part-6

Zip and Zip_Longest Functions in Python

Description

The zip() and zip_longest() functions in Python are used to iterate over multiple iterables simultaneously. These functi ons can be particularly useful when working with lists, tuples, or other data structures of different lengths.

Key Points:

- **zip()**: Combines multiple iterables into a single iterator of tuples, where each tuple contains the corresponding elem ents from each iterable.
- **zip_longest()**: Similar to zip(), but it fills in the missing values with a specified fill value if the iterables have different l engths.

Using the zip() Function

- The zip() function takes one or more iterables as arguments and returns an iterator of tuples.
- If the input iterables have different lengths, the resulting iterator will have the length of the shortest iterable.
- Example: Zipping two lists of names and printing the results.
- Example: Zipping three lists (first names, last names, and numbers) and printing the results.

Using the zip_longest() Function

- The zip_longest() function is part of the `itertools` module and behaves similarly to the zip() function, but it handles iterables of different lengths.
- If one of the iterables is exhausted before the others, the remaining values are filled with the specified `fill_value.`
- Example: Zipping two lists of names using zip_longest() and printing the results.
- Example: Zipping two lists of names using zip_longest() with a custom `fill_value.`

Table: Comparison of zip() and zip_longest()

Function	Behavior
`zip()`	Combines iterables into a single iterator of tuples, stopping at the end of the shortest iterable.
`zip_longest()`	Combines iterables into a single iterator of tuples, filling in missing values with a specified `fill_value` if the iterables have different lengths.