Question 1.  [7 marks]

Each subquestion on this page has a piece of code that is supposed to work as described in the comment statement but has a small part missing. For each one, add the missing part inside the box. Your solution must follow the instructions in the comment statement. Each subquestion is independent.

Part (a)  [1 mark]

d = {'k1': 'no', 'k3': 'nope'}
# Add one key-value pair to d so that the code below the box prints 'YES'.

d['k2'] = 'YES'
print d['k2']

Part (b)  [1 mark]

d = {}
# Add one key-value pair to d so that the code below the box prints 'Computer Science'.

d['Science'] = 'Computer'
for item in d:
    print d[item], item

Part (c)  [1 mark]

d = {(1, 2): [3, ['Anonymous', 5]]}
# Write a print statement that outputs only 'Anonymous' and does it by accessing
# the correct piece of d.

print d[(1,2)][1][0]

In the box beside each piece of code below, write its output. If it would generate an error, say so, and give the reason for the error.

Part (d)  [2 marks]

L1 = ['CSC', [108, 148, 165]]
L2 = L1[:]
L1[1].append(209)
print L1
print L2

Part (e)  [2 marks]

L1 = ['CSC', [108, 148, 165]]
L2 = L1[:]
L2[1] = ['209', '207', '263']
print L1
print L2
Question 2. [7 marks]

Suppose we are keeping track of who is working with whom on a course assignment. We could represent the
groups using a nested list of student numbers like this: [[2, 9], [4], [3, 1]]. (Here we use one-digit
student numbers to make the example easier to read.) In this example, we have two groups of two students,
and one group of one student.

Part (a) [1 mark]

Consider the following function.

```python
def no_group(group_list, class_list):
    '''Return a list containing the student number of everyone in list class_list
    who is not in any group according to group_list.'''
```

Write a call to the function that should return a list of length 2, and involves a class list of 6 students.

Solution: There are many. Here is one possibility.

```python
no_group([[2, 4], [5], [6]], [1, 2, 3, 4, 5, 6])
```

Part (b) [6 marks]

Now write the function. You do not need to repeat the `def` line or the docstring.

Solution:

```python
ungrouped = []
for student in class_list:
    this_student_grouped = False
    for group in group_list:
        if student in group:
            this_student_grouped = True
    if not this_student_grouped:
        ungrouped.append(student)
return ungrouped
```
Question 3.  [5 marks]

Write the following function according to its docstring.

Hint: Use str.find

def nth(small, big, n):
    '''Return the index of the nth non-overlapping occurrence of string small within string big.
    For example, nth('oo', 'A Coool pool look', 1) returns 3 and
    nth('oo', 'A Coool pool look', 2) returns 9. Return -1 if there are fewer than n occurrences.
    int n is > 0.'''

Solution:

    # start the location here so once we add len(small) first search starts at 0
    location = - len(small)
    for i in range(n):
        location = big.find(small, location + len(small))
        if location == -1:
            return -1
    return location