Hyperiondev

TASK

# Handling Strings, Lists, and Dictionaries

**Model-Answer Approach**

Visit our website


Copyright © 2023 HyperionDev. All rights reserved.

# Practical task 1

Practical task 1 consists of two parts: The first involves modifying the original string, which we get from the user using the `input()` function to alternate letters between upper and lowercase, while the second alternates words. Here's the approach taken:

**Part 1:** To alternate letters between upper and lowercase within a string, begin by creating an empty string. Then, iterate over the indexes of the original string using a `for` loop. With an `if-else` statement, check if the index is even (if `i % 2 == 0`) or odd (`else`). If even, convert the letter to uppercase; if odd, convert to lowercase. Finally, print the modified string after the loop.

**Part 2:** To alternate words between upper and lowercase in a string, split the original string into a list of words using the `.split()` method. Create a new empty list to store modified words. Use a `for` loop with enumerate to access both index and word. Enumerate is preferred over just using range (`len(original_string)`) because it allows you to access both index and word directly. Again, use an `if-else` statement to check if the index is even or odd. If even, convert the word to lowercase and append it to the list; if odd, convert it to uppercase and append it. Finally, join the modified words list into a string using the `.join()` method and store it in a variable. Print it as requested.

This approach is well-suited for this task because it is easy to implement, and applies familiar string methods and concepts covered in the lecture. The use of the `input()` function to allow the program to use for any given sentence makes the program dynamic. The innovation here is the usage of enumerate in part 2, which comes in handy when you need to access the index and element of an iterable.

# Practical task 2

The task involved creating a list of menu items and two dictionaries: one for stock levels and another for prices, aimed at calculating the total value of the café's stock.

The approach includes iterating over the menu list and retrieving corresponding values from the stock and price dictionaries. Each menu item's value is calculated by multiplying its stock quantity by the unit price, and these values are summed to determine the total stock value. This approach provides practical experience in iterating through lists and efficiently accessing dictionary values for calculations.

This straightforward yet effective approach offers valuable practice in iterating through lists and accessing dictionary values to perform calculations based on


Copyright © 2023 HyperionDev. All rights reserved.

stock and price values. However, it assumes that all menu items have matching entries in both the stock and price dictionaries.

To handle potential mismatches between the stock and price dictionaries, a second example demonstrates an `if-else` statement within the loop. This statement checks to ensure that the menu item is listed within both the stock and price dictionaries. By ensuring both price and stock quantities are available for each menu item, this approach guarantees accurate valuation of the café's inventory, minimising errors due to missing or inconsistent data.

Rate us

## Share your thoughts

HyperionDev strives to provide internationally excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

**Click here** to share your thoughts anonymously.


Copyright © 2023 HyperionDev. All rights reserved.