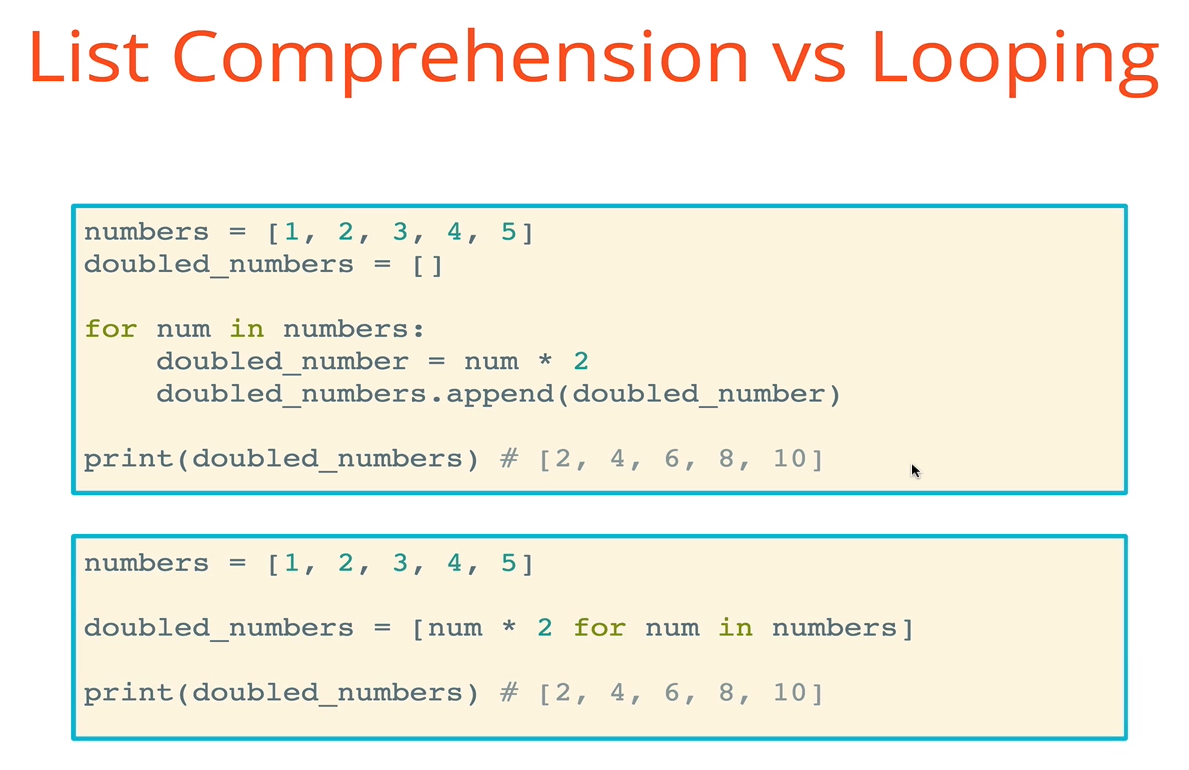
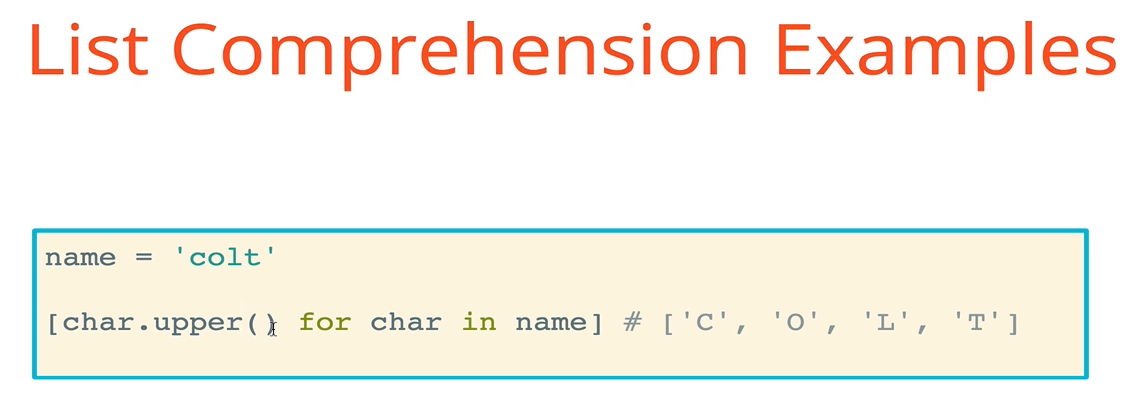
* A cool shorthand syntax that allows you to make new lists, make direct copies of other lists, or generated a modified list from an existing list
* Syntax: for each thing in a list, do something (or do nothing) for that thing and append the modified thing to a new list
* Performs the same function as a for loop, but with less code



* Iterating over strings



* Capitalize the first letter of each word in a list



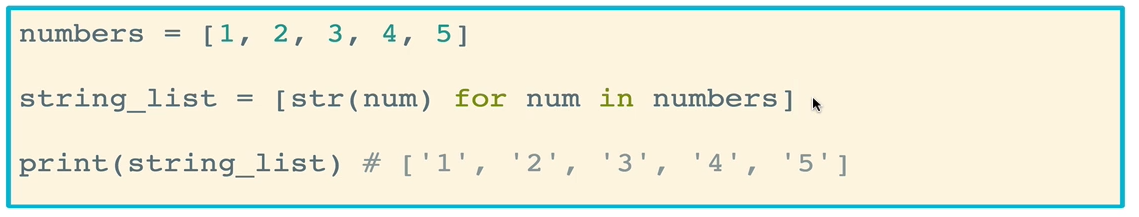
* Multiply numbers in a list (generated by the range function)



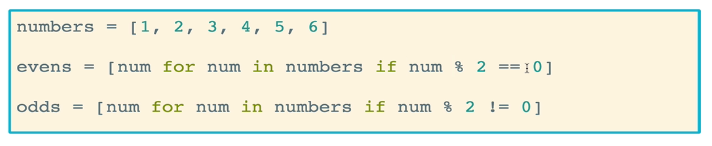
* Return the Boolean representation of a value (True or False) for each item in a list



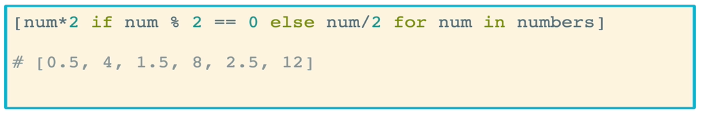
* Convert a list of numbers to a list of stringified numbers



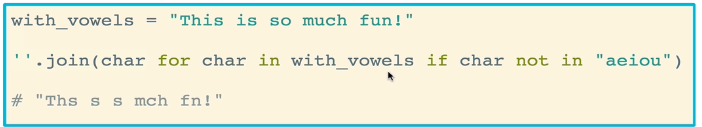
* Conditional logic can be added to list comprehension. The code below will loop through each number in the numbers list, test whether the number is even or odd, and append the number to the “evens” list if it is even and to the “odds” list if it is odd



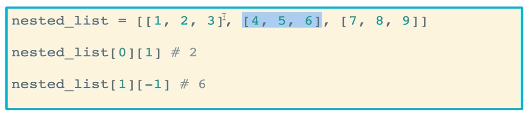
* You can use if/else statements in list comprehension. It’s a little confusing, but with practice it gets easier. Just look it up if you forget. In the code below, even numbers will be multiplied by 2, while odd numbers will be divided by 2. The results are appended sequentially to a new list



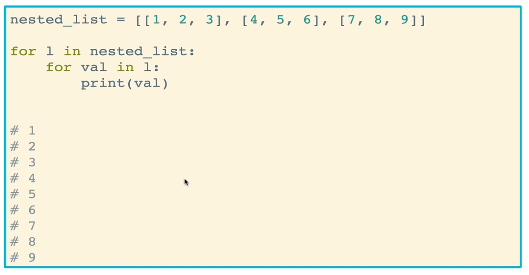
* We can also do checks on using *in* and *not in*. In the code below, the string .join() method is used to join each character (including spaces) in the with\_vowels string together by an empty string. The .join(*string*) method which usually joins the elements of *string*. In this case however, each character in the with\_vowels string is tested to determine whether it is a vowel. If it is not, then the character gets added to the new string. The end effect is removing all vowels from the string



* **Nested lists** are just lists that has lists as elements in them
  + Used for complex data structures, such as matrices
  + Data science structures in Python
  + Game boards, mazes
  + Rows and columns for visualizations, tabulation, and grouping data
* Accessing nested lists is done by grabbing the index of the list you want, then the index of the element within the list



* You can also iterate through nested loops by looping through one item at a time



* Nested list comprehension

