Lab 11,

**Goals**

Upon successful completion of this lab, you should be able to understand and write programs using

* functions
* file I/O
* Python's built-in index function
* hand-coded linear search

**What To Turn In**

* Writeup: in Moodle
* Submit: code from Parts D and E
* Demo: code from Part D

**Instructions**

Follow the links below to complete each part of this lab.

1. [Setup](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11a.html)
2. [Introduction to file I/O](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11b.html)
3. [Searching a list with the index function](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11c.html)
4. [Linear search](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11d.html)
5. [Drawing user-specified flags](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11e.html)
6. [Submitting your assignment](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11f.html)

# Part A: Setup

## Setup

1. Before you start writing code, you will set up a directory for this assignment on cwolf. Log on to cwolf using PuTTY or Terminal.
2. Enter your CS 115 subdirectory by typing

cd cs115

1. Within your CS 115 subdirectory, create a directory for Lab 11:

mkdir lab11

1. Enter your Lab 11 directory by typing

cd lab11

1. In this lab, you will use a list of California's largest cities as an input file. Copy this file from my cwolf directory to yours by typing the following command. Do not omit the '.' at the end of the command:

cp ~srivoire/cs115/pickup/cities.txt .

1. Type

ls

(that's a lowercase L followed by a lowercase S). If you copied the files correctly, you should see cities.txt in the list.

1. For the remainder of this lab, you will create, edit, and test Python programs. You can work locally on your lab machine or remotely on cwolf.
2. **[remote]** Open cities.txt in emacs and take a look at the contents of the file. Do not make any changes to the file.
3. **[local]** Right-click to download [cities.txt](http://rivoire.cs.sonoma.edu/cs115/labs/cities.txt) into your default Python directory (C:\Python32 or C:\Python33 on Windows or Documents on a Mac). Take a look at the contents of the file, but do not make any changes to it.
4. [Continue to Part B.](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11b.html)

# Part B: Intro to files

## Instructions

1. Create and open a new Python source code file called lab11b.py:
2. """
3. Program: CS 115 Lab 11b/c/d
4. Author: *Your name*
5. Description: This program will open a file and then search its contents.
6. """
7. def readfile(filename):
8. """
9. Reads the entire contents of a file into a single string using
10. the read() method.
11. Parameter: the name of the file to read (as a string)
12. Returns: the text in the file as a large, possibly multi-line, string
13. """
14. infile = open(filename, "r") # open file for reading
15. # Use Python's file read function to read the file contents
16. filetext = infile.read()
17. infile.close() # close the file
18. return filetext # the text of the file, as a single string
19. def main():
20. """ Read and print a file's contents. """
21. pass
22. main()
23. Read this code carefully before continuing and answer Question 1.
24. Right now, this code doesn't do anything because the definition of main is empty. To fill out the definition of main, replace the pass with the following 3 lines of code:
    * A statement that asks the user to supply the name of a file to read. This statement should save the user's answer as a string.
    * A statement that calls the readfile function to retrieve the contents of the user's file.  
      You should pass readfile the user-supplied filename (as a string).  
      You should save readfile's return value in a different variable.
    * Write a statement to print the value of the variable that holds readfile's return value.
25. Run your program. Type cities.txt when prompted. Verify that your program prints the names of cities.
26. Run your program again. Type something like blah blah blah. Answer Question 2 in your writeup.
27. Right now, your program is crashing in readfile when the file you asked it to open does not exist. Modify readfile to handle this exception:
    * Create a try block.The three statements involving infile should go inside the try block, since they are the ones that depend on the data being read correctly.
    * Create an except block to handle the exception you identified. Within this block, instead of printing an error message or exiting the program, just set the filetext variable to the empty string.  
      Remember that an empty string is just two quotes right next to each other, without a space between them.
    * Be sure that your function returns the value of the filetext variable, whether or not an exception occurs.
28. Run your program again and verify that it does not crash when you type arbitrary text for the filename.
29. Answer Question 3 in your writeup. To help you answer these questions, you can print the value and length of the information you read from the file.
30. Inside the definition of readfile, add a call to split() at the end of the call to read:

filetext = infile.read().split()

Answer Question 4 in your writeup.

1. To split the text from the file into a list of lines rather than a list of words, modify your line of code as shown:

filetext = infile.read().splitlines()

Answer Question 5 in your writeup.

1. Here are some samples of what your program should be doing at this point. Modify your wording to match the example, and add code to print the number of lines in the file if you do not have it already.

#### Sample 1

Name of input file: blah

Number of lines in file: 0

#### Sample 2

Name of input file: cities.txt

Number of lines in file: 50

['Los Angeles', 'San Diego', 'San Jose', 'San Francisco', 'Fresno', 'Sacramento', 'Long Beach', 'Oakland', 'Bakersfield',

'Anaheim', 'Santa Ana', 'Riverside', 'Stockton', 'Chula Vista', 'Fremont', 'Irvine', 'San Bernardino', 'Modesto', 'Oxnard',

Fontana', 'Moreno Valley', 'Glendale', 'Huntington Beach', 'Santa Clarita', 'Garden Grove', 'Santa Rosa', 'Oceanside', 'Rancho

Cucamonga', 'Ontario', 'Lancaster', 'Elk Grove', 'Palmdale', 'Corona', 'Salinas', 'Pomona', 'Torrance', 'Hayward',

Escondido', 'Sunnyvale', 'Pasadena', 'Orange', 'Fullerton', 'Thousand Oaks', 'Visalia', 'Simi Valley', 'Concord', 'Roseville',

Santa Clara', 'Vallejo', 'Victorville']

1. [Continue to Part C.](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11c.html)

# Part C: Searching a list with the index function

## Instructions

1. Add the following function definition to your program:
2. def print\_list(list\_to\_print):
3. """
4. Print the contents of a list.
5. Parameter: the list to print
6. Returns: nothing
7. """
8. for i, item in enumerate(list\_to\_print):

print(i, ': ', item, sep="")

1. Modify your main function to call your print\_list function instead of printing the list directly. Sample output:
2. Name of input file: cities.txt
3. Number of lines in file: 50
4. 0: Los Angeles
5. 1: San Diego
6. 2: San Jose
7. 3: San Francisco

[etc]

1. Answer Question 6 in your writeup.
2. We will modify this program to allow the user to ask for the name of a city, and we will tell the user its position in the list. Add the following function definition to your program:
3. def built\_in\_search(search\_list, value\_to\_find):
4. """
5. Uses Python's built-in search function to find the position of an item
6. in a list
7. Parameters: the list; the item to search for
8. Returns: the position of the item in the list
9. (or None if it is not in the list)
10. """
11. return search\_list.index(value\_to\_find)
12. Modify your main function to ask the user for the name of a city and print the city's position in the list by calling this function.  
    You should only modify main and not the function!  
    For example:
13. Name of input file: cities.txt
14. Number of lines in file: 50
15. 0: Los Angeles
16. 1: San Diego
17. 2: San Jose
18. 3: San Francisco
19. [etc]
20. Enter the name of a city: Santa Rosa
21. The position of Santa Rosa is 25.
22. Answer Questions 7 and 8 in your writeup.
23. Modify the definition of built\_in\_search to add exception handling.  
    The exception handler should return None if value\_to\_find is not in the list. Test your program with a city that is not in the list to be sure that it does not crash. For example:
24. Name of input file: cities.txt
25. Number of lines in file: 50
26. 0: Los Angeles
27. 1: San Diego
28. 2: San Jose
29. 3: San Francisco
30. [etc]
31. Enter the name of a city: Cotati
32. The position of Cotati is None.
33. Modify your main function to continue asking the user for the names of cities until the user types quit:
34. Name of input file: cities.txt
35. Number of lines in file: 50
36. 0: Los Angeles
37. 1: San Diego
38. 2: San Jose
39. 3: San Francisco
40. [etc]
41. Enter the name of a city: Cotati
42. The position of Cotati is None.
43. Enter the name of a city: Fresno
44. The position of Fresno is 4.
45. Enter the name of a city: Santa Rosa
46. The position of Santa Rosa is 25.
47. Enter the name of a city: quit
48. [Continue to Part D.](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11d.html)

# Part D: Linear search

## Instructions

1. Add the following function definition to your program:
2. def linear\_search(search\_list, value\_to\_find):
3. """
4. Uses a hand-coded linear search to find the position of an item in a list
5. Parameters: the list; the item to search for
6. Returns: the position of the item in the list
7. (or None if it is not in the list)
8. """
9. pass
10. Write code in linear\_search to do the following. Do NOT use index!! We are trying to duplicate what index does by writing our own code.
    * Loop over your list (hint: use enumerate)
    * Compare each item, in turn, to value\_to\_find
    * If an item matches the value you are trying to find, return its position in the list *immediately*.

You do not need to explicitly return None, but you should make sure that your code does not return anything if the item is not in the list.

1. Add code to your loop to print each city's position in the list using both of your functions. At this point, you should remove the code that prints the length and contents of the entire list.
2. Answer Question 9 in your writeup.
3. Make sure your code matches the sample inputs and outputs below exactly, including the blank lines:
4. Name of input file: cities.txt
5. Enter the name of a city: Santa Rosa
6. The position of Santa Rosa is:
7. Built-in search: 25
8. Linear search: 25
9. Enter the name of a city: Oakland
10. The position of Oakland is:
11. Built-in search: 7
12. Linear search: 7
13. Enter the name of a city: Cotati
14. The position of Cotati is:
15. Built-in search: None
16. Linear search: None

Enter the name of a city: quit

1. Call an instructor to demo your program. If you are running short on time, skip to [Part F](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11f.html) to submit your code.
2. [Continue to Part E.](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11e.html)

# Part E: Getting flag specifications from the user

## Setup

Because this part of the lab is graphical, you will need to do it on your local machine. Follow these instructions to get set up.

* Right-click on this link: <http://mcsp.wartburg.edu/zelle/python/graphics.py>. Select "Save Link As..." or "Save Target As..." If you are using Mac OS, save the file into your Documents directory. If you are on Windows, save it into C:\Python32 or C:\Python33.
* In a separate browser tab, open [the reference document for the graphics library](http://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html), just in case you need to refer to it during this lab.
* Make a copy of your lab10d.py file, and name the copy lab11e.py.
  + If you did not complete [Lab 10, Part D](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab10d.html), you will need to do that first.
  + If you need any help copying the file, please ask the course staff for assistance.

## Instructions

1. In lab11e.py, modify your main() function to prompt the user for the following information:
   * Country name
   * Flag width
   * Whether the stripes are horizontal or vertical

Remove the rest of the code inside main. Sample input and output:

Which national flag do you want to draw? France

Enter desired flag width: 600

Are the stripes horizontal (h) or vertical (v)? v

You can assume that the user will give valid answers to your questions, but the h and v should be case-insensitive.

1. Next, ask the user to enter the colors of the flag's stripes on separate lines, followed by a single blank line.  
   You should read the user's inputs into a list; see [Lab 10, Part A](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab10a.html) if you need a refresher on building a list.
2. Print out the length and elements of the list.  
   Make sure that the blank line was not included in the list.  
   Sample output:
3. Which national flag do you want to draw? France
4. Enter desired flag width: 600
5. Are the stripes horizontal (h) or vertical (v)? v
6. Enter the colors of the stripes in order (one per line).
7. End with a blank line.
8. DarkBlue
9. white
10. red
11. Number of colors: 3
12. Color list: ['DarkBlue', 'white', 'red']
13. Inside of main(), write a line of code to create a graphical window with the following properties:
    * The width is the value supplied by the user
    * The height is 2/3 of the width
    * The title is the name of the country the user supplied
14. Call draw\_stripes to draw the stripes specified by the user. Don't forget to convert the user's 'h' or 'v' input to a True or False before you call draw\_stripes.
15. Remove the extra output you added in Step 2. Then run the following sample inputs and outputs:
16. Which national flag do you want to draw? Indonesia
17. Enter desired flag width: 600
18. Are the stripes horizontal (h) or vertical (v)? h
19. Enter the colors of the stripes in order (one per line).
20. End with a blank line.
21. red

white



Which national flag do you want to draw? Belgium

Enter desired flag width: 600

Are the stripes horizontal (h) or vertical (v)? v

Enter the colors of the stripes in order (one per line).

End with a blank line.

black

yellow

red



Which national flag do you want to draw? Mauritius

Enter desired flag width: 600

Are the stripes horizontal (h) or vertical (v)? h

Enter the colors of the stripes in order (one per line).

End with a blank line.

red

DarkBlue

yellow

green



1. When your main function is calling draw\_stripes correctly for all of these examples, call a lab instructor to demo your code for Part E.
2. [Continue to Part F.](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11f.html)

# Part F: Submitting the assignment

[[Back to Lab 11](http://rivoire.cs.sonoma.edu/cs115/labs/cs115_lab11.html)] [[Back to CS 115 schedule](http://rivoire.cs.sonoma.edu/cs115/schedule.html)]

## Instructions

1. Open FileZilla. Fill out the blanks at the top of the screen as follows:  
   Host: cwolf.cs.sonoma.edu  
   User: *your cwolf username*  
   Password: *your cwolf password*  
   Port: 22  
   Then click "QuickConnect".
2. The left half of the screen shows the files on your local machine. Navigate the directory tree to find your program:
   * In Mac OS X... Navigate to /users/student/Documents.
   * In Windows 7... Navigate to C:\Python32 or C:\Python33.

You should be able to see your Lab 11 python files in the file listing at the lower left.

1. The right half of the screen shows your files on cwolf. Navigate to your cs115/lab11 subdirectory.
2. Drag your Lab 11 .py files from the left-hand window to the empty directory in the right-hand window. Then exit FileZilla.
3. If you are not already logged into cwolf, log back in using PuTTY or Terminal, and navigate to your Lab 11 directory:

cd cs115/lab11

1. From your Lab 11 directory, run the following command:

~srivoire/cs115/pickup/test.sh lab11 lab11b.py

The script tests your program on a list of Sonoma County cities.  
If all of the tests say "OK", then your output matches the sample output and you should proceed to submit your code (skip the next step).  
If one or more of the tests gives you a URL, paste that URL into your browser to see the differences between your code and the sample code.

1. On cwolf, type the following command to submit your lab11b program:

~srivoire/bin/submit 115

When prompted for the assignment, type

L11

When prompted for the file to submit, you can hit ENTER to submit lab11b.py. If your file has a different name, type that name and then hit ENTER.

1. Wait about two minutes, and then check the submission web pages to verify that your file was submitted correctly:  
     
   If your files do not show up on this page after two minutes, talk to the instructor.
2. Answer the evaluation question in your writeup and submit it.
3. Complete the separate Week 11 survey in Moodle, and check out of lab.