Computer Science 1 — CSCI-1100  
Spring Semester 2015  
Exam 2 Overview and Practice Questions

SOLUTIONS
The following are the solutions to the practice problems. Please be aware that there may be more than one way to solve a problem and so your answer may be correct despite being different from ours.

Questions
1. Assume \( v \) is a list containing numbers. Write Python code to find and print the highest two values in \( v \). If the list contains only one number, print only that number. If the list is empty, print nothing. For example, if we assigned

\[
v = [7, 3, 1, 5, 10, 6]\]

then the output of your code should be something like

7 10

If we are given that

\[
v = [7]\]

then the output of your code should be

7

Solution:

```python
v.sort()
if len(v) == 1:
    print v[0]
elif len(v) > 1:
    print v[-2], v[-1]
```

2. Consider a simplified version of the Yelp lab data, where just the name of the restaurant, the type of restaurant, and the ratings are provided. Assume these values have already been read into a list of lists of the form below:

```python
restaurants = [ ['Acme', 'Italian', 2, 4, 3, 5],
                ['Flintstone', 'Steak', 5, 2, 4, 3, 3, 4],
                ['Bella Troy', 'Italian', 1, 4, 5] ]
```

Write a segment of Python code that prints all Italian restaurants in the \texttt{restaurants} list that have no ratings of value 1 and at least one rating of value 5. In the above example, \textit{Acme} would be printed in the output, but \textit{Flintsone} and \textit{Bella Troy} would not. \textit{Flintsone} is not Italian and \textit{Bella Troy} has a 1 rating. Your code should work for any legal version of \texttt{restaurants}.

Solution:
for rest in restaurants:
    if rest[1] == 'Italian' and (1 not in rest[2:]) and (5 in rest[2:]):
        print rest[0]

3. Continuing with the Yelp lab, assume that you have the code

    in_file = open('yelp.txt')
    for line in in_file:
        p_line = parse_line(line)
        print p_line

and that you parse_line will return a list that looks like

    ["Meka's Lounge", 42.74, -73.69, "Bars", [5, 2, 4, 4, 3, 4, 5], 3.857142857142857 ]

Modify the for loop above to create a list called high that stores the names of all restaurants that have an average rating of at least 4.0. You do not have to print high.

Solution: Don’t forget to initialize high to be the empty list:

    in_file = open('yelp.txt','r')
    high = []
    for line in in_file:
        p_line = parse_line(line)
        if p_line[5] >= 4.0:
            high.append( p_line[0] )

4. In the game of chess you can often estimate how well you are doing by adding the values of the pieces you have captured. The pieces are Pawns, Bishops, Knights, Rooks and Queens. Their values are

    P - (P)awn, value = 1
    B - (B)ishop, value = 3
    K - (K)night, value = 3
    R - (R)ook, value = 5
    Q - (Q)ueen, value = 9

Write a Python function called chess_score that takes a single string as an argument and returns the combined values represented by the pieces in the string. You may assume that only 'P', 'B', 'K', 'R', and 'Q' appear in the string. You may not use any if statements and you may not use any loops. As an example,

    print chess_score('BQBP')

should output the value 16 because there are 2 Bishops (3 points each), 1 Queen (9 points each), and 1 Pawn (1 point each).

Solution:

    def count_pieces(pieces):
        pawns = pieces.count('P')
        bishops = pieces.count('B')
        knights = pieces.count('K')
        rooks = pieces.count('R')
        queens = pieces.count('Q')
        return pawns + 3*bishops + 3*knights + 5*rooks + 9*queens;
5. Write a function called `in_between` that takes a list as a parameter and returns a new list that contains only the numbers that are greater than the first value in the list (at index 0) and less than the last value in the list. As an example

```
print in_between([2, 3, 6, 5, 1, 6])
```

should output

```
[3, 5]
```

**Solution:**

```python
def in_between(v):
    t = []
    for x in v:
        if v[0] < x and x < v[-1]:
            t.append(x)
    return t
```

6. Write a function that takes as a parameter a list, `v`. Using a **while loop**, it should count and return the number of values in `v` that are greater than the previous value in the list. For example, if `v` is

```
[ 21.5, 32.0, 16.5, 18.7, 33.2, -1.1, -0.5, 6.7, 19.4, 14.2 ]
```

the returned value should be 6 because of the 32.0, 18.7, 33.2, -0.5, 6.7 and the 19.4. After you are finished, rewrite the solution to using a **for loop**.

**Solution:** While loop version

```python
def count_misordered_while(v):
    i = 1
    count = 0
    while i < len(v):
        if v[i-1] < v[i]:
            count += 1
        i += 1
    return count
```

For loop version

```python
def count_misordered_for(v):
    count = 0
    for i in range(1, len(v)):
        if v[i-1] < v[i]:
            count += 1
    return count
```
7. Write the output for each of the following sections of Python code:

**Part a**

```python
for i in range(10,1,-2):
    print i
```

**Part b**

```python
pickle = 2
for j in range(2):
    for k in range(2):
        pickle = pickle + 2
        pickle = pickle * 2
    print pickle
```

**Part c**

```python
i = 0
while i<10:
    if i < 5:
        i += 5
    elif i%4 == 0:
        i -= 1
    else:
        i += 3
    print i
```

**Solution:** Please test these for yourself.

8. You are given a file that contains, on each line of input, three integers separated by commas. Write a Python program that sums all of the first integers, the second integers, and the third integers, outputting the resulting sums all on one line, separated by commas. As a simple example, if the input is

```
2, 5, 7
3, 6, 10
1, 2, -3
2, 4, 1
```

Then the output should be

```
8, 17, 15
```

**Solution:** The following solution does not require the creation of three lists to store the values prior to summation.

```python
sums = [0,0,0]
for line in open('infile.txt'):
    dat = line.split(',')
    sums[0] += int(dat[0])
    sums[1] += int(dat[1])
    sums[2] += int(dat[2])

print "%d, %d, %d" %tuple(sums)
```
The second solution does create three lists. The first solution is slightly preferred because of this, although you would not lose points on a test unless we explicitly disallowed the creation of the three lists.

```python
firsts = []
seconds = []
thirds = []

for line in open('infile.txt'):
    dat = line.split(',')
    firsts.append(int(dat[0]))
    seconds.append(int(dat[1]))
    thirds.append(int(dat[2]))

print "%d, %d, %d" % (sum(firsts), sum(seconds), sum(thirds))
```

9. Write Python code to generate the following ranges

(a) (100, 99, 98, ..., 0)
(b) (55, 53, 51, ..., -1)
(c) (3, 5, 7, 9, ..., 29)
(d) (-95, -90, -85, ..., 85, 90)

**Solutions:**

# Part a
range(100, -1, -1)

# Part b
range(55, -2, -2) # could have 55, -3, -2 as well

# Part c
range(3, 30, 2) # 30 could be replaced by 31

# Part d
range(-95, 91, 5) # 91 could be replaced by any of 91..95

10. Write a **while** loop to add all of the numbers in a list `v` until it reaches a negative number or until it reaches the end of the list. Store the sum in the variable `result`. Your code should work for any version of `v` containing only numbers. For example, the value of `result` should be 25 after the loop for both of the following lists:

   v = [ 10, 12, 3, -5, 5, 6 ]

   v = [ 0, 10, 3, 6, 5, 1 ]

**Solutions:**

```python
i = 0
result = 0
while i < len(v):
    if v[i] < 0:
        break
    result += v[i]
    i += 1
```

Version combining logic
i = 0
result = 0
while i < len(v) and v[i] >= 0:
    result += v[i]
i += 1

11. Write Python code that takes a list of numbers, v, and outputs the **positive** values that are in v in increasing order, one value per line. If there are no positive values, then the output should be the string 'None'. You may assume there is at least one value in the list. As an example,

\[ v = [17, -5, 15, -3, 12, -5, 0, 12, 22, -1]\]

Then the output of your code should be

12
12
15
17
22

As a second example, if

\[ v = [-17, -5, -15, -3, -12, -5, 0, -12, -22, -1]\]

then then output should be just

None

**Solution:**

```python
v.sort()
if v[-1] <= 0:  # largest number is not positive
    print 'None'
else:
    for x in v:
        if x > 0:
            print x
```

Here is a second solution that involves a second list and two loops:

```python
pos = []
for x in v:
    if x > 0:
        pos.append(x)
pos.sort()
if len(pos) == 0:
    print 'None'
else:
    for x in pos:
        print x
```

12. What is the output of the following operations:


```python
>>> mylist = [1,4,8,12,6]
>>> x = mylist.sort()
>>> print x

>>> mylist = [1,4,8,12,6]
>>> slice1 = mylist[2:4]
>>> slice1[0] = 20
>>> print slice1

>>> print mylist

Solution: Please test them for yourself.

13. Suppose you are given a list of integers, v. In other words, v is already provided with a value in your program, such as

```
v = [ 126, -1, 347, -14, 14, 15, 29, 12, 9, 8 ]
```

Write Python code that:

- asks the user for one integer, reads it in and stores it in the variable x0,
- asks the user for a second integer, reads it in and stores it in the variable x1, and
- prints all values in v that are above x0 and below x1. For example, if the user gave the input -14 and 15 then the output should be

```
-1
14
12
9
8
```

Note: if the input was 15 and 5, then the output is empty! There is no number greater than 15 and less than 5.

Solution:

```python
x0 = int(raw_input("Enter x0 =>"))
x1 = int(raw_input("Enter x1 =>"))
for y in v:
    if x0 < y and y < x1:
        print y
```
14. Using the methods from the images lab, do the following:

(a) Resize an image so that it is the same height but half the width.
(b) Take an image, and create a new image that repeats this image \(m\) times horizontally and \(n\) times vertically.

**Solution:** We'll let you figure out and test these for yourself!

15. Write a Python `for` loop to print out the values from the list \(v\) that are positive (0 is NOT a positive number).

**Solution:** This assumes that list \(v\) already exists

```python
for x in v:
    if x > 0:
        print x
```

16. What is the output of the following code?

```python
L1 = [0,1,2,4,1,0]
s1 = set(L1)
L1.pop()
L1.pop()
L1.pop()
L1[0] = 5
s1.add(6)
s1.discard(1)
print L1
for v in sorted(s1):
    print v
```

**Solutions:** Please generate them for yourself.

17. \(v\) is a list containing an **odd** number of values. Write Python code (no need for a whole function) to compute and print the average of the first, middle and last values in \(v\), accurate to two decimal places. For example, if we assigned

\[
v = [ 6, -5, 3.1, 7, 12, -5, 9 ]
\]

then the output of your code should be

**Average:** 7.33

Note that your code needs to work for any odd-length \(v\).

**Solution:**

```python
mi = len(v) / 2
avg = ( v[0] + v[mi] + v[-1] ) / 3.0
print "Average: %.2f" % avg
```
18. Write a Python program that copies an input file called `in.txt` to an output file called `out.txt`. All lines from `in.txt` should be copied to `out.txt` except the first line and the last line. Try to think of some different ways to approach this.

**Solution:** The first version stores the lines in a list and then outputs all but the first and last values of the list:

```python
lines = []
for line in open("in.txt"):
    lines.append(line)

outf = open("out.txt","w")
for i in range(1,len(lines)-1):
    outf.write(lines[i])
```

The second version skips the first line and then delays the output of each line until the next pass through the loop. In this way, the loop ends before the last line of the file is written.

```python
inf = open("in.txt")
outf = open("out2.txt", "w")
skip_line = inf.readline()  # read in the first line, this is never output
prev_line = inf.readline()  # read in and remember the "previous" line

for line in inf:
    outf.write(prev_line)
    prev_line = line
```
19. What is the output of the following program?

```
def spam(a1,b1,a2,b2):
    if (a1 == a2) and (b1 > b2):
        return 1
    else:
        return 0

def egg(a1,b1,a2,b2):
    if (a1 > a2) and (b1 == b2):
        return 0
    else:
        return 1

a1 = 3
b1 = 4
a2 = 6
b2 = 4

print spam(a2, b2, a1, b1)
print egg(a1, b1, a2, b2)
c = spam(a1, b2, a2, b1)
print c
c += egg(a1, b2, a2, b1)
print c
```

Solution: Please test them for yourself.

20. (16 pts) Write a function called `copy_half` that takes the name of two files as arguments. The function should copy the first, third, fifth, etc. lines (i.e. odd lines only) from the first file to the second file. For example, if the file names are `in.txt` and `out.txt` and if `in.txt` contains

```
starting line
not this line
middle line is here
    skip this line too
    I like this line
```

then after the call

```
copy_half( 'in.txt', 'out.txt' )
```

the file `out.txt` should contain

```
starting line
middle line is here
    I like this line
```

Solution:
def copy_half( in_name, out_name ):
    inf = open(in_name)
    outf = open(out_name,'w')
    i = 0
    for line in inf:
        i += 1
        if i%2 == 1:
            outf.write(line)
    outf.close()

Solution using a list

def copy_half( in_name, out_name ):
    inf = open(in_name)
    lines = []
    for line in inf:
        lines.append(line)

    outf = open(out_name,'w')
    for i in range(0,len(lines),2):
        outf.write(lines[i])
    outf.close()
21. Write a program that extracts all of the names of functions from the Python program stored in a file called prog.py. For example, if prog.py contains

```
def a1 (y):
    def b__ (x):
        return x*x
    return b__ (y) * 3
def _a134():
    print "This is _a134"
print a1(3)
```

then your program should output

```
a1
b__
_a134
```

Write this in two separate parts:

(a) Write a function called `name_of_function` that takes a string representing a line of input from a Python program. If there is a function def on the line, it should return a string that contains the name of the function. If there is no function definition on the line, it should return the empty string, '', which is length 0. For example,

```
name_of_function("  def b__ (x):")
```

should return 'b__'. Hint: string method `isalnum` returns True if the string (could be just a single character) contains all letters and digits.

Solution:

```
def name_of_function( in_line ):
    in_line = in_line.strip()
    if not in_line.startswith('def '):
        return ''
    else:
        i = 4
        while in_line[i].isspace():
            i+= 1
        name = ''
        while in_line[i].isalnum() or in_line[i] == '_':
            name += in_line[i]
            i += 1
        return name
```

A small variation:

```
def name_of_function( in_line ):
    in_line = in_line.strip()
    if not in_line.startswith('def '):
        return ''
    else:
        i = 4
        while in_line[i].isspace():
            i+= 1
        start = i
        while in_line[i].isalnum() or in_line[i] == '_':
            i += 1
        return in_line[start:i]
```
(b) Write the rest of the program, making use of name_of_function.

```python
for line in open("prog.py"):
    fname = name_of_function(line)
    if len(fname) > 0:
        print(fname)
```

22. Write a segment of code that outputs the indices and the values in a list, v. For example, the output for the list

```python
v = [ 15, 'bicycle', 'xray', -3.14 ]
```

should be

0: 15
1: bicycle
2: xray
3: -3.14

**Solution:**

```python
for i in range(len(v)):
    print "%d: " %i, v[i]
```

23. Write a function called `count_in_range` that counts and returns the number of values in a list that are between the values stored in parameters `x0` and `x1`. (Values equal to `x0` or `x1` should not be included in the count.) The function begins with

```python
def count_in_range( v, x0, x1):
```

**Solution:**

```python
def count_in_range( v, x0, x1):
    count = 0
    for x in v:
        if x0 < x and x < x1:
            count += 1
    return count
```
24. Write a segment of code that reads integers from a file called test2.txt and stores the positive values in one list, the negative values in a second list, and skips blank lines and zeros. The order of the values in each list should match the order of the input. Each line of input will contain either spaces or spaces and an integer. For example, if test2.txt contains

```
11
-3
5
0
```

Then after your code, the list P should be [ 11, 5 ] and the list N should be [ -3 ].

**Solution:**

```python
P = []
N = []
for s in open("test2.txt"): 
s = s.strip()
if len(s) > 0:
i = int(s)
if i > 0:
P.append(i)
elif i < 0:
N.append(i)
```

25. Give the output of each of the following

(a) 
i = 4
L = [ 0, 12, 3, 5, 2, -1 ]
while 0 <= i and i < len(L):
    if L[i] < 0:
        break
    else:
        i = L[i]
print i, L[i]

**Solution:**

```
2 3
3 5
5 -1
```

(b) 
tough = 2
for i in range(2):
s = 1
    for j in range(i, tough):
s *= tough
print s
print tough
tough = s
print tough
```
26. Suppose time is represented by a tuple containing the hour and minute, both as integers. (We are ignoring AM and PM in this problem.) For example,

\[
t_1 = (12, 3)
\]

is 12:03. Write a function that takes two time tuples and returns True if and only if the first tuple represents an earlier time in a day than the second. As examples,

\[
>>> \text{earlier\_than}( (12,3), (1,5) )
\]

True

\[
>>> \text{earlier\_than}( (5,45), (1,5) )
\]

False

\[
>>> \text{earlier\_than}( (5,45), (5,45) )
\]

False

\textbf{Solution:}

\textbf{Version 1:}

\[
\begin{align*}
def \text{is\_earlier\_than}( t, u ) : \\
    \text{return } t[0] \% 12 \times 60 + t[1] < u[0] \% 12 \times 60 + u[1]
\end{align*}
\]

\textbf{Version 2:}

\[
\begin{align*}
def \text{is\_earlier\_than}( t, u ) : \\
    \text{if } t[0] == 12 \text{ and } u[0] != 12: \\
        \text{return True} \\
    \text{elif } t[0] != 12 \text{ and } u[0] == 12: \\
        \text{return False} \\
    \text{else:} \\
        \text{return } t[0] < u[0] \text{ or } (t[0] == u[0] \text{ and } t[1] < u[1])
\end{align*}
\]

\textbf{Version 3:}

\[
\begin{align*}
def \text{is\_earlier\_than}( t, u ) : \\
    \text{if } t[0] == 12 \text{ and } u[0] != 12: \\
        \text{return True} \\
    \text{elif } t[0] != 12 \text{ and } u[0] == 12: \\
        \text{return False} \\
    \text{elif } t[0] < u[0]: \\
        \text{return True} \\
    \text{elif } t[0] > u[0]: \\
        \text{return False} \\
    \text{elif } t[1] < u[1]: \\
        \text{return True} \\
    \text{else} \\
        \text{return False}
\end{align*}
\]

27. Suppose a list of words in alphabetical order has been assigned to the variable called \texttt{words}. For example, we might have the assignment
words = [ 'aardvark', 'abaka', 'expedite', 'experience', 'shoetrees', 'tastetest', 'test' ]

Write code to find and output the first and the last string in words that start and end with the same letter and are at least 8 characters long. You may assume that at least one word in words satisfies this condition. You may write a function if you wish. For the above example, the output should be

expedite
tastetest

**Solution:** Version using two for loops:

```python
for w in words:
    if w[0] == w[-1] and len(w) >= 8:
        print w
        break

for w in words.reverse():
    if w[0] == w[-1] and len(w) >= 8:
        print w
        break
```

Version using indexing

```python
n = len(words)
for i in range(0,n):
    if words[i][0] == words[i][-1] and len(words[i]) >= 8:
        print words[i]
        break

for i in range(n-1,-1,-1):
    if words[i][0] == words[i][-1] and len(words[i]) >= 8:
        print words[i]
        break
```

Version using while loop

```python
n = len(words)
i = 0
while words[i][0] != words[i][-1] or len(words[i]) < 8:
    i += 1
print words[i]

i = n-1
while words[i][0] != words[i][-1] or len(words[i]) < 8:
    i -= 1
print words[i]
```

These loops are simple because of the assumption that at least one word satisfying the condition is there...

28. Recall that our Sudoku board from lab was represented as a list of lists. For example,

```python
bd = [ ['1', '.', '.', '.', '.', '2', '7', '3', '7'],
      [ '.', '6', '.', '.', '1', '4', '8', '9'],
      [ '.', '5', '.', '.', '.', '.', '1', '9'],
      [ '.', '1', '4', '3', '.', '.', '5', '9'],
      [ '.', '4', '1', '9', '.', '.', '3', '7'],
      [ '2', '8', '.', '.', '4', '.', '.', '6'] ]
```
Your goal in this problem is to write a function called `min_and_max_filled` that outputs the minimum and maximum number of values that are filled in the rows. For the above example, the output values for `bd` would be 2 and 5 because rows 3 and 5 both have just two values ('9' and '4' in row 3 and '1' and '4' in 5), while row 1 has 5 values ('6', '5', '1', '4' and '8'). Solve this in 2 parts:

(a) Write a short function called `num_filled` that takes a single list from a row, such as

```
[ '.', '6', '.', '.', '.', '5', '1', '4', '8']
```

and returns the number of values filled in (in this case 5). This can be done using just two lines of code, without a `for` loop. If you need to use a `for` loop, slightly less credit will be given.

**Solution:**

Version 1:

```
def num_filled( row ):
    return 9 - row.count('.
```

Version 2:

```
def num_filled( row ):
    count = 0
    for c in row:
        if c != '.
            count += 1
    return count
```

(b) Now write function `min_and_max_filled`, which takes a Sudoku board as its only argument. It must call function `num_filled` for each row, and make use of the values returned by these function calls to solve the problem. The output from `min_and_max_filled` for the example board above should simply be

```
2 5
```

**Solution:**

```
counts = []
for r in bd:
    counts.append( num_filled(r) )
print min(counts), max(counts)
# or...
# counts.sort()
# print counts[0], counts[-1]
```

Version 2:

```
min_value = 9
max_value = 0
for r in bd:
    cnt = num_filled(r)
    if cnt > max_value:
        max_value = cnt
    if cnt < min_value:
        min_value = cnt
print min_value, max_value
```
29. Please show the output from the following code?

```python
def get_min(v):
    v.sort()
    return v[0]

def get_max(v):
    x = max(v)
    return x

v = [ 14, 19, 4, 5, 12, 8 ]
if len(v) > 10 and get_min(v) > 6:
    print "Hello"
    print v[0]
    print v[4]
else:
    print "So long"
    print v[0]
    print v[-1]
    if len(v) < 10 or get_max(v):
        print get_max(v)
        print v[0]
        print get_min(v)
        print v[0]
```

**Solution:** The trick to getting this completely right is to realize that the first if conditional never calls the function `get_min`. The first half of the test, `len(v) > 10` is False so the and can never be True, which means that Python does not bother to evaluate the second half of the logical expression — `get_min(v) > 6`.

```
So long
14
8
19
14
4
```

30. Write code that uses a `range` (and NO loops) to generate the following lists:

```python
v0 = [ 10, 9, 8, 7, 6, 5, 4, 3 ]

v1 = [ -10, -3, 4, 11, 18, 25, 32, 39 ]
```

**Solution:**

```python
v0 = range(10, 2, -1)
v1 = range(-10, 40, 7)
```

31. Consider the following list of lists of strings:

```python
wordy = [ [ 'impala', 'malibu', 'camry', 'jetta'], [ 'zebra', 'impala', 'lion', 'impala', 'malibu', 'zebra' ], [ 'tiger', 'lion', 'cowboy', 'jet', '49er' ], [ ], [ 'five', 'seven', 'nine'] ]```
(a) Show the output:

\[
\text{print wordy[2][1]}
\]

\[
\text{print wordy[1][2][3]}
\]

\[
\text{print len(wordy)}
\]

\[
\text{print len(wordy[1])}
\]

\[
\text{print sorted( set(wordy[0]) & set(wordy[1]) )}
\]

Solution:

lion
n
5
6
[ impala, malibu ]

(b) Write a loop to print the last word of each list in wordy, stopping when either an empty list is found or when there are no more lists. For the above example, the output should be:

jetta
zebra
49er

Solution:

\[
\text{for w in wordy:}
\]

\[
\quad \text{if len(w) > 0:}
\]

\[
\quad \text{print w[-1]}
\]

\[
\quad \text{else:}
\]

\[
\quad \text{break}
\]

or

\[
i = 0
\]

\[
\text{while i < len(wordy) and len(wordy[i] > 0):}
\]

\[
\quad \text{print wordy[i][-1]}
\]

\[
i += 1
\]

32. Suppose census data has been read into a list called census. Here is a short version of this list.

\[
\text{census = [ ['Albany', 'county', 294571, 304204 ], ['Albany', 'city', 94444, 97856], ['Berne', 'town', 2853, 2794], ['Coeymans', 'town', 8161, 7418], ['Cohoes', 'city', 15607, 16168], ['Spamalot', 'county', 81100, 99999 ], ['Colonie', 'town', 79327, 81591], ['Green Island', 'town', 2283, 2620] ]}
\]

Recall that the first number in each list in census is the population in 2000 and the second number is the population in 2010. Write code to determine the average town size in 2000 and the average town size in 2010. Write code to determine the average town size in 2000 and the average town size in 2010. Integer arithmetic is fine. For the version of census, the average town sizes are \((2853 + 8161 + 79327 + 2283) / 4 = 23156\) in 2000 and \((2794 + 7418 + 81591 + 2620) / 4 = 23605\), so the output should be.
2000: 23156
2010: 23605

Solution:

num_town = 0
town0 = 0
town1 = 0
for entity in census:
    if entity[1] == 'town':
        num_town += 1
        town0 += entity[2]
        town1 += entity[3]
print '2000:', town0/num_town
print '2010:', town1/num_town

33. Show the output from the following code:

def elephant(height):
    time_step = 1
    steps = 0
    while steps < height:
        steps += time_step
        steps -= time_step/3  # note: this is integer division
        time_step += 1
    print "%d, %d" %(time_step, steps)
elephant(0)
elephant(5)
elephant(6)

Solution:

1, 0
4, 5
5, 8
34. Show the output of the following code. Make sure we can determine what is output and what is scratch work.

```python
def remove_something(z):
    z.remove(z[z[0]])

v = [1, 8, [12, 8], 'hello', 'car']
x = 'salad'

if len(v[2]) >= 2:
    if x > v[3]:
        if v[0] == 1:
            print 'Three'
        else:
            print 'Two'
    elif len(v) == 5:
        print 'Six'
    else:
        v.append('five')
        print 'Ten'

remove_something(v)
print v[1]
print v[2]
v.append(x)
print len(v)
```

Solution:

```
One
Three
[12, 8]
hello
5
```

35. Write a function that takes as input a string and returns a new string that has a plus sign in the input string after every two characters. Example runs are given below.

```python
>>> addplus('abracadabra')
'ab+ra+ca+da+br+a'

>>> addplus('steven')
'st+ev+en+'
```

Solution:

```python
def addplus(s):
    w = ''
    for i in range(0,len(s),2):
        w += s[i:i+2]
        if len(s[i:i+2]) == 2:
            w += '+'
    return w
```
36. Write a function

   \texttt{compare\_semester(sem1,sem2)}

that takes as input two tuples \texttt{sem1,sem2} representing two academic semesters, each given in the form of a semester (either Fall or Spring) and year pair.

Your function should return 1 if \texttt{sem2} comes after \texttt{sem1}, -1 if \texttt{sem1} comes after \texttt{sem2}, and 0 otherwise. Remember, Fall 2014 comes after Spring 2014.

\begin{verbatim}
>>> compare_semester(('Fall',2013), ('Spring',2014))
1
>>> compare_semester(('Fall',2013), ('Spring',2012))
-1
>>> compare_semester(('Fall',2012), ('Spring',2012))
-1
>>> compare_semester(('Fall',2012), ('Fall',2012))
0
\end{verbatim}

**Solution:**

```python
def compare_semester(sem1, sem2):
    if sem1 == sem2:
        return 0
    elif sem2[1] > sem1[1]:
        return 1
    elif sem1[1] > sem2[1]:
        return -1
    elif sem2[0] == 'Fall':
        return 1
    else:
        return -1
```

37. You are given in variable \(x\) a list of lists represented as an \(N\times N\) grid in which each list corresponds to one row of the grid. For example, a 4x4 grid is given by:

\[ x = [[1,2,3,4], [4,3,2,1], [2,1,4,2], [2,1,4,5]] \]

Write a piece of code to print the grid in the following format with a vertical and horizontal line right in the middle:

\[
\begin{array}{cc|cc}
1 & 2 & 3 & 4 \\
4 & 3 & 2 & 1 \\
----|----
2 & 1 & 4 & 2 \\
2 & 1 & 4 & 5 \\
\end{array}
\]

Solution:

```python
n = len(x)
for i in range(n):
    l = ''
    for j in range(n):
        l += str(x[i][j]) + ' '  
        if j == (n/2-1):
            l += '| '  
    print l
    if i == (n/2-1):
        print '-'*n + '|' + '-'*n
```

38. Write a piece of code that repeatedly asks the user for numbers using `raw_input` until the user enters 'stop'. Then, the program reports the sum of the values entered by the user and the total number of values strictly greater than zero. You can assume that the user enters a valid number until she enters stop.

An example run of this code is given below.

Enter a value ==> 1.2
Enter a value ==> 0
Enter a value ==> 2
Enter a value ==> -1
Enter a value ==> stop
Sum: 2.2
Values > 0: 2

Solution:

```python
sum = 0
numgt = 0
while (True):
    val = raw_input('Enter a value ==> ')  
    if val == 'stop':
        break
    val = int(val)
    if val > 0:
        numgt += 1
        sum += val
print "Sum:", sum
print "Values > 0:", numgt
```
39. You are given a list of restaurants in the RPI region in a variable called `rest` in the form:

```python
rest = [ ["De Fazio's", "Troy", "Pizza", 4.5],
         ["Dinosaur BBQ", "Troy", "BBQ", 3.5],
         ["Van's", "Albany", "Vietnamese", 4],
```

For each restaurant, the first item is the name of a restaurant, the second item is the location, the third item is the type of restaurant it is, and the fourth item is the score.

Write a piece of code that uses this variable to find all restaurants in Troy with a score of at least 4 and print the name and the category. For example, for the above list, it would print:

De Fazio's (Pizza)
Lucas Confectionery (Wine Bar)

**Solution:**

```python
scores = []
for item in rest:
    if item[1] == 'Troy' and item[3] >= 4:
        print "%s (%s)" % (item[0], item[2])
```

40. Write a function `remove_val(l, val)` that removes all copies of `val` from list `l`.

Suppose you are given a variable `x` containing numbers as shown below:

```python
x = [1, 4, 2, 1, 2, 4, 4, 2, 5, 5, 2]
```

Then, your function should work as follows:

```python
>>> remove_val(x, 4)
>>> x
[1, 2, 1, 2, 2, 5, 5, 2]
```

Note: if your function returns a new list with this content instead of modifying it as given, you will lose 3 points.

**Solution:**

```python
def remove_val(l, val):
    while l.count(val) > 0:
        l.remove(val)
```

41. Suppose you are given the scores of two athletes in various competitions, given in two separate lists. Assume there are unknown number of competitions numbered 1,2,3, etc. and the length of the two lists is the same.

```python
a1 = [11,8,11,9]
a2 = [11,9,8,12]
```

For example according this to list, both athletes got a score of 11 in competition 1. Print the index of all the competitions in which `a2` did better. For example, for the above lists, we would print:

```
a2 is better in 2 4
```

If there is no value in which `a2` is better, then you should print:

```
a2 is never better
```
Solution:

found = False
line = "a2 is better in "
for i in range(len(a1)):
    if a2[i] > a1[i]:
        line += str(i) + ' '  
        found = True
if not found:
    line = "a2 is never better"
print line

42. What is the output of the following programs:

<table>
<thead>
<tr>
<th>Part a</th>
<th>Scratch area:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 25</td>
<td></td>
</tr>
<tr>
<td>b = 11</td>
<td></td>
</tr>
<tr>
<td>while True:</td>
<td></td>
</tr>
<tr>
<td>print a, b</td>
<td></td>
</tr>
<tr>
<td>if a &lt;= 0 or b &lt;= 0:</td>
<td>break</td>
</tr>
<tr>
<td>if a &gt; b:</td>
<td>a = a - b</td>
</tr>
<tr>
<td>else:</td>
<td>b = b - a</td>
</tr>
<tr>
<td>b -= 1</td>
<td></td>
</tr>
<tr>
<td>a += 1</td>
<td></td>
</tr>
</tbody>
</table>

Solution:

25 11
15 10
6 9
7 2
6 1
6 0
Part b

def spam(l,s):
    m = len(s)/2
    s1 = s[:m]
    s2 = s[m:]
    if l.count(s1) == 0:
        l.append(s1)
    if l.count(s2) == 0:
        l.append(s2)

l = ['ab','cd','de','fg']
s1 = 'abcde'
s2 = 'fghi'
spam(l,s1)
print s1
print l
l = spam(l,s2)
print s2
print l

Solution:

abcde
['ab', 'cd', 'de', 'fg', 'cde']
fghi
None