Rampup Session Exercises

1 Dictionaries

1.1 Simple Formatting

Write code that takes dictionary with Keys as student numbers (int), and values as names (str). You should print out all records, nicely formatted.

```
>>> record = {1234: 'Tony Stark', 1138: 'Steve Rogers'}
```

Your output should be:

```
Tony Stark (#1234)
Steve Rogers (#1138)
```

Hint:

```
print('{} (#{})'.format(var1, var2))
```

1.2 Loops and Dictionaries

Write code that takes open file open_file, and creates a dictionary with key/value pairs of each word and the number of occurrences of that word. (a word is a white-space delimited token, and can have punctuation)

```
>>> open_file = io.StringIO('a b a a c c a.')
```

(Think of an open file as a list of strings containing each file line (can iterate by line))

Your dictionary should be:

```
{'a': 3, 'b': 1, 'a.': 1, 'c': 2}
```

Hints:

```
in
str.split
```

2 Functions

2.1 Simple Function Reuse

The following function takes in two strings: a first and last name, and returns them as a string in the format: last_name, first_name

```
def format_name(first_name, last_name):
    """
    @type first_name: str
    @type last_name: str
    @rtype: str

    >>> format_name('David', 'Cohen')
    'Cohen, David'
    """
    return last_name + ', ' + first_name
```

Write a function that takes in three strings: a first name, last name, and phone number, and returns a string in the format: last_name, first_name: phone_number. Call the above format_name function in your own function. Make sure you include a docstring for your function!

```
>>> print(to_listing('Julianna', 'Paprakis', '416-555-5555'))
```

Your output should be:

```
    Paprakis, Julianna: 416-555-5555
```

3 Memory & Mutability

3.1 Variable Assignment

Write the values of each variable once the following piece of code is done executing:

```
    >>> a = [0, 1, 2, 3, 4]
    >>> b = a
    >>> b[2] = 10
    >>> c = a[1]
    >>> c = 20
    >>> d = [5, 6, 7, 8]
    >>> d = b
```

Values:

a _____ b _____ c _____ d _____

4 Testing the Code

Use the most_employees function for this section. Assume it is saved in a file 'employees.py'

```
def most_employees(companies_with_employees):
    """
    @type companies_with_employees: {str: [str]}
    @rtype: [str]

    Precondition: companies_with_employees is not empty

    Return the company (or companies) with the most employees.

    >>> most_employees({'Walmart':['Trish', 'Bob', 'Sam'], 'Subway':['Joe', 'Anne']}
    ['Walmart']
    """
```

4.1 Unit Tests

Complete the following two test methods for most_employees:

```
import unittest
import employees

class TestMostEmployees(unittest.TestCase):
    def test_most_employees_one_item(self):
        """ Test most_employees with a dictionary of length 1."""









    def test_most_employees_mutation(self):
        """ Confirm that most_employees does not mutate the dict it is given."""
```

4.2 Doctests

Write some more appropriate Doctests to add to the existing docstring for most_employees

```
>>>
```