Homework for Lists - Part 1

1. **Sorted Order**[1]: Write a program that reads integers from the user and stores them in a list. Your program should continue reading values until the user enters 0. Then it should display all of the values entered by the user (except for the 0) in order from smallest to largest, with one value appearing on each line. Use the `sort` method to sort the list.

   > Hint: We can read values forever using a loop, just like we have done in the past lessons.

2. **Reverse Order**[1]: Write a program that reads integers from the user and stores them in a list. Use 0 as a sentinel value to mark the end of the input. Once all of the values have been read your program should display them (except for the 0) in reverse order, with one value appearing on each line.

   > Hint: Don't forget that most of the list methods are acting *in place*, changing the list itself.

3. **Negatives, Zeros and Positives**[1]: Create a program that reads integers from the user until a blank line is entered. Once all of the integers have been read, your program should display all of the negative numbers, followed by all of the zeros, followed by all of the positive numbers. Within each group the numbers should be displayed in the same order that they were entered by the user. For example, if the user enters the values 3, -4, 1, 0, -1, 0, and -2 then your program should output the values -4, -1, -2, 0, 0, 3, and 1. Your program should display each value on its own line.

   > Hint: You can use seprate lists to store the values as they are coming in from the user.

4. **List of Proper Divisors**[1]: A proper divisor of a positive integer, *n*, is a positive integer less than *n* which divides evenly into *n*. Write a function that computes all of the proper divisors of a positive integer. The integer will be passed to the function as its only parameter. The function will return a list containing all of the proper divisors as its only result. Complete this exercise by writing a main program that demonstrates the function by reading a value from the user and displaying the list of its proper divisors.

   > Hint: You can search for the proper divisor in a loop and append each divisor to a result list and return it at the end of the function.

5. **Perfect Numbers**[1]: An integer, *n*, is said to be perfect when the sum of all of the proper divisors of *n* is equal to *n*. For example, 28 is a perfect number because its proper divisors are 1, 2, 4, 7 and 14, and 1 + 2 + 4 + 7 + 14 = 28. Write a function that determines whether or not a positive integer is perfect. Your function will take one

parameter. If that parameter is a perfect number then your function will return true. Otherwise it will return false. In addition, write a main program that uses your function to identify and display all of the perfect numbers between 1 and 10,000. Import your solution to Exercise 4 when completing this task.

> Hint: You can use the built-in function `sum(foo)` where `foo` is a list to get the sum of its elements. Use the result of the function that returns the list of proper divisors and sum its elements to check against the numbers going from 1 to 10,000.

Don't forget to use the `main()` function when building your programs!

---

1 - *Ben Stephenson, The Python Workbook*