List Comprehension

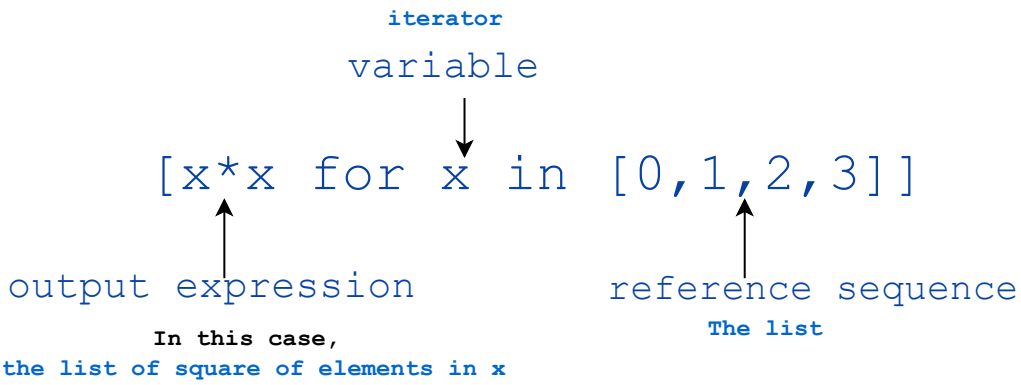
Now that we are done with lists, let's have a look at list comprehension.

WE'LL COVER THE FOLLOWING

List Comprehensions

List comprehensions are a concise way to create lists. They consist of square brackets containing an expression followed by the `for` keyword; the result will be a list whose results match the expression.

The general syntax is:



In python, here's how to create a list with the **squared numbers** of another list.

```
print([x*x for x in [0, 1, 2, 3]])
```

Given their flexibility, list comprehensions generally make use of the `range` function that returns a range of numbers.

For example,

```
range(4) # returns values from 0 to 3.
```

> **Note:** default value of range(n) starts from 0

If you want to define the starting value, write the following:

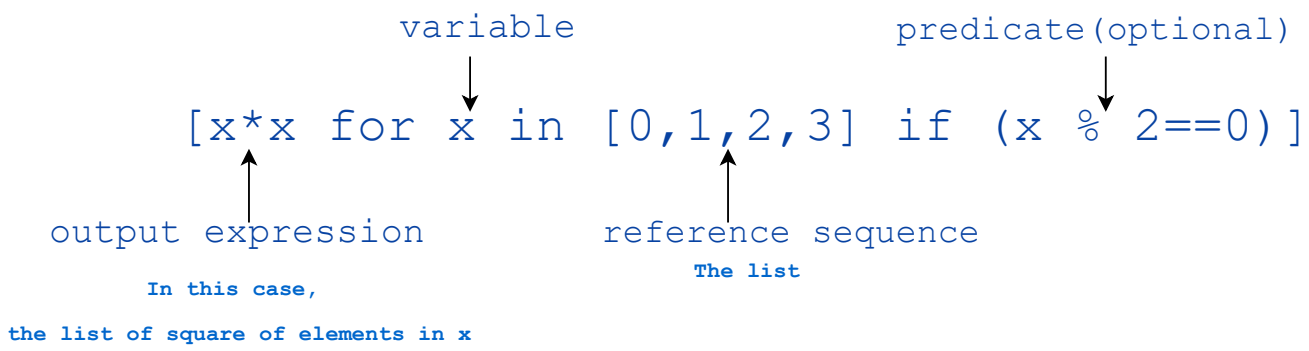
```
range(startingvalue, n) # returns value from startingvalue to n-1
```

> if range iterator is not specified it iterates with increments of 1, but if it is defined it increments by that value.

```
range(startingvalue, n, i) # returns value from startingvalue to n-1 wit
h i increment
```

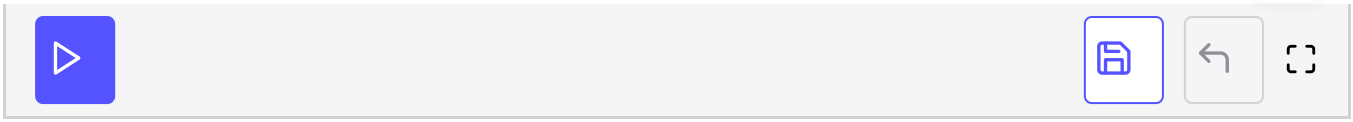
```
print [x*x for x in range(4)]
```

Sometimes you may want to filter the elements by a given condition; the `if` keyword can be used in those cases.



The following python code displays all elements from 0 to 9 which are divisible by 2.

```
print [x for x in range(10) if x % 2 == 0]
```



The example above returns all even values in range 0...10.

More about list comprehensions can be found at Python Documentation on List Comprehension.

Now that the concept of a list comprehension is clear, let's check your knowledge in the upcoming exercises before moving on to the next 'Modules and Functions' chapter.