CoGrammar

Week 5 – Open Class 2

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- 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 Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture 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**

Lecture Objectives

1. Fundamental characteristics of Lists.
2. Multi-Dimensional list indexing.
3. Open floor Q&A

CoGrammar

List Comprehension

★ List comprehension is a condensed method for creating lists in Python. In comparison to conventional for-loops, it offers a more condensed syntax for creating lists.

```
List Comprehension:
# Basic Structure
new_list = [expression for item in iterable]

# Squaring numbers from 0 to 9
squares = [x**2 for x in range(10)]
# Result: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

CoGrammar

```
List Comprehension:
# Basic Structure
new_list = [expression for item in iterable]

# Squaring numbers from 0 to 9
squares = [x**2 for x in range(10)]
# Result: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

★ **Expression:** The expression to be evaluated and included in the new list.
★ **Item:** The variable representing an element in the iterable (e.g., a range, list, string).
★ **Iterable:** The source of data to iterate over.

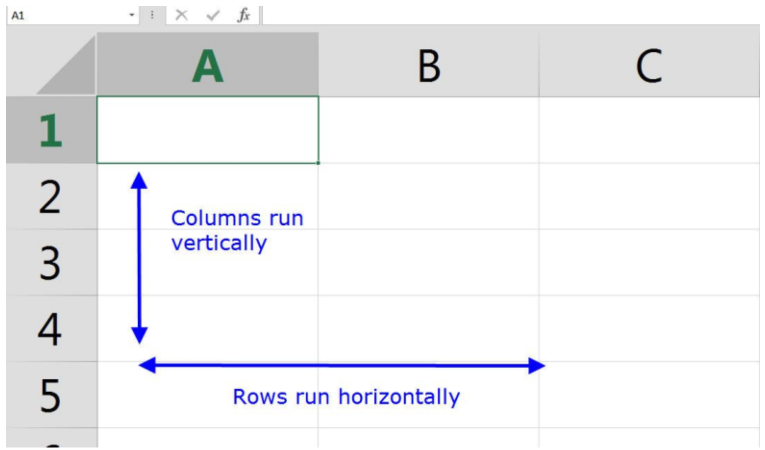
CoGrammar

Benefits & Precautions

- **Benefits:**
  - ***Conciseness:*** Achieve the same result with less code.
  - ***Readability:*** Express your intent more clearly and compactly.
  - ***Efficiency:*** List comprehensions are often faster than equivalent for-loops.
- **Considerations:**
  - ***Avoid Complexity:*** While list comprehensions are powerful, avoid making them overly complex for the sake of readability.
  - ***Conditional Expressions:*** You can use ternary expressions for conditional inclusion.

2D List

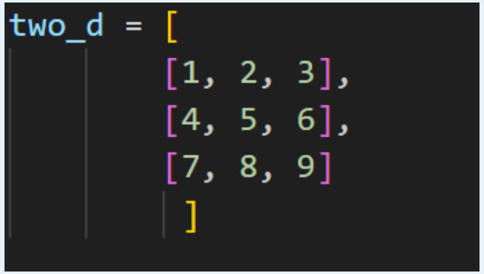
- ★ A List within a List.
- ★ Outer List (1 Dimension) + Inner List (1 Dimension) = 2D



CoGrammar

Rows and Columns

★ Elements are essentially accessed using rows and column indices.

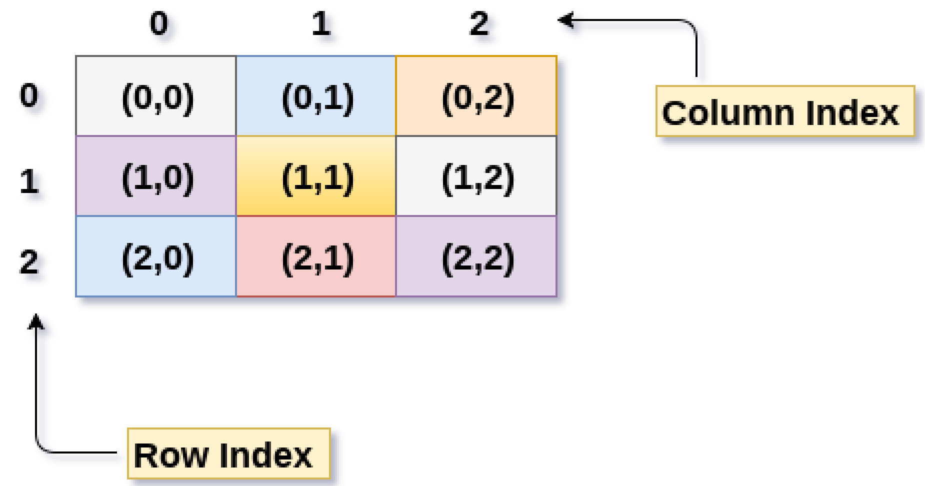


```
two_d = [
            [1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]
             ]
```

CoGrammar

Traversing

★ Nested Loops (iterate through rows and columns)
★ List comprehension



Wrapping Up

2D Lists

2D lists in Python offer a powerful mechanism for organising and manipulating data in a structured manner.

Rows and Columns

Rows represent individual lists within the main list, while columns denote elements within each of these lists.

Traversal

Whether it's accessing specific elements, performing operations on the entire list, or searching for particular values, traversing techniques are central to unleashing the full potential of 2D lists.

CoGrammar

Thank you for joining

CoGrammar