



# Welcome to this **CoGrammar** tutorial: Functions, Sequences and Debugging

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** 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](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident: [www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)





# 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**.
- 



# 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**.





# 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.



**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

# CoGrammar

## Functions, Sequences and Debugging

September 2024

# Learning Outcomes

- Define and call functions with parameters and return values.
- Implement functions to modularise and organise code effectively.
- Apply scope rules to avoid common errors related to variable access and modification.
- Interpret stack traces to debug and identify the source of errors in their code.



# Learning Outcomes

- Perform basic string operations such as concatenation, slicing, and formatting.
- Use built-in string methods to manipulate and analyse text data.
- Perform basic list operations such as indexing, slicing, appending, and removing elements.
- Use list methods and functions to manipulate and process list data.

# Learning Outcomes

- Perform basic dictionary operations such as adding, updating, and removing key-value pairs.
- Use dictionary methods to access and manipulate data efficiently.
- Perform basic operations on 2D lists such as accessing, modifying, and iterating over elements.

# Functions & Scope



# Functions Recap

- In Python, a function is like a recipe or a set of instructions that you can use over and over again in your code. It's a way to group together a block of code that performs a specific task.
- In Python, there are two main types of functions: **built-in functions** and **user-defined functions**.

# Functions Recap cont.

- **Built-in functions:** These are functions that are already provided by Python, so you can use them without having to define them yourself. For example, `print()`, `len()`, and `input()` are all built-in functions.
- **User-defined functions:** These are functions that you create yourself to perform a specific task in your code. You use the `def` keyword to define these functions. You give your function a name, specify any parameters it needs, and then write the code that the function should execute when it's called.

# Scope

- Scope is a set of rules that determine where and how you can access variables in your code.
- **Global Scope:** Variables that are defined outside of any function are said to be in the global scope. This means they can be accessed from anywhere in your code, including inside functions.
- **Local Scope:** Variables that are defined inside a function are said to be in the local scope. This means they can only be accessed within that specific function.

*Understanding scope is crucial because it helps you keep your code organised and prevents unexpected errors caused by variables being accessed in the wrong places.*



# Stack Traces & Debugging



# Stack Traces

- In programming, a stack trace is like that treasure map. It helps you trace your steps through your code to find out what went wrong when an error occurs.
- When an error happens in your Python code, Python creates a **stack trace** to help you debug it. The stack trace shows you the sequence of function calls that led to the error, starting from the main program and going deeper into any functions or methods that were called along the way.



# Stack Traces & Debugging

- Stack traces are incredibly helpful for debugging because they give you valuable information about what caused an error and where to look to fix it. This is usually where debugging comes in handy.
- **Debugging** is the process of finding and fixing errors, or "bugs," in computer programs. Just like detectives investigate and solve mysteries, programmers debug to track down and eliminate problems in their code.
- **Debugging** is an essential skill for programmers, as it allows them to create software that works correctly and reliably.

# Handling Strings



# What is String Handling?

- **Strings** are sequences of characters, such as letters, numbers, and symbols, which are often used to represent words, sentences, or other textual information.
- **String handling** refers to the manipulation and management of text data in a programming context.
- Efficient and effective string handling is essential for tasks such as text processing, data parsing, user input validation, and generating formatted output.

# String Handling Recap

- We create strings by enclosing characters within either single (' ') or double (" ") quotes.
- Combining strings together is called **concatenation**. You can use the + operator to concatenate strings
- However, Python offers multiple ways to format strings, such as using the **format()** method or **f-strings**.
- To access (index) individual characters in a string we make use of square brackets [].

# String Handling Recap

- Python provides many built-in methods to manipulate strings, such as:
  - **len()**: Returns the length of a string.
  - **upper()**, **lower()**: Convert strings to uppercase or lowercase.
  - **strip()**: Removes leading and trailing whitespace.
  - **split()**: Splits a string into a list of substrings based on a delimiter.
  - **join()**: Joins elements of a list into a single string using a specified delimiter.

# Lists & Dictionaries





# What are Lists?

- Lists are **fundamental data structures** in Python and are widely used in various programming tasks, from simple data storage to complex data manipulation and analysis.
- They are **incredibly versatile** and widely used in programming for storing and manipulating data.
- Mastering list manipulation techniques is essential for effective programming in Python.

# Manipulating Lists

- **Creating Lists:** Lists are created by enclosing a comma-separated sequence of items within square brackets [ ].
- **Accessing Elements:** You can access individual elements of a list using zero-based indexing (counting starts from 0).
- **Slicing:** Slicing allows you to extract a portion of a list by specifying a start and end index.
- **Modifying Lists:** Lists are mutable, meaning you can change their elements after they've been created.



# Manipulating Lists cont.

- **Adding Elements:** You can append new elements to the end of a list using the `append()` method or insert them at a specific position using the `insert()` method.
- **Removing Elements:** You can remove elements from a list using methods like `remove()`, `pop()`, or `del`.
- **List Methods:** Python provides many built-in methods for working with lists, such as `sort()`, `reverse()`, `count()`, and `index()`.

# Dictionaries

- **Dictionaries** in Python are like real-life dictionaries; they store pairs of keys and their associated values.
- Each value in a dictionary is accessed by its corresponding key, making dictionaries useful for mapping relationships between different pieces of data.
- **Dictionaries** are incredibly useful for representing structured data, such as information about users, configuration settings, or any other data that can be organised into key-value pairs.

# Manipulating Dictionaries

- **Accessing Elements:** Values in a dictionary are accessed by their keys.
- **Modifying Dictionaries:** Dictionaries are mutable, so you can change their values, add new key-value pairs, or remove existing ones.
- **Dictionary Methods:** Python provides various built-in methods for working with dictionaries, such as `keys()`, `values()`, `items()`, `get()`, `pop()`, and `update()`.

**Let's take a short  
break**

**CoGrammar**



**Demo Time!**



# Conclusion and Recap

In this lesson, we explored the importance and application of functions and sequences in Python programming.

- **Importance of Functions:**

**Purpose:** Encapsulate code into reusable blocks.

**Benefits:** Enhance code organisation, maintainability, and error management.

- **Importance of Sequences:**

**Types:** Lists, tuples, and strings.

**Uses:** Store and manipulate ordered collections of data.

# Conclusion and Recap

- Combining functions with sequences allows for flexible and dynamic data handling.
- Key operations include searching, sorting, and transforming data.



# Resources

- Official Python Documentation - Functions:
  - <https://docs.python.org/3/tutorial/controlflow.html#defining-functions>
- Online Tutorials:
  - <https://realpython.com/defining-your-own-python-function/>
  - [https://www.w3schools.com/python/python\\_functions.asp](https://www.w3schools.com/python/python_functions.asp)
- Additional Reading: "Automate the Boring Stuff with Python" by Al Sweigart
- Python Official Documentation - Data Structures:
  - <https://docs.python.org/3/tutorial/datastructures.html>
- Online Tutorials:
  - <https://realpython.com/python-sequences/>



# Questions and Answers



# Thank you for attending



Department  
for Education

CoGrammar

