🐍 List Comprehension in Python - Interview Preparation

🎯 Objective:

Prepare for Python interviews with an interactive and practical approach to **list comprehensions**, including key concepts, hands-on examples, and frequently asked interview questions.

✅ 1. What is List Comprehension in Python?

- **List comprehension** is a concise way to create lists in Python.
- It reduces the need for loops and makes the code more readable.

📚 2. Basic Syntax of List Comprehension

```
[expression for item in iterable if condition]
```

📖 Explanation:

- **expression:** Operation or transformation performed on each element.
- **item:** The current element from the iterable.
- **iterable:** A collection like list, tuple, or range.
- **condition (optional):** A filter to include only specific elements.

📂 3. Basic Example of List Comprehension

📝 **Traditional Way:**

```
numbers = [1, 2, 3, 4, 5]
squared = []
for num in numbers:
    squared.append(num ** 2)
print(squared)
```

👉 *Output:*

```
[1, 4, 9, 16, 25]
```

🎯 **Using List Comprehension:**

t

```
numbers = [1, 2, 3, 4, 5]
squared = [num ** 2 for num in numbers]
print(squared)
```

👉 *Output:*

```
[1, 4, 9, 16, 25]
```

🚀 4. List Comprehension with Conditions

➢ 1. Filtering Even Numbers

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_numbers = [num for num in numbers if num % 2 == 0]
print(even_numbers)
```

👉 *Output:*

```
[2, 4, 6, 8]
```

➢ 2. Filtering Odd Numbers

```
odd_numbers = [num for num in numbers if num % 2 != 0]
print(odd_numbers)
```

👉 *Output:*

```
[1, 3, 5, 7, 9]
```

➢ 3. Creating a List of Strings with Length Greater than 3

```
words = ["cat", "elephant", "dog", "mouse"]
long_words = [word for word in words if len(word) > 3]
print(long_words)
```

👉 *Output:*

```
['elephant', 'mouse']
```

🧠 5. Nested List Comprehension

➢ 1. Flattening a 2D List

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened = [num for row in matrix for num in row]
print(flattened)
```

👉 *Output:*

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

➢ 2. Multiplying Elements of Two Lists

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
product = [a * b for a in list1 for b in list2]
print(product)
```

👉 *Output:*

```
[4, 5, 6, 8, 10, 12, 12, 15, 18]
```

📌 6. Advanced List Comprehension

➤ 1. Using `if-else` in List Comprehension

```
numbers = [1, 2, 3, 4, 5]
result = ["Even" if num % 2 == 0 else "Odd" for num in numbers]
print(result)
```

👉 *Output:*

```
['Odd', 'Even', 'Odd', 'Even', 'Odd']
```

➤ 2. Using `enumerate()` with List Comprehension

```
words = ["apple", "banana", "cherry"]
indexed_words = [(index, word) for index, word in enumerate(words)]
print(indexed_words)
```

👉 *Output:*

```
[(0, 'apple'), (1, 'banana'), (2, 'cherry')]
```

➤ 3. Using Multiple Conditions

```
numbers = [10, 15, 20, 25, 30, 35, 40]
filtered = [num for num in numbers if num > 20 and num % 5 == 0]
print(filtered)
```

👉 *Output:*

```
[25, 30, 35, 40]
```

🧠 7. Dictionary and Set Comprehension

➢ 1. Dictionary Comprehension

```
squares = {num: num ** 2 for num in range(1, 6)}
print(squares)
```

👉 *Output:*

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

➢ 2. Set Comprehension

```
unique_squares = {num ** 2 for num in [1, 2, 3, 2, 3, 4]}
print(unique_squares)
```

👉 *Output:*

```
{16, 1, 4, 9}
```

🔥 8. Common Interview Questions on List Comprehension

🔥 1. What is the difference between list comprehension and a traditional loop?

- **List Comprehension:** Concise and faster.

- **Traditional Loop:** More explicit and easier for beginners.

🔥 2. Can we use multiple `if` conditions in list comprehension?

Yes, multiple conditions can be combined using logical operators.

📝 **Example:**

```
numbers = [10, 20, 30, 40, 50, 60]
filtered = [num for num in numbers if num > 20 and num < 50]
print(filtered)
```

🔥 3. How do you perform conditional operations inside list comprehension?

You can use a ternary operator inside list comprehension.

📝 **Example:**

```
numbers = [1, 2, 3, 4, 5]
result = ["Even" if num % 2 == 0 else "Odd" for num in numbers]
print(result)
```

🔥 4. How do you convert a list of strings to uppercase using list comprehension?

```
words = ["hello", "world", "python"]
uppercase_words = [word.upper() for word in words]
print(uppercase_words)
```

🔥 5. Can list comprehensions be used with tuples?

No, list comprehension returns a list. Use **tuple comprehension** with `tuple()` to convert.

```
numbers = (1, 2, 3, 4)
```

```
squared_tuple = tuple(num ** 2 for num in numbers)
print(squared_tuple)
```

🕹️ 9. Practice Challenges

🚀 1. Create a List of Squares for Even Numbers Only

```
numbers = [1, 2, 3, 4, 5, 6]
squared_even = [num ** 2 for num in numbers if num % 2 == 0]
print(squared_even)
```

🚀 2. Generate a List of Multiples of 3 from 1 to 20

```
multiples_of_3 = [num for num in range(1, 21) if num % 3 == 0]
print(multiples_of_3)
```

🚀 3. Create a List of Vowels from a Given String

```
string = "Python Programming"
vowels = [char for char in string if char.lower() in "aeiou"]
print(vowels)
```

🚀 4. Create a List of Prime Numbers from 1 to 50

```
primes = [num for num in range(2, 51) if all(num % i != 0 for i in
range(2, int(num ** 0.5) + 1))]
print(primes)
```

🎁 10. Advanced Concepts in List Comprehension

➤ 1. Nested List Comprehension with Conditions

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
even_numbers = [num for row in matrix for num in row if num % 2 == 0]
print(even_numbers)
```

➤ 2. Flattening a List of Lists with Conditions

```
nested_list = [[1, 2], [3, 4, 5], [6, 7]]
flattened = [num for sublist in nested_list for num in sublist if num
% 2 != 0]
print(flattened)
```

➤ 3. Using List Comprehension with `zip()`

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
summed_list = [a + b for a, b in zip(list1, list2)]
print(summed_list)
```

🎯 11. Quick Tips for Interview Success

- ✅ Practice both **basic and advanced list comprehensions.**
- ✅ Understand how to use `if`, `else`, and multiple conditions.
- ✅ Learn to apply list comprehensions with `zip()`, `enumerate()`, and nested loops.
- ✅ Be comfortable with set and dictionary comprehensions.
- ✅ Know when to choose list comprehension over traditional loops.
