**The Search Algorithm**

An **algorithm** is a list of steps required to accomplish a task. A recipe is an example. The list of instructions in a manual is another example of an algorithm. This unit of the course we look at some very common and useful algorithms in computer science. The first lesson is on the most common algorithm of all: the search algorithm.

The **search** **algorithm** searches for a particular element in a list of elements. At the end of the algorithm we want to state if the element was found or not found.

We already know how to do this using the **in** command. Suppose we have an array of names:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

To see if our name is in the list, we use a simple IF statement:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

if search **in** names:

print("The name", search, "is in the list!")

And if the name isn’t there, we add an ELSE statement:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

if search **in** names:

print("The name", search, "is in this list!")

**else:**

print("The name", search, "is NOT in this list!")

Python has a special search algorithm inside the **in** command. Our task will be to write our own version of this algorithm. How can you do the above code ***without*** using **in**?

**The Search Algorithm**

How do you search for an element in a list without using the **in** command?

For a search, we will follow this algorithm:

1. Start with a list of names
2. Ask the user for a name to search for.
3. Go through each name in the list, starting with the first name in the list:
4. Is this the name we are looking for? If so, remember that we found our name and quit searching.
5. Repeat step 4 with the next name until we finish all the names in the list.
6. State whether we found the name or not.

We have our list of names, and we know who to search for. Now you have to go through the list one element at a time (step 3). To do this, we use a FOR loop. We can use an indexed loop or a non-indexed loop. To make things simple, we will use a non-indexed loop. This will print out the names one at a time:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

for name in names:

print (name)

We are already at step 4. Now we need to declare if we find a matching name:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

for name in names:

if name == search:

print("We found it!")

break

Try this out. Does it work?

It works, but what if the name is not in the list? We can try using an ELSE statement:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

for name in names:

if name == search:

print("We found it!")

else:

print("Sorry, the name is not on the list")

Oh-oh! What happens? The program is reacting to each name on the list, stating if it is a match. We don’t want to do this exactly. Step 4 says we want to remember if we find a match, without printing anything yet. To do this we need a **flag**.

Just to remind you, a **flag** is a Boolean variable that keeps track of a state. In this case, the state is whether the item was found yet or not. We will call the flag “found”. If we find our item, we will set it to True.

Our first step is to set the flag to false before we start our search:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

**found = False # we haven’t found the name… yet!**

for name in names:

if name == search:

print("We found it!")

else:

print("Sorry, the name is not on the list")

Next we take out the print statements, for now, and just set the flag. When do we set the flag to True? The flag is set in the if statement (note I did not add the first part of the program):

for name in names:

if name == search:

found = True # remember that we found it

break

Finally, when the loop is finished, we check the flag by adding an if...else statement at the end:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

**found = False # we haven’t found the name… yet!**

for name in names:

if name == search:

**found = True # we found it**

break # quit looking

**if flag == True: # did we find it...**

print("We found it!")

else: # or not?

print("Sorry, the name is not on the list")

I have added a small error to the code. Can you fix it? [hint: the flag is not named “flag”]

Now we know how to complete a search in Python. Now go to the next page to see how to do this with indexing.

**Searching Deeper: Using an indexed search**

Sometimes you just want to know if an element is in a list, as we did in the previous example. Sometimes you need to know more information - for example, which element is it? For this you will need to use an indexed loop. Since we’ve done indexed loops before, I will just show you the result:

names = ["Joe", "Julie","Jamila","Jerry"]

search = input("Enter a name to search: ")

found = False # we haven’t found the name… yet!

for i in range(len(names)):

print (i)

if names[i] == search:

found = True # we found it

index = i # remember the index

break # quit looking

if found == True: # did we find it...

print("We found it! It was name", index, "in the list")

else: # or not?

print("Sorry, the name is not on the list")

Exercise

1. Create an array called *names* and an array called *marks*. Put 4 names in the names array and 4 corresponding marks (from 0 to 100) in the marks array. Have the program ask the user for a name. If the name is in the array, print the name and their mark. If the name isn’t in the array, print an appropriate message.