# Simple Python Exercises

## Implement these Python functions.

1. Define a function `max()` that takes two numbers as arguments and returns the largest of them. Use the if-then-else construct available in Python. (It is true that Python has the `max()` function built in, but writing it yourself is nevertheless a good exercise.)

2. Define a function `max_of_three()` that takes three numbers as arguments and returns the largest of them.

3. Define a function that computes the *length* of a given list or string. (It is true that Python has the `len()` function built in, but writing it yourself is nevertheless a good exercise.)

4. Write a function that takes a character (i.e. a string of length 1) and returns `True` if it is a vowel, `False` otherwise.

5. Define a function `sum()` and a function `multiply()` that sums and multiplies (respectively) all the numbers in a list of numbers. For example, `sum([1, 2, 3, 4])` should return `10`, and `multiply([1, 2, 3, 4])` should return `24`.

6. Define a function `reverse()` that computes the reversal of a string. For example, `reverse("I am testing")` should return the string `"gnitset ma I"`.

7. Define a function `is_palindrome()` that recognizes palindromes (i.e. words that look the same written backwards). For example, `is_palindrome("radar")` should return `True`.

8. Write a function `is_member()` that takes a value (i.e. a number, string, etc) `x` and a list of values `a`, and returns `True` if `x` is a member of `a`, `False` otherwise. (Note that this is exactly what the `in` operator does, but for the sake of the exercise you should pretend Python did not have this operator.)

9. Define a function `overlapping()` that takes two lists and returns True if they have at least one member in common, False otherwise. You may use your `is_member()` function, or the `in` operator, but for the sake of the exercise, you should (also) write it using two nested for-loops.

10. Define a function `generate_n_chars()` that takes an integer `n` and a character `c` and returns a string, `n` characters long, consisting only of `c`:s. For example, `generate_n_chars(5,"x")` should return the string `"xxxxx"`. (Python is unusual in that you can actually write an expression `5 * "x"` that will evaluate to `"xxxxx"`. For the sake of the exercise you should ignore that the problem can be solved in this manner.)

11. Write a function `find_longest_word()` that takes a list of words and returns the length of the longest one.

12. Write a function `filter_long_words()` that takes a list of words and an integer `n` and returns the list of words that are longer than `n`.

13. Write a function `char_freq()` that takes a string and returns a dictionary with characters as keys and their frequency as the values. Try it with something like `char_freq("abbabcbdbabdbdbabababcbcbab")`.

14.