Lab 5: For loops and lists

Labs are graded for participation rather than correctness. Keep all your lab code in your course GitHub repo to receive credit for your work. We'll be looking to see that you have at least partially completed all problems in each lab.

If you finish the lab assignment early, you may get started on the homework.

Getting started

In your local repo, create a folder for Lab 5. Create new .py files for this week's lab problems and save them in the Lab 5 folder.

Documentation and format guidelines

The guidelines from previous labs still apply. No new guidelines this week!

Problem 1: Scores

Write a program to compute some statistics for a class of students' scores on an assignment. The students' scores should be stored in a list. Your program should be able to compute the following:

- the average (mean)
- the median
- the lowest score
- the highest score

Write separate functions for each statistic and, write Pytest tests to practice unit testing functions with list parameters. A few suggestions of things to test: a list with an even number of grades, a list with an odd number of grades, a sorted list, an unsorted list, and an empty list.

Note: Python provides built-in functions `min()`, `max()`, and `sum()`, which enable you to get those statistics without using loops. However, the goal of this lab is to practice using loops and working with lists so write your functions without using `min()`, `max()`, or `sum()`. You may use any of the list methods given at the bottom of this resource: https://www.w3schools.com/python/python_lists.asp

Optional extension: Allow the user to enter scores one at a time then get the statistics for the class. You will need to use a while loop to allow the user to keep entering scores. You will also

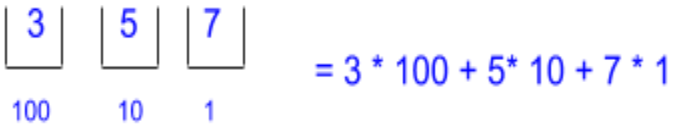
need a way for the user to indicate that they are finished entering grades so you can exit the while loop.

Problem 2: binary to decimal

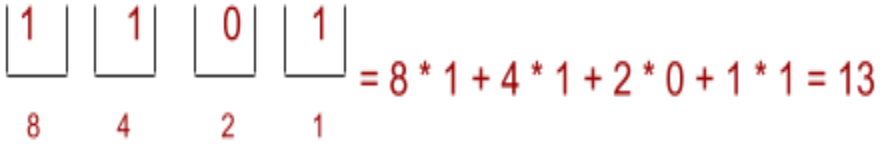
In the last lab you wrote a function to compute the log base 2 of powers of 2. We care about logarithm base 2 and powers of 2 in computer science because of the nature of circuits: They understand on/off, zero/one, True/False. Values in memory (numeric, strings, machine code, all of it) are stored as a collection of 0s and 1s.

Of course, there's a translation that happens -- between 0s and 1s, also called *base 2* or *binary*, and what we might recognize as a more "normal" number, also called *base 10* or *decimal.*

The value of a binary number is calculated a lot like the way we learned about numbers as kids. Looking at a decimal number, there's a ones place, a tens place, a hundreds place, etc.



Binary numbers work exactly the same way. Except, instead of the digits 0-9, we have 0 and 1, that's it. And instead of powers of 10 (1, 10, 100, 1000, etc.), we use powers of 2 (1, 2, 4, 8, 16, 32, etc.).



We would say that 1101 base 2 is equal to 13 base 10. Or: $1101_2 = 13_{10}$

For this problem:

- Prompt the user for a binary number (assume good input).
- Convert it to its decimal equivalent and print it out.

Helpful hints:

- Keep the user's input as a string, not an int. It's easier to iterate through it that way.
- We haven't used the exponent operator much in this class, but you can use it to get the next power of 2. 2 ** 0 is 1, 2 ** 1 is 2, 2 ** 2 is 4, etc.
- In my solution, I started at the rightmost character in the string and worked my way to the left. Python has a built-in function for reversing strings but we ask that you do not use it for this assignment!

Problem 3: draw a rectangle

Write a program to draw a rectangle on the command line.

- Prompt the user for a desired width in columns (assume good input).
- Prompt the user for a desired height in rows (assume good input).
- Prompt the user for a character that you will use to draw the rectangle. The character can be any single letter, number, or punctuation mark except space (assume good input).

For example, given a width of 6 columns, a height of 4 rows, and the character *, your output would look like this:

```
******
*    *
*    *
******
```

This is a simple program—try to make it as efficient as possible!