Welcome to this CoGrammar Tutorial: Task Walkthrough

The session will start shortly...

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

CoGrammar

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

Task Walkthrough: Recursion | Searching & Sorting

October 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**

CoGrammar

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

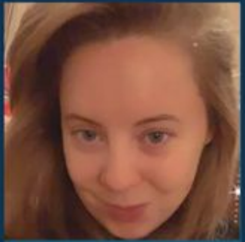
We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

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

Rafiq Manan


Charlotte Witcher

Nurhaan Snyman

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

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.

CoGrammar

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.

CoGrammar

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.

CoGrammar

Advised Resources

- ❖ HyperionDev PDF notes
- ❖ Lectures: 7, 9 & 10 October 2024
- ❖ Example code files
- ❖ Task walkthrough lecture
- ❖ Research

Learning Outcomes

- ❖ Break down problems into recursive subproblems and implement base and recursive cases effectively.
- ❖ 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 Recursion and Searching & Sorting tasks.

CoGrammar

Recursion

- **Definition**: A technique where a function calls itself in order to solve a problem.
- **Base Case**: The condition under which the recursion stops; it prevents infinite recursion.
- **Recursive Case**: The part of the function that includes the recursive call; it breaks the problem down into smaller subproblems.
- **Stack Overflow**: Occurs when there are too many recursive calls without hitting a base case, exhausting the call stack.
- **Advantages**: Simplifies code for problems that have a natural recursive structure, leading to cleaner and more understandable solutions.
- **Disadvantages**: Can lead to increased memory usage due to function call overhead and risk of stack overflow if not designed carefully.

Searching & 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.
- **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.

CoGrammar

Recursion Task Walkthrough: Auto-graded Task 1

CoGrammar

Auto-graded task 1

Follow these steps:

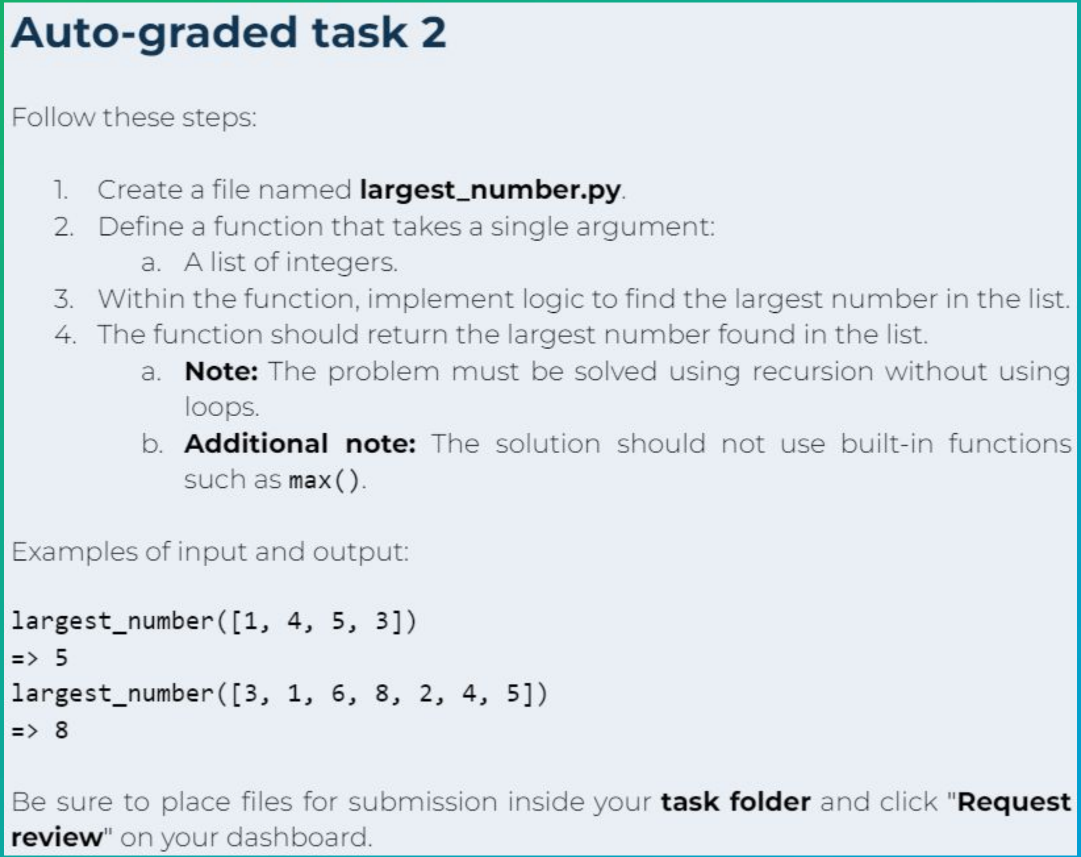
1. Create a file named **sum_recursion.py**.
2. Define a function which takes two arguments:
   a. A list of integers.
   b. A single integer that represents an index point.
3. The single integer will represent the index point up to which the function should sum all the numbers in the list.
   a. **Note:** List indices start at 0. The number at the specified index should be included in the calculation.
4. The function is required to sum all the numbers in the list up to and including the given index point.
5. The function should calculate the sum using recursion as opposed to using loops.

Examples of input and output:

```
adding_up_to([1, 4, 5, 3, 12, 16], 4)
=> 25
```
=> adding the numbers all the way up to index 4 (1 + 4 + 5 + 3 + 12)

Recursion Task Walkthrough: Auto-graded Task 2

CoGrammar



Auto-graded task 2

Follow these steps:

1. Create a file named **largest_number.py**.
2. Define a function that takes a single argument:
   a. A list of integers.
3. Within the function, implement logic to find the largest number in the list.
4. The function should return the largest number found in the list.
   a. **Note:** The problem must be solved using recursion without using loops.
   b. **Additional note:** The solution should not use built-in functions such as `max()`.

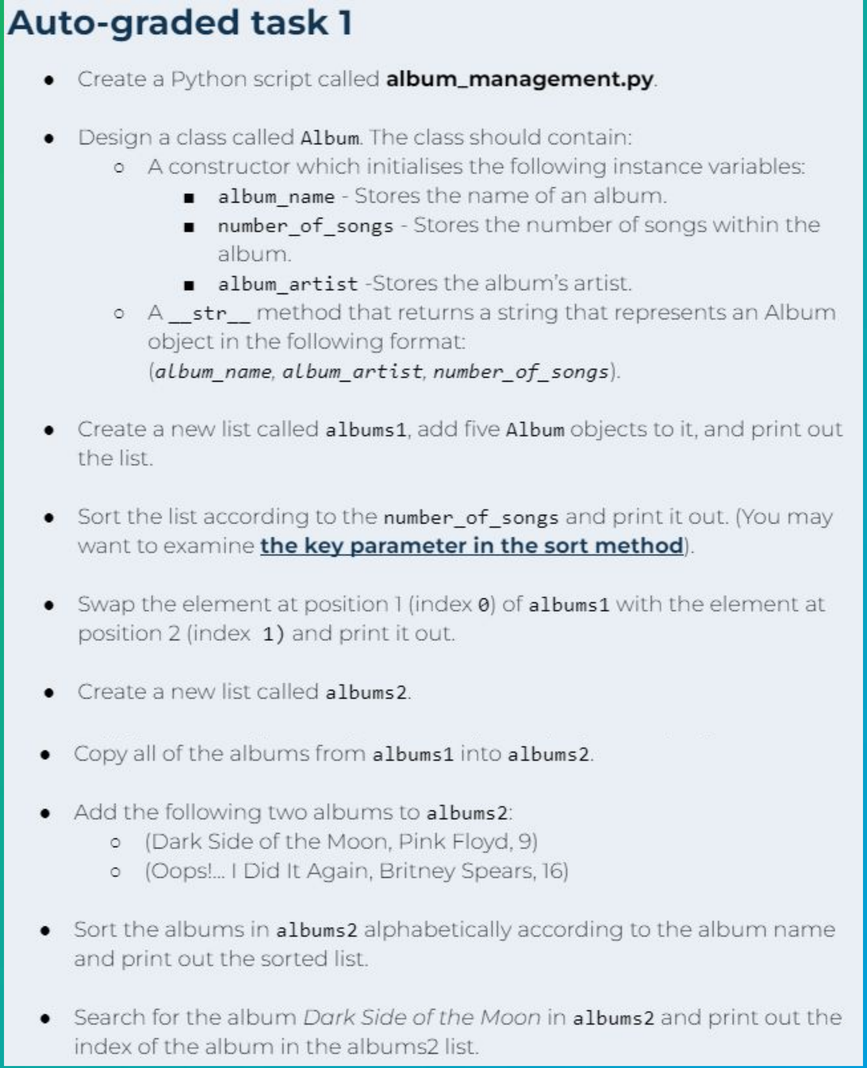
Examples of input and output:

```
largest_number([1, 4, 5, 3])
=> 5
largest_number([3, 1, 6, 8, 2, 4, 5])
=> 8
```

Be sure to place files for submission inside your **task folder** and click "**Request review**" on your dashboard.

Searching & Sorting Task Walkthrough: Auto-graded Task 1

CoGrammar

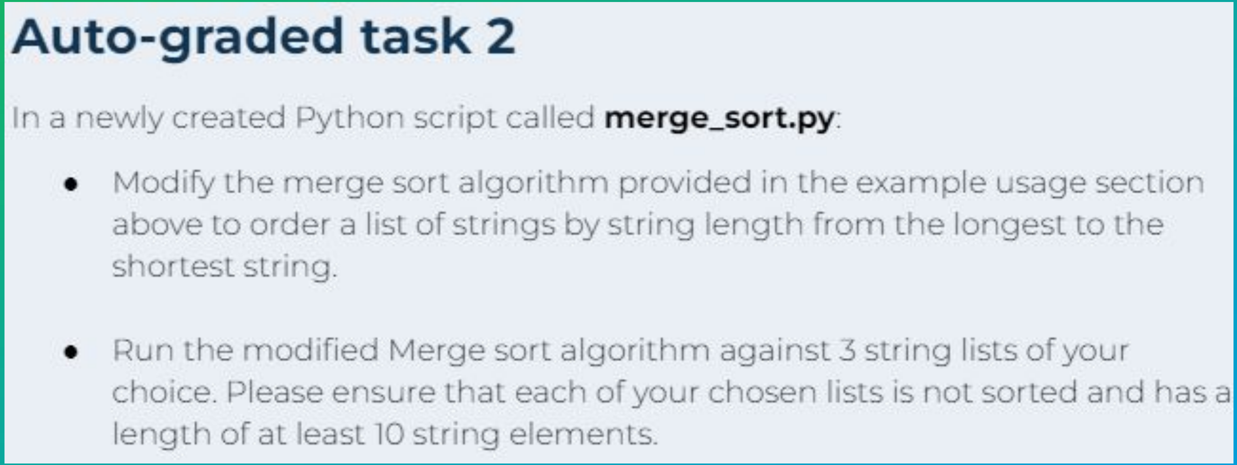


Auto-graded task 1

- 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**:

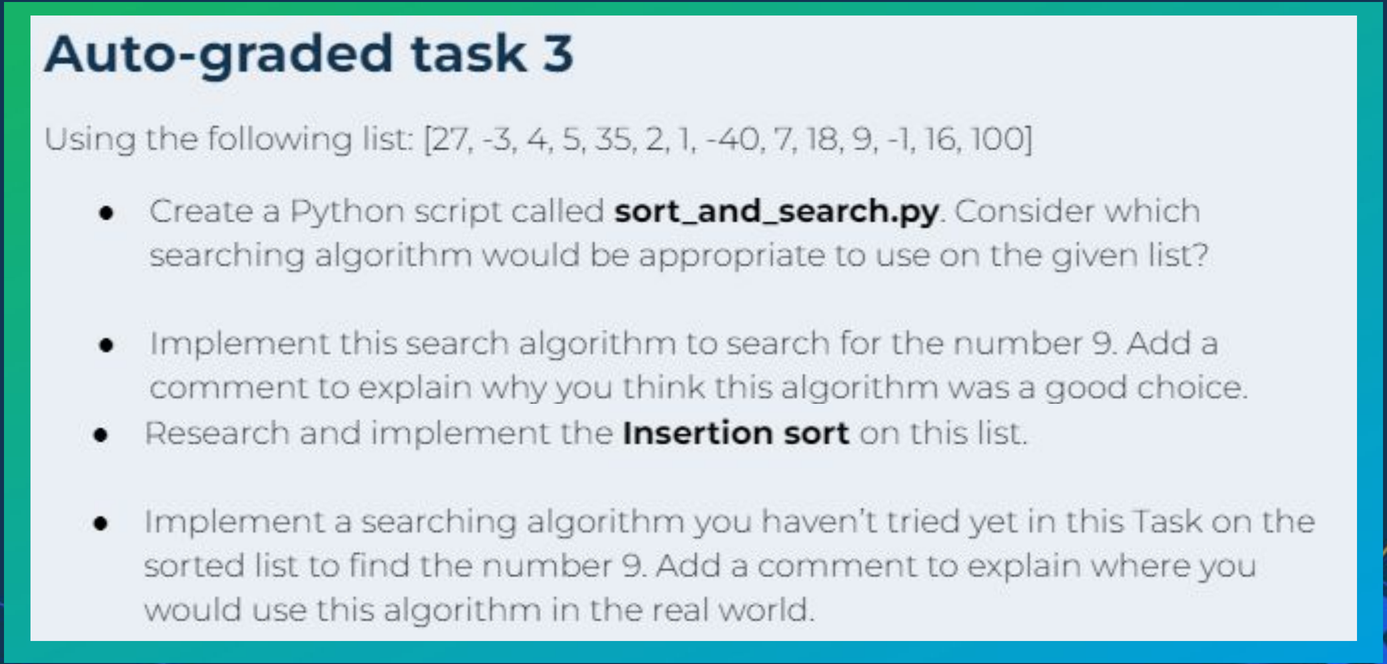
- Modify the merge sort algorithm provided in the example usage section above to order a list of strings by string length from the longest to the shortest string.

- 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.

CoGrammar

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

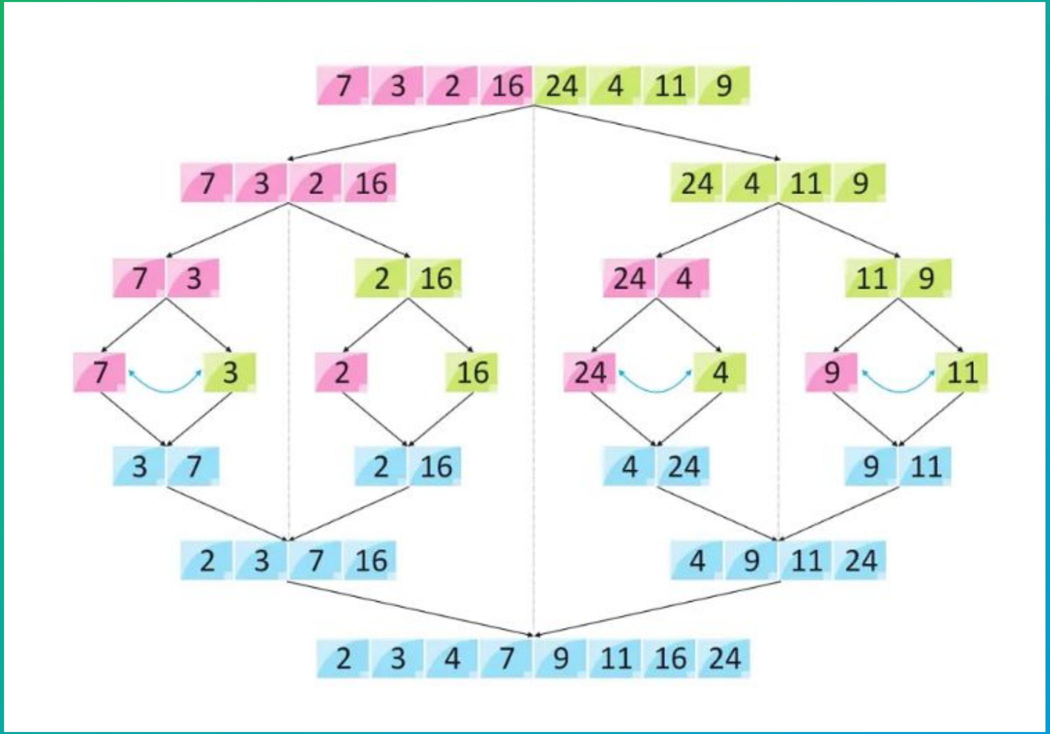


Image source:
https://medium.com/@ozgurmehmetakif/merge-sort-algorithm-ff52822f5608

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar