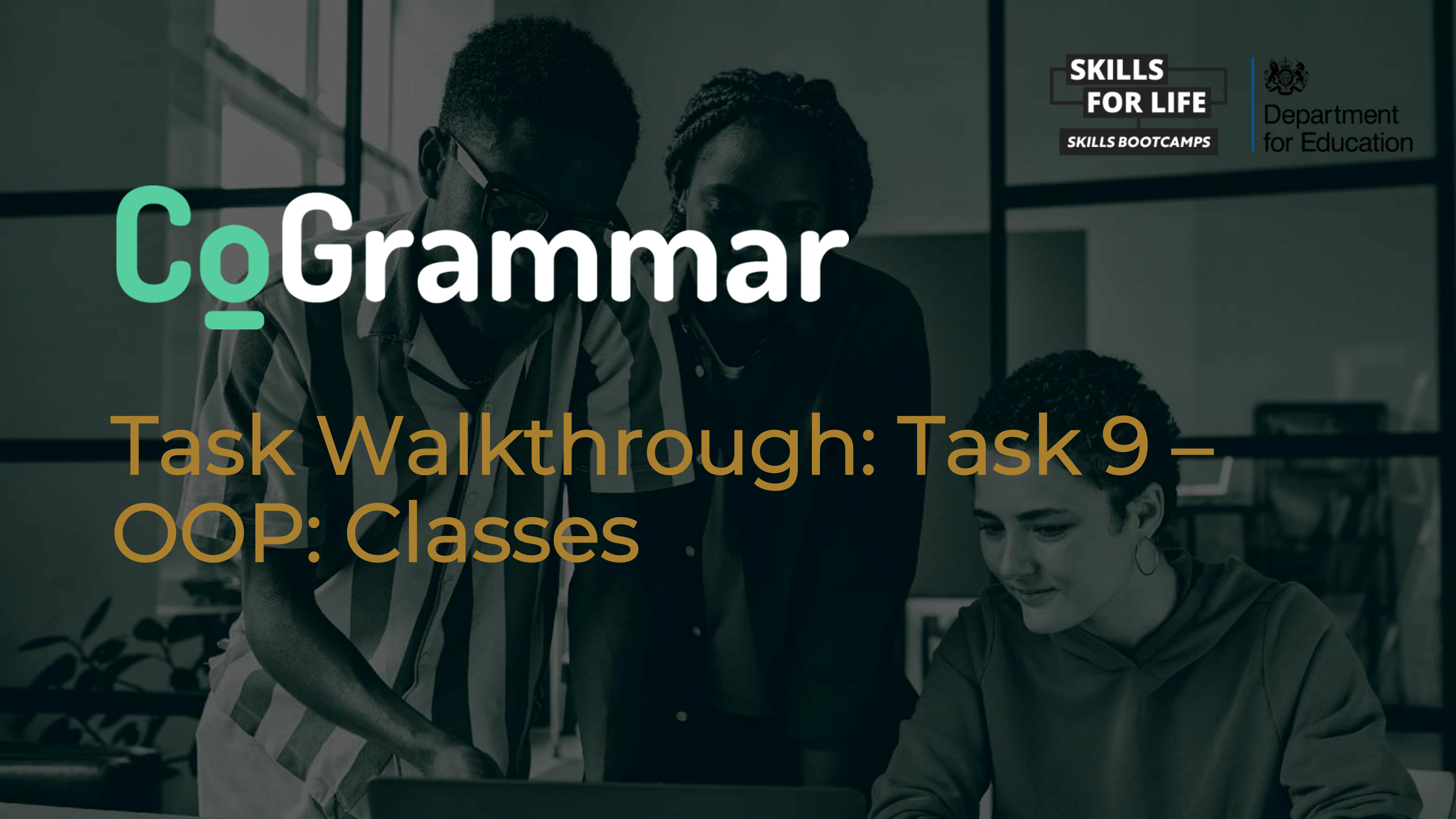
or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Learning Outcomes

- Understand and implement Object-Oriented Programming concepts.
- Utilise dictionaries for efficient data storage and retrieval.
- Implement functions, conditional statements and loops for user interaction with a user-driven library system.
- Transfer your learnings to complete the task.

CoGrammar



SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

Task Walkthrough: Task 9 – OOP: Classes

OOP

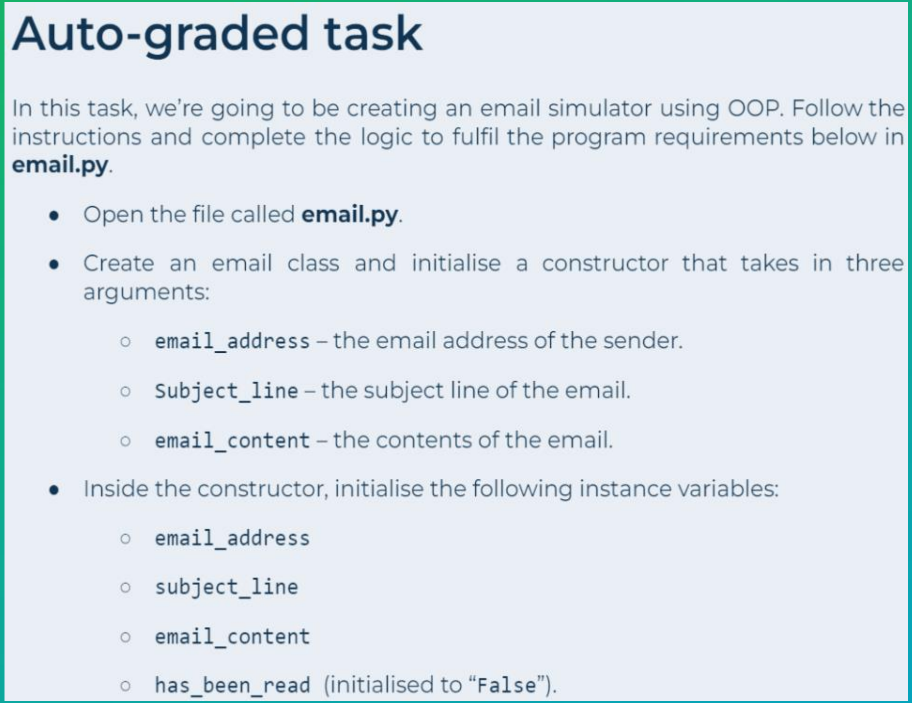
- A class is a blueprint for creating objects, defining attributes and methods that objects from this class can use.
- An object is an instance of a class that contains actual values for the attributes defined by the class.
- Attributes are variables that belong to an object defined in the class's constructor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

OOP

- A constructor is a special method that is automatically called when an object is created.
- Encapsulation is the practice of bundling related attributes and methods into a structured unit which also involves restricting direct access to data.
- Self-referencing: self is a reference to the current instance of the class, allowing methods to access and modify the object’s attributes and call other methods within the class.

Auto-graded Task Walkthrough

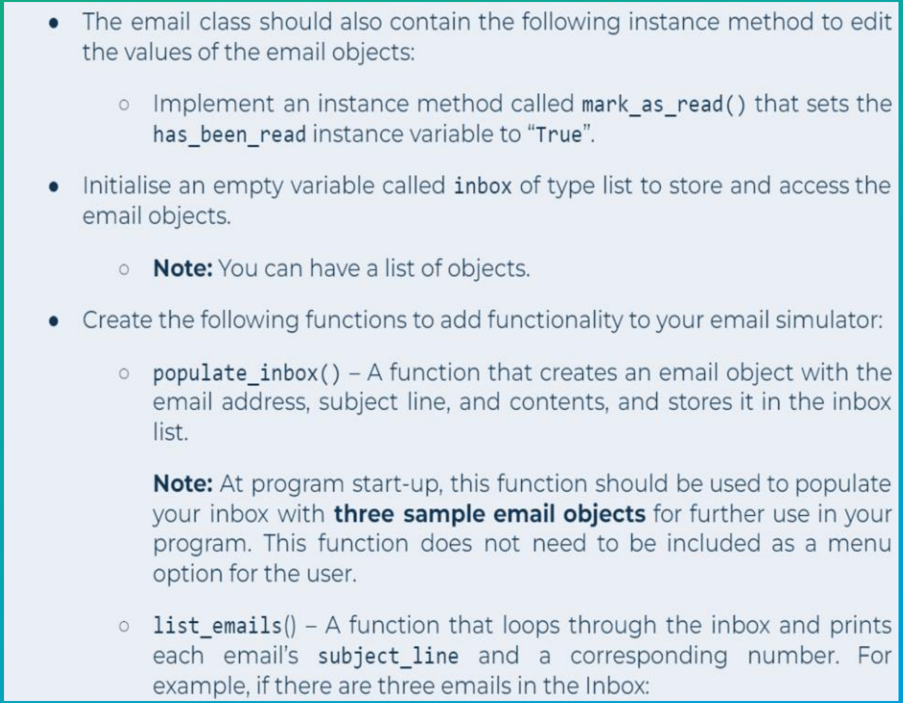
CoGrammar



Auto-graded task

In this task, we're going to be creating an email simulator using OOP. Follow the instructions and complete the logic to fulfil the program requirements below in **email.py**.

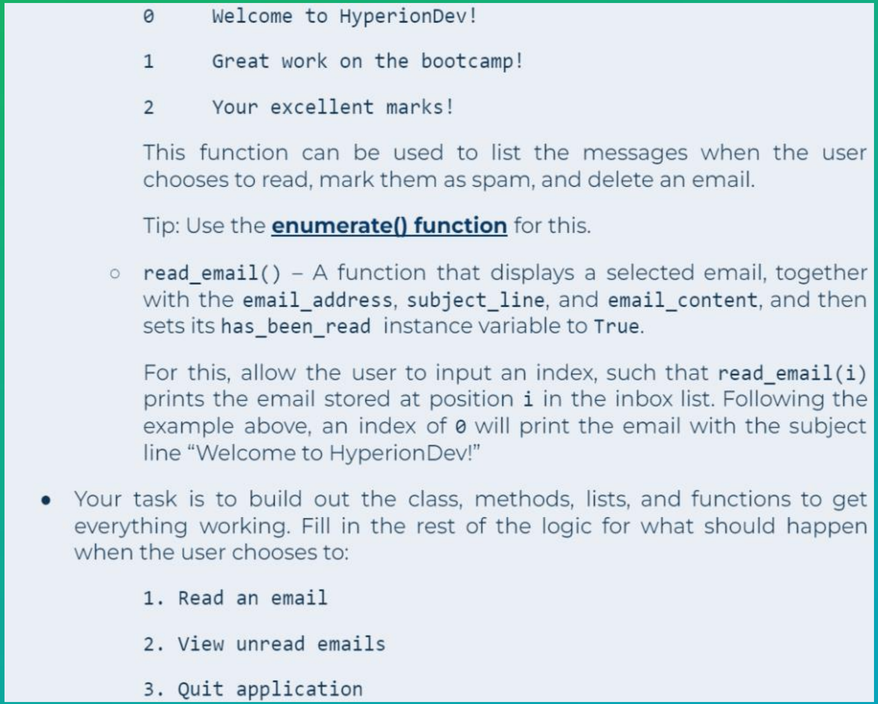
- Open the file called **email.py**.
- Create an email class and initialise a constructor that takes in three arguments:
    - `email_address` – the email address of the sender.
    - `Subject_line` – the subject line of the email.
    - `email_content` – the contents of the email.
- Inside the constructor, initialise the following instance variables:
    - `email_address`
    - `subject_line`
    - `email_content`
    - `has_been_read` (initialised to "`False`").



- The email class should also contain the following instance method to edit the values of the email objects:
    - Implement an instance method called `mark_as_read()` that sets the `has_been_read` instance variable to "`True`".
- Initialise an empty variable called `inbox` of type list to store and access the email objects.
    - **Note:** You can have a list of objects.
- Create the following functions to add functionality to your email simulator:
    - `populate_inbox()` – A function that creates an email object with the email address, subject line, and contents, and stores it in the inbox list.

      Note: At program start-up, this function should be used to populate your inbox with **three sample email objects** for further use in your program. This function does not need to be included as a menu option for the user.
    - `list_emails`() – A function that loops through the inbox and prints each email's `subject_line` and a corresponding number. For example, if there are three emails in the Inbox:

CoGrammar



```
0    Welcome to HyperionDev!

1    Great work on the bootcamp!

2    Your excellent marks!
```

This function can be used to list the messages when the user chooses to read, mark them as spam, and delete an email.

Tip: Use the **enumerate() function** for this.

- `read_email()` – A function that displays a selected email, together with the `email_address`, `subject_line`, and `email_content`, and then sets its `has_been_read` instance variable to `True`.

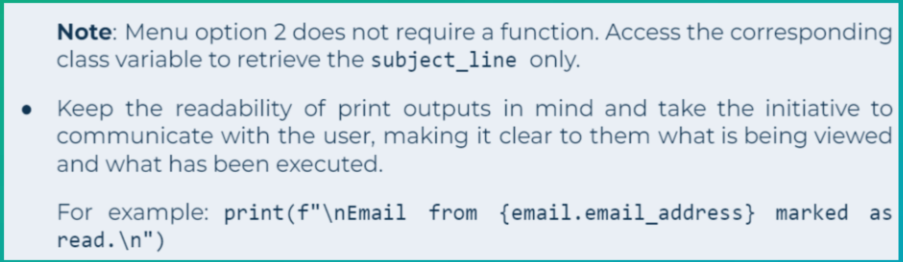
  For this, allow the user to input an index, such that `read_email(i)` prints the email stored at position `i` in the inbox list. Following the example above, an index of `0` will print the email with the subject line "Welcome to HyperionDev!"

- Your task is to build out the class, methods, lists, and functions to get everything working. Fill in the rest of the logic for what should happen when the user chooses to:

```
1. Read an email

2. View unread emails

3. Quit application
```



Note: Menu option 2 does not require a function. Access the corresponding class variable to retrieve the `subject_line` only.

- Keep the readability of print outputs in mind and take the initiative to communicate with the user, making it clear to them what is being viewed and what has been executed.

  For example: `print(f"\nEmail from {email.email_address} marked as read.\n")`

CoGrammar

email.py

Task Objective

- The objective of this task is to demonstrate your knowledge of Object-Oriented Programming (OOP) concepts, including classes, methods, constructors, and instance variables, while applying conditional logic and list operations to build an email simulator. You will:
    - Use a class to model the attributes and behaviours of an email.
    - Dynamically manage a list of email objects using functions that allow operations such as reading, listing, and interacting with emails.
    - Apply logic to control instance variables (e.g., marking emails as read) and improve user interaction by providing an intuitive interface.
- This task is designed to enhance your understanding of OOP principles, list management, and method implementation, as well as strengthen your problem-solving skills in Python.

CoGrammar

Questions and Answers

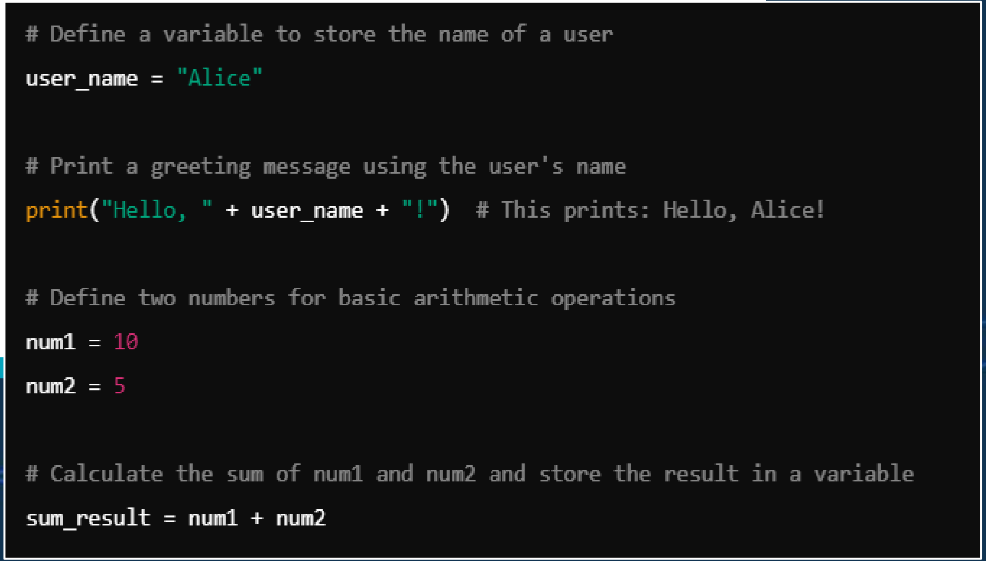
CoGrammar

Documentation and Style

- Add comments to your code. Explain your approach, and/or how your code works.
- Consult the Python PEP8 guidelines: https://peps.python.org/pep-0008/

Pay close attention to:

- Variable names
- Spacing around operators
- Separating logical sections
- Indentation



```python
# Define a variable to store the name of a user
user_name = "Alice"

# Print a greeting message using the user's name
print("Hello, " + user_name + "!")  # This prints: Hello, Alice!

# Define two numbers for basic arithmetic operations
num1 = 10
num2 = 5

# Calculate the sum of num1 and num2 and store the result in a variable
sum_result = num1 + num2
```

CoGrammar

Learner Challenge

For those who are looking for an additional challenge:

Enhance the functionality of the email simulator by implementing the following features:

- Delete an Email
  - Add a **delete_email()** function that allows the user to delete an email by specifying its index.
  - Ensure that the email is removed from the inbox list.
  - Provide user feedback: *print(f"Email '{subject_line}' has been deleted.")*
- Mark Emails as Spam
  - Introduce an **instance variable** is_spam (initialised to False) in the Email class.
  - Implement a **mark_as_spam()** instance method to set is_spam to True.
  - Add an option to mark an email as spam by specifying its index.
  - Provide user feedback: *print(f"Email '{subject_line}' marked as spam.")*

CoGrammar

Learner Challenge

For those who are looking for an additional challenge:

- View Spam Emails
  - Allow the user to view all emails marked as spam by looping through the inbox and printing the subject lines of emails where is_spam == True.
- Update the menu with the new options
  - Delete an email
  - Mark an email as spam
  - View all spam emails

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar