Python Comprehensions

List Comprehensions are a special kind of syntax that let us create lists out of other lists, and are incredibly useful when dealing with numbers and with one or two levels of nested for loops.

List comprehension

This is how we create a new list from an existing collection with a For Loop:

```
>>> names = ['Charles', 'Susan', 'Patrick', 'George']

>>> new_list = []
>>> for n in names:
...     new_list.append(n)
...
>>> new_list
# ['Charles', 'Susan', 'Patrick', 'George']
```

And this is how we do the same with a List Comprehension:

```
>>> names = ['Charles', 'Susan', 'Patrick', 'George']

>>> new_list = [n for n in names]
>>> new_list
# ['Charles', 'Susan', 'Patrick', 'George']
```

Adding conditionals

If we want `new_list` to have only the names that start with C, with a for loop, we would do it like this:

```
>>> names = ['Charles', 'Susan', 'Patrick', 'George', 'Carol']

>>> new_list = []
>>> for n in names:
...     if n.startswith('C'):
...         new_list.append(n)
...
>>> print(new_list)
# ['Charles', 'Carol']
```

In a List Comprehension, we add the `if` statement at the end:

```
>>> new_list = [n for n in names if n.startswith('C')]
>>> print(new_list)
# ['Charles', 'Carol']
```

Set comprehension

```
>>> b = {"abc", "def"}
>>> {s.upper() for s in b}
{"ABC", "DEF"}
```

Dict comprehension

```
>>> c = {'name': 'Pooka', 'age': 5}
>>> {v: k for k, v in c.items()}
{'Pooka': 'name', 5: 'age'}
```

A List comprehension can be generated from a dictionary:

```
>>> c = {'name': 'Pooka', 'first_name': 'Oooka'}
>>> ["{}:{}".format(k.upper(), v.upper()) for k, v in c.items()]
['NAME:POOKA', 'FIRST_NAME:OOOKA']
```