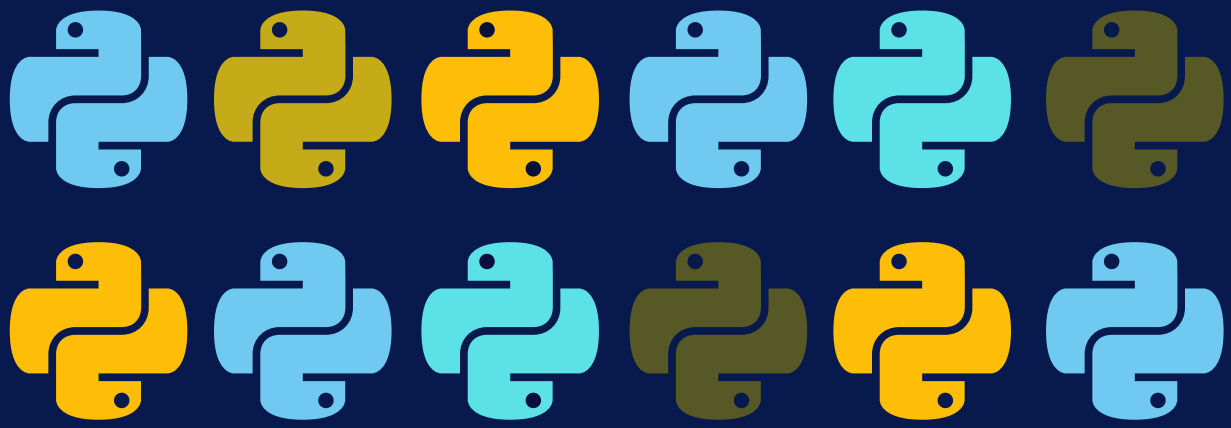


Mastering Python List Comprehensions

With Code Examples

A Powerful
Way to Create Lists

Python series

save for later

Python List Comprehensions

Python List Comprehensions are a powerful feature that allows you to create lists in a concise and expressive manner.
They consist of square brackets containing an expression followed by a for clause, which iterates over elements, and optionally, one or more for or if clauses for filtering or additional iterations

- Simplifies code and reduces the number of lines.
- Enhances readability and clarity, making code more maintainable.
- Often performs better in terms of execution speed compared to traditional loops.
- Encourages the use of Pythonic idioms, promoting elegant and efficient code.

Swipe next ⟶

follow for more

Basic Syntax

new_list = [expression for item in iterable]

```python
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x**2 for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Swipe next ⟶

save for later

With Condition

new_list = [expression for item in iterable if condition]

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = [x for x in numbers if x % 2 == 0]
print(even_numbers)  # Output: [2, 4, 6, 8, 10]
```

Swipe next ⟶

follow for more

Nested Loop

new_list = [expression for item1 in iterable1 for item2 in iterable2]

```python
fruits = ['apple', 'banana', 'cherry']
colors = ['red', 'yellow', 'green']
combinations = [fruit + ' ' + color for fruit in fruits for color in colors]
print(combinations)  # Output: ['apple red', 'apple yellow', 'apple green', 'banana red', 'banana yellow', 'banana green', 'cherry red', 'cherry yellow', 'cherry green']
```

Swipe next ⟶

save for later

With Function

new_list = [expression(item) for item in iterable]

```python
numbers = [1, 2, 3, 4, 5]
squared_numbers = [pow(x, 2) for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Swipe next ⟶

follow for more

With Conditional Expression

new_list = [expression1 if condition else expression2 for item in iterable]

```
numbers = [1, -2, 3, -4, 5]
absolute_values = [abs(x) if x < 0 else x for x in numbers]
print(absolute_values)  # Output: [1, 2, 3, 4, 5]
```

Swipe next ⟶

save for later

Flattening Lists

new_list = [item for sublist in iterable for item in sublist]

```python
matrix = [[1, 2], [3, 4], [5, 6]]
flattened = [item for sublist in matrix for item in sublist]
print(flattened)  # Output: [1, 2, 3, 4, 5, 6]
```

Swipe next ⟶

save for later

Flattening Lists

new_list = [item for sublist in iterable for item in sublist]

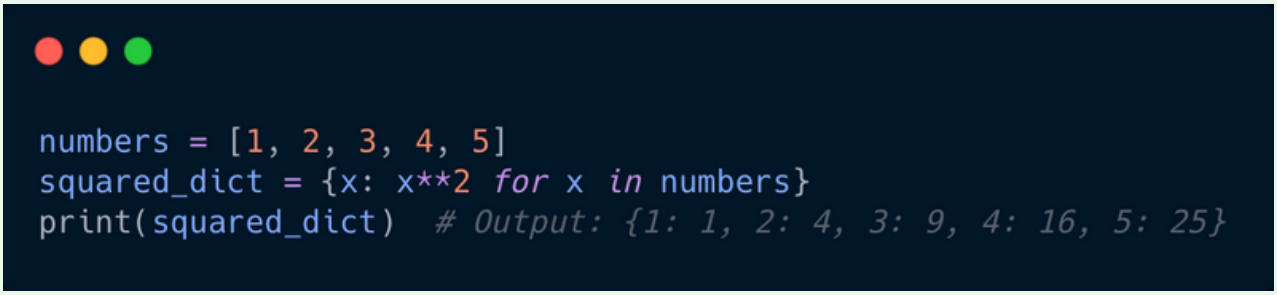
```python
matrix = [[1, 2], [3, 4], [5, 6]]
flattened = [item for sublist in matrix for item in sublist]
print(flattened)  # Output: [1, 2, 3, 4, 5, 6]
```

Swipe next ⟶

follow for more

With Dictionary Comprehension

new_dict = {key_expression: value_expression for item in iterable}



```python
numbers = [1, 2, 3, 4, 5]
squared_dict = {x: x**2 for x in numbers}
print(squared_dict)  # Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Swipe next ⟶

save for later

String Manipulation

new_list = [expression(item) for item in iterable]

```
words = ['apple', 'banana', 'cherry', 'date']
uppercase_words = [word.upper() for word in words]
print(uppercase_words)  # Output: ['APPLE', 'BANANA', 'CHERRY', 'DATE']
```

Swipe next ⟶

follow for more

With if-else

new_list = [expression1 if condition else expression2 for item in iterable]

```
numbers = [10, 20, 30, 40, 50]
result = ['Even' if x % 2 == 0 else 'Odd' for x in numbers]
print(result)  # Output: ['Even', 'Even', 'Odd', 'Even', 'Odd']
```

Swipe next ⟶

save for later

With Function

new_list = [function(expression) for item in iterable]

```python
import math

numbers = [1, 4, 9, 16, 25]
square_roots = [math.sqrt(x) for x in numbers]
print(square_roots)  # Output: [1.0, 2.0, 3.0, 4.0, 5.0]
```

Swipe next ⟶