**What are Comprehensions in Python?**

Comprehensions are a **short, elegant, and readable way** to create new sequences (lists, dictionaries, sets) or perform filtering/mapping operations, **all in one line**.

They are essentially **shortcuts** for loops with optional conditions.

Types:

1. **List Comprehension**
2. **Dictionary Comprehension**
3. **Set Comprehension**
4. **Generator Expression**

**1. List Comprehension**

**Syntax:**

[expression for item in iterable if condition]

* **expression** → What to do with each item
* **iterable** → Sequence like list, tuple, range, etc.
* **condition** → (Optional) Filter items

**Example 1 – Basic loop vs comprehension:**

# Without comprehension

squares = []

for i in range(1, 6):

squares.append(i \*\* 2)

# With comprehension

squares = [i \*\* 2 for i in range(1, 6)]

print(squares) # [1, 4, 9, 16, 25]

**Example 2 – With condition:**

even\_squares = [i \*\* 2 for i in range(1, 11) if i % 2 == 0]

print(even\_squares) # [4, 16, 36, 64, 100]

**2. Dictionary Comprehension**

**Syntax:**

{key\_expression: value\_expression for item in iterable if condition}

**Example 1 – Mapping numbers to their squares:**

squares\_dict = {x: x \*\* 2 for x in range(1, 6)}

print(squares\_dict)

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

**Example 2 – Filtering:**

even\_squares\_dict = {x: x \*\* 2 for x in range(1, 11) if x % 2 == 0}

print(even\_squares\_dict)

# {2: 4, 4: 16, 6: 36, 8: 64, 10: 100}

**3. Set Comprehension**

**Syntax:**

{expression for item in iterable if condition}

* Works like list comprehension but creates a **set** (no duplicates, unordered).

**Example:**

nums = [1, 2, 2, 3, 4, 4, 5]

unique\_squares = {x \*\* 2 for x in nums}

print(unique\_squares)

# {1, 4, 9, 16, 25}

**4. Generator Expression**

**Syntax:**

(expression for item in iterable if condition)

* Like list comprehension but **uses () instead of []**.
* Returns a **generator object** (lazy evaluation → uses less memory).

**Example:**

gen = (x \*\* 2 for x in range(1, 6))

print(next(gen)) # 1

print(next(gen)) # 4

**Nested Comprehensions**

You can use **comprehensions inside comprehensions**.

**Example – Flattening a matrix:**

matrix = [[1, 2, 3], [4, 5, 6]]

flat = [num for row in matrix for num in row]

print(flat) # [1, 2, 3, 4, 5, 6]

**When to Use & When Not to Use**

✅ Use when:

* The logic is **short** and **easy to read**
* You want **concise** code

❌ Avoid when:

* The expression becomes **too long** or **complex** (hurts readability)
* Multiple nested loops with complex conditions → better use normal loops