Your Name: ____________________________

CS 1301 Summer 2009 Exam 2/2

Problem	Earned Points	Points Possible
Vocabulary Matching		22
Python Expressions		20
Comedy & Drama		8
Phonebook		6
Fill in the Blank		4
LongWords		12
Pixel Swap		14
Robot Photographer		14
Total:		100

Your Name: ________________________

1. Vocabulary Matching (22 points)

Write the number before the definition on the right on the line before the matching vocabulary word.

__10__ aliases	1. Smallest addressable element of a picture.
__17__ clone	2. A variable that can only be accessed within the function that it was defined in.
__9__ compound data type	3. The % operator takes a format string and a tuple of values and generates a string by inserting the data values into the format string at the appropriate locations.
__21__ decrement	4. When a boolean expression is evaluated the evaluation starts at the left hand expression and proceeds to the right, stopping when it is no longer necessary to evaluate any further to determine the final outcome.
__22__ dictionary	5. A block of code which can be executed as if it were a function but without a name.
__8__ exception	6. Can be seen through a program module, even inside of functions.
__7__ file	7. A named entity, usually stored on a hard drive, floppy disk, or CD-ROM, that contains a stream of characters.
__3__ format operator	8. Raised by the runtime system if something goes wrong while the program is running.
__6__ global variables	9. A data type that is itself made up of elements that are themselves values.
__19__ immutable type	10. Multiple variables that contain references to the same object.
_20___ increment	11. A data type that is made up of elements organized linearly, with each element accessed by an integer index.
__15__ iteration	12. The process of calling the currently executing function.
__5__ lambda	13. To repeat an operation on all members of a set from the start to the end.
__2__ local variables	14. A copy of part of a sequence specified by a series of indices.
_18___ mutable type	15. To repeat a section of code.
_16___ nested list	16. A list that is itself contained within a list.
__1__ pixel	17. To create a new object that has the same value as an existing object.
_12___ recursion	18. A compound data type whose elements can be assigned new values.
_11___ sequence	19. A compound data type whose elements can NOT be assigned new values.
_4___ short circuit evaluation	20. To add one to a variable.
_14___ slice	21. To subtract one from a variable.
_13___ traverse	22. A collection of key/value pairs that maps from keys to values.

Your Name: ________________________

2. Python Expression Evaluation (20 points)

For this question, assume the following statements have already been entered and interpreted:

```
a = [ 5, 10, 15, True, ["Cherry", "Apple","Plum"], 56, [4, 5, 6], 84 ]
b = a
c = a[0:4]
d = a[4]
d[2] = "Peach"
x = { 1: "one", 2 : "two"}
```

Pretend that you are the Python Interpreter (IDLE window). Watch out for the difference between aliases and clones! What do you print or return when each of the following statements are entered?

Example: a[0]	*Result:*___**5**____
Example: a[1:4]	*Result:* _ [10, 15, True]_
1. **a[6][0]**	*Result:_4*_______
2. **d**	*Result:["Cherry", "Apple", "Peach"]*
3. **c**	*Result: ___[5,10,15,True]*_____
4. **a[4][2]**	*Result: __"Peach"*______
5. **b[:2]**	*Result: _[5,10]*_______
6. **x[2]**	*Result: _"two"*_______
7. **b[-2]**	*Result: __[4,5,6]*______
8. **c[-2]**	*Result: ___15*_____
9. **x.get(0 , 5)**	*Result: ____5*____
10. **print "Pumpkin %.3f" %3.1459**	*Result: ___Pumpkin 3.146*_____

-2 for "plumb", -1 for 1 extra item at end of slices, for missing "'s or missing []'s. Don't count off added "'s around Pumpkin 3.146. -1 for Missing Pumpkin but getting 3.146

Your Name: ________________________

3. Comedy & Drama (8 points)

a. Write a function called addComedy that takes a list as input, adds the string ":)" to the end of the list, and returns the modified list. This function should modify *and return* the original list.

Example:

```
>>> a = [True, 4.0, "Saturday"]
>>> addComedy(a)
[True, 4.0, "Saturday", ":)"]
>>> a
[True, 4.0, "Saturday", ":)"]
```

```
def addComedy( aList):
     aList.append( “:)” )
     return(aList)
```

```
Grading: 1 point each for correct definition, returning the list,
appending “:)” to the list, and appending the string at the END of
the list.
```

b.Write another function called addDrama that takes a list as input, makes a duplicate of the list, adds the string ":(" to the end of the duplicate, and returns the modified list. Note that unlike addComedy, this function should NOT modify the original list!

Example:

```
>>> a = [2.85, 98, "Othello"]
>>> addDrama(a)
[2.85, 98, "Othello", ":("]
>>> a
[2.85, 98, "Othello"]
```

```
def addDrama( aList):
     aCopy = aList[:]  # or aCopy = copy( aList)
     aCopy.append(“:(“)
     return( aCopy)
```

```
Grading: One point each for correct header, making a copy of the
list,  appending the string correctly, and returning the copy of the
list.
```

Your Name: ________________________

4. PhoneBook (6 points)
You have a list of names and telephone numbers stored in a dictionary called phoneBook. The names are the keys, and the numbers are the values. Both the keys and values (names and numbers) are stored as strings. What ***single line of code*** would you need to execute in each of the following scenarios to update the phoneBook dictionary correctly?

a. Your old friend Steve has changed his number from “123-4567” to “987-6543”. (You may assume the key “Steve” already exists in the phone book with the value “123-4567” associated with it.)

phoneBook[“Steve”] = “987-6543”

b. Steve introduces you to his younger sister, Jenny, whom you've never met before. (Her name is not in your phone book.) Her number is “867-5309”, and you add it to your phonebook.

phoneBook[“Jenny”] = “867-5309”

c. Steve informs you that he has been selected by the UN to be an undercover secret peace agent, keeping the world safe from megalomaniacs and mad scientists. Unfortunately, this means you won't be able to contact him by telephone any more. Remove his entry from your phone book.

del phoneBook[“Steve”]

`Grading: 2 points for a correct answer. -1 point for minor syntax error, or using the value as the key once.`

5. Fill in the Blank (4 points)

Python has several compound data types that we have learned about. A __String_ can be used to store a sequence of characters, while a __Tuple____ can store a sequence of any type of data (but is immutable). A _List____ can also store any type of data, and allows you to change elements within it. Finally, a __Dictionary__ can associate a value to a key.

Your Name: ________________________

6. LongWords (12 points)
The function longWords(aList) accepts a list of strings and prints out each string with more than five letters in it. You may assume that only lists containing nothing but strings will be passed into this function.
Example:

```
>>> a = [ 'a', 'to', 'two', 'reallybigstring', 'anotherlongstring']
>>> longWords(a)
reallybigstring
anotherlongstring
```

a. Write longWords using a while loop.

```
def longWords( aList):
    index = 0
    while( index < len( aList) ):
        aString = aList[index]
        if ( len(aString) > 5 ):
            print aString
        index = index + 1
```

+1 point each for correct header, while checking the index vs. the length of the list, if checking the length of the item, and incrementing index.

b. Write longWords using a for loop.

```
def longWords( aList ):
    for x in aList:
        if (len(x) > 5):
            print x
```

+1 point each for the correct header, for loop, length of item check, and print.

c. Write longWords using a small helper function (named printIfBig) and map.

```
def printIfBig( x ):
    if len(x) > 5:
        print x

def longWords( aList ):
    map( printIfBig, aList)
```

+1 point for def of printIfBig, if test inside it, correct header for longWords, and correct use of the map function.

Your Name: ____________________________

7. PixelSwap (14 points)
Write a function called pixelSwap() that will have your robot take a picture and then swap the red and green values of every 3rd pixel. After it swaps the red and green pixel value of every third pixel, it should return the modified picture.

```
def pixelSwap():
    p = takePicture()
    counter = 1
    for pix in getPixels(p):
        if (counter = = 3):
            r = getRed(pix)
            g = getGreen(pix)
            setRed(pix,g)
            setGreen(pix,r)
            counter = 1
        counter = counter + 1
    return(p)
```

```
Grading:
2 - correct header definition
1 – take picture
1 – iterate through pixels
2 – correctly modifying every 3rd
2 – correctly getting R,G values
2 – swapping the r & g values
2 – return picture
```

Your Name: ______________________

8. Robot Photographer (14 points)
Write a program that makes your robot move forward and take pictures. Every time it takes a picture, it should turn to the right and then move forward again before taking another picture. Right after it takes a picture, it should use the getLight("center") function to sample the light value in that location. Only show a picture if the light level reading returned by the center light sensor is smaller than 150. Your robot should move around and keep taking pictures until it has ***shown*** 20 pictures (no matter how many pictures it has taken!)

```
picShown = 0
while( picShown < 20):
        p = takePicture()
        lv = getLight("center")
        turnRight(0.5,0.5)
        forward(0.5,0.5)
        if( lv < 150):
                show(p)
                picShown = picShown + 1
```

2 – Taking the pictures
2 – Moving (right/forward)
2 – get light value
3 – Show Picture if light value < 150
3 – Keep track of number of pictures shown
2 – show 20 pictures (1 if they show 19 or 21)