Welcome to this CoGrammar Lecture: Sequences

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

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

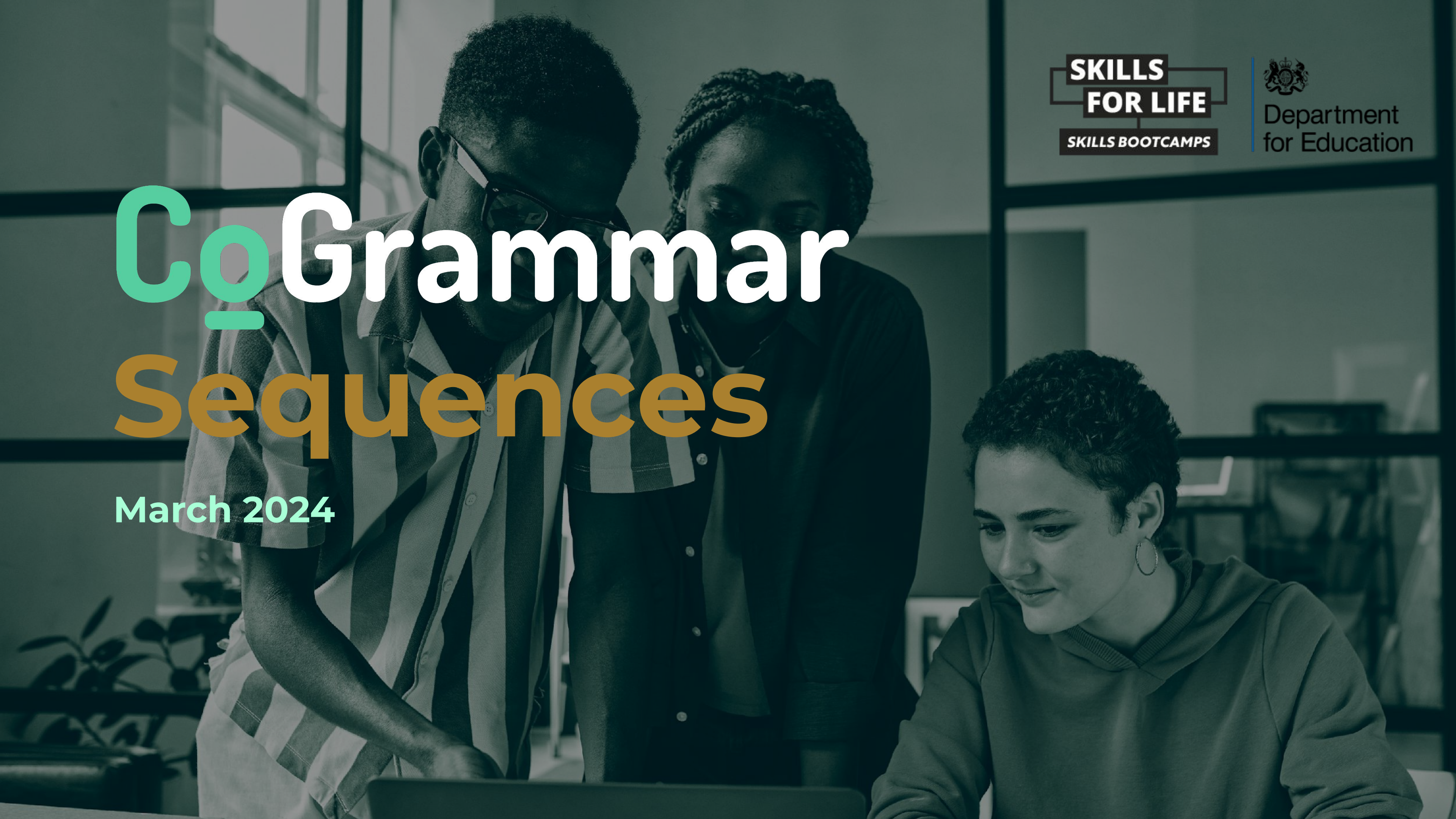
✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar



SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

Sequences

March 2024

String Handling

CoGrammar

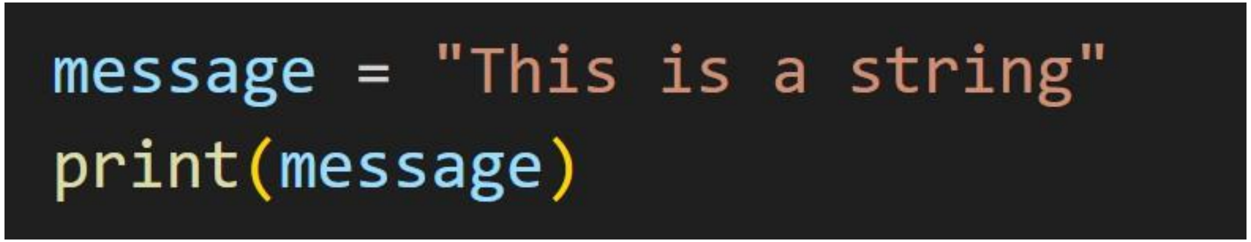
Agenda

- ❖ 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.

CoGrammar

String Creation & Initialization

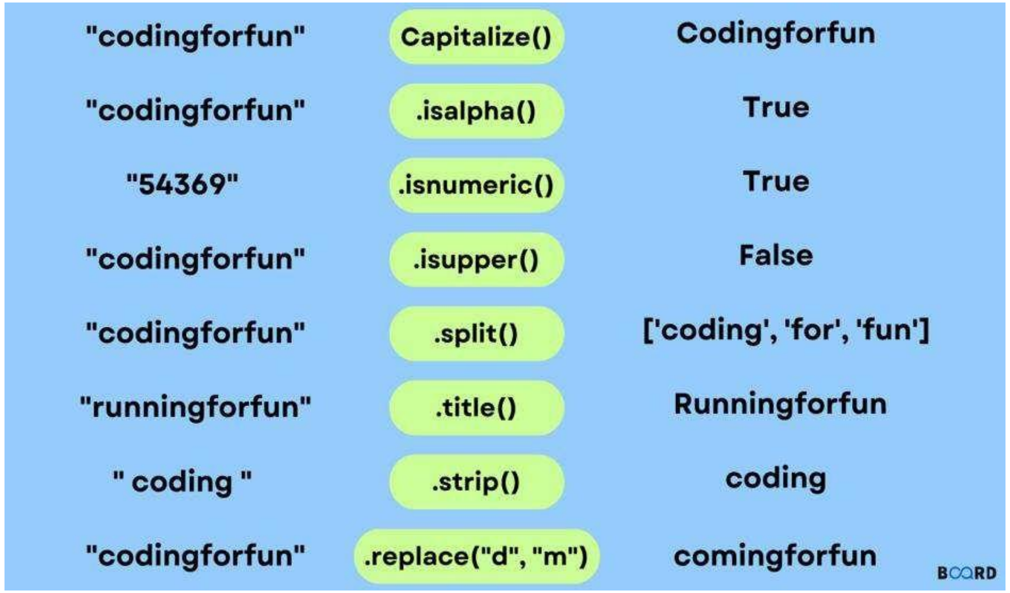
Strings in Python are sequences of characters, enclosed within either single quotes (' '), double quotes (" "), or triple quotes (''' ''')



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

CoGrammar

Basic String Methods



"codingforfun"	Capitalize()	Codingforfun
"codingforfun"	.isalpha()	True
"54369"	.isnumeric()	True
"codingforfun"	.isupper()	False
"codingforfun"	.split()	['coding', 'for', 'fun']
"runningforfun"	.title()	Runningforfun
" coding "	.strip()	coding
"codingforfun"	.replace("d", "m")	comingforfun

BOARD

CoGrammar

Strings cont.

Strings are Immutable

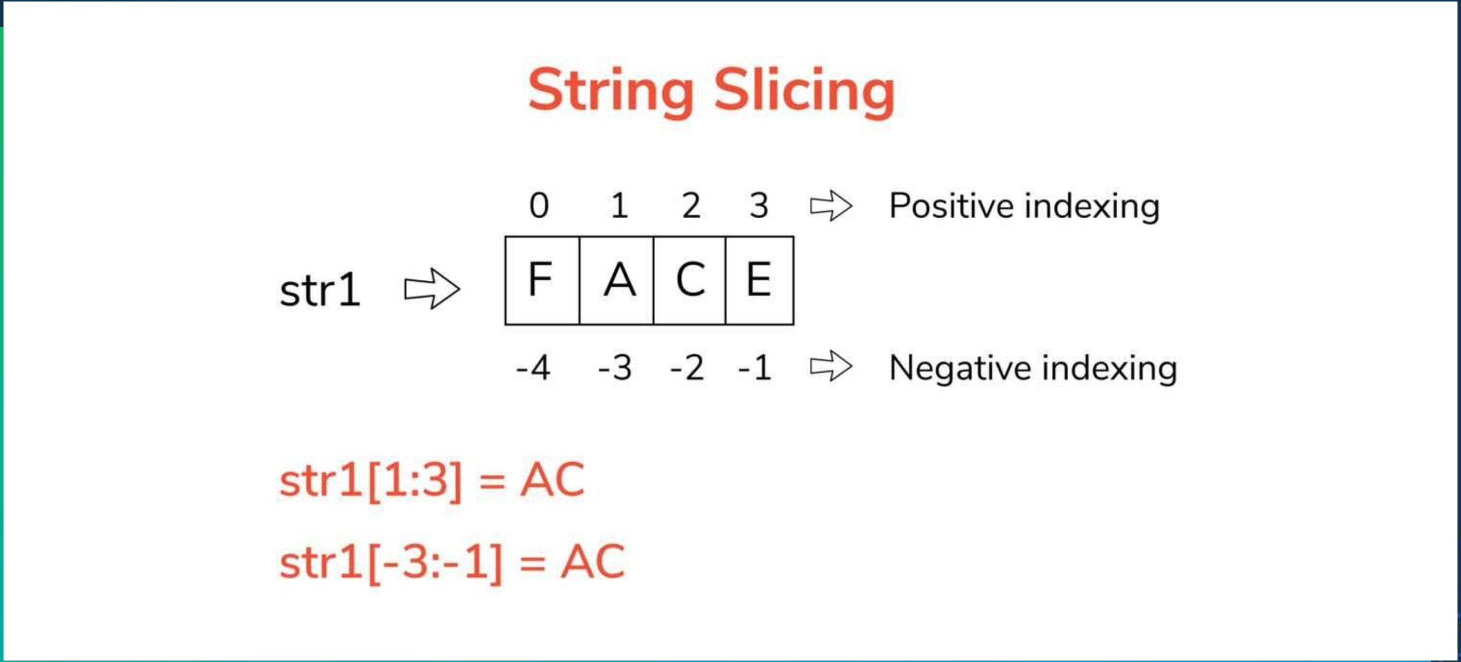
❖ When an object is immutable it means the object cannot be changed.

❖ 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.

CoGrammar

Strings Indexing

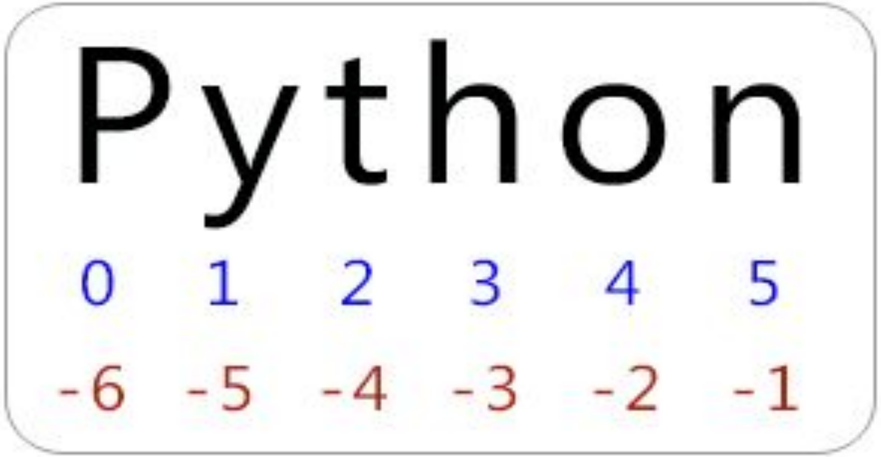


str1[1:3] = AC

str1[-3:-1] = AC

CoGrammar

Strings Indexing



CoGrammar

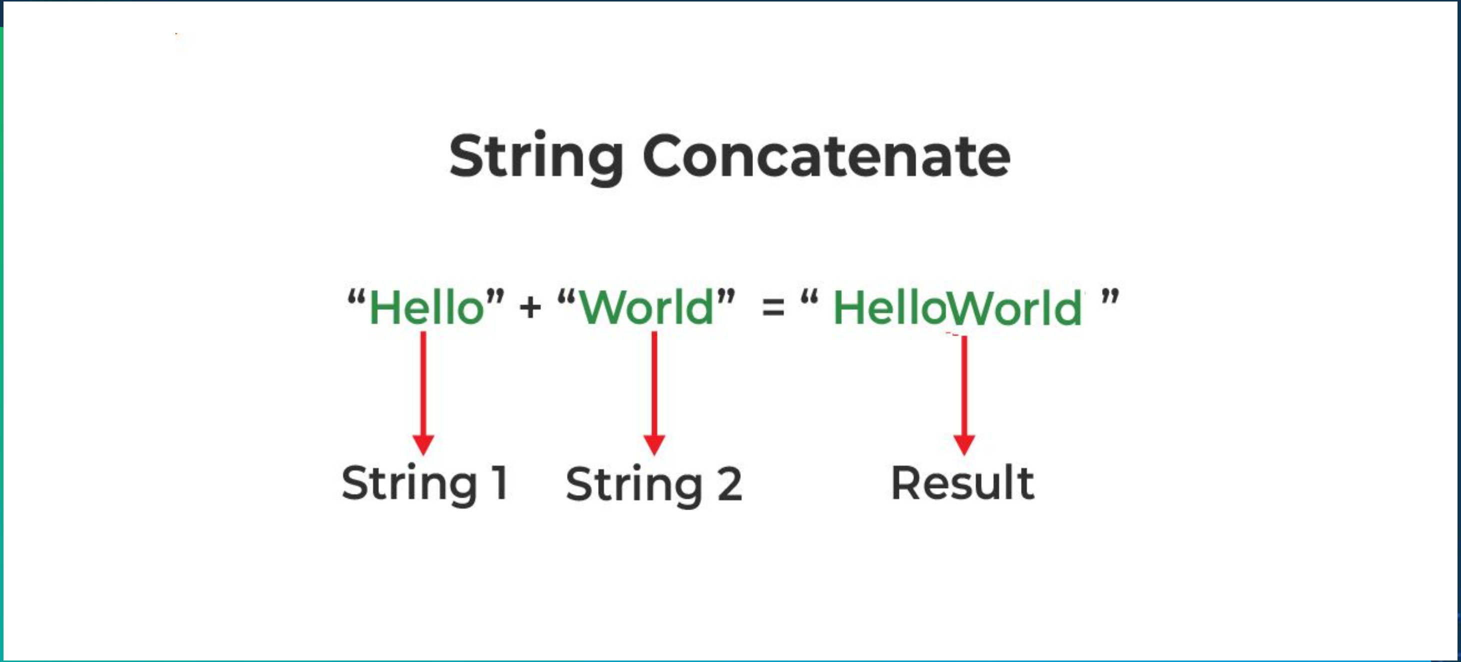
String Concatenation & Formatting

- 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.

CoGrammar

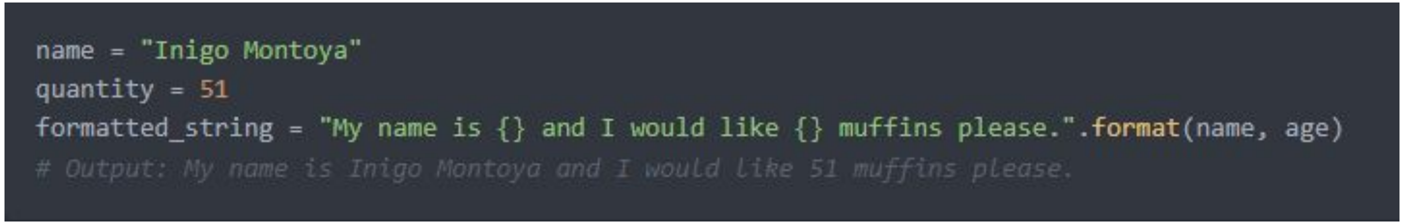
String Concatenation



CoGrammar

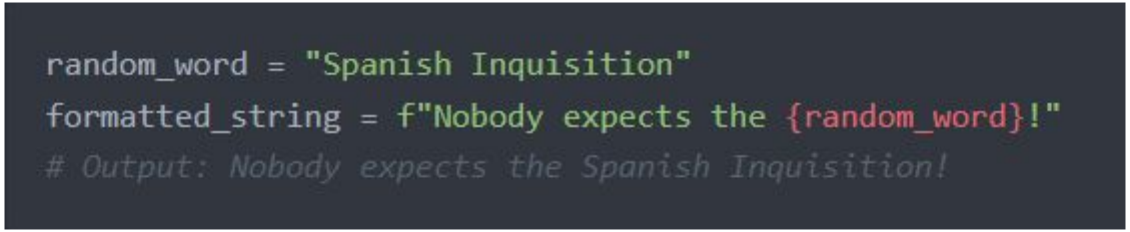
String Formatting

Format() Function



```python
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



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

CoGrammar

Summary

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.

CoGrammar

Lists in Python

CoGrammar

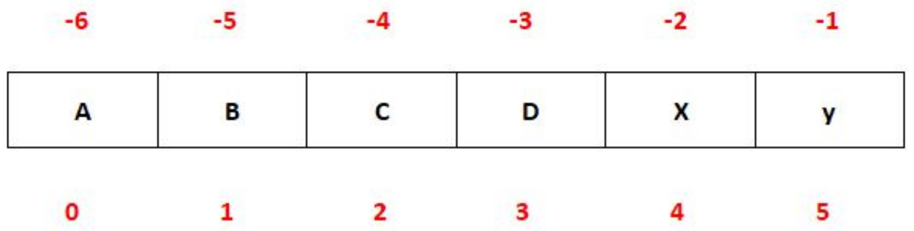
Agenda

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

CoGrammar

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]

CoGrammar

Lists cont.

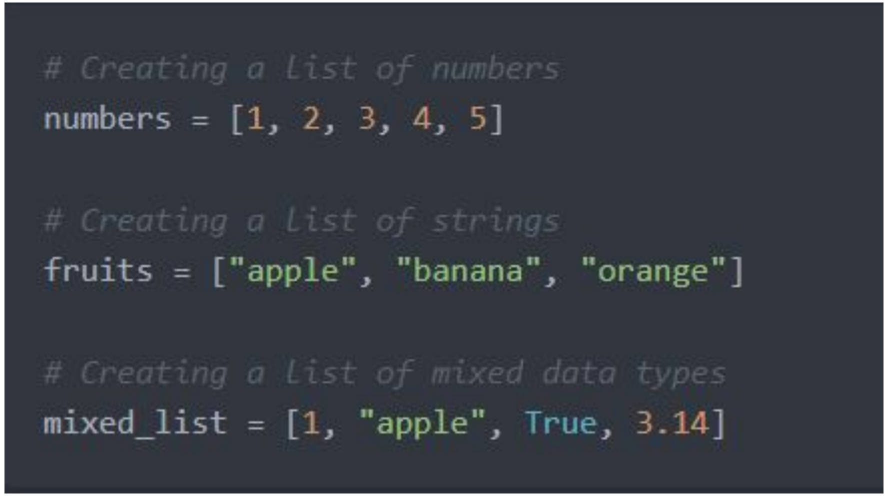
Key List functions

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

CoGrammar

Lists Examples

Creating lists

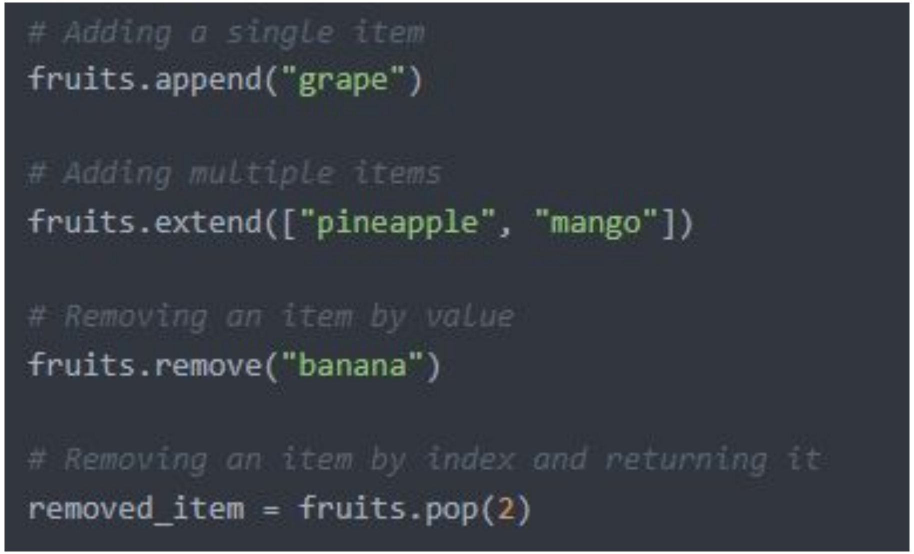


```python
# Creating a list of numbers
numbers = [1, 2, 3, 4, 5]

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

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

CoGrammar

Lists Examples

Adding & Removing Items

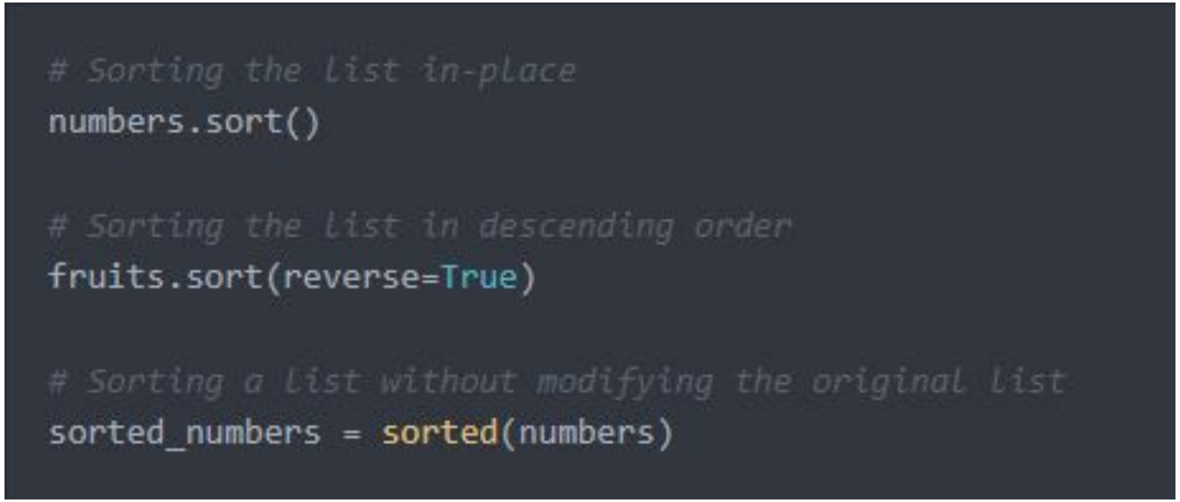


```
# Adding a single item
fruits.append("grape")

# 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)
```

CoGrammar

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)
```

CoGrammar

Dictionaries in Python

CoGrammar

Agenda

- ❖ 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

CoGrammar

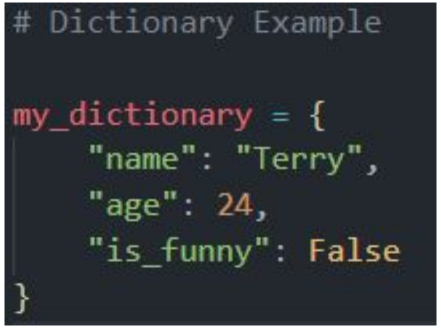
Dictionaries

- ❖ 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.

CoGrammar

Dictionary Example

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



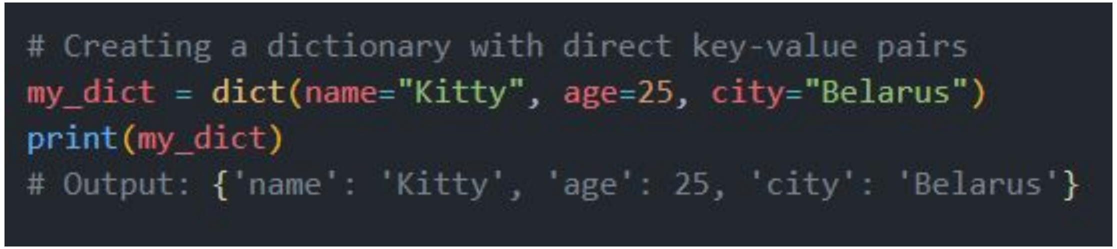
```
# Dictionary Example

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

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

CoGrammar

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.

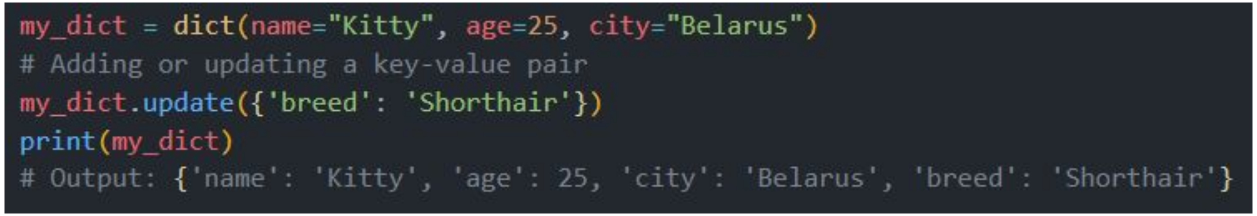


```
# Creating a dictionary with direct key-value pairs
my_dict = dict(name="Kitty", age=25, city="Belarus")
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus'}
```

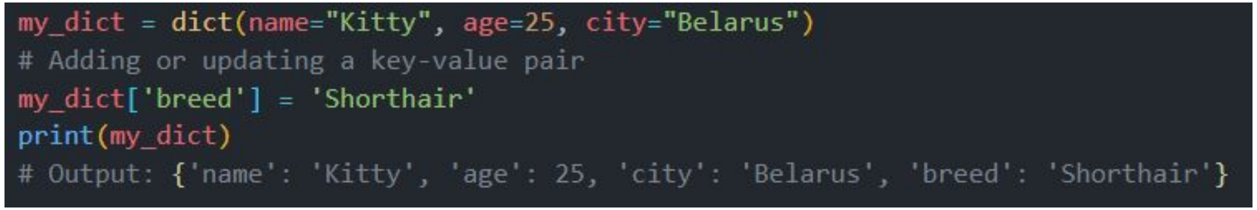
CoGrammar

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.



```python
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'}
```



```python
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()*

CoGrammar

Summary

Lists

❖ Lists in Python offer a powerful mechanism for organizing 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.

Let’s take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar