Understanding Dictionary Comprehension

MUKESH KUMAR

AGENDA

- Introduction to Dictionary Comprehension
- Syntax and Explanation
- Examples with Comparison to Loops
- Nested Dictionary Comprehension
- Use Cases and Best Practices

Introduction to Dictionary Comprehension

- **Definition:**
  - Dictionary comprehension is a concise way to create dictionaries in Python.

- **Why Use It?**
  - Improves code readability.
  - Reduces lines of code compared to traditional loops.
  - Efficient for transforming data.

Basic Syntax

```
{key_expr: value_expr for item in iterable}
```

- **key_expr**: Expression for dictionary
- **keysvalue_expr**: Expression for dictionary
- **valuesiterable**: Any iterable object (list, tuple, set, etc.)

Example – Basic Comprehension

- Mapping number to their squares from 0 to 4.

```
squares = {x: x**2 for x in range(5)}
print(squares)
```

- Output:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Code without comprehension

- Mapping number to their squares from 0 to 4.

```
squares = {}
for x in range(5):
    squares[x] = x**2
```

Practice Questions – Basic

- Create a dictionary mapping numbers 1 to 5 with their cubes as values.
- Generate a dictionary where keys are letters in "hello" and values are their ASCII values.
- Create a dictionary of numbers from 1 to 10 and check if they are prime (use True/False).
- Make a dictionary of numbers 1 to 5 where values are their factorials.
- Generate a dictionary where keys are numbers 1-5 and values are their binary representation.

Comprehension with if Condition

- **Syntax – if Condition**

```
{key_expr: value_expr for item in iterable if condition}
```

- **key_expr:** Expression for dictionary keys.
- **value_expr:** Expression for dictionary values.
- **iterable:** Any iterable object (list, tuple, set, etc.).
- **condition (optional):** Filters elements.

Example : with if Condition

- Create a dict with number as key and their squares a value only for even numbers from 0 to 10.

```
even_squares = {x: x**2 for x in range(10) if x % 2 == 0}
print(even_squares)
```

- Output:

```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```

Without Dictionary Comprehension

- Create a dict with number as key and their squares a value only for even numbers from 0 to 10.

```
even_squares = {}
for x in range(10):
    if x % 2 == 0:
        even_squares[x] = x**2
```

Practice Questions – if Condition

- Create a dictionary of numbers 1-10 where only odd numbers are included as keys with their cubes as values.
- Generate a dictionary where keys are words in a list ["apple", "banana", "cherry"] and values are their lengths, but only for words longer than 5 characters.
- Make a dictionary of numbers 1-20 where values are True if the number is divisible by 3.
- Create a dictionary where keys are numbers 1-10, and values are squares but only if the number is greater than 5.
- Generate a dictionary of numbers 1-10 with their square roots, but only for even numbers.

Comprehension with if-else Condition

Syntax:

```
{key_expr: value_if_true if condition else value_if_false for item in iterable}
```

- **key_expr:** Expression for dictionary keys.
- **value_if_true, value_if_false:** Expression for dictionary values based on condition
- **iterable:** Any iterable object (list, tuple, set, etc.).
- **condition (optional):** Filters elements

Example :Comprehension with if-else Condition

- **Map number to ‘even’ and ‘odd’ in range of 0 to 5.**

```
number_type = {x: 'even' if x % 2 == 0 else 'odd' for x in range(5)}
print(number_type)
```

- Output:

```
{0: 'even', 1: 'odd', 2: 'even', 3: 'odd', 4: 'even'}
```

Without Dictionary Comprehension

- **Map number to ‘even’ and ‘odd’ in range of 0 to 5.**

```
number_type = {}
for x in range(5):
    if x % 2 == 0:
        number_type[x] = 'even'
    else:
        number_type[x] = 'odd'
```

Practice Questions – if-else Condition

- Create a dictionary of numbers 1-10 where values are "positive" if the number is greater than 5, otherwise "negative".
- Generate a dictionary mapping numbers 1-10 to "prime" or "composite".
- Make a dictionary where numbers 1-10 are keys, and values are "small" if less than 5, "medium" if between 5-7, and "large" otherwise.
- Generate a dictionary where words in a list ["car", "bike", "bus"] are keys, and values are "long" if length >3, otherwise "short".
- Create a dictionary mapping numbers 1-10 to "odd" if they are odd, and "even" if they are even, but for multiples of 3, store "multiple of 3".

Summary

- Dictionary comprehension makes dictionary creation concise
- Supports conditions and nested comprehensions
- Improves performance compared to traditional loops
- Use it wisely to maintain code readability