List Comprehension

Let's learn how to create a new list from an existing list using for loops.

We'll cover the following

- Definition
- Structure
- Creating a List Comprehension
  - Adding a Condition
- Using Multiple Lists

Definition

> List comprehension is a technique that uses a `for` loop and a condition to create a new list from an existing one.

The result is always a **new** list, so it's a good practice to assign list comprehension to a new variable.

Structure

A list comprehension statement is always enclosed in square brackets, `[]`.

The comprehension consists of three main parts:

[expression for loop if condition]

- The `expression` is an operation used to create elements in the new list.
- The `for loop` will iterate an existing list. The iterator will be used in the `expression`.
- New elements will only be added to the new list when the `if condition` is fulfilled. This component is optional.

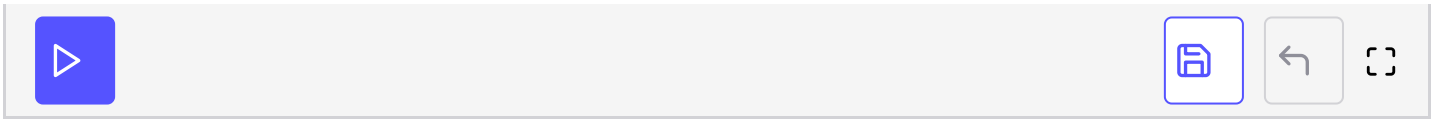
Creating a List Comprehension

Let's create a new list whose values are the doubles of the values of an existing list.

```
nums = [10, 20, 30, 40, 50]
nums_double = []

for n in nums:
    nums_double.append(n * 2)

print(nums)
print(nums_double)
```



Let's break down the loop above into the three components of a list comprehension.

The expression is equivalent to `n * 2` since it's used to create each value in the new list.

Our `for` loop is `for n in nums`, where `n` is the iterator.

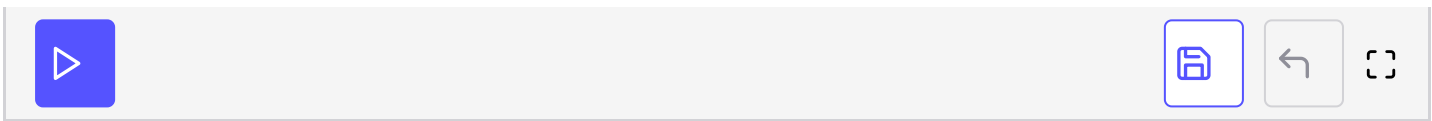
An `if` condition doesn't exist in this case.

So, let's convert the loop above into a list comprehension:

```
nums = [10, 20, 30, 40, 50]

# List comprehension
nums_double = [n * 2 for n in nums]

print(nums)
print(nums_double)
```



This looks more concise and clean! We can make a new list in a single line.

Adding a Condition

Our previous comprehension did not have a condition. All the values of the `nums` list were simply doubled and added to `nums_double`.

What if we only wanted our new list to have elements which were divisible by `4`?

We'd simply add an `if` condition at the end of our list comprehension:

```
nums = [10, 20, 30, 40, 50]

# List comprehension
nums_double = [n * 2 for n in nums if n % 4 == 0]

print(nums)
print(nums_double)
```

Now, only `20` and `40` were selected from `nums` since they fulfill the `if` condition.

Using Multiple Lists

List comprehension can also be performed on more than one list. The number of `for` loops in the comprehension will correspond to the number of lists we're using.

Let's write a list comprehension which creates tuples out of the values in two lists when their sum is greater than `100`. These tuples are the elements of the new list.

Here's the code:

```
list1 = [30, 50, 110, 40, 15, 75]
list2 = [10, 60, 20, 50]

sum_list = [(n1, n2) for n1 in list1 for n2 in list2 if n1 + n2 > 100]

print(sum_list)
```

We could solve the problem above using a nested `for` loop as well, but this approach seems much simpler.

The next data structure we'll study is the **tuple**.