

Remembering lists of values lists



Susan Ibach | Technical Evangelist
Christopher Harrison | Content Developer

Sometimes you have to remember lists of values

- I want to remember the names of everyone coming to a party
- I want to remember the scores I got in all my courses
- I want to remember the directions to get to my doctor appointment

Multiple values

Lists allow you to store multiple values

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
scores = [78, 85, 62, 49, 98]
```

You can create an empty list and add values later

```
guests = []
```


```
scores = []
```

You can reference any value in the list by specifying it's position in the list

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#print the first guest  
#the first value is in position 0  
print(guests[0])
```

```
scores = [78,85,62,49,98]  
#Print the fourth score  
print(scores[3])
```

A screenshot of a Python terminal window. The title bar at the top is brown and contains a small icon on the left and the file path 'C:\Python34\python' on the right. The terminal area has a black background with white text. It shows the output of the first two print statements: 'Christopher' on the first line and '49' on the second line. Below the output, the text 'Press any key to continue . . .' is displayed.

```
C:\Python34\python  
Christopher  
49  
Press any key to continue . . .
```

Geek Tip

- We call the position of an item in the list the index

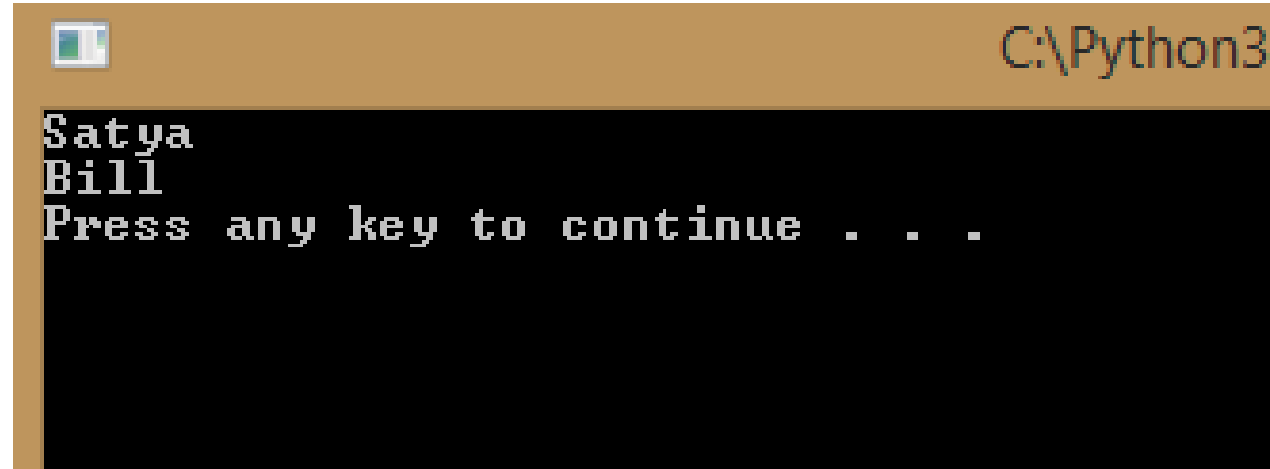


You can even count backwards

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#print the last entry in the list  
print(guests[-1])
```

```
#print the second last entry in the list  
print(guests[-2])
```



```
C:\Python3  
Satya  
Bill  
Press any key to continue . . .
```


DEMO

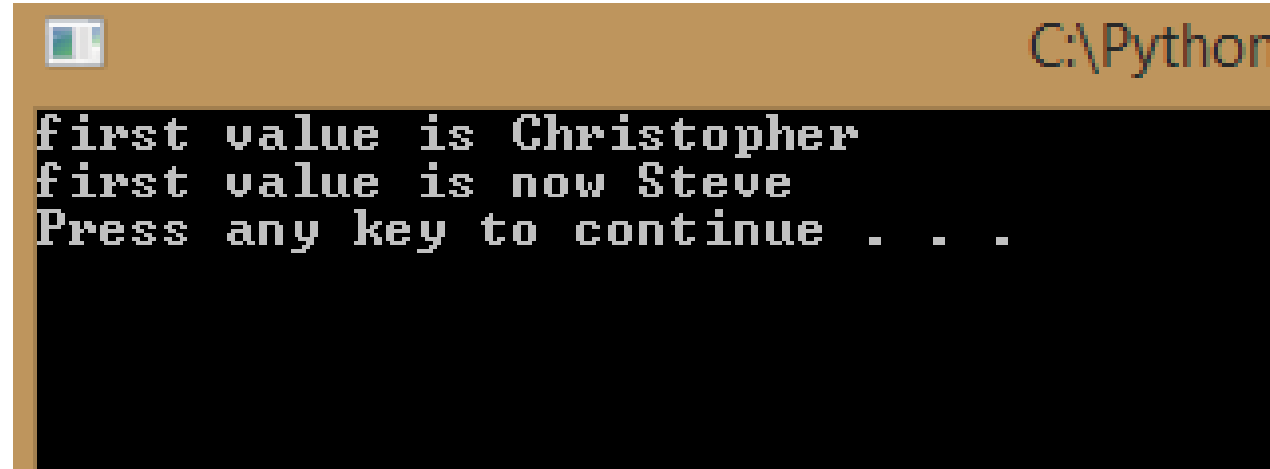
Creating and populating a list

Updating lists

You can change a value in a list

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']  
print("first value is " + guests[0])
```

```
#change the first value in the list to Steve  
guests[0] = 'Steve'  
print("first value is now " + guests[0])
```



```
C:\Python  
first value is Christopher  
first value is now Steve  
Press any key to continue . . .
```

You can add a value to a list with `append()`

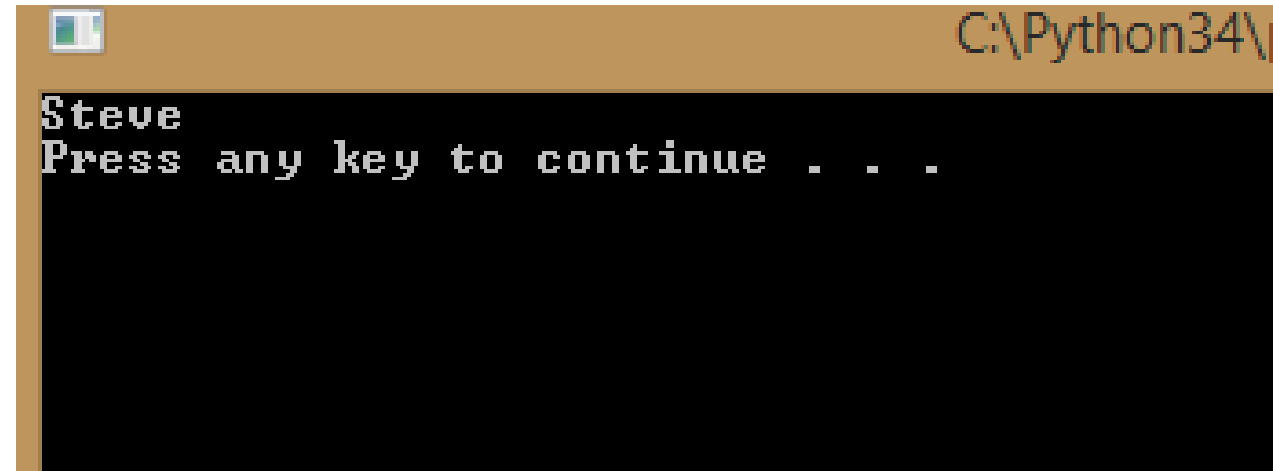
```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#add a new value to the end of the list
```

```
guests.append('Steve')
```

```
#display the last value in the list
```

```
print(guests[-1])
```



```
C:\Python34\  
Steve  
Press any key to continue . . .
```

You can remove a value from a list with remove()

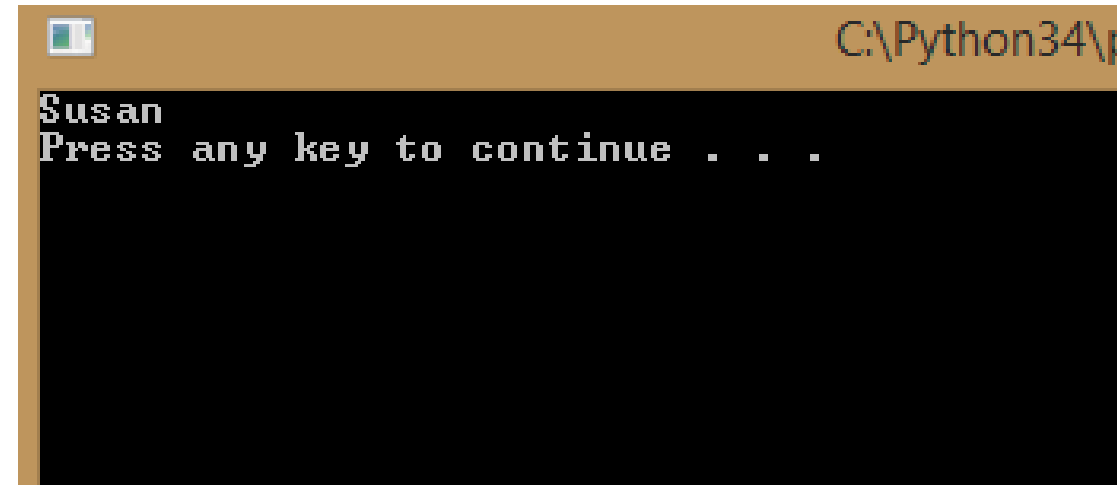
```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#add a new value to the end of the list
```

```
guests.remove('Christopher')
```

```
#display the last value in the list
```

```
print(guests[0])
```



```
C:\Python34\p  
Susan  
Press any key to continue . . .
```

You can use the del command to delete an entry

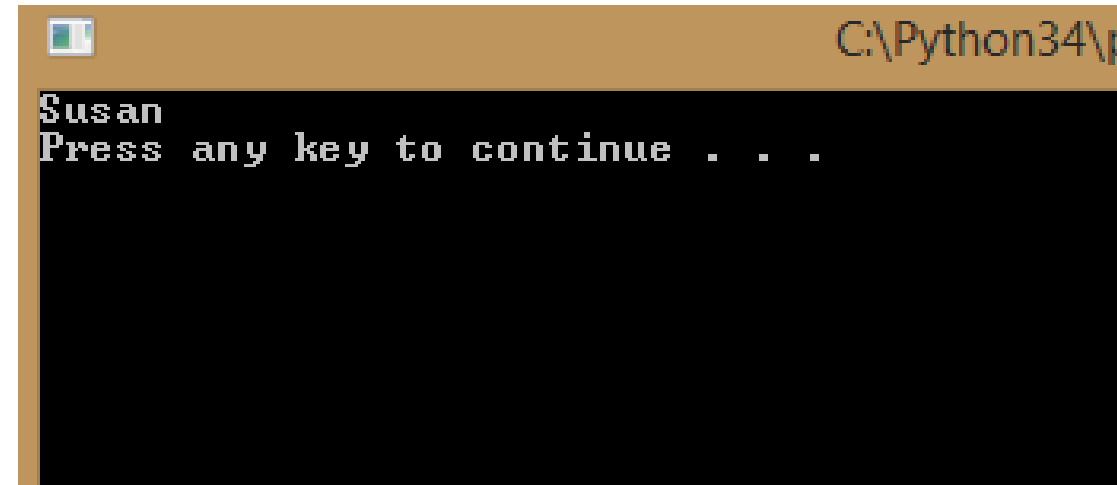
```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#delete the first item in the list
```

```
del guests[0]
```

```
#print the first item in the list
```

```
print(guests[0])
```



```
C:\Python34\p  
Susan  
Press any key to continue . . .
```

DEMO

Modifying list contents

Finding values

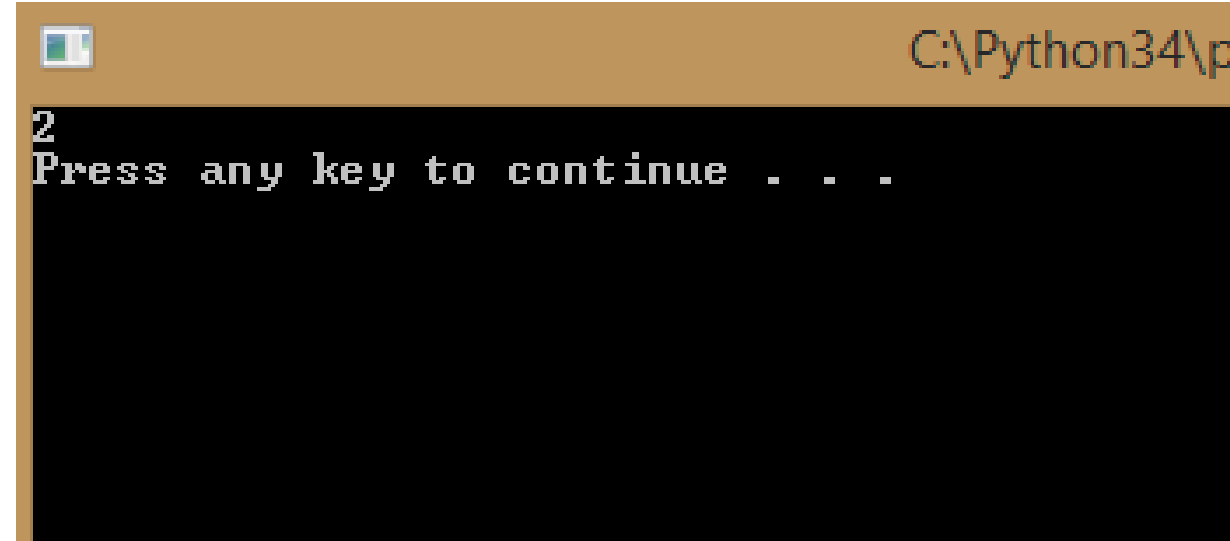
The `index()` function will search the list and return the index of the position where the value was found

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#this will return the index in the list
```

```
#where the name Bill is found
```

```
print(guests.index('Bill'))
```



A screenshot of a Windows command prompt window with a brown title bar. The title bar contains a small icon on the left and the text "C:\Python34\p" on the right. The command prompt itself has a black background with white text. The first line shows the number "2", which is the output of the `index()` function. The second line shows the prompt "Press any key to continue . . .".

```
C:\Python34\p
2
Press any key to continue . . .
```

DEMO

Searching a list

What do you think will happen if we search for a value that doesn't exist in the list?

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#this will return the index in the list
```

```
#where the name Steve is found
```

```
print(guests.index('Steve'))
```

The code crashes!

We need to add error handling

Or find another way to go through the list and find a value

Displaying values

Use a loop!

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#Create a loop that executes four times
```

```
#Since we have four values
```

```
for steps in range(4) :
```

```
    #Remember the value of steps goes up by one
```

```
    #Each time the loop executes
```

```
    print(guests[steps])
```



```
Christopher
```

```
Susan
```

```
Bill
```

```
Satya
```

```
Press any key to continue . . .
```

DEMO

Looping through all the values in a list

What if I don't know how many values are in the list?

Use the len() function to find out how many entries are in your list

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#Find out how many entries are in the list
```

```
nbrEntries = len(guests)
```

```
#Create a loop that executes once for each entry
```

```
for steps in range(nbrEntries) :  
    print(guests[steps])
```



```
Christopher  
Susan  
Bill  
Satya  
Press any key to continue . . .
```


Shhhh, don't tell anyone but
there is an even easier way to
go through all the items in a list

You can just tell the for loop to go through your list!

```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#specify the name of your list and a variable name
```

```
#to hold each entry as you go through the loop
```

```
for guest in guests :
```

```
    #the variable guest will contain the values
```

```
    #as we go through the loop
```

```
    print(guest)
```



```
Christopher
```

```
Susan
```

```
Bill
```

```
Satya
```

```
Press any key to continue . . .
```

Want to sort your list?

You can sort your list with the sort() function

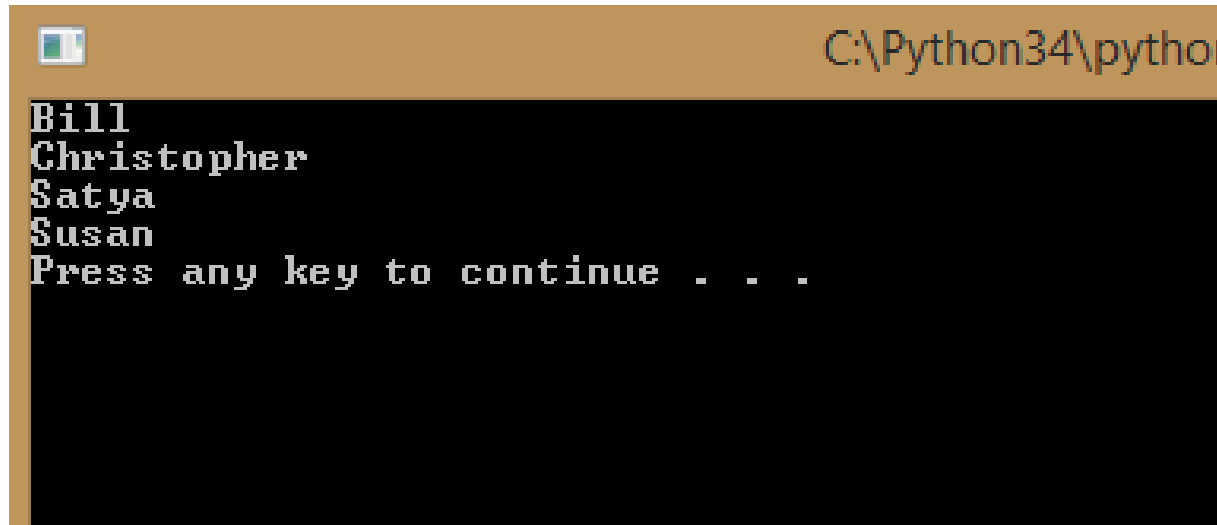
```
guests = ['Christopher', 'Susan', 'Bill', 'Satya']
```

```
#Sort the names in alphabetical order
```

```
guests.sort()
```

```
#print the list
```

```
for guest in guests :  
    print(guest)
```



```
C:\Python34\python  
Bill  
Christopher  
Satya  
Susan  
Press any key to continue . . .
```

DEMO

Sort a list and print the results

Your challenge... Starting to get harder...

- Ask the user to enter the names of everyone attending a party
- Then return a list of the party guests in alphabetical order
- This will require pulling together everything we have learned so far, so let's walk through the thought process of idea to code

Break the problem into steps

1. Ask the users to enter the names of everyone attending a party
2. Put those values in a list
3. Sort the list
4. Print the sorted list

1. Ask the user to enter the names of everyone attending a party

- What command do we use to ask a user for a value?
 - input function
- What type of variable will we need to store all the names?
 - A list
- How can I ask the user for more than one name?
 - Use a loop

Should we use a for loop or while loop?

- Do you know how many names the user will enter?
 - No, that means we don't know how many times the loop needs to execute, so we should use a while loop
- How will the loop know when to stop executing?
 - We could have user enter a special keyword when they are done (as long as we tell them to do it!)

2. Put those values in a list

- Declare an empty list
- Each time a new name is entered, add it to the list

3. Sort the list

- Once the values are in a list, use the sort function to sort the list alphabetically

4. Print the sorted list

- Use a loop to go through the values in the list
- For each value, print the name

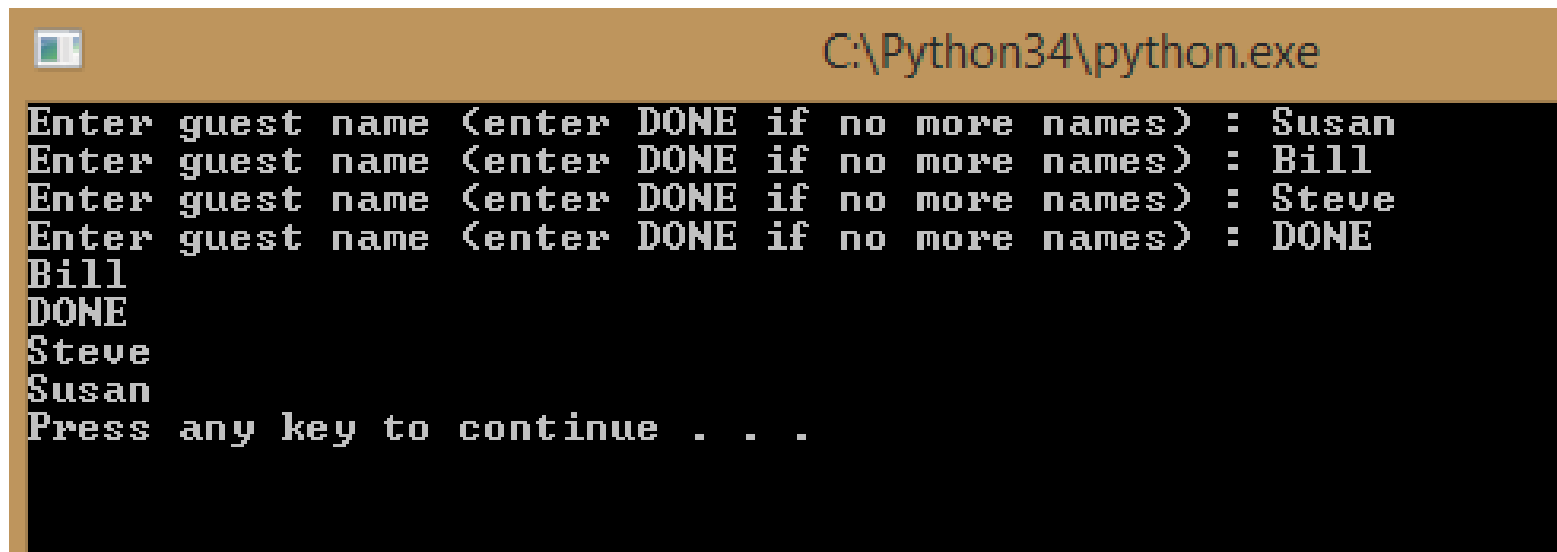
So... something like this?

```
guests = []  
name = ""
```

```
while name != "DONE" :  
    name = input("Enter guest name (enter DONE if no more names) : ")  
    guests.append(name)
```

```
guests.sort()  
for guest in guests :  
    print(guest)
```

We are close but our code added the name DONE to our list of guests
How can we tell the program that if the name is "DONE" not to add it?



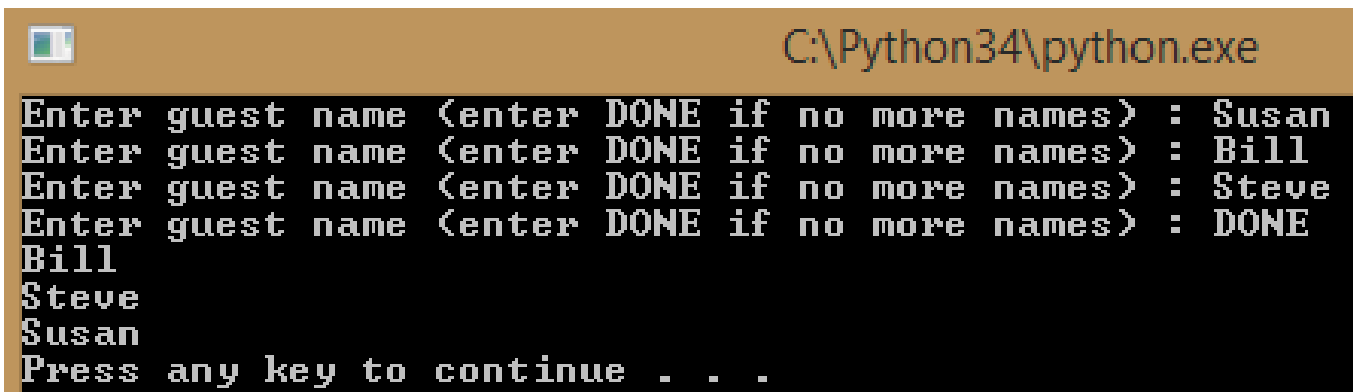
```
C:\Python34\python.exe  
Enter guest name (enter DONE if no more names) : Susan  
Enter guest name (enter DONE if no more names) : Bill  
Enter guest name (enter DONE if no more names) : Steve  
Enter guest name (enter DONE if no more names) : DONE  
Bill  
DONE  
Steve  
Susan  
Press any key to continue . . .
```

Use an if statement. You are gradually building a toolkit to solve different problems!

```
guests = []  
name = ""
```

```
while name != "DONE" :  
    name = input("Enter guest name (enter DONE if no more names) : ")  
    if name.upper() != "DONE" :  
        guests.append(name)
```

```
guests.sort()  
for guest in guests :  
    print(guest)
```



A screenshot of a Windows command prompt window titled "C:\Python34\python.exe". The window shows the execution of a Python script. The prompt "Enter guest name (enter DONE if no more names) :" is displayed four times. The user enters "Susan", "Bill", "Steve", and "DONE" in sequence. After the fourth prompt, the program prints the sorted list of names: "Bill", "Steve", and "Susan". The prompt "Press any key to continue . . ." is shown at the bottom.

```
C:\Python34\python.exe  
Enter guest name (enter DONE if no more names) : Susan  
Enter guest name (enter DONE if no more names) : Bill  
Enter guest name (enter DONE if no more names) : Steve  
Enter guest name (enter DONE if no more names) : DONE  
Bill  
Steve  
Susan  
Press any key to continue . . .
```

Congratulations!

- You can now remember a list of different values
- You can search the list for a specific value
- You can sort the list
- You can read through all the values in the list





Microsoft

©2013 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.