CS 2316
Exam 3 Spring 2012

Name : _____

Grading TA: _____

- INTEGRITY: By taking this exam, you pledge that this is your work and you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech. Do NOT sign nor take this exam if you do not agree with the honor code.
- DEVICES: If your cell phone, pager, PDA, beeper, iPod, or similar item goes off during the exam, you will lose 10 points on this exam. Turn all such devices off and put them away now. You cannot have them on your desk.
- ACADEMIC MISCONDUCT: Academic misconduct will not be tolerated. You are to uphold the honor and integrity bestowed upon you by the Georgia Institute of Technology.
  - Keep your eyes on your own paper.
  - Do your best to prevent anyone else from seeing your work.
  - Do NOT communicate with anyone other than a proctor for ANY reason in ANY language in ANY manner.
  - Do NOT share ANYTHING during the exam. (This includes no sharing of pencils, paper, erasers).
  - Follow directions given by the proctor(s).
  - Stop all writing when told to stop. Failure to stop writing on this exam when told to do so is academic misconduct.
  - Do not use notes, books, calculators, etc during the exam.
- TIME: Don't get bogged down by any one question. If you get stuck, move on to the next problem and come back once you have completed all of the other problems. This exam has 6 questions on 9 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Georgia Tech community. I have also read and understand the requirements outlined above.

Signature: _____

Question	Points	Score
1. Vocabulary	9	
2. Multiple Choice	3	
3. Regular Expressions	11	
4. Weather	10	
5. Canvas Fun	10	
6. Every Other	10	
Total:	53	

1. *(9 points)*
   For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] object

   Solution: A compound data type that is often used to model a thing or concept in the real world. It bundles together the data and the operations that are relevant for that kind of data. Instance and object are used interchangeably.
   Looking for: Compound Data Type, Includes data & operations/functions/methods.

   (b) [3 pts] syntax error

   Solution: An error in a program that makes it impossible to parse – and therefore impossible to interpret.

   (c) [3 pts] delimiter

   Solution: A character that is used to separate (or delimit) separate data items in a string. The default delimiter for comma separated value (CSV) files is the comma.
   Looking for: Separation of data items.

2. *(3 points)*
   For each of the following multiple choice questions, indicate the most correct answer! Indicate your selected answer by circling it.

   (a) [1 pt] You can pack a label and then grid a button in the same frame and have both displayed.

   A. True **B. False**

(b) [1 pt] You can create a frame inside another frame.

A. True B. False

(c) [1 pt] You must have a variable representing the instance as the first parameter of any method in a class.

A. True B. False

3. *(11 points)*

For each regular expression given below, determine which of the lines that follow it are completely matched by the regex pattern. (That is, there are no characters in the string that will not be matched by the given pattern...) Circle **all** lines that are fully matched.

(a) [2 pts] Pattern: `-?\d*.\d+`

-.1301
1
-42+9001
2316
3.14

Solution: -.1301
-42+9001
2316
3.14
Grading: +2 for 100% correct. Minus 1 for each extra or missing circle, minimum score of zero on this part.

(b) [2 pts] Pattern: `[a-z][a-z0-9]+\(\)`

myFunc1()
str(7)
print()
astr.strip()
x()

Solution: print()
Grading:
+2 for 100% correct. Minus 1 for each extra or missing circle, minimum score of zero on this part.

(c) [2 pts] Pattern: `[:;8]-?[DOX(P)]`

X(
:-P
;-]
8X
:?

Solution: : - P
8X

Grading: +2 for 100% correct. Minus 1 for each extra or missing circle, minimum score of zero on this part.

(d) [5 pts] Write a regular expression that will match a string formatted as: "City, ST ZIPCO". The city name can contain any letters or spaces, but will always be followed by a comma and a space before the state abbriviation. The state abbriviation is exactly two upper case letters, followed by a space, followed by exactly 5 digits. Remember that city names can have two parts separated by spaces! For example:
Atlanta, GA 30002
Yakima, WA 98903
Boca Raton, FL 33427

Solution: One possible solution:
`[A-Za-z ]+, [A-Z]{2} [0-9]{5}`
Grading:
+1 for matching one or more letters for the city
+1 for matching uppercase letters for the ST
+1 for matching exactly 2 uppercase letters for the ST.
+1 for matching the spaces.
+1 for matching exactly 5 digits in the zipcode.

4. *(10 points)*
Examine the following code:

```
class GA:
    population = 9800000
    weather = "hot"

    def __init__(self, loc):
        print("Georgia")
        self.city = loc
        if self.city == "Atlanta":
            self.Atlanta()
        elif self.city == "Athens":
            print("Bad!")
            GA.weather = "cold"

    def Atlanta(self):
        self.weather = "humid"

place = GA("Atlanta")
place2 = GA("Athens")
print("The weather in {0} is {1}.".format(place.city, place.weather))
print("The weather in {0} is {1}.".format(place2.city, GA.weather))
```

What is printed on the screen as the code above is executed?

Solution: One point for each correct line:
Georgia
Georgia
Bad!
The weather in Atlanta is humid.
The weather in Athens is cold.

Different identifiers above are used to name a **class, class variable, object, and object variable**. For each of the following identifiers, write what it was used to name:

GA -

population -

city -

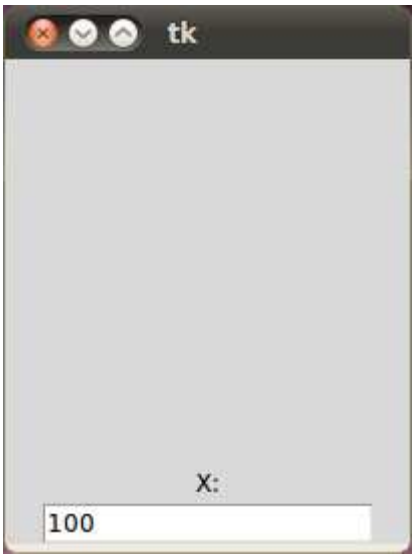
place -

Solution: +1 point for each correct answer, +1 bonus if all 4 are correct.:
GA is a class
population is a class variable
city is an object (instance) variable
place is an object

5. *(10 points)*
Examine the following code:

```
from tkinter import *
class CanvasFun:

    def __init__(self, root):
        self.root = root
        canvas = Canvas(self.root, height = 200, width = 200)
        canvas.bind("<Button-1>", self.leftClick)
        canvas.bind("<Button-3>", self.rightClick)
        self.canvas = canvas.pack()

    def rightClick(self, event):
        if self.canvas == None:
            self.createScore()

    def leftClick(self, event):
        self.sv.set(event.x)

    def createScore(self):
        Label(self.root, text = "X:").pack()
        self.sv = StringVar()
        self.entry = Entry(self.root, textvariable= self.sv)
        self.entry.pack()

root = Tk()
CanvasFun(root)
root.mainloop()
```

Draw the single GUI created by the code above after both the Right and then the Left mouse button have been clicked in the exact center of the window. Include any extra text printed in the console (IDLE Shell Window) on the side of the GUI.

Solution: 

Grading:
+2 entry drawn in correct position (at bottom, under X:)
+2 label drawn in the correct position (just above entry, centered)
+2 space for canvas above entry/label (correct dimensions)
+2 the number 100 (or anything 98-102) appears in the entry
+2 nothing is printed in the console.

6. *(10 points)*

Write a function, **everyOther**, which will take in two parameters: fromFile and toFile. fromFile is a string containing the file name on your computer you wish to read from. toFile is a string of the file name on your computer you wish to write to. You should read in the contents of fromFile, and write every other line to toFile. For example, if from File contains:

```
Hi
Isn't
CS
Fun
```

Then after your function runs, toFile should contain:

```
Hi
CS
```

Solution:

Example solution:

```
def everyOther(fromFile, toFile):
    fromF = open(fromFile)
    toF = open(toFile, "w")
    myLines = fromF.readlines()
    for i in range(0, len(myLines)):
        if i % 2 == 0:
            toF.write(myLines[i])
    fromF.close()
    toF.close()
```

Grading Rubric:

```
+1 opens fromFile correctly
+1 opens toFile with "w" as mode
+2 correctly reads lines from file
+2 correctly writes lines to file
+2 writes only every other line to file
+2 closes both files correctly
```