Welcome to this CoGrammar tutorial: Unit Testing and Modules

The session will start shortly...

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

CoGrammar

Software Engineering Session Housekeeping

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

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 throughout the 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]

CoGrammar

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

To be eligible for a certificate of completion, students must fulfil three specific criteria. These criteria ensure a high standard of achievement and alignment with the requirements for the successful completion of a Skills Bootcamp.

✓ Criterion 1 - Meeting Initial Requirements

Criterion 1 involves specific achievements within the first two weeks of the program. To meet this criterion, students need to:

- Attend a minimum of 7-8 hours per week of guided learning (lectures, workshops, or mentor calls) within the initial two-week period, for a total minimum of 15 guided learning hours (GLH), by no later than 15 September 2024.
- Successfully complete the Initial Assessment by the end of the first 14 days, by no later than 15 September 2024.

CoGrammar

Skills Bootcamp Progression Overview

✓ Criterion 2 - Demonstrating Mid-Course Progress

Criterion 2 involves demonstrating meaningful progress through the successful completion of tasks within the first half of the bootcamp.
To meet this criterion, students should:

- Complete 42 guided learning hours and the first half of the assigned tasks by the end of week 7, no later than 20 October 2024.

CoGrammar

Skills Bootcamp Progression Overview

✓ Criterion 3 - Demonstrating Post-Course Progress

Criterion 3 involves showcasing students' progress after completing the course.
To meet this criterion, students should:

- Complete all mandatory tasks before the bootcamp's end date. This includes any necessary resubmissions, no later than 22 December 2024.
- Achieve at least 84 guided learning hours by the end of the bootcamp, 22 December 2024.

CoGrammar

Learning Outcomes

- Describe the importance of unit testing in software development.
- Implement unit tests for Python code using the unittest framework.
- Explain the concept of modules in Python and their role in code organisation and reuse.
- Create and use modules in Python projects effectively.

CoGrammar

Unit Testing

Polls

CoGrammar

Poll

- *Refer to the polls section to vote for you option.*

1. **What is the primary purpose of unit testing in software development?**
   a. To find and fix bugs in the code.
   b. To verify that individual units of code work correctly.
   c. To validate the overall system functionality.
   d. To improve code performance.

Poll

- *Refer to the polls section to vote for you option.*

2. Which assertion method is used to check if two values are equal in a unittest?
    a. assertEqual()
    b. assertNotEqual()
    c. assertTrue()
    d. assertFalse()

CoGrammar

Unit Testing

CoGrammar

What is unit testing?

- A software testing method where individual units or components of a software application, are tested in isolation to ensure they work as intended.
- The goal is to verify that each unit of the software performs as designed and that all components are working together correctly.
- Help developers catch bugs early in the development process, when they are easier and less expensive to fix.
- Helps ensure that any changes made to the code do not cause unintended consequences or break existing functionality.

CoGrammar

Unit Testing - Advantages

- Catch errors early
- Improve code quality
- Refactor with confidence
- Document code behaviour
- Facilitate collaboration

Arrange, Act, Assert

CoGrammar

Arrange, Act, Assert

- The AAA pattern is a common pattern used in unit testing to structure test cases. It stands for Arrange, Act, Assert.
  - Arrange: Set up any necessary preconditions or test data for the unit being tested.
  - Act: Invoke the method or code being tested.
  - Assert: Verify that the expected behaviour occurred.

CoGrammar

Arrange, Act, Assert

Let’s have a look at an example of how to write a unit test in Python using the AAA pattern.

- Consider a simple function that adds two numbers:

```
def add_numbers(a, b):
    return a + b
```

CoGrammar

Arrange, Act, Assert

- To test this function, we would create a new function called test_add_numbers (note that the name must start with test_ for the Python test runner to find it).

```
def test_add_numbers(self):
    # Arrange
    a = 2
    b = 3

    # Act
    result = add_numbers(a, b)

    # Assert
    self.assertEqual(result, 5)
```

We've set up the test data (Arrange) by creating two variables a and b with the values 2 and 3.

We then invoke the function being tested (Act) and store the result in a variable called result.

Finally, we assert that the result is equal to the expected value of 5 (Assert).

CoGrammar

What is FIRST?

- Set of rules created by uncle bob also known for the SOLID principles and TDD.
- We follow these rules when creating tests to make sure our tests are clear, simple and accurate.

FIRST - Fast

- Tests should be fast and can run at any point during the development cycle.
- Even if there are thousands of unit tests it should run and show the desired outcome in seconds.

CoGrammar

FIRST - Independent

- Each unit test, its environment variables and setup should be independent of everything else.
- Our results should not be influenced by other factors.
- Should follow the 3 A’s of testing: Arrange, Act, Assert.

CoGrammar

FIRST - Repeatable

- Tests should be repeatable and deterministic, their values shouldn't change based on being run on different environments.
- Each test should work with its own data and should not depend on any external factors to run its test.

FIRST – Self Validation

- You shouldn't need to check manually, whether the test passed or not.

FIRST - Thorough

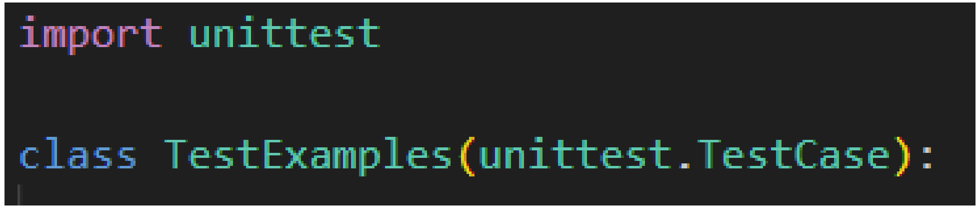
- Try covering all the edge cases.
- Test for illegal arguments and variables.
- Test for security and other issues
- Test for large values, what would a large input do.
- Should try to cover every use case scenario and not just aim for 100% code coverage.

Unit Tests

CoGrammar

Unit Tests

- Different packages for unit testing - Pytest, unittest, testify and Robot.
- We will use unittest. It is built into python and does not require additional installations.
- To use unittest we simply import the module and create a class for our testing.



```python
import unittest


class TestExamples(unittest.TestCase):
```

CoGrammar

Let’s get coding!

CoGrammar

Questions and Answers

CoGrammar

Let’s take a short break

CoGrammar

Conclusion and Recap

CoGrammar

Recap – Unit Testing

- Unit Testing
  - Process of testing the behaviours of our program to make sure it behaves as intended.
- Arrange, Act, Assert
  - Pattern used to structure our unit tests.
- FIRST Principles
  - A set of rules we follow to create quick simple and accurate unit tests.

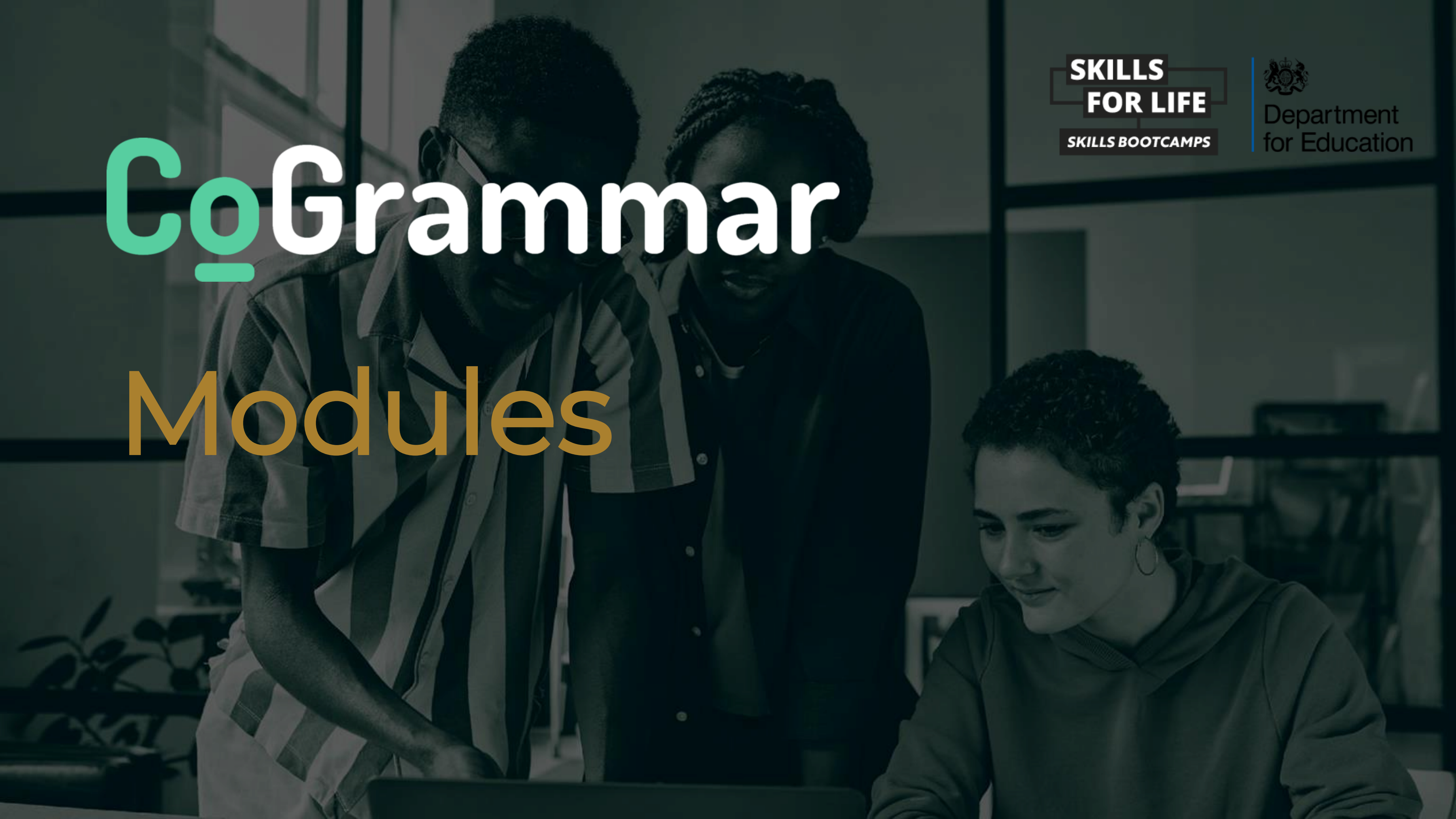
CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Modules



Polls

CoGrammar

Poll

- *Refer to the polls section to vote for you option.*

1. **What is a module in Python?**
    a. A built-in function provided by Python.
    b. A collection of related classes and functions.
    c. A file containing Python code that can be imported and used in other Python programs.
    d. An external library that extends Python's functionality.

CoGrammar

Poll

- *Refer to the polls section to vote for you option.*

2. What is the purpose of using modules in Python?
   a. To organise code into reusable components.
   b. To improve code performance.
   c. To execute code in parallel.
   d. To create graphical user interfaces.

CoGrammar

Poll

- *Refer to the polls section to vote for you option.*

3. Which statement correctly imports the math module in Python?
    a. import math.module
    b. from math import *
    c. include math.module
    d. import module as math

Modules

What are Modules?

- Modules in Python are files containing Python code that can define variables, functions, classes, or other Python constructs.
- The primary purpose of modules is to organise code into reusable and manageable units, facilitating better code organisation, maintenance, and reuse.

Key Aspects

CoGrammar

Key Aspects: Encapsulation

- Modules encapsulate related code into separate files, allowing developers to group similar functionality together.
- This promotes code organisation and helps maintain a clear structure within a project.

CoGrammar

Key Aspects: Reuse

- Modules enable code reuse by allowing developers to import and use functions, classes, and variables defined in one module within other modules or scripts.
- This promotes the "Don't Repeat Yourself" (DRY) principle, as common functionality can be defined once and reused multiple times across different parts of the codebase.

CoGrammar

Key Aspects: Namespacing

- Modules provide a namespace for the variables, functions, and classes they define, preventing naming conflicts between different parts of the codebase.
- By importing modules, developers can access the names defined within them using the dot notation (e.g., module_name.variable_name).

CoGrammar

Key Aspects: Abstraction

- Modules abstract away implementation details, allowing developers to interact with functionality at a higher level without needing to understand the underlying implementation.
- This promotes code readability and maintainability by hiding complex implementation details behind simple interfaces.

CoGrammar

Key Aspects: Separation of Concerns

- Modules promote the separation of concerns by allowing developers to partition code based on its functionality or purpose.
- This modular approach makes it easier to understand, maintain, and update code, as changes made to one module are less likely to affect other unrelated parts of the codebase.

CoGrammar

Best Practices

Best Practices: Module Structure

- Organise modules logically based on their functionality or purpose.
- Group related modules together in directories or packages to create a clear and intuitive hierarchy.

Best Practices: Module Names

- Choose descriptive and meaningful names for modules that reflect their purpose or functionality.
- Use lowercase letters and underscores to separate words in module names.
- Avoid using special characters or spaces in module names.

CoGrammar

Best Practices: Naming Conflicts

- Ensure that module names are unique and do not conflict with built-in Python modules or third-party libraries.
- Use names that are specific to your project or organisation to minimise the risk of naming conflicts.

CoGrammar

Best Practices: Package Names

- If you're working with packages (directories containing multiple modules), use short, lowercase names for package directories.
- Avoid using underscores or special characters in package names.

CoGrammar

Best Practices: Documentation

- Include clear and concise documentation within modules to explain their purpose, functionality, and usage.
- Use docstrings to provide inline documentation for classes, functions, and modules.

Let’s get coding!

CoGrammar

Questions and Answers

CoGrammar

Conclusion and Recap

CoGrammar

Recap – Modules

1. Modular design breaks down complex systems into smaller, manageable parts. It makes code easier to understand, maintain, and reuse.
2. Practice is crucial for mastering module usage in Python. By regularly exploring new modules, experimenting with different functionalities, and incorporating them into your projects, you can enhance your proficiency in leveraging modules to streamline your code and enhance its functionality.

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar