Activities: List Comprehension

```
{i for i in range(2,100) if i not in
{i*j for i in range(2,51) for j in range(2,51) if i*j < 100}}
```

1. We used this code to generate a set of all prime numbers between 1 and 100. The code first creates a set of all composite numbers and then just picks the numbers that are not in this set as the prime numbers. Implement the same construction without using list comprehension.

2. Use list comprehension to generate a list of all square numbers between 1 and 1000^2 that have remainder 9 when dividing by 10. What about a list of all squares that have remainder 7 modulo 10?

3. Use list comprehension on the letters of a string to create a new string that repeats each letter in the original string twice. For example, if the string is "A nice afternoon, isn't it?", then the altered string is "AA nniiccee aaffttteerrnnoooonn,, iissnn''tt iitt??". Hint: You want to use the `"".join([ ])` construct.
4. Use list comprehension to generate a dictionary that associates to a number between 1 and 10000 its exact cubed root, if it exists. The dictionary should contain {1:1, 8:2, 27:3, …}

5. Use list comprehension to generate a list of all prime numbers between 1 and 10 000. What happens if you just use the code above with the proper alterations? (Answer: the creation of composite numbers takes 5000 x 5000 iterations, which takes up a lot of time. This happens for the determination of primeness for every number.) If we define a set composites first and then use list comprehension in order to create prime, what happens? (Answer: it is still very slow, but at least do-able). You should learn from this that elegance does not equal effectiveness.

6. Use the implementation of the sieve of Eratosthenes as a better generator of the set of all prime numbers between 1 and 10000. Then write a function that takes a number between 10000 and generates a list of all of its prime factors. For example, the function applied to 72 should yield [2,2,2,3,3].