List Comprehension

This notebook introduces you to the basics of list comprehension.

List comprehensions provide a concise way to create lists. It consists of square brackets containing an expression followed by a for clause, then zero or more `for` or `if` clauses. The result will be a new list.

List comprehension starts with a `[` and `]`, to help you remember that the result is going to be a list.

The syntax is: `[expression for item in list if conditional ]`

In [2]:

```
'''
Create a list of integers between 18 and 27 (inclusive)
'''
my_lst = [x for x in range(8, 14)]
print(my_lst)
```

```
[8, 9, 10, 11, 12, 13]
```

In [3]:

```
'''
Create a list that contains the squares of the integers between 18 and 27 (inclusive)
'''
my_lst = [x**2 for x in range(8, 14)]
print(my_lst)
```

```
[64, 81, 100, 121, 144, 169]
```

In [6]:

```
'''
Create a new list consisting of all the non-string values from my_lst
'''
my_lst = ['ABC', 23.4, 7, 'Wow', 16, 'xyz', 10]
new_lst = [x for x in my_lst if type(x) != str]
print(new_lst)
```

```
[23.4, 7, 16, 10]
```

In [8]:

```
'''
print the first character of each word in the input list
'''
listOfWords = ["this","is no there","apples","List","Tf","words"]
new_list = [word.lower() if len(word) % 2 == 0
            else word.upper()
            for word in listOfWords
            if len(word) <=5]
print(new_list)
```

```
['this', 'list', 'tf', 'WORDS']
```

In [1]:

```
'''
list comprehension can also be used on functions
'''

'''
Suppose we have the following function which accepts an integer and returns the cube of the intege
r
'''
```

```
'''
def cube_value(n):
    return n*n*n

'''
Now suppose we have a list of integers called my_lst and we want a new list that contains the cube
of the integers
'''
my_lst = [8,12,15,19,22,34]
cubed_lst = [cube_value(x) for x in my_lst]
print(cubed_lst)

'''
Now suppose we want a new list that contains the cube of only the odd integers in my_lst
'''
cubed_odd_lst = [cube_value(x) for x in my_lst if x%2 == 1]
print(cubed_odd_lst)
```

```
[512, 1728, 3375, 6859, 10648, 39304]
[3375, 6859]
```

In [9]:

```
'''
You can create new lists by iterating across two lists simultaneously using two ieteration variabl
es.
In the example below, each element in the first list is added to every element in the second list
resulting in a new list
whose length is the product of the lengths of the two input lists.
'''
list1 = [5, 8, 15, 19, 25]
list2 = [6, 19, 23, 10, 5, 7]
list3 = [x + y for x in list1 for y in list2]
print(list3)
print(len(list3))
```

```
[11, 24, 28, 15, 10, 12, 14, 27, 31, 18, 13, 15, 21, 34, 38, 25, 20, 22, 25, 38, 42, 29, 24, 26,
31, 44, 48, 35, 30, 32]
30
```

In [10]:

```
'''
Create a new list by summing the corresponding elements in the two input lists
'''
list1 = [5, 8, 15, 19, 25]
list2 = [6, 19, 23, 10, 5,7]
list3 = [x + y for x in list1 for y in list2 if list1.index(x)==list2.index(y)]
print(list3)
```

```
[11, 27, 38, 29, 30]
```