



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 Sequences

September 2024

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 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**.

Poll

1. Which of the following correctly define and call a function with parameters and a return value in Python?
 - a. `def greet(name): return "Hello " + name, greet("Alice")`
 - b. `def greet(): return "Hello " + name, greet("Alice")`
 - c. `def greet(name): print("Hello " + name), greet("Alice")`
 - d. `def greet(name): return "Hello " + name, result = greet("Alice")`

Poll

2. Which of the following demonstrate correct function calls and parameter usage in Python?
- a. `def power(base, exp): return base ** exp, power(2, 3)`
 - b. `def power(base, exp): return base ^ exp, result = power(2, 3)`
 - c. `def power(base, exp): print(base ** exp), result = power(2, 3)`
 - d. `def power(base, exp): return base ** exp, result = power(2, 3)`

String Handling



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.

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)
```

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

BOORD

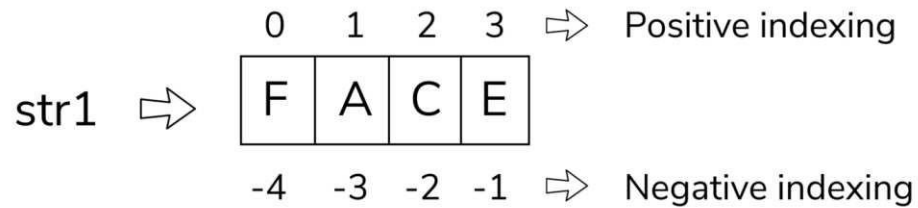
Strings cont.

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 Slicing



`str1[1:3] = AC`

`str1[-3:-1] = AC`

Strings Indexing

Python

0 1 2 3 4 5

-6 -5 -4 -3 -2 -1

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 Concatenate

"Hello" + "World" = "HelloWorld"

String 1 String 2 Result

String Formatting

Format() Function

```
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!
```

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.

Lists in Python



Agenda

- 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.

-6	-5	-4	-3	-2	-1
A	B	C	D	X	y
0	1	2	3	4	5

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	<i>append(), insert()</i>
● Removing Elements	<i>remove(), pop() and 'del'</i>
● 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



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

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	<i>items(), keys(), values()</i>
• Fetching	<i>get()</i>
• Updating	<i>update()</i>
• Deleting	<i>pop(), popitem()</i>

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.

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")`

Follow-up Activity

- **Title:** Inventory Management System
- **Objective:** Create a small inventory management system using functions, lists, and dictionaries.
- **Instructions:**
 - **Functions:** Write functions to add items to the inventory, remove items, and display the current inventory.
 - **Lists:** Store items in a list, and use a dictionary to keep track of each item's quantity.
 - **Scope:** Ensure that inventory updates are handled correctly using local and global variables.
 - **Stack Traces:** Intentionally introduce an error (e.g., referencing a non-existent item) and use the stack trace to debug the issue.

Practical

- **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.
 - 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.
 - 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.
 - Functions:
 - `add_hobby(hobbies_list, hobby)`: Add a new hobby to the employee's list of hobbies.
 - `remove_hobby(hobbies_list, hobby)`: Remove a specified hobby from the list.
 - `format_employee_info()`: Create a formatted string to display the employee's details.
 - `calculate_average_sales(sales_data)`: Calculate the weekly sales average based on the sales data.

Questions and Answers



Resources

- **Software:**

- <https://www.python.org/downloads/>
- <https://code.visualstudio.com/download>

- **Additional Resources**

- [HackInScience — Python Exercises](#)
- <https://www.codewars.com/>

- **Books:**

- <https://greenteapress.com/wp/think-python-2e/>
- <https://python.land/introduction-to-python/variable>

Thank you for attending



Department
for Education

CoGrammar

