Let's talk about functions for a minute.... Suppose you want to print the lyrics to the song Happy Birthday! You could write code like this:

```
print("Happy Birthday to you!")
print("Happy Birthday to you!")
print("Happy Birthday dear Dave!")
print("Happy Birthday to you!")
```

But what if you wanted your program to wish five of your friends a happy birthday? You would need twenty lines of code. Most of it is redundant. Programmer's hate redundancy. They like to simplify. You could begin by creating a simple function as shown below.

```
def HappyBirthday():
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday dear Dave!")
    print("Happy Birthday to you!")


HappyBirthday()
```

But what happens if all your friends aren't named Dave? You need to make your function more flexible. Python can help with parameters. You simply declare a parameter for your friend's name as shown below. Now you can wish a happy birthday to anybody!

```
def HappyBirthday(name):
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday dear {0}!".format(name))
    print("Happy Birthday to you!")

# only python code that is aligned to the left will run on its own.
# Call the function by simply calling its name and passing the variable called name
# which contains your friend's name from the prompt.
HappyBirthday(name)
```

So now we know something about functions. Let's create a function for calculating the scrabble score of a word. We'll apply this function to your movie titles and see which title has the best scrabble score.

```
history = open("myfile.txt",'a') #use w for write, a for append, r for read
while True:
    answer = input("Enter a movie you like or press 'q' to quit: ")
    if answer.lower() == 'q':
        break
    else:
        history.write(answer + "\n")
        #print("You entered " + answer)
history.close()

#now open the file and read it into a dictionary
history = open("myfile.txt",'r')
while True:
    line = history.readline()
    if line == "":
        break
    print(line,end="")
#It is not necessary to close the file


print("")
print("You will now exit the program.")
```

```
# How can Python tell me What directory my file is in???
# can you find your file? Open it up in Notepad++ and add another movie at the end.
# when you read your list the next time you will see your additional movie.
import os
print("The current working directory is " + os.getcwd())

#--------------------------------------------------------------------------------
'''
Your next assignment is to read the history file into a list then sort the list
'''
movielist = [] #create an empty list
history = open("myfile.txt",'r')
while True:
    #read each line of the file into the variable called line
    line = history.readline()
    # if the line contains no data (it is an empty string) then exit the loop. You're done.
    if line == "":
        break
    else:
        # if the line contains data then append it to the list
        movielist.append(line)



# Sorting a list is simple
movielist.sort()

# loop through each movie title in your list and print each, along with a number
# I use index as my loop counter and I start with -1 so I loop through all movies
# Try changing index to 0 or 1... what happens? Do you see all your movies?
index = -1
while index < len(movielist)-1:
    print("{0}. {1}".format(index+2,movielist[index]),end="")
    index +=1


# if you simply pass the list variable to the print statement it will print but not in a
format you may like.
# what are the \n's?
print("\n")
print("Another way to print a list:\n")
print(movielist)
```

```
"""
File: concordance.py
Project 5.8

Prints the unique words in a text file and their frequences.
"""

# Take the input file name
inName = input("Enter the input file name: ")

# Open the input file and initialize list of unique words
inputFile = open(inName, 'r')
uniqueWords = {}

# Add the unique words in the file to the list
for line in inputFile:
    words = line.split()
    for word in words:
        freq = uniqueWords.get(word, None)
        if freq == None:
            uniqueWords[word] = 1
        else:
            uniqueWords[word] = freq + 1

# Prints the unique words and their frequencies,
# in alphabetical order
words = list(uniqueWords.keys())
words.sort()
for word in words:
    print(word, uniqueWords[word])
```

```
quit = False
amounts = {}
while quit == False:
    nmbr = input("Enter a number or press (q) to quit: ")

    if nmbr.lower()=='q'.lower():
        quit = True
    else:
        key = int(nmbr)
        if amounts.get(key):
            print("You have entered " + amounts[key])
        else:
            value = input("Enter the value or press (q) to quit: ")
            if value.lower() != 'q'.lower():
                amounts[key] = value
            else:
                quit = True

print("\nHere are the values you have entered:")
theKeys = list(amounts.keys())
theKeys.sort()
for key in theKeys:
    print(key,amounts[key])
```

```
quit = False
amounts = {}
while quit == False:
    nmbr = input("Enter a number or press (q) to quit: ")

    if nmbr.lower()=='q'.lower():
        quit = True
    else:
        key = int(nmbr)
        if amounts.get(key):
            print("You have entered " + amounts[key])
        else:
            value = input("Enter the value or press (q) to quit: ")
            if value.lower() != 'q'.lower():
                amounts[key] = value
            else:
                quit = True

print("\nHere are the values you have entered:")
theKeys = list(amounts.keys())
theKeys.sort()
for key in theKeys:
    print(key,amounts[key])
```

Dictionaries A python dictionary contains keys and values. Sometimes they're called key-value pairs. Often they're called hash tables. By any name they organize information by association. So the values can be located by use of the corresponding key. A list, on the other hand, organizes its elements by position. So you could easily find the third item in a list. A dictionary doesn't have a third item. It does, actually, but you don't access it that way. You would access the value by knowing the corresponding key you gave it. The keys and values in a python dictionary can be of any type, not just integers and strings. You could even have a dictionary that contains values which are lists or other dictionaries.

A python dictionary, like a regular dictionary makes a great look up table. So if you wanted to store the cities of some states you could create a dictionary like this: cities = {'CA': 'San Francisco','MI': 'Detroit','FL': 'Jacksonville'}

Notice that the dictionary uses the {} characters (sometimes called brackets). A list uses the [] characters. I don't know what they are called - square brackets?!

You can access a state's city as: city = cities['CA']. Now the variable called city will contain the value 'San Francisco'. You can add a city as: cities['OR'] = 'Portland'.

If you want to print a sorted list of the cities for each state in your program you would do something like this: Note that I converted the dictionary to a list so I could sort it because you can't sort a dictionary.

theKeys = list(amounts.keys()) theKeys.sort() for key in theKeys: print(key,amounts[key])

There are more examples in the book and online. Here is a link if you want more information: http://learnpythonthehardway.org/book/ex39.html.

So, let's use the dictionary in a program.

1. Create a program that will prompt the user to enter an amount. When the user enters the amount, the program will respond with the text spelling out of the amount. For example, the user enters '10', the program responds with 'ten'.
2. If the user enters an amount that the program does not have stored then the program shall prompt the user to enter the text. So if the user enters '201' and the program doesn't know that value then the program will prompt the user to type in the text, 'two-hundred one'.
3. Once the user types in the text, the program shall store the text in a dictionary. This will enable the program to quickly locate the value the next time the user enters it.