List Comprehension

What Is List Comprehension

List comprehension is a concise way to create lists by applying an expression to elements of an iterable, optionally filtering with a condition.

Why Should We Use List Comprehension?

- **Shorter Code: Write less to do more.**
- **Cleaner Syntax: Easy to read and understand.**
- **Faster Execution: Performs better than loops in many cases.**
- **Combines Actions: Transform and filter in one step.**

Basic Syntax Explained

[expression for i in iterable if condition]

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

Using Nested List Comprehension

Definition:
Nested list comprehension is used to handle multiple loops or create 2D lists like grids or tables.

Syntax Explanation: [[inner_expression for inner_loop] for outer_loop]

- **Outer loop: Creates the main structure (e.g., rows in a table).**
- **Inner loop: Fills the structure (e.g., columns in a row).**

Example: Multiplication Table:
table=[[x* y for x in range(1, 11)] for y in range(1, 11)]

List Comprehension With Functions

Definition:
A function can be applied to each element during list comprehension for cleaner and reusable code.

Example 1: Squaring Numbers:

```
def square(x):
    return x**2
result=[square(x) for x in range(5)]
print(result) #Output:[0, 1, 4, 9, 16]
```

Basic Use Case

- A feature that makes a new list by transforming/filtering of an iterable.
- Syntax

```
[expression for item in iterable if
condition#optional)
```

- Basic usage

```
numbers= [0,1,2,3]
squares= [x**2 for x in numbers]
print(squares)

Output [0,1,4,9]
```

Output [0,1,4,9]

When NOT to Use List Comprehension

When to avoid it?

1- High memory usage

2- Complexity

- Beginners.
- Nested lists.

```
result = [x*y for x in range(5) for y in range(3) if x + y > 4]
```

3- Conditions and readability.

```
result = [x2 if x% 2== 0 else(x**3 if x % 3 == 0 else x**4) for x in range(10)]
```

Thanks!

Rahaf Alotaibi
Manar AlShaykh
Ahmed Alhassar