Histograms

Histograms provide a convenient graphical representation of numerical data by showing the number of measurements that fall in a given range by the height of a vertical bar.

To construct a histogram, you begin by deciding how many bars you want. You then divide the range of the data up into subintervals or *bins* and count the number of measurements that fall in each bin. For example, suppose the measurements are

0.5, 0.5, 0.7, 0.9, 3.4, 2.7, 1.8, 1.9, 1.7, 2.4, 3.0, 6.1

Also suppose you’ve decided that the data that you’re interested in lies in the range from 0 to 4, and that you want four bins. Then dividing 0–4 into four equal-sized subintervals gives us the bins

0–1, 1–2, 2–3, and 3–4

So we immediately see that 6.1 is going to be ignored, since it doesn’t belong to any bin. We also see that 3.0 can go into either the 2–3 bin or the 3–4 bin. We’ll use the convention that the bins contain their left endpoint, but not their right. So the bin 2–3 should contain all the measurements ≥ 2 and < 3, while the bin 3–4 should contain all the measurements ≥ 3 and < 4. So 3.0 should go into the 3–4 bin.

Now we can just go through the data and count the number of measurements falling into each bin. We see that

- Bin 0–1 contains 4 measurements
- Bin 1–2 contains 3 measurements
- Bin 2–3 contains 2 measurements
- Bin 3–4 contains 2 measurements

So a histogram of the data should look something like the diagram in Figure 1.

2 Program 3

For programming assignment 3, you should write a Python program that builds a histogram from user-input data. Input to the program will consist of the following information:

- `n`: the number of measurements in the user’s data.
- `vals`: the user’s measurements. So this will be a list of `n` floats.
Figure 1: A histogram

- b: the number of bars or bins.
- min_val: the minimum value for the smallest bin.
- max_val: the maximum value for the largest bin.

These values will be entered in this order, and each value (including the elements in vals) will be on a separate line. You can assume that all of the input data is correct and “makes sense.” So, for example, n and b will be positive ints. The elements of vals and min_val and max_val will be floats and min_val < max_val. However, it may happen that there are elements of vals that should be ignored because they’re either less than min_val or greater than or equal to max_val. After getting the user input, your program should determine the boundaries of the bins and count the number of measurements that fall into each bin.

Although you can get extra credit if you use TurtleWorld to draw the histogram (see below), it’s only necessary to use “ASCII graphics.” Rather than drawing vertical rectangles, you can draw horizontal bars composed of capital X’s. You should also identify each bar with the range of data that it contains. So the program output for the data in the example above might look something like this:

```
0.0 -- 1.0 XXXX
1.0 -- 2.0 XXX
2.0 -- 3.0 XX
3.0 -- 4.0 XX
```

### 3 Some Details

Getting the bars to start in the same location on each line of output can be a little tricky if the number of digits in the bin boundaries varies. For example, suppose the bins are 0–20, 20–40,
40–60, 60–80, and 80–100. If you’re not careful, your histogram might look something like this:

- 0.0 -- 20.0 XXXX
- 20.0 -- 40.0 XXXXXXXXXXXXXX
- 40.0 -- 60.0 XXX
- 60.0 -- 80.0 XXXXX
- 80.0 -- 100.0 XX

Of course, what you want is something like this

- 0.0 -- 20.0 XXXX
- 20.0 -- 40.0 XXXXXXXXXXXXXX
- 40.0 -- 60.0 XXX
- 60.0 -- 80.0 XXXXX
- 80.0 -- 100.0 XX

The problem is that 0.0 and 100.0 have different numbers of digits from the other values. There’s a function on the class website called `create_format` which can be used to create a format string that will give all of the values the same “width” when they’re printed. It can be used in your program to print a line of the histogram as follows:

```python
fmt_s = create_format(min_val, max_val)
.
.
print fmt_s % lower_bound, "--", fmt_s % upper_bound, bar_of_xs
```

Testing your program will probably involve running it many times, and typing in the data over and over can be very tedious. Fortunately, Unix shells allow you to redirect input. So if all of the input data is in the file `input_tst` and your Python program is in the file `histo.py`, you can run the program without typing any input by using the command

```
$ python histo.py < input_tst
```

Recollect that this tells the system to get the input from the file `input_tst` instead of the keyboard. You can generate more or less random input data using the program `rand_data.py` on the class website. This will generate input in the format specified above.

### 4 Due Date

In order to receive full credit, your program must be in the `p3` subdirectory of your Subversion repository by 2:00 pm on Friday, October 28, and you must turn in a print out of your program by 5 pm on the 28th.

### 5 Grading

Your program will be graded on the basis of its correctness and its “static features.”

1. Correctness will be 60% of your grade. Does your program produce a correct histogram for various sets of input? Does it ignore values outside the range specified by the user? Does it correctly count values that fall on the boundary of a bin?

2. The following static features will be graded.
(a) Documentation will be 10% of your grade. Does your header documentation include the author’s name, the purpose of the program, and a description of how to use the program? Are the identifiers meaningful? Are any obscure constructs clearly explained?

(b) Source format will be 10% of your grade. Is the indentation consistent? Have blank lines been used so that the program is easy to read?

(c) Quality of solution will be 20% of your grade. Does your solution contain unnecessary calculations? Is your solution too clever — e.g., has the solution been condensed to the point where it’s incomprehensible? Are functions and data structures (e.g., lists and strings) used properly? Are any functions (including the main program) more than 20 lines long?

3. It’s also possible to get up to 10 points extra credit by using TurtleWorld to draw the histogram. In addition to drawing the bars, you’ll also need to label the horizontal and vertical axes. The author has some code in which he uses a Turtle to draw letters. You can use this as the basis of code for drawing numbers. (For a link, see the class website.)

6 Collaboration

It is OK for you to discuss solutions to this program with your classmates. However, no collaboration should ever involve looking at one of your classmate’s source programs! It is usually extremely easy to determine that someone has copied a program, even when the individual doing the copying has changed identifiers and comments. If we discover that someone has copied a program, the authors of both programs will receive an F in the course.