Welcome to this CoGrammar Task Walkthrough: Task 14

The session will start shortly...

Questions? Drop them in the chat.

Task Walkthrough: Searching and Sorting

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

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

Learning Outcomes

- Retrieve, sort, swap and search for items in a list.
- Give a detailed explanation of how the merge sort algorithm works.
- Transfer learnings to complete the Searching & Sorting tasks.

Searching and Sorting

- **Searching Algorithms**: Methods to find an element in a collection.
  - **Linear Search**: Sequentially checks each element until the desired element is found; best for small lists.
  - **Binary Search**: Efficiently searches in a sorted list by repeatedly dividing the search interval in half; requires sorted data.
- **Sorting Algorithms**: Methods to arrange elements in a specified order.
  - Bubble sort, insertion sort, merge sort, quick sort and more.

Searching and Sorting

- **Time complexity**: Important to understand the efficiency of algorithms; typically expressed in Big O notation
- **Real-world applications**: Used in databases, searching through web pages, and more.

Searching & Sorting Task Walkthrough: Auto-graded Task 1

CoGrammar

Auto-graded task 1

- Create a Python script called **album_management.py**.

- Design a class called `Album`. The class should contain:
  - A constructor which initialises the following instance variables:
    - `album_name` - Stores the name of an album.
    - `number_of_songs` - Stores the number of songs within the album.
    - `album_artist` -Stores the album's artist.
  - A `__str__` method that returns a string that represents an Album object in the following format:
    (*`album_name`*, *`album_artist`*, *`number_of_songs`*).

- Create a new list called `albums1`, add five `Album` objects to it, and print out the list.

- Sort the list according to the `number_of_songs` and print it out. (You may want to examine **the key parameter in the sort method**).

- Swap the element at position 1 (index `0`) of `albums1` with the element at position 2 (index `1`) and print it out.

- Create a new list called `albums2`.

- Copy all of the albums from `albums1` into `albums2`.

- Add the following two albums to `albums2`:
  - (Dark Side of the Moon, Pink Floyd, 9)
  - (Oops!... I Did It Again, Britney Spears, 16)

- Sort the albums in `albums2` alphabetically according to the album name and print out the sorted list.

- Search for the album *Dark Side of the Moon* in `albums2` and print out the index of the album in the albums2 list.

Searching & Sorting Task Walkthrough: Auto-graded Task 2

CoGrammar

Auto-graded task 2

In a newly created Python script called **merge_sort.py**:

- Modify the merge sort algorithm provided in the example usage section above to order a list of strings by string length from the longest to the shortest string.

- Run the modified Merge sort algorithm against 3 string lists of your choice. Please ensure that each of your chosen lists is not sorted and has a length of at least 10 string elements.

Searching & Sorting Task Walkthrough: Auto-graded Task 3

CoGrammar

Auto-graded task 3

Using the following list: [27, -3, 4, 5, 35, 2, 1, -40, 7, 18, 9, -1, 16, 100]

- Create a Python script called **sort_and_search.py**. Consider which searching algorithm would be appropriate to use on the given list?
- Implement this search algorithm to search for the number 9. Add a comment to explain why you think this algorithm was a good choice.
- Research and implement the **Insertion sort** on this list.
- Implement a searching algorithm you haven't tried yet in this Task on the sorted list to find the number 9. Add a comment to explain where you would use this algorithm in the real world.

Merge Sort Algorithm

CoGrammar

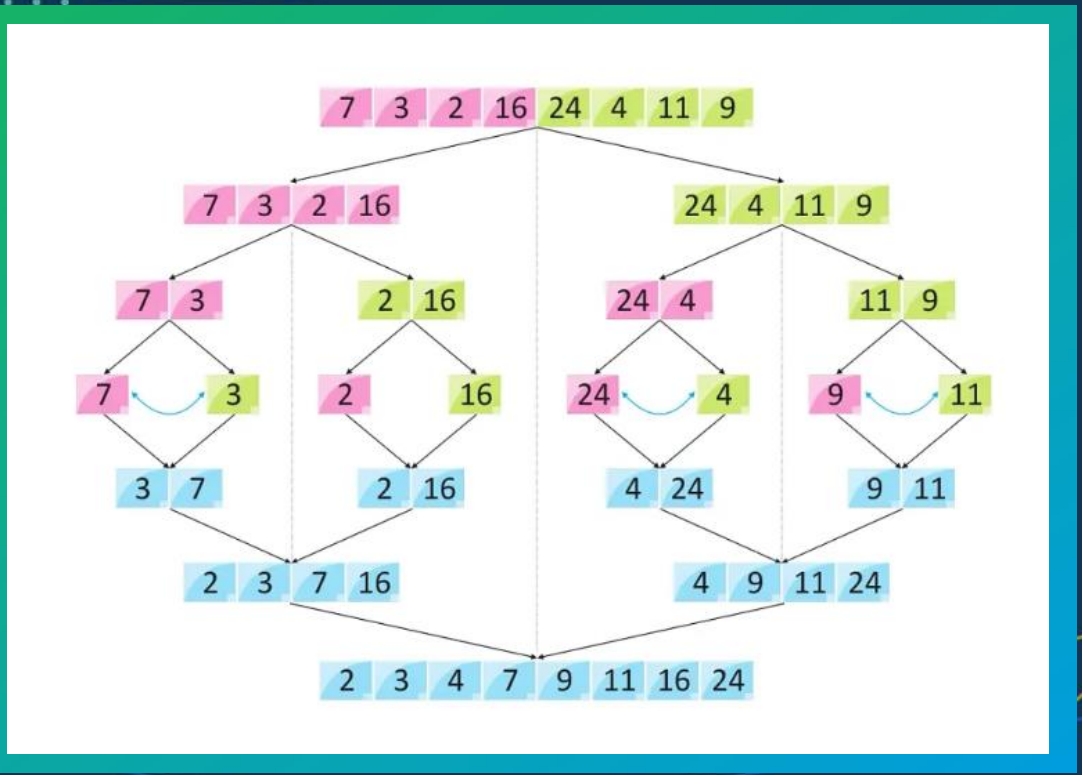

Image source:
https://medium.com/@ozgurmehmetakif/merge-sort-algorithm-ff52822f5608

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar