1) Lists. Familiarize yourself with the Python list operations in the table on page 343 of your text, as well as the list functions in the table on page 345.

a. [~15 minutes] Complete Discussion Exercise 1 on pages 379 of the text. Use the interactive Python Shell to verify your answers. When you understand them all, you are ready to earn your check.

   ☟ Get check 1 ☟

b. [~15 minutes] Complete Discussion Exercise 2 on page 380 of the text. Same instructions as above apply. For this one we will ask you to explain a couple of your answers to us.

   ☟ Get check 2 ☟

2) Programming Exercise 5 from Chapter 11. Hint: use for loops to achieve each of these.

a. [~15 minutes] Part a. (This should return an integer value, as myList.count(x) would.)

   ☟ Get check 3 ☟

b. [~15 minutes] Part b. (This should return a boolean value, as x in myList would.)

   ☟ Get check 4 ☟

c. [~15 minutes] Part c. (This should return an integer value, as myList.index(x) would.)

   ☟ Get check 5 ☟

3) Today in class we wrote code to count how frequently each letter of the alphabet appears in your names. Let’s re-save it as letterFreq.py. This was our code so far:
a. [~5 minutes] Test it and run it, make sure it works. Add comments explaining what is going on at the lines marked with #### so that this can be a good source for future reference.

☺ Get check 6 ☺

b. Our goal now (over the next three checks) is to sort these characters by decreasing frequency, and output a readable report, like this. (Note that these are not the correct values for our class, they are from a previous class.) Also, since we don’t want to distinguish here between upper and lower-case occurrences of a letter, convert to upper-case the entire string of names after reading it from the file.

A : 54
E : 45
N : 41
R : 39
: 37 #(this is how the tabs will output)
: 36 #(can you guess what’s going on here?)
I : 30
S : 22
O : 20
T : 19
M : 19
L : 17
G : 15
H : 14
U : 13
C : 12
B : 10
P : 10
V : 8
Z : 8
Y : 8
D : 8
K : 8
J : 5
W : 4
X : 4
- : 2
Q : 1

[~10 minutes] But how will we achieve this sorting? We learned in class today that dictionaries are not ordered as lists are... they can be thought of as a “bag” of key-value pairs, there is no underlying linear structure. So dictionaries cannot be sorted! We will therefore have to “export” the data into a list.

i. Refer to the table on page 371 of Zelle to learn that python has a function that returns the contents of a dictionary as a sequence of key-value tuples: the <dict>.items() function.

ii. Call this function on the dictionary we created in letterFreq.py to extract the data into a list so we can sort it. Note that a sequence in Python is not exactly the same thing as a list. To convert a sequence seq to a list, we simply do: list(seq), which “casts” the generic “sequence” into an official list data type. (We did this in class so that we could then do list operations on the sequence, like sorting!)

iii. Print the list of these key-value pairs to make sure you’ve achieved your goal so far.

☺ Get check 7 ☺

iv. [~15 minutes] Now that you have the dictionary data in a list, you can sort the list based on frequency value. Hint: we practiced sorting lists of tuples in class with our highScores.py module. This is very similar to that.

☺ Get check 8 ☺
v. [~15 minutes] Output the resulting sorted list in a nicely printed format as shown above. Also explain to the TAs why in the print out there are some lines with strange blanks/spacing.

Get check 9 ☺

Extra time? Complete the following extra credit checks!

**Bonus A:** More of programming Exercise 5 from Chapter 11.

i. Part d

ii. Part e

**Bonus B:** Programming Exercise 10 from Chapter 11 (Sieve of Eratosthenes).

**Bonus C:** Programming Exercises 11 and 12 from Chapter 11. (Censorship: 2 separate bonus checks.)