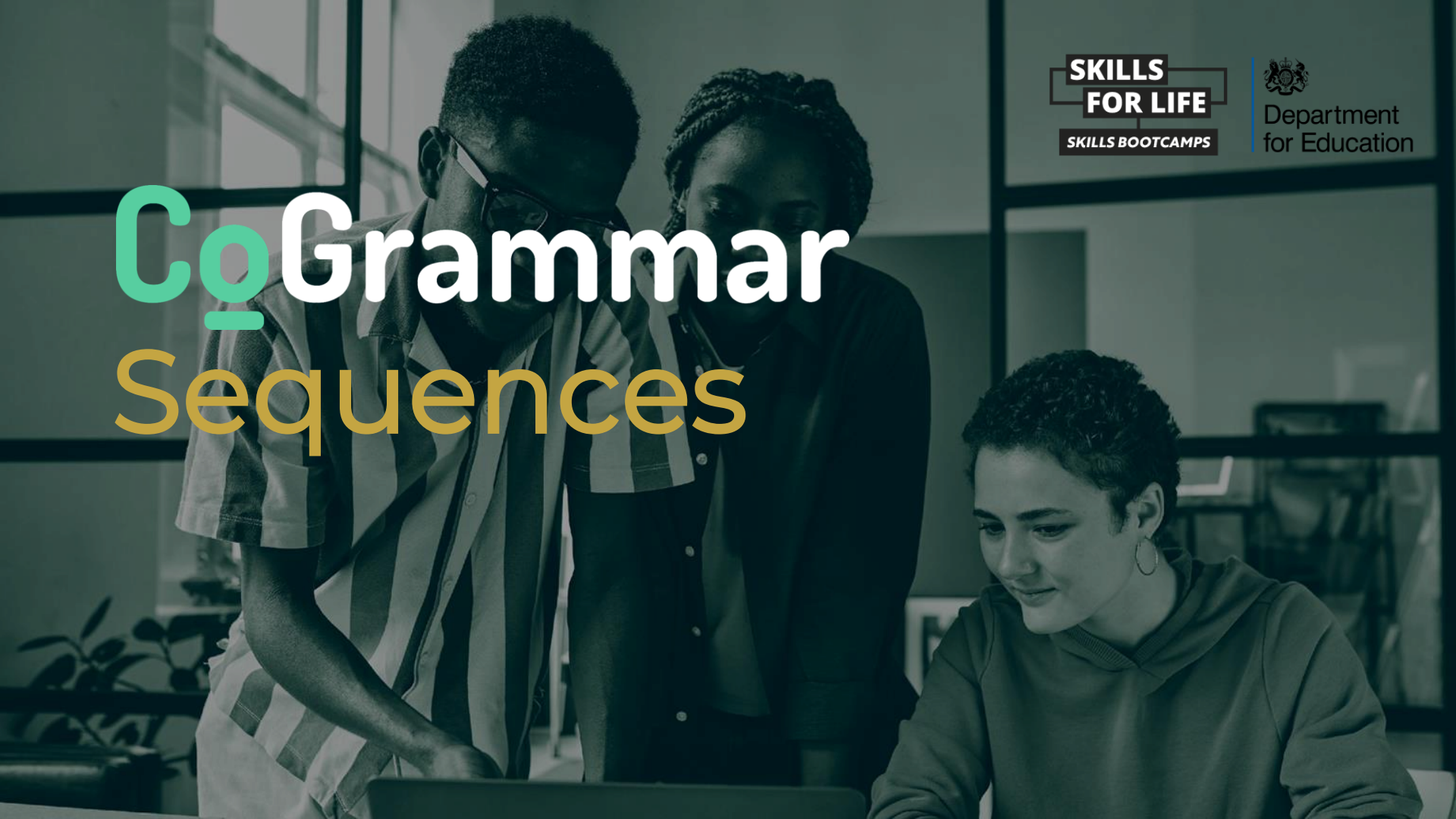
Welcome to this CoGrammar lecture:

Sequences

The session will start shortly...

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


SKILLS
FOR LIFE
SKILLS BOOTCAMPS
Department
for Education
CoGrammar
Sequences

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

Polls

CoGrammar

Poll

1. **True or False: Strings in Python are mutable (i.e., they can be changed in place).**
   A. True
   B. False

CoGrammar

Poll

2. Which of these is the correct way to create a list?

A. list = {1, 2, 3}

B. list = (1, 2, 3)

C. list = [1, 2, 3]

D. list = <1, 2, 3>

Poll

3. What is an important feature of a dictionary in Python?

A. Keys can have multiple values.

B. Keys are unique, and values are stored as key-value pairs.

C. Keys must always be strings.

D. Dictionaries store data in a sorted order.

String Handling

CoGrammar

Learner Outcomes

- Define and construct strings in Python.
- Master key string methods for effective text manipulation in Python.
- Effectively extract characters and substrings from strings using indexing and slicing.
- Utilise string concatenation and formatting techniques in Python.

String Creation & Initialisation

Strings in Python are sequences of characters, enclosed within either single quotes (' '), double quotes (" "), or triple quotes (''' ''')

```
message = "This is a string"
print(message)
```

Basic String Methods

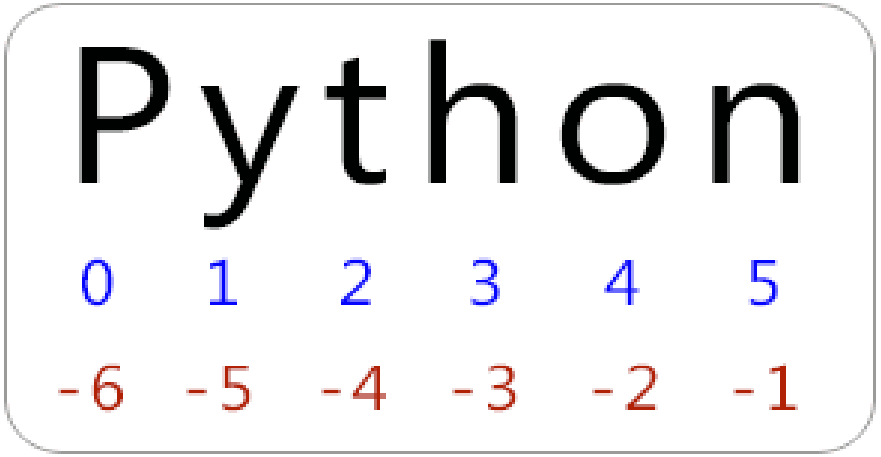
“coding for fun”	string.capitalize()	Coding for fun
“coding for fun”	string.isalpha()	True
“54369”	string.isnumeric()	True
“coding for fun”	string.isupper()	False
“coding for fun”	string.split()	[“coding”, “for”, “fun”]
“running for fun”	string.title()	Running For Fun
“£coding£”	string.strip()	coding
“coding for fun”	string.replace(‘f’, ‘s’)	coding sor sun

Strings cont.

Strings are Immutable

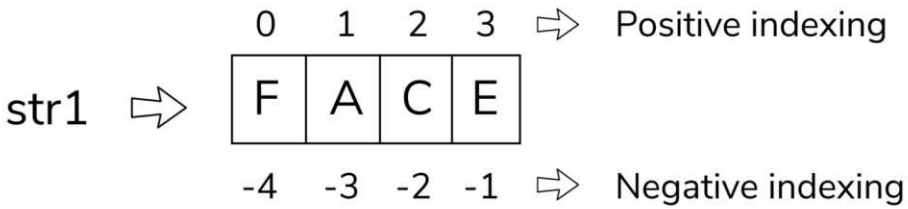
- When an object is immutable it means the object cannot be changed.
- When we apply methods to a string that appear to make changes, they are actually creating and returning new string objects.
- This means we have to store the changes we make in a variable to be reused.

Strings Indexing



String Indexing

String Slicing



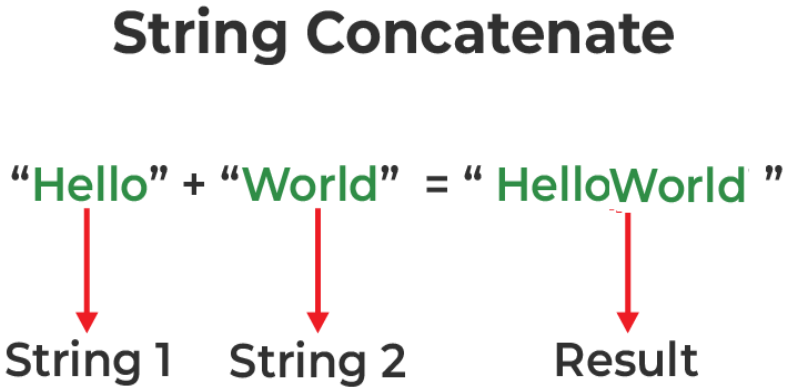
str1[1:3] = AC

str1[-3:-1] = AC

String Concatenation & Formatting

- String concatenation is the process of joining strings together, while formatting allows you to insert dynamic values into strings.
- String formatting in Python refers to the process of creating strings where dynamic values are inserted into predetermined locations within the string.

String Concatenation



String Formatting

.format() Method

```
name = "Inigo Montoya"
quantity = 51
formatted_string = "My name is {} and I would like {} muffins please.".format(name, age)
# Output: My name is Inigo Montoya and I would like 51 muffins please.
```

f-String

```
random_word = "Spanish Inquisition"
formatted_string = f"Nobody expects the {random_word}!"
# Output: Nobody expects the Spanish Inquisition!
```

Strings Recap

String Methods

- Built-in functions that operate on strings, providing various functionalities such as manipulating case, finding substrings, determining length and many more.

String Indexing and Slicing

- It's all about accessing characters within a string using their position and extracting a substring from a string.

String Formatting

- The process of inserting values and expressions into a string to create informative output.

Lists in Python

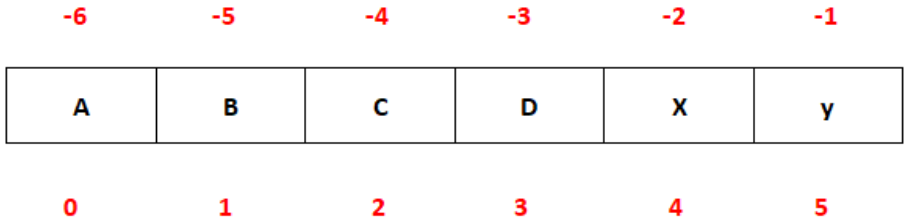
CoGrammar

Learner Outcomes

- Recall the fundamental characteristics of Lists.
- Explain the concept of indexing in a list.
- Apply knowledge of lists to manipulate elements.

Lists

- A list is a data type that allows us to store multiple values of any type together and a list can contain duplicates.
- We can access individual values using indexing and multiple values using slicing.
- We can iterate over lists using a for loop.



Lists cont.

- Lists are mutable. This means the values inside a list can be changed and unlike a string won't return a new list when changes have been made.
- We can apply methods to our lists without having to restore them inside our variables.
- To create a list we can surround comma separated values with square brackets. []
    - E.g.
        - `my_list = [value1, value2, value3]`

Lists cont.

Key List functions

• Adding Elements	*append(), insert()*
• Removing Elements	*remove(), pop() and 'del'*
• Manipulating elements	sorting, reversing and slicing

Lists Examples

Creating lists

```
# Creating a list of numbers
numbers = [1, 2, 3, 4, 5]

# Creating a list of strings
fruits = ["apple", "banana", "orange"]

# Creating a list of mixed data types
mixed_list = [1, "apple", True, 3.14]
```

Lists Examples

Adding & Removing Items

```
# Adding a single item
fruits.append("grape")

# Adding multiple items
fruits.extend(["pineapple", "mango"])

# Removing an item by value
fruits.remove("banana")

# Removing an item by index and returning it
removed_item = fruits.pop(2)
```

Lists Examples

Sorting Lists

```
# Sorting the list in-place
numbers.sort()

# Sorting the list in descending order
fruits.sort(reverse=True)

# Sorting a list without modifying the original list
sorted_numbers = sorted(numbers)
```

Dictionaries in Python

CoGrammar

Learner Outcomes

- Distinguish between the functionality of a Lists and Dictionaries.
- Expand on key operations relevant to Dictionaries.
- Apply the above knowledge to improve data management in programs

Dictionaries

- In Python, dictionaries function akin to the dictionaries we commonly used in English class, such as those from Oxford.
- Python dictionaries are similar to a list, however each item has two parts, a key and a value.
- To draw a parallel, consider an English dictionary where the key represents a word, and the associated value is its definition.

Dictionary Example

- Dictionaries are enclosed in curly brackets; key value pairs are separated by colon and each pair is separated by a comma.

```
# Dictionary Example

my_dictionary = {
    "name": "Terry",
    "age": 24,
    "is_funny": False
}
```

- On the left is the key, on the right is the value.

Dict() Functions

- The dict() function in Python is a versatile way to create dictionaries.
- Create dictionaries through assigning values to keys by passing in keys and values separated by an = sign.

```
# Creating a dictionary with direct key-value pairs
my_dict = dict(name="Kitty", age=25, city="Belarus")
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus'}
```

Dictionary Update()

- To append or add elements to a dictionary in Python, you can use the update() method or simply use the square bracket notation.

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict.update({'breed': 'Shorthair'})
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict['breed'] = 'Shorthair'
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

Dictionaries cont.

Key Dictionary functions

• Key-Value Pairs	*items(), keys(), values()*
• Fetching	*get()*
• Updating	*update()*
• Deleting	*pop(), popitem()*

Let’s take a short break

CoGrammar

Let’s get coding!

CoGrammar

Polls

CoGrammar

Poll

1. **Which of the following lines correctly creates a string, a list, and a dictionary?**

   A. my_string = "Hello", my_list = [1, 2, 3], my_dict = {"key": "value"}

   B. my_string = 123, my_list = "abc", my_dict = (1, 2)

   C. my_string = "Hello", my_list = {1, 2, 3}, my_dict = ["key", "value"]

   D. my_string = "Hello", my_list = [1, 2, 3], my_dict = {"key": 1}

Poll

2. **Which snippets correctly concatenate two strings, merge two lists, and add a key-value pair to a dictionary?**

   A. "Hello" + "World", list1 + list2, my_dict.update({"new_key": "new_value"})

   B. "Hello" & "World", list1.append(list2), my_dict["new_key"] = "new_value"

   C. "Hello".append("World"),list1.extend(list2), my_dict.add({"new_key": "new_value"})

   D. "Hello".concat(" World"), list1.insert(0, list2), my_dict.add("new_key": "new_value")

Lists and Dictionaries Recap

- Lists
  - Lists in Python offer a powerful mechanism for organising and manipulating data in a structured manner.
- Indexing
  - We can access elements in our list with indexing and can use slicing to grab multiple values.
- Dictionaries
  - Dictionaries in Python are mutable collections of key-value pairs, allowing efficient storage and retrieval of data.
  - They provide a mapping between unique keys and their associated values.

Learner Challenge

- Title: Employee Onboarding System with String, List, and Dictionary Operations
- Objective: Build a system to manage employee onboarding details, applying string manipulation, list operations, and dictionary usage to handle employee data efficiently.
  - **Lists:**
    - Store a list of employee hobbies.
    - Manipulate the hobbies list through basic operations: indexing, slicing, appending, and removing.
    - Manage sales data for multiple weeks in a 2D list and perform operations on the weekly sales.
  - **String Manipulation:**
    - **Concatenation:** Combine employee first name and last name into a full name.
    - **Slicing:** Extract initials from employee names for identification.
    - **Formatting:** Display employee data (name, age, department, hobbies) in a clear format.
  - **Dictionaries:**
    - Store employee details like name, age, department, address, and hobbies as key-value pairs.
    - Manage employee information, allowing updates to specific fields like department or address.

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar