Learning assignment-4

zip() vs enumerate() in Python

Both zip() and enumerate() are built-in Python functions that deal with iterating over collections, but they serve different purposes.

1. zip()

Used to combine multiple iterables (e.g., lists, tuples) element-wise.

Returns an iterator of tuples.

Stops at the shortest iterable if they are of different lengths.

Code-

names = ['Alice', 'Bob', 'Charlie']

ages = [25, 30, 35]

zipped = zip(names, ages) # Pairs corresponding elements together

print(list(zipped)) # [('Alice', 25), ('Bob', 30), ('Charlie', 35)]

2. enumerate()

Used to iterate over an iterable while keeping track of the index.

Returns an iterator of index-element pairs.

Code-

names = ['Alice', 'Bob', 'Charlie']

for index, name in enumerate(names, start=1): # start=1 makes it 1-based index

print(index, name)

When to Use Which?

Use zip() when you need to pair elements from multiple iterables.

Use enumerate() when you need both the index and value of a single iterable.

List Comprehension in Python

List comprehension is a concise way to create lists using a single line of code.

Basic Syntax:

[expression for item in iterable if condition]

Examples:

1.Creating a list of squares

squares = [x\*\*2 for x in range(5)]

print(squares) # [0, 1, 4, 9, 16]

2.Filtering even numbers

evens = [x for x in range(10) if x % 2 == 0]

print(evens) # [0, 2, 4, 6, 8]

3.Using zip() in list comprehension

names = ['Alice', 'Bob', 'Charlie']

ages = [25, 30, 35]

pairs = [(name, age) for name, age in zip(names, ages)]

print(pairs) # [('Alice', 25), ('Bob', 30), ('Charlie', 35)]

4.Using enumerate() in list comprehension

names = ['Alice', 'Bob', 'Charlie']

indexed\_names = [(i, name) for i, name in enumerate(names, start=1)]

print(indexed\_names) # [(1, 'Alice'), (2, 'Bob'), (3, 'Charlie')]