Related List

The lists called *name* and *mark* are **related** because the information in corresponding elements (ie. elements with the same index or subscript) are about the same person or thing. In this case, “Opey” is located in element *name [2]* and his mark is located in element *mark [2]*.

Obviously, the lists would only stay related as long as both lists remain exactly the same they can't be sorted or changed in any way.

A Program Loading and Using Related Lists

''' A program to load the names and marks of 5 people into two lists called name and mark from the user. The user will then be asked to input a name and have thecorresponding mark outputted.'''

```
# Creates two empty lists one for a student's name and one for the student's mark
name = []
mark = []

for i in range (0, 5):              # A for loop to load the list by having the user input the data
        name.append (input ("Enter student name: "))
         mark.append (input ("Enter student mark: "))

# Section asking the user to have a name inputted and have the corresponding mark outputted
while True:
        choice = input ("Enter student name you are searchig for: ")
        element = name.index (choice)       # Finds the index of the name inputted in the list name
                                            # and stores it in the variable called element

        # The next line outputs the name of the student stored in the variable choice and the
        # corresponding mark stored in the element list [ choice ]
        print (choice,"'s mark is", mark [element])

        ans = input ("Check another? ")
         if ans == "no" or ans == "No":
                break
```

Notes on the Program Above

If the user runs the program above and inputs the lists shown on the example at the top of the page, they will then be asked to input a name. If they input the name "Axle", the variable choice will have the value of 3. The message outputted will be "Axle's mark is 81"because the value of *mark [3]* is 81.

This program has two problems with it:

1. If the name inputted by the user is not in the list, the program will crash. This can be solved by inserting a condition (if) before you find the index to see if there is at least one instance of the name the user is looking for using the count function.
2. If there are more than one instance of the name you are looking for (ie. 2 Axle's), only the mark of the first one would be outputted. Solving this is a much more difficult program and requires sublist.