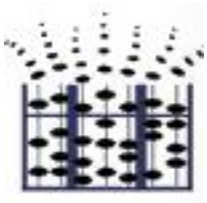

CIS 211
Spring 2014

Project 7: Functional Programming

Due Friday May 30
Upload via Blackboard by 11:00 P.M.

Reading

Introduction to Computing Using Python, Section 12.3.

Code Snippets

The class web sites have some Python code you can use for this project:

- `in_class.py` (functions created during lecture on May 21)
- `test_fp.py` (unit test program)

Programming Projects

The project this week is to write several small Python functions that each use one or more functional programing (FP) constructs. The grading standards for this project are:

- 50% for correctness (the function produces the expected results)
- 50% for programming style (you use FP constructs where appropriate, plus other good programming style)

You are allowed to write additional "helper functions" if you think it will make your code easier to understand.

Put all of your functions in a single file named `fp.py`.

1. Write a function named `codes` that returns a list of numeric codes for the characters in a string. If `ch` is a character (i.e. a 1-letter string) the builtin function named `ord` will return its character code:

   ```
   >>> ord('A')
   65
   ```

   ```
   >>> codes('Aloha')
   [65, 108, 111, 104, 97]
   ```

2. Write a function named `vowels` that will return a string made from the vowels (letters A, E, I, O, and U) in a string:

   ```
   >>> vowels('Aloha')
   'Aoa'
   ```

   Note that the vowels are returned in order, and that case is preserved.

3. Write a function named `tokens` that will split an input string into individual words and remove the punctuation marks from the ends of the words. The result returned from the function should be a `map` object; to see the individual words pass this object in a call to `list`:

```
>>> m = tokens("Buy now! Only $29.95.  Wait, there's more!!")
>>> m
<map object at 0x10260cf90>
>>> list(m)
['Buy', 'now', 'Only', '29.95', 'Wait', "there's", 'more']
```

   Hint: you can use the `strip_punctuation` function demonstrated in class as a helper function.

4. Write a function named `numbers` that will use your `tokens` function to break a line into words and then return the tokens that contain nothing but digits:

```
>>> numbers('Want all 5?  Get them now for only $99!')
['5', '99']
```

5. Write a function named `sq_ft` that will compute the total area of a house by adding up the areas of the individual rooms. The argument passed to `sq_ft` will be a file containing the dimensions of the rooms. To test your function you can download a file named `house.txt`. This is the expected result:

```
>>> sq_ft('house.txt')
1539.0
```

   The file will have one line per room, where each line has the room name and two numbers representing the width and depth the room, *e.g.*

```
kitchen 10 14
laundry 4 5
...
```

   Suggestion: define a class named Room, and define the constructor so it will initialize an object from a string with the name and dimensions:

```
>>> r1 = Room('kitchen 10 14')
```

   Add a method named `area` that will compute the area of a room:

```
>>> r1.area()
140.0
```

   Now your `sq_ft` function can create a Room object from the description of each line in the input file and the `sq_ft` function call `area` to compute the area of each room.

Unit Tests

If you want to check the correctness of your `codes`, `vowels`, `tokens`, and `numbers` functions with the unit test program download the program from the web site and run it with this shell command:

```
python3 -m unittest test_fp.py
```

If all your functions are working the output will be

```
....
Text
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```

Extra Credit Ideas

- Write a function name `checksum` that will compute the bitwise exclusive OR of all the binary codes in a string. For example, here are the individual codes of the letters in 'Aloha' (displayed in hexadecimal):

```
>>> for x in 'Aloha':
...   print("{:0X}".format(ord(x)))
...
41
6C
6F
68
61
```

  Here is the checksum, also displayed in hex:

```
>>> print("{:0X}".format(checksum('Aloha')))
4B
```

What to Turn In

Documentation Write a short description (two or three paragraphs total) of what you did for this project. The documentation should be in a file named `writeup` with an extension that identifies the file format (.doc for Microsoft Word, .pdf for Adobe PDF, .txt for plain text, .rtf for rich text format).

If you did anything extra for this project describe it in a separate paragraph at the end of your writeup.

Create a package (tar or zip format) that includes your writeup and your fp.py file and upload the package via Blackboard.