6.189 Day 4

Readings

*How To Think Like A Computer Scientist*, chapters 7 and 8

*6.01 Fall 2009 Course Notes* page 27 - 29 (‘Lists’ and ‘Iterations over lists’; ‘List Comprehensions’ is optional); sections 3.2 - 3.4 (up to 3.4.6).

Exercise 4.0 – Recollections

Recall that strings and tuples are *immutable* datatypes, while lists are *mutable* datatypes. What does this mean?

Exercise 4.1 – String Operations

String operators might be a little less intuitive than those on numbers. This exercise will give you a chance to practice those. Given the following variables:

```python
look = 'Look at me!
now = 'NOW'
```

What are the values of the following expressions? Try to guess on your own before using your interpreter; turn in your answers to this problem.

1. `look[:4]`
2. `look[-1]`
3. `look*2`
4. `look[-1] + now + look[-1]`
5. `now[1]`
6. `now[4]`
7. `look*2 + look[-1] + now + look[-1]`
Exercise 4.2 – Working With Lists

Look at the function below and test it out. Save the code in Day4.py.

```python
def sum_all(t):
    # t is a list of numbers
    total = 0
    for x in t:
        total += x
    return total
```

The function takes a list of numbers as a parameter (note how we specify, with a comment, what type of parameter to use), and returns the sum of all the numbers.

Now make a new function cumulative_sum that modifies sum_all so that instead of returning the sum of all the elements, it returns the cumulative sum; that is a new list where the i\textsuperscript{th} element is the sum of the first i + 1 elements from the original list. For example, the cumulative sum of \([1, 2, 3]\) is \([1, 3, 6]\).

Exercise 4.3 – Report Card with GPA

Write a function report_card where the user can enter each of his grades, after which the program prints out a report card with GPA. Remember to ask the user how many classes he took. Example output is below.

```python
>>> report_card()
How many classes did you take? 4
What was the name of this class? English
What was your grade? 94
...
REPORT CARD:
English - 94
Math  96
Science 91
Social Studies - 88
Overall GPA 92.25
```

**Hints:** You’ll want to use a for loop, and you’ll probably want to keep track of names and grades separately; there are a couple ways to do this. Remember, add to lists with `my_list.append(elt)`.

Exercise 4.4 – Pig Latin

Write a function pig_latin that lets the user enter in some English text, then converts the text to Pig-Latin. To review, Pig-Latin takes the first letter of a word, puts it at the end, and appends “ay”. The only exception is if the first letter is a vowel, in which case we keep it as it is and append “hay” to the end.

E.g. “hello” → “ellohay”, and “image” → “imagehay”.

It will be useful to define a list or tuple at the top of your code file called VOWELS. This way, you can check if a letter x is a vowel with the expression `x in VOWELS`.

It’s tricky for us to deal with punctuation and numbers with what we know so far, so instead, ask the user to enter only words and spaces. You can convert their input from a string to a list of strings by calling `split` on the string; also, you can use `lower` to make a string all lowercase:
>>> phrase = 'My name is John Smith'
>>> word_list = phrase.split()
>>> print word_list
['My', 'name', 'is', 'John', 'Smith']
>>> lowercase_phrase = phrase.lower()
>>> print lowercase_phrase
'my name is john smith'

Using this list, you can go through each word and convert it to Pig-Latin. Also to get a word except for the first letter, you can use word[1:].

Hints: It will make your life much easier - and your code much better - if you separate tasks into functions, e.g. have a function that converts one word to Pig-Latin rather than putting it into your main program code.

Optional extensions: Once you have your program working, make it interactive such that it keeps translating phrases into pig latin until the user enters in the phrase stop. Or, you could try and deal with punctuation by looking for it within a string and moving it to the end of the word (the solutions I wrote only handle commas, periods, !, ?, : and ; as they are pretty simple to handle).

Exercise 4.5 – [Tuples] Collision Detection of Balls

Many games have complex physics engines, and one major function of these engines is to figure out if two objects are colliding. Weirdly-shaped objects are often approximated as balls. In this problem, we will figure out if two balls are colliding. You’ll need to know how to unpack tuples; look online or ask an LA if this is confusing to you.

We will think in 2D to simplify things, though 3D isn’t different conceptually. For calculating collision, we only care about a ball’s position in space and its size. We can store position with its center x-y coordinates, and we can use its radius for size. So a ball is a tuple of (x, y, r).

To figure out if two balls are colliding, we need to compute the distance between their centers, then see if this distance is less than the sum of their radii. If so, they are colliding.

Write a function b_collide that takes two balls as parameters and computes if they are colliding. Then call the function with two sets of balls. The first set is (0, 0, 1) and (3, 3, 1); these should not be colliding. The second set is (5, 5, 2) and (2, 8, 3); these should be colliding.

Exercise 4.6 – Environment Diagrams

For this exercise, you’ll be asked to work through problems in the 6.01 course notes. Write your answers to these neatly, and in a reasonably sized font (your LAs have to read 130 of these!), on a separate sheet of paper and turn in tomorrow stapled to your code. The last two are tricky; don’t hesitate to ask for help, and remember that this is a pass/fail class. We only care that you put your best effort into solving each problem- not that it’s 100% correct!

1. Variables, page 35: Exercise 3.1
2. Lists, page 37: Exercises 3.2, 3.3, 3.4
3. List mutation, page 41: Exercises 3.6, 3.7
4. Procedures, page 49: Exercise 3.10
Exercise 4.7 – List Comprehensions

OPTIONAL!! (but recommended) List comprehensions follow naturally from set builder notation and lambda calculus. They are very cool and make your life a lot easier, but they aren’t critical to understanding how to program. Do this problem if you want to challenge yourself; we will not be covering list comprehensions in class otherwise, but you will see them in 6.01.

Read about list comprehensions in chapter 2 of the 6.01 course notes; the Wikipedia article on them are good, and this site is concise and good: http://www.se netix.de/olli/Python/list_comprehensions.hawk - or just Google “Python list comprehensions” and find a site that makes sense to you.

Problems: Put these exercises in Day4.py. Some are trickier than others; move on if you’re stumped on one and come back to it later!

1. Write a list comprehension that prints a list of the cubes of the numbers 1 through 10.

2. Write a function that takes in a list of elements of different types and uses a list comprehension to return all the elements of the list of type int. Note: The function isinstance will be of help here. Google “Python isinstance” and see if you can figure out what it does, or type help(isinstance) at the Python shell.

3. Write a function that takes in a string and uses a list comprehension to return all the vowels in the string.

4. Run this list comprehension in your prompt:

[x+y for x in [10,20,30] for y in [1,2,3]]

Figure out what is going on here, and write a nested for loop that gives you the same result. Make sure what is going on makes sense to you! This list comprehension does not do what you might initially think it would; hopefully the next problem will illustrate why a two variable list comprehension would do this.

5. Write a list comprehension which solves the equation $y = x^2 + 1$. Your solution should print out a list of $(x, y)$ tuples; use the domain $x \in [-5, 5]$ and the range $y \in [0, 10]$. 