List Comprehensions in Python

Suppose we need to create a list with first 10 multiple of 6 in it, So we may do this with a normal for loop or with list comprehensions, Let's see both of them and understand the difference.

Normal For loop

```
list1 =[]
for n in range(1,11):
    list1.append(n*6)
print(list1)
```

Output:
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]

List comprehension

```
list1 = [n*6 for n in range(1,11)]
print(list1)
```

Output:
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]

We got the same output using list comprehensions just by writing a line of code.

In general list comprehension

[<the_expression> for <the_element> in <the_iterable>]

Comparing this with our example n*6 is **the expression**, n is **the element**, range(1,11) is **the iterable**.

Applying list comprehension with a condition
Now, Suppose we need to create a list of multiple of 6 for just even numbers between 1 to 10.

```
list1 =[]
for n in range(1,11):
    if n%2==0:
        list1.append(n*6)
print(list1)
```

Output:
[12, 24, 36, 48, 60]

Using list comprehensions
```
list1 = [n*6 for n in range(1,11) if n%2==0]
print(list1)
```

Output:
[12, 24, 36, 48, 60]

In general list comprehension

[<the_expression> for <the_element> in <the_iterable> if <the_condition>]

Comparing this with our example n*6 is **the expression**, n is **the element**, range(1,11) is **the iterable** and n%2==0 is **the condition.**

Applying list comprehension with if-else condition
Now, Suppose we need to create a list of multiple of 6 for even numbers between 1 to 10 and multiple of 5 for rest of the numbers.

```
list1 =[]
for n in range(1,11):
   if n%2==0:
      list1.append(n*6)
   else:
      list1.append(n*5)
print(list1)
```

Output:
[5, 12, 15, 24, 25, 36, 35, 48, 45, 60]

Using list comprehensions
```
list1 = [n*6 if n%2==0 else n*5 for n in range(1,11)]
print(list1)
```

Output:
[5, 12, 15, 24, 25, 36, 35, 48, 45, 60]

In general list comprehension

[<the_expression> if <the_condition> else <other_expression> for <the_element> in <the_iterable>]

Comparing this with our example n*6 is **the expression**, n%2==0 is **the condition,** n*5 is **the other expression**, n is **the element** and range(1,11) is **the iterable.**

Applying list comprehension with Nested loops
Now, Suppose we need to multiply n ranging from 1 to 10 with first 1 then 2 and then 3.

```
list1 =[]
for i in range(1,4):
    for j in range(1,11):
        list1.append(i*j)
print(list1)
```

Output:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

```
list1 = [i*j for i in range(1,4) for j in range(1,11) ]
print(list1)
```

Output:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

In general list comprehension

```
[ <the_expression> for <element_a> in <iterable_a> (optional if <condition_a>)
            for <element_b> in <iterable_b> (optional if <condition_b>)
            for <element_c> in <iterable_c> (optional if <condition_c>)
            ... and so on ...]
```

Comparing this with our example i*j is **the expression**, i is **the element_a**, j is **the element_b**, range(1,4) is **the iterable_a** and range(1,11) is **the iterable_b.**