Lab Exercise #9

Assignment Overview

This lab exercise provides practice with lists, functions and namespaces in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

Part A: Local Scope and Namespaces

1. Consider the Python statements shown below and the partial output they produce.

```
def display( y ):

    x = 2
    print( "In display, x:", x )
    print( "In display, y:", y )

    return

def main():

    x = 6
    display( x+2 )

    print( "In main, x:", x )

    return

main()
```

Assuming the "Namespace for function…" information below is displayed at the instant a function returns, predict what the remaining output will be.

```
In display, x: ?

In display, y: ?

        Namespace for function display:
        x   ?
        y   ?

In main, x: ?

        Namespace for function main:
        x   ?
```

2. After completing (1) above, download and execute the program ("lab09.parta.py") to check your predictions. Note that the program has been instrumented with calls to function "locals" to access each function's namespace dictionary.

Part B: Nested Scope and Namespaces

Consider the Python statements shown below and the partial output they produce. Assuming the "Namespace" information is displayed at the instant a function returns, predict what the remaining output will be. Download and execute the program ("lab09.partb.py") to check your predictions.

```
def show( x, y ):

    def square( n ):
        return n*n

    def sum( x, y ):

        s = square( x )
        t = square( y )
        return s + t

    z = sum( 2*x, 2*y )
    print( "Value of x:", x )
    print( "Value of y:", y )
    print( "Value of z:", z )

def main():

    a = 3
    b = 7
    show( a+1, b-2 )

main()
```

```
         Namespace for function square:
         n         ?

         Namespace for function square:
         n         ?

         Namespace for function sum:
         x         ?
         y         ?
         s         ?
         t         ?

Value of x: ?
Value of y: ?
Value of z: ?

         Namespace for function show:
         x         ?
         y         ?
         square    ?
         sum       ?
         z         ?

         Namespace for function main:
         a         ?
         b         ?
```

Part C: Lists and Functions

1. Consider the following Python program.

```
import random

def sub1( size ):

    values = list()
    for i in range(size):
        values.append(0)
    return values

def sub2( values, size ):

    for n in range(100000):
        i = random.randint(0,size-1)
        values[i] += 1

def sub3( values, size ):

    for i in range(size):
        print( i, values[i] )

def main():

    num = 10

    count = sub1( num )

    sub2( count, num )

    sub3( count, num )

main()
```

a) What is the purpose of function "sub1"? How many elements are in the list which is created by that function? What is the initial value of each element?

b) What is the purpose of function "sub2"? What does the call to function "`random.randint`" return? What does the statement "`values[i] += 1`" accomplish?

c) What is the purpose of function "sub3"?

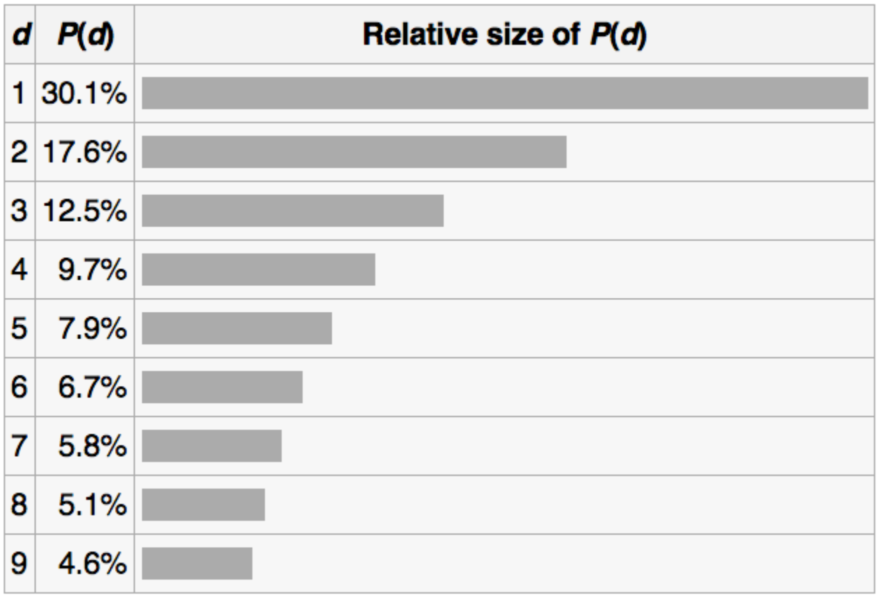
2. Download the program ("lab09.partc.py"), then repeatedly execute the program to check your predictions. Note that you can re-execute the program by simply typing "`main()`" at the shell prompt.

3. Revise the program to display the summary information in a more readable format. Each line should begin with a tab, followed by the value of "`i`", the string "`:   `", and the value of "`values[i]`" (right justified with a field width of 7).

4. Revise the program to prompt the user for the number of elements in the list. If the user makes a mistake when entering that number, the program will display a message and assume the number is 10.

Part D: Benford's Law

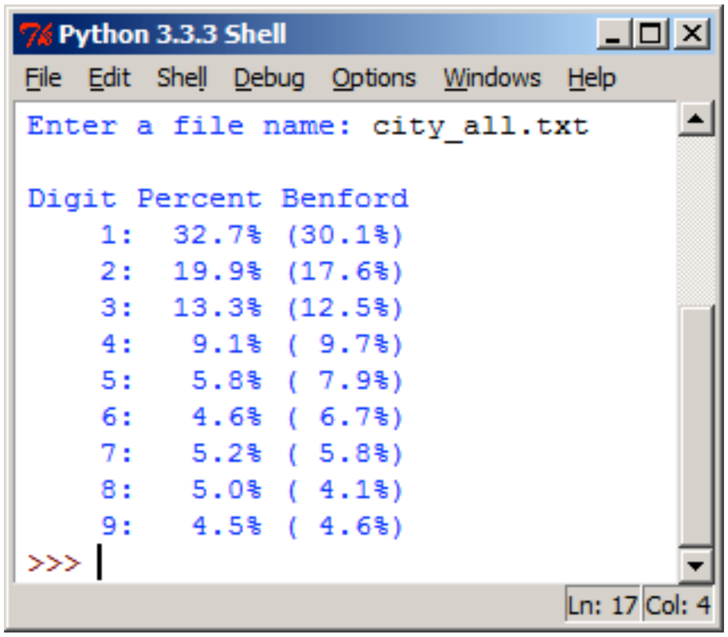
Benford's Law (sometimes called the First-Digit Law) is an observation about the distribution of first digits in many data sets: the number 1 occurs as the leading digit about 30% of the time, while larger numbers occur in that position less frequently (for example, 9 occurs as the first digit less than 5% of the time). As noted on Wikipedia: This result has been found to apply to a wide variety of data sets, including electricity bills, street addresses, stock prices, population numbers, death rates, lengths of rivers, physical and mathematical constants, and processes described by power laws (which are very common in nature).



d	*P*(*d*)	Relative size of *P*(*d*)
1	30.1%	
2	17.6%	
3	12.5%	
4	9.7%	
5	7.9%	
6	6.7%	
7	5.8%	
8	5.1%	
9	4.6%	

Develop a program which will allow you to experiment with Benford's Law. The program will prompt the user for the name of an input file, read the contents of the file and count the leading digits, and then display the results. For simplicity, the program will assume that the data file contains integer numbers, with one value per line.

The program's output will be formatted for readability (see below).



```
Enter a file name: city_all.txt

Digit Percent Benford
    1:   32.7% (30.1%)
    2:   19.9% (17.6%)
    3:   13.3% (12.5%)
    4:    9.1% ( 9.7%)
    5:    5.8% ( 7.9%)
    6:    4.6% ( 6.7%)
    7:    5.2% ( 5.8%)
    8:    5.0% ( 4.1%)
    9:    4.5% ( 4.6%)
>>> 
```

Notes and Suggestions

1. The directory for this lab exercise contains three data files which you can use as you develop your program:

`city_part.txt`	the first 10 lines of the following file
`city_all.txt`	the population of every city and town in Michigan (1500+ lines)
`warblers.txt`	decades of counts of warbler observations (540,000+ lines)

Use the first file during your initial development since it only has 10 lines.

2. There are three phases to this program. What are they? Which should you do first?

3. The string method "`strip`" might be useful to discard leading and trailing whitespace.

4. The string method "`isdigit`" might be useful to ensure that your program only processes integer numbers.

5. The string method "`format`" might be useful to align the output in columns.

6. Your program does not need to somehow compute the Benford values. Instead, you may use the values from the chart (or sample output).

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab09.partd.py") for grading via the CSE handin system.**