List Comprehension

Table of contents

01
Introduction

02
Comparison

03
Examples

04
Summary

01

Introductioan

Introductioan

- List comprehension is a **concise** and **efficient** way to create lists in Python.

- It provides an elegant alternative to traditional **for loops** for generating lists from iterables (like lists, tuples, strings, sets, and range()).

Why Use List Comprehension?

- **Conciseness** → Reduces code length compared to loops.
- **Efficiency** → Faster execution as Python optimizes list comprehensions.
- **Readability** → More expressive and easier to understand.
- **Flexibility** → Supports filtering, transformations, and multiple iterations.
- **Reduced Errors** → Less room for mistakes compared to loops.

Syntax Basic

Python

```python
new_list = [expression for item in iterable]
```

Explanation:

- **expression** → The operation or transformation applied to each element.
- **item** → The variable representing each element in the iterable.
- **iterable** → The source of elements (list, tuple, string, range(), etc.).

Syntax with if

Python

```python
new_list = [expression for item in iterable if condition]
```

Explanation:

- **expression** → The operation or transformation applied to each element.
- **item** → The variable representing each element in the iterable.
- **iterable** → The source of elements (list, tuple, string, range(), etc.).
- **if condition (optional)** → Filters elements based on a condition.

Syntax with if-else

Python

```
new_list = [true_exp if condition else false_exp for item in iterable]
```

Explanation:

- **true_exp** → The operation or transformation applied if condition is true.
- **false_exp** → The operation or transformation applied if condition is false.
- **item** → The variable representing each element in the iterable.
- **iterable** → The source of elements (list, tuple, string, range(), etc.).
- **if condition (optional)** → Filters elements based on a condition.

Syntax with Nested list

Python

```
new_list = [element for sublist in nested_list for element in
sublist]
```

Explanation:

- **element** → The variable representing each element in the iterable.
- **sublist** → Part of a nested list
- **nested_list** → List within another list.
- **if condition (optional)** → Filters elements based on a condition.

02

Comparison

For Loop Vs List Comprehension

Python

Using For Loop	**Using List Comprehension**

```python
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
```

Using For Loop

```python
newlist = []
for x in fruits:
    if "a" in x:
        newlist.append(x)
print(newlist)
```

Using List Comprehension

```python
newlist = [x for x in fruits if "a" in x]
print(newlist)
```

Output

```
['apple', 'banana', 'mango']
```

03

Examples

Example 1:

Objective:

Create a list of numbers from 0 to 9

Python

```python
numbers = [x for x in range(10)]
print(numbers)
```

Output

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Example 2:

Objective:

Filtering elements (Only include numbers less than 5)

Python

```python
small_numbers = [x for x in range(10) if x < 5]
print(small_numbers)
```

Output

```
[0, 1, 2, 3, 4]
```

Example 3:

Objective:

Convert lowercase letters to uppercase

Python

```python
text = "hello python"
uppercase_letters = [char.upper() for char in text if char.isalpha()]
print(uppercase_letters)
```

Output

```
['H', 'E', 'L', 'L', 'O', 'P', 'Y', 'T', 'H', 'O', 'N']
```

Example 4:

Objective:

Extract even numbers from 1 to 20

Python

```
even_numbers = [num for num in range(1, 21) if num % 2 == 0]
print(even_numbers)
```

Output

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

Example 5:

Objective:

Exclude a specific item ("apple")

Python

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
filtered_fruits = [x for x in fruits if x != "apple"]
print(filtered_fruits)
```

Output

```
['banana', 'cherry', 'kiwi', 'mango']
```

Example 6:

Objective:

Replace "banana" with "orange"

Python

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
new_fruits = [x if x != "banana" else "orange" for x in fruits]
print(new_fruits)
```

Output

```
['apple', 'orange', 'cherry', 'kiwi', 'mango']
```

Example 7:

Objective:

Generate all possible pairs from two lists

Python

```python
list1 = [1, 2, 3]
list2 = ['A', 'B', 'C']
pairs = [(x, y) for x in list1 for y in list2]
print(pairs)
```

Output

```
[(1, 'A'), (1, 'B'), (1, 'C'), (2, 'A'), (2, 'B'), (2, 'C'), (3,
'A'), (3, 'B'), (3, 'C')]
```

Example 8:

Objective:

Convert all fruit names to uppercase

Python

```python
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
upper_fruits = [x.upper() for x in fruits]
print(upper_fruits)
```

Output

```
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

Example 9:

Objective:

Set all values to "hello"

Python

```
fruits = ["apple", "banana", "cherry"]
newlist = ["hello" for x in fruits]
print(newlist)
```

Output

```
['hello', 'hello', 'hello']
```

Example 10:

Objective:

Flatten a nested list (convert a 2D list into a 1D list)

Python

```python
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
new_list = [element for sublist in nested_list for element in sublist]
print(new_list)
```

Output

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

04

Summary

Summary

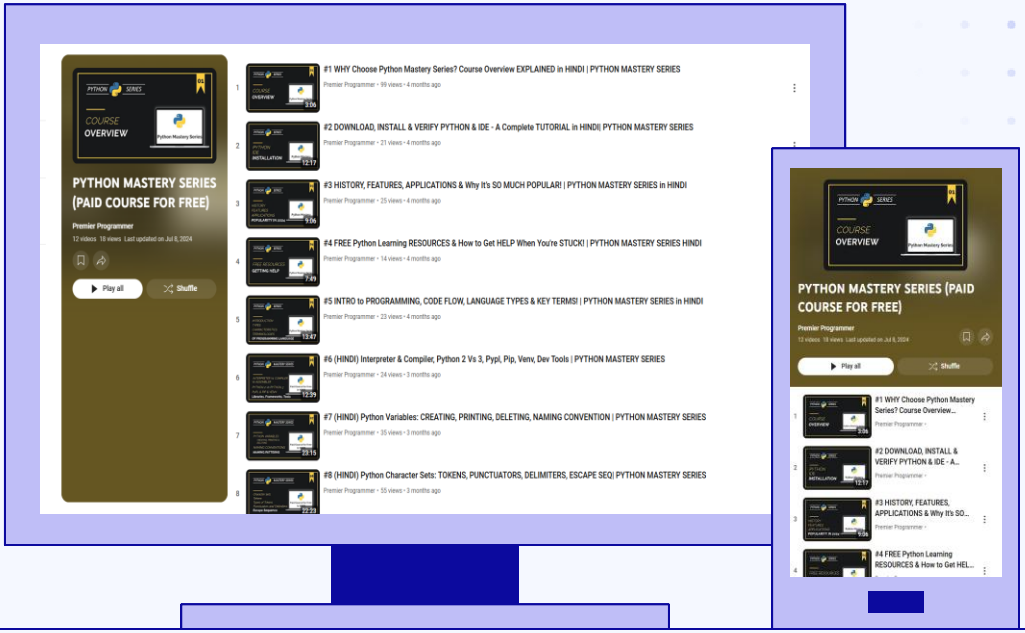
- **List Comprehension** is a concise way to create lists from iterables.
- It allows **filtering elements** using conditions.
- We can **modify elements** using if-else inside expressions.
- Supports **nested loops** for combinations.
- Generally **faster and more readable** than traditional loops.

Download Link in **Description** and **Pinned Comment**

List Loop

WATCH

Level up your coding with each episode in this focused Python series.



PYTHON MASTERY SERIES (PAID COURSE FOR FREE)

Premier Programmer

12 videos 18 views Last updated on Jul 8, 2024

Play all Shuffle

1 #1 WHY Choose Python Mastery Series? Course Overview EXPLAINED in HINDI | PYTHON MASTERY SERIES
Premier Programmer • 99 views • 4 months ago

2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A Complete TUTORIAL in HINDI| PYTHON MASTERY SERIES
Premier Programmer • 21 views • 4 months ago

3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO MUCH POPULAR! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 25 views • 4 months ago

4 #4 FREE Python Learning RESOURCES & How to Get HELP When You're STUCK! | PYTHON MASTERY SERIES HINDI
Premier Programmer • 14 views • 4 months ago

5 #5 INTRO to PROGRAMMING, CODE FLOW, LANGUAGE TYPES & KEY TERMS! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 23 views • 4 months ago

6 #6 (HINDI) Interpreter & Compiler, Python 2 Vs 3, Pypl, Pip, Venv, Dev Tools | PYTHON MASTERY SERIES
Premier Programmer • 24 views • 3 months ago

7 #7 (HINDI) Python Variables: CREATING, PRINTING, DELETING, NAMING CONVENTION | PYTHON MASTERY SERIES
Premier Programmer • 35 views • 3 months ago

8 #8 (HINDI) Python Character Sets: TOKENS, PUNCTUATORS, DELIMITERS, ESCAPE SEQ| PYTHON MASTERY SERIES
Premier Programmer • 55 views • 3 months ago

PYTHON MASTERY SERIES (PAID COURSE FOR FREE)
Premier Programmer
12 videos 18 views Last updated on Jul 8, 2024
Play all Shuffle
1 #1 WHY Choose Python Mastery Series? Course Overview...
2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A...
3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO...
4 #4 FREE Python Learning RESOURCES & How to Get HEL...

Next Video!

List Loop
Practice Set Solution