CS1110 Lab 2 (Feb 3-4, 2015)

First Name: ______________ Last Name: ______________ NetID: ______

The lab assignments are very important and you must have a CS 1110 course consultant "tell CMS" that you did the work. (Correctness does not matter.) This can be done any time up until the start of next week's lab (Feb 10-11). Thus, if you have trouble with a problem, then you have a week to get help from the teaching staff. If you finish before the hour is over, then you can leave early or you can work on the current assignment. Indeed, you are not required to physically attend the labs at all. Just make sure your work is "checked off" by a consultant. And remember this: *The lab problems feed into the assignments and the assignments define what the exams are all about.*

1 Boolean Expressions

For each example in this section, write down what you think is the value of the expression. Then use Python in interactive mode to find out if your mental reasoning is OK or not. If not, try to figure out why Python gave the answer that it did. Ask a staff member for clarification if necessary. Leave a paper trail of your mistakes and recoveries in the "notes column" as a reminder not to make the same mistakes in the future!

1.1 Numerical Examples

Assume that `x`,`y` and `z` are initialized as follows

```
>>> x = 1
>>> y = 5
>>> z = 10
```

Now complete this table:

	Expression	**I Think the Value Is**	**Python Says**	**Notes**
1	`x < z`	`True`	`True`	
2	`2*y >= z`			
3	`2*y < z`			
4	`(x>1) or (z!=7)`			
5	`y != (z/2)`			
6	`(x>0) or ((y>0) and (z<0))`			
7	`((x>0) or (y>0)) and (z<0)`			

Recall that if you just enter an expression in Interactive Mode, then Python will display the value of that expression. Thus, something like >>> `x < z` will result in the display of either "`True`" or "`False`".

1.2 String Examples

Assume that `x`, `y`, and `z` are initialized as follows

```
>>> x = 'Cornell'
>>> y = 'Harvard'
>>> z = 'Yale'
```

Now complete this table:

	Expression	I Think the Value Is	Python Says	Notes
1	`x != z`	True	True	
2	`x == 'cornell'`			
3	`len(x) > len(y)`			
4	`y[1:]  > z[1:]`			
5	`len(x+z) > len(y)`			

1.3 A Truth Table

Complete this truth table[1]:

A	B	A and B	not (A and B)	not A	not B	(not A) or (not B)
True	True	True				
True	False	False				
False	True	False				
False	False	False				

2 If Constructions

The problems in this section are all about how various if constructions work. You will want to download `Sticks.py` and `AllDiff.py` from the Labs page on the course website.

2.1 If-Else With Numerical Comparisons

Suppose you have three sticks. Using them you can form a triangle provided no stick has a length that is greater than or equal to the sum of the lengths of the other two sticks. Thus, if the three stick lengths are 7, 3, and 9, then we can form a triangle. On the other hand, if the three stick lengths are 1, 2, and 4, then we cannot form a triangle. Fill in the blanks with Boolean expressions so that the messages printed give the correct indications.

[1]You will observe that columns 4 and 7 are the same. This means that `not(A and B)` and `(not A) or (not B)` always have the same value. This connection between the `and` and `or` operations is famous. It is called *de Morgan's Law*.

```
a = input('Enter the first stick"s length (assumed positive): ')
b = input('Enter the second stick"s length (assumed positive): ')
c = input('Enter the third stick"s length (assumed positive): ')

if ______________________________________________________________________________:

    print 'Cannot be arranged to make a triangle!'
else:
    print 'Can be arranged to make a triangle!'

if ______________________________________________________________________________:

    print 'Can be arranged to make a triangle!'
else:
    print 'Cannot be arranged to make a triangle!'
```

Develop your solution through a combination of pencil and paper work, mental reasoning and experimentation with `Sticks.py`. Inspire confidence in your solution by trying out a wide variety of input values. For example, if you input 1, 2, and 4 then the script should print

```
Cannot be arranged to make a triangle!
Cannot be arranged to make a triangle!
```

2.2 If-Else with String Comparisons

Fill in the blanks with Boolean expressions so that the messages printed correctly indicate whether or not the input string consists of three different characters or whether there is a repeated character. Thus , `'abb'`, `'xxx'`, and `'bab'` have repeated characters while `'abc'`, `'Aa2'`, and `'(2)'` do not.

```
s = raw_input('Enter a length-3 string: ')

if _______________________________________________________________________________:

    print 'There is a repeat'
else:
    print 'All different'

if _______________________________________________________________________________:

    print 'All different'
else:
    print 'There is a repeat'
```

Develop your solution through a combination of pencil and paper work, mental reasoning and experimentation with `AllDiff.py`. Inspire confidence in your solution by trying things out on a wide variety of cases. Remember, that `raw_input` interprets the response as a string and that quotes are *not* used. Here is a sample dialog

```
Enter a length-3 string:  abc
All Different
All Different
```

2.3 If-Elif-Else

Consider the following

```
if (1<=x<=2) and (1<=y<=2):
   print 'A'
elif  (x<1) or (x>2):
   print 'B'
elif (y<1) or (y>2):
   print 'C'
else:
   print 'D'
```

Fill in the blanks:

If the value of `x` is __1.3__ and the value of `y` is __1.9__ then the output is `'A'`.

If the value of `x` is ________ and the value of `y` is ________ then the output is `'B'`.

If the value of `x` is ________ and the value of `y` is ________ then the output is `'C'`.

If the value of `x` is ________ and the value of `y` is ________ then the output is `'D'`.

3 Pretty printing

Down load `FormatPlay.py` from the Labs page. Here it is

```
from math import pi
x = 355
y  = 113
z = float(x)/float(y)
err = abs(z - pi)
print '\nNumerator   Denominator   Quotient      Error'
print '-----------------------------------------------'
print '%3d %3d    %22.15f %10.6e' % (x,y,z,err)
```

It produces the following output:

```

Numerator   Denominator   Quotient      Error
-----------------------------------------------
355 113         3.141592920353983 2.667642e-07
```

Modify the last line in `FormatPlay.py` so that the following output is *exactly* reproduced:

```

Numerator   Denominator   Quotient      Error
-----------------------------------------------
    355          113      3.1415929    2.67e-07
```

Do this by playing with the format specifications and blanks in the string `'%3d %3d %22.15f %10.6e'`.

4 Errors

If you do something illegal, Python complains. So let's learn more about about some of those naughty mistakes that we all make. Download the Jan 29 Lecture Demo `Hyphenator.py`. Here it is for your convenience

```
""" Inputs a string and inserts hyphens.
If the string has even length, the hyphen splits
the first and second halves. Otherwise, a hyphen
is inserted on either side of the middle character.
"""

s = input('Enter a string: ')
n = len(s)
if n%2==0:
    # s has even length
    m = n/2
    h = s[0:m] + '-' + s[m:]
else:
    # s has odd length
    m = n/2
    h = s[0:m]+'-'+s[m]+'-'+s[m+1:]
print s,h
```

In the following, we ask you to make a change that induces an error. For each example (1) make the change and save, (2) run the modified `Hyphenator.py`, (3) report the error message and explain why Python is complaining, (4) undo the change thereby restoring the the original `Hyphenator.py`.

1. In the doc string, change one of the `"""` to `""`.

2. Remove the colon after the `else`.

3. Change `n%2==0` to `n%2=0`

4. Change `n%2==0` to `n%2 = = 0`.

5. Remove one of the quotes in the `input` statement.

6. Change `n = len(s)` to `n = length(s)`

7. Remove the `"#"` from any one of the comments.

8. "Unindent" the first `m=n/2` statement.