Just a for loop

```python
>>> [i for i in range(5)]
[0, 1, 2, 3, 4]
```

This is your friendly neighborhood list comprehension. It works like a very simple for loop.

For with an if

```python
>>> [i for i in range(5) if i%2 == 0]
[0, 2, 4]
```

The if statement here is *filtering* the elements you want to consider.

If-else inside of the for

```python
>>> [i if i%2 == 0 else -i for i in range(5)]
[0, -1, 2, -3, 4]
```

This time, the if-else statement is deciding *what the element will be*, not *whether it should be included*.

Double for loop

```python
>>> [(i,j) for i in range(2) for j in range(2)]
[(0, 0), (0, 1), (1, 0), (1, 1)]
```

The fancy name for this is an *iterated comprehension*. It gives a 1D array. Notice the precedence: first we get `[(0, j) for j in range(2)]` and then we get `[(1, j) for j in range(2)]`. In this way the “i” loop is the “outer” one.

2D comprehension

```python
>>> [[(i,j) for i in range(2)] for j in range(2)]
[[[0, 0], [0, 1]], [[1, 0], [1, 1]]]
```

This is actually just two friendly neighborhood list comprehensions but inside nested arrays. Notice how this means that the precedence is shifted from the previous example, the “j” loop is the outer one.
Construct a non-array

```python
>>> {i for i in [0,0,0,1,1,2,0,0]}
{0,1,2}

>>> {i: i % 2 for i in range(5)}
{0: 0, 1: 1, 2: 0, 3: 1, 4: 0}
```

Any of the list comprehensions here can be used to construct a set, dictionary, or tuple.