List comprehension in Python is a concise and efficient way to create lists. It's a way to perform operations on each element of an iterable (like a list or range) and construct a new list based on the results. The syntax you've provided is a good example of how list comprehension works. Let's break it down:

pythonCopy code

newlist = [expression for item in iterable if condition == True]

1. **Expression**: This is the part of the list comprehension where you specify what you want each item in the new list to be. The expression is applied to each item in the iterable (if it meets the condition, if one is specified). It can be a simple operation like item itself, a function of the item like item\*2 or more complex expressions involving the item.
2. **For item in iterable**: This is the loop part of the list comprehension. You iterate over each item in the iterable. The iterable can be any Python iterable, like a list, tuple, set, or even a string (where each character is an item).
3. **If condition == True**: This is an optional part. It's a filter that only includes items in the new list if they meet a certain condition. If this part is omitted, all items in the iterable are included in the new list.

Note; In the expression we will say what we want in our newly created list. i.e either the squared of the iterator x or the squared even number of the x.

Let's look at an example to clarify:

pythonCopy code

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

squared\_even\_numbers = [x\*\*2 for x in numbers if x % 2 == 0]

In this example:

* The **expression** is x\*\*2, which squares each number.
* The **iterable** is numbers.
* The **condition** is x % 2 == 0, which checks if the number is even.

So, squared\_even\_numbers will be [4, 16] because it contains the squares of the even numbers (2 and 4) from the original list.

In summary, the **expression** in a list comprehension is the transformation you apply to each item in the iterable to create the new list, possibly filtering with a condition.



numbers = [1, 2, 3, 4, 5] squared\_even\_numbers = [x\*\*2 for x in numbers if x % 2 == 0] show this example in normal loop without list comprehension

To rewrite the list comprehension example in the form of a normal loop without using list comprehension, you would create an empty list and then add items to it using a for loop and an if statement. Here's how the example you provided would be translated:

pythonCopy code

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

squared\_even\_numbers = []

for x in numbers:

if x % 2 == 0:

squared\_even\_numbers.append(x\*\*2)

This code does the following:

* Initializes an empty list squared\_even\_numbers.
* Iterates over each element x in the numbers list.
* Checks if x is even (x % 2 == 0). If it is, it computes the square of x (x\*\*2) and appends it to the squared\_even\_numbers list.

After executing this loop, squared\_even\_numbers will contain the same values as the list comprehension example, which are the squares of even numbers from the original numbers list. In this case, squared\_even\_numbers will be [4, 16].