EDULiTO

A Complete Guide to Python Programming Part 3

7. The use of Lists (arrays), including one and two dimensional arrays
8. File handing operations (open, read, write and close)

[illegible]

Photocopiable Resources

Terms and Conditions of Use

7. The use of Lists (arrays), including one and two dimensional arrays

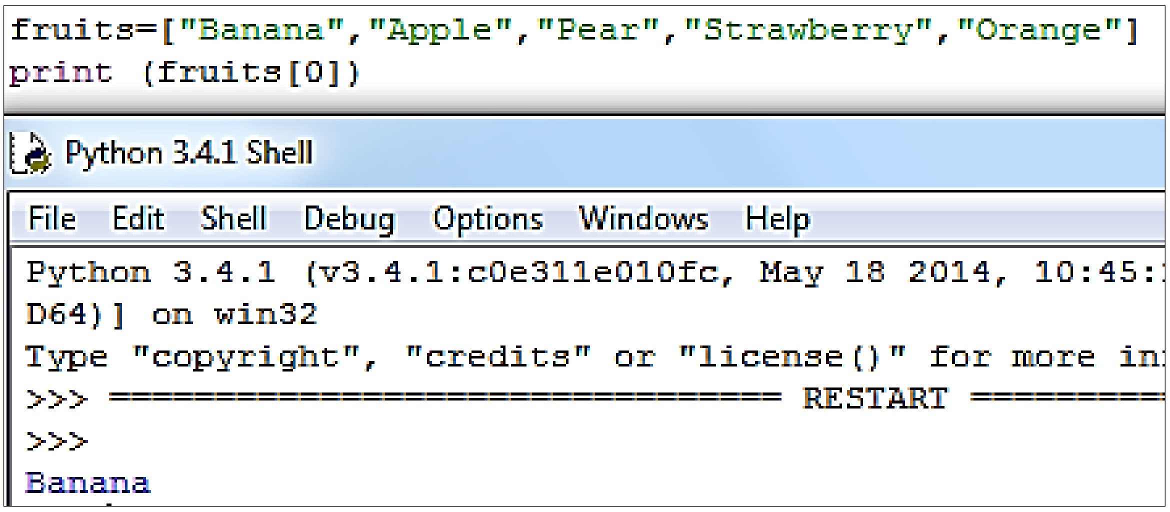
The limitation of using variables is that they only store one item of data at any one time. An Array, which is called a **List** in python, is a data structure that can store many items of data. This data can then be viewed, manipulated and accessed.

When creating a list in Python you must use square brackets. As with **variables**, an item in a list can be the data type: string, integer, float or Boolean.

Here is a list of fruits. In this case each item has the data type **string**. An index is used to locate the position of an item in the string. Banana is at the index location 0 and Pear is at the index location 2.

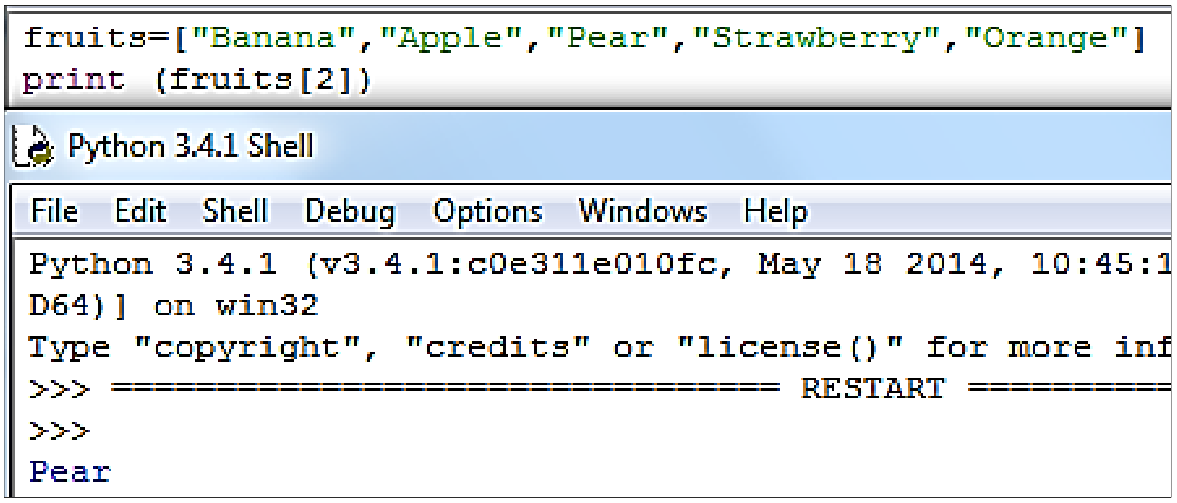
```
Index position:              0          1        2            3            4
                   fruits=[“Banana”,”Apple”,”Pear”,”Strawberry”,”Orange”]
```

In Python you can use the print statement and the index number to find an item.



```
fruits=["Banana","Apple","Pear","Strawberry","Orange"]
print (fruits[0])
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:
D64)] on win32
Type "copyright", "credits" or "license()" for more in
>>> ================================ RESTART =========
>>>
Banana
```

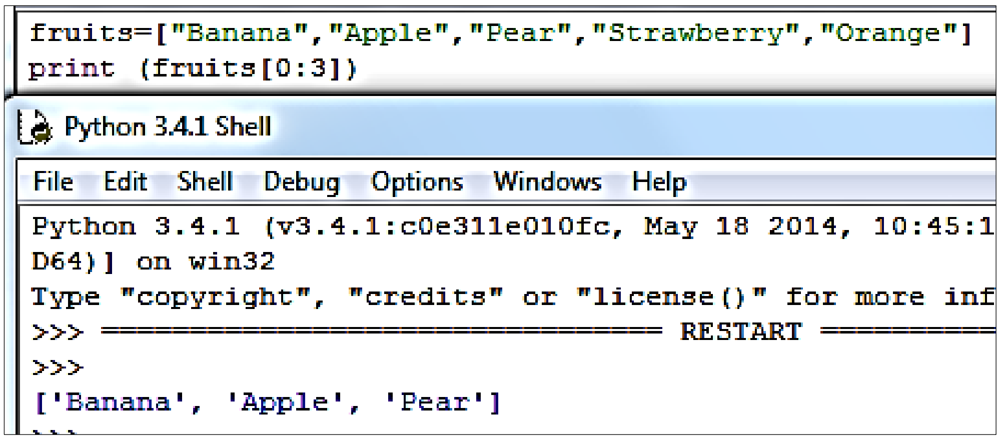


```
fruits=["Banana","Apple","Pear","Strawberry","Orange"]
print (fruits[2])
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:1
D64)] on win32
Type "copyright", "credits" or "license()" for more inf
>>> ================================ RESTART =========
>>>
Pear
```

Slicing items in a list

It is possible to select a particular section of the list. This is called **slicing**.
The example below shows a sliced list that just shows items at index 0, 1 and 2 (This is 0 to the item before 3)



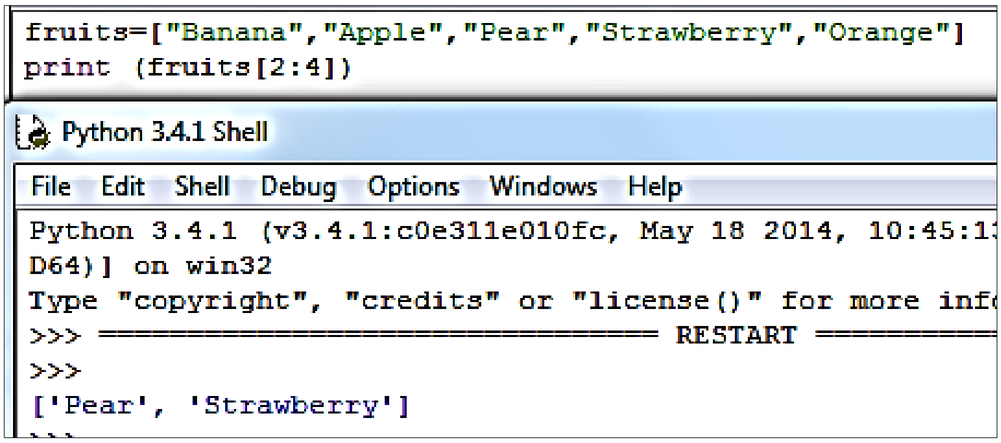
```
fruits=["Banana","Apple","Pear","Strawberry","Orange"]
print (fruits[0:3])
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:1
D64)] on win32
Type "copyright", "credits" or "license()" for more inf
>>> ================================ RESTART ==========
>>>
['Banana', 'Apple', 'Pear']
```

The example below shows a sliced list that just shows items at index 2 and 3. (It stops before the last number is reached).



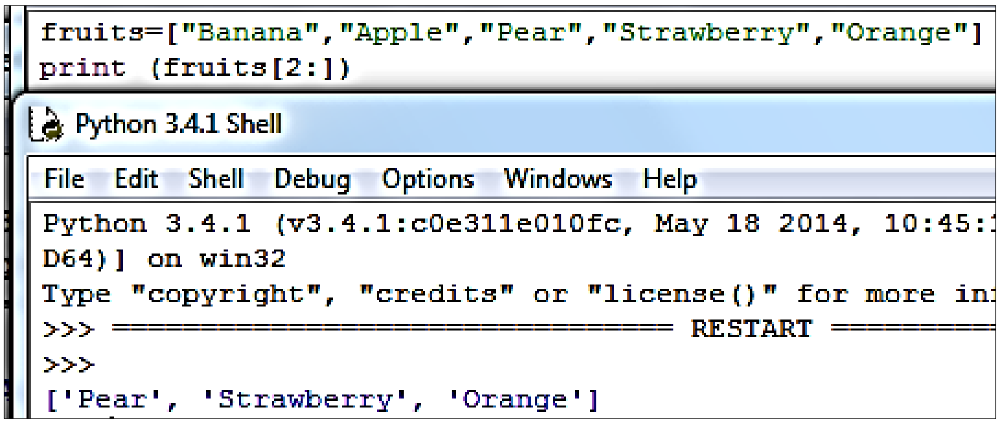
```
fruits=["Banana","Apple","Pear","Strawberry","Orange"]
print (fruits[2:4])
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:1
D64)] on win32
Type "copyright", "credits" or "license()" for more inf
>>> ================================ RESTART ==========
>>>
['Pear', 'Strawberry']
```

The example below shows a sliced list that displays from index 2 to the end of the list.



```
fruits=["Banana","Apple","Pear","Strawberry","Orange"]
print (fruits[2:])
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

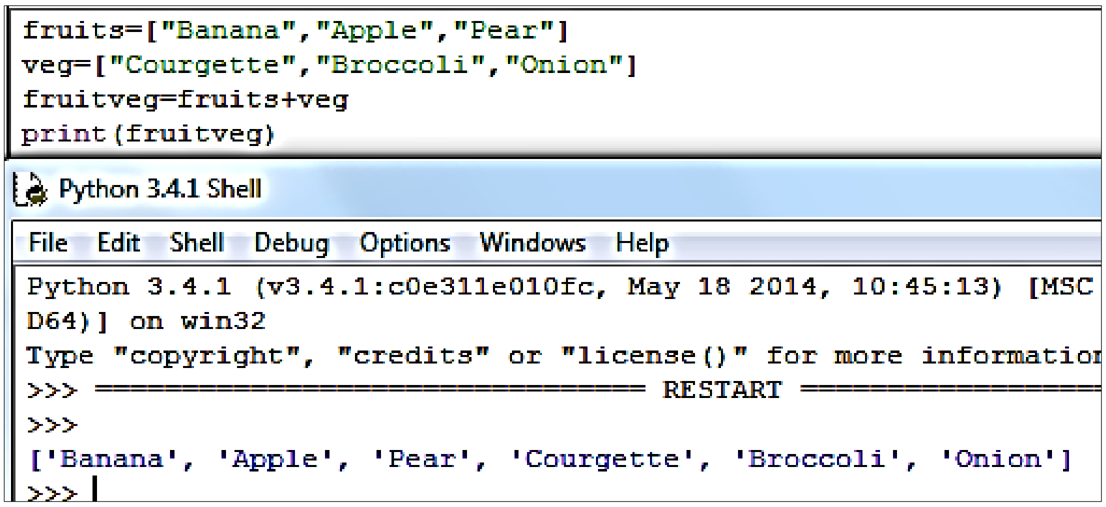
```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:
D64)] on win32
Type "copyright", "credits" or "license()" for more in
>>> ================================ RESTART =========
>>>
['Pear', 'Strawberry', 'Orange']
```

Activity 7.1

1 a. Create a list (Array) of planets that are part of our solar system. The planet nearest to the Sun must be located at index 0 and as the index number increases each planet is further from the Sun.
b. Slice the list to show the 6 planets that are nearest to the Sun.
c. Slice the list to show the 3 planets that are furthest away from the Sun.
d. Slice the list to show all of the planets that are further from the Sun than Earth.

Concatenate (join together) lists and sorting lists

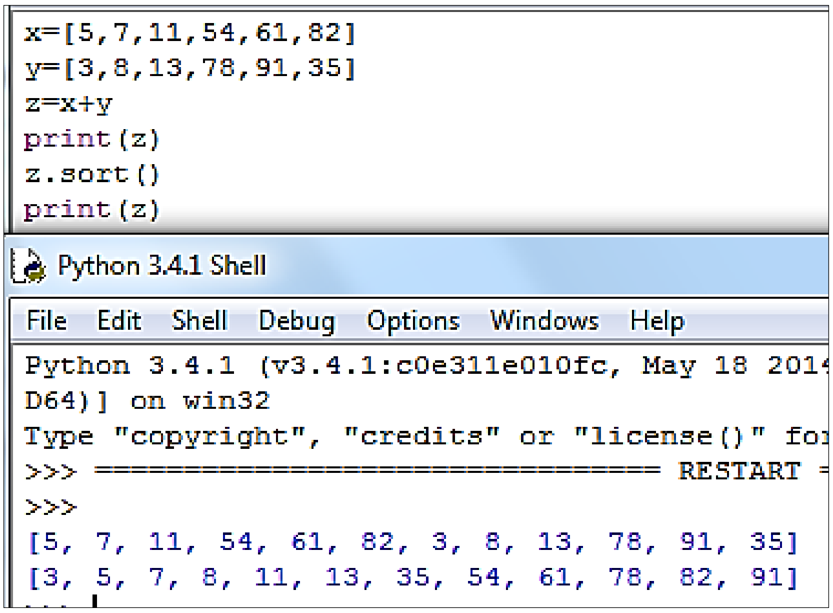
It is possible to use concatenate to join lists together.



```
fruits=["Banana","Apple","Pear"]
veg=["Courgette","Broccoli","Onion"]
fruitveg=fruits+veg
print(fruitveg)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC
D64)] on win32
Type "copyright", "credits" or "license()" for more informatio
>>> ================================ RESTART ==================
>>>
['Banana', 'Apple', 'Pear', 'Courgette', 'Broccoli', 'Onion']
>>> 
```

You can also concatenate lists that are made up of Integers (see below) and in this case you can also use another built-in function (**sort**) to sort the list into numerical order.



```
x=[5,7,11,54,61,82]
y=[3,8,13,78,91,35]
z=x+y
print(z)
z.sort()
print(z)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 201
D64)] on win32
Type "copyright", "credits" or "license()" fo
>>> ================================ RESTART =
>>>
[5, 7, 11, 54, 61, 82, 3, 8, 13, 78, 91, 35]
[3, 5, 7, 8, 11, 13, 35, 54, 61, 78, 82, 91]
```

Activity 7.2

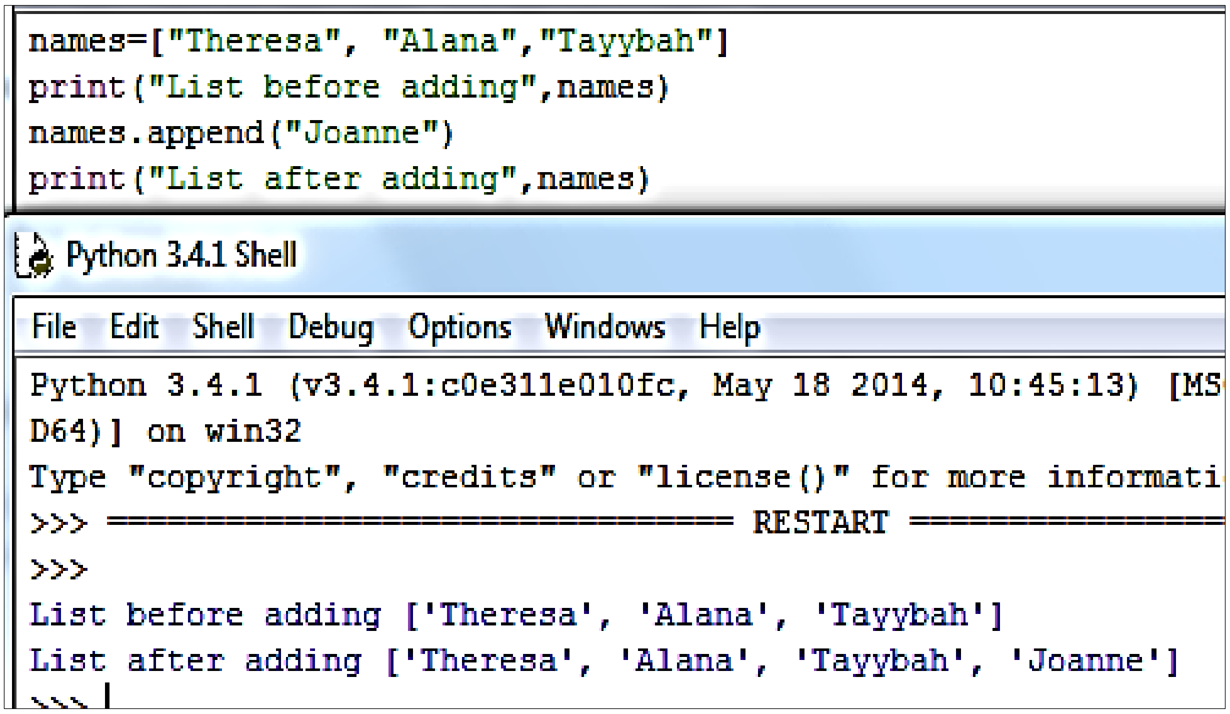
1 a. Create a list (Array) of planets that are part of our solar system (or use the list from activity 7.1). The planet nearest to the Sun must be located at index 0 and as the index number increases each planet is further from the Sun.
b. Now use the sort function to sort the planets into alphabetical order and then display them as a sorted list.

2. Here are two lists:
form10a=[”Jones”,”Ahmed”,”Adams”,”Graham”]
form10b=[”Collins”,”Peters”,”Khan”,”Langley”]
The forms have been merged and you have been asked to display the surnames of all of the students in the new form in alphabetical order.

3. Concatenate these two lists:
a=[1,2,3,4,5]
b=[6,7,8,9,10]

Adding and Inserting items into a list

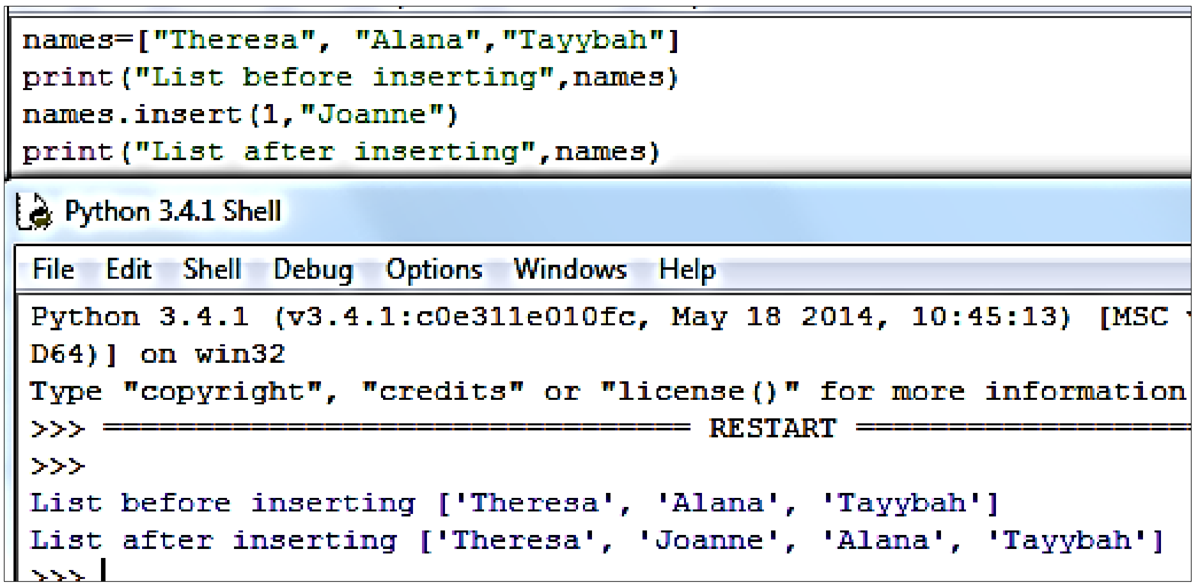
Adding - Sometimes when creating a program you start with an empty or incomplete list. You can easily add to the list – This is called **appending**.



```
names=["Theresa", "Alana","Tayybah"]
print("List before adding",names)
names.append("Joanne")
print("List after adding",names)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MS
D64)] on win32
Type "copyright", "credits" or "license()" for more informati
>>> ================================ RESTART ===============
>>>
List before adding ['Theresa', 'Alana', 'Tayybah']
List after adding ['Theresa', 'Alana', 'Tayybah', 'Joanne']
>>>
```

Inserting - You can also insert an item into a particular location in a list. Notice that in the example below the name has been inserted at index location 1.



```
names=["Theresa", "Alana","Tayybah"]
print("List before inserting",names)
names.insert(1,"Joanne")
print("List after inserting",names)
```

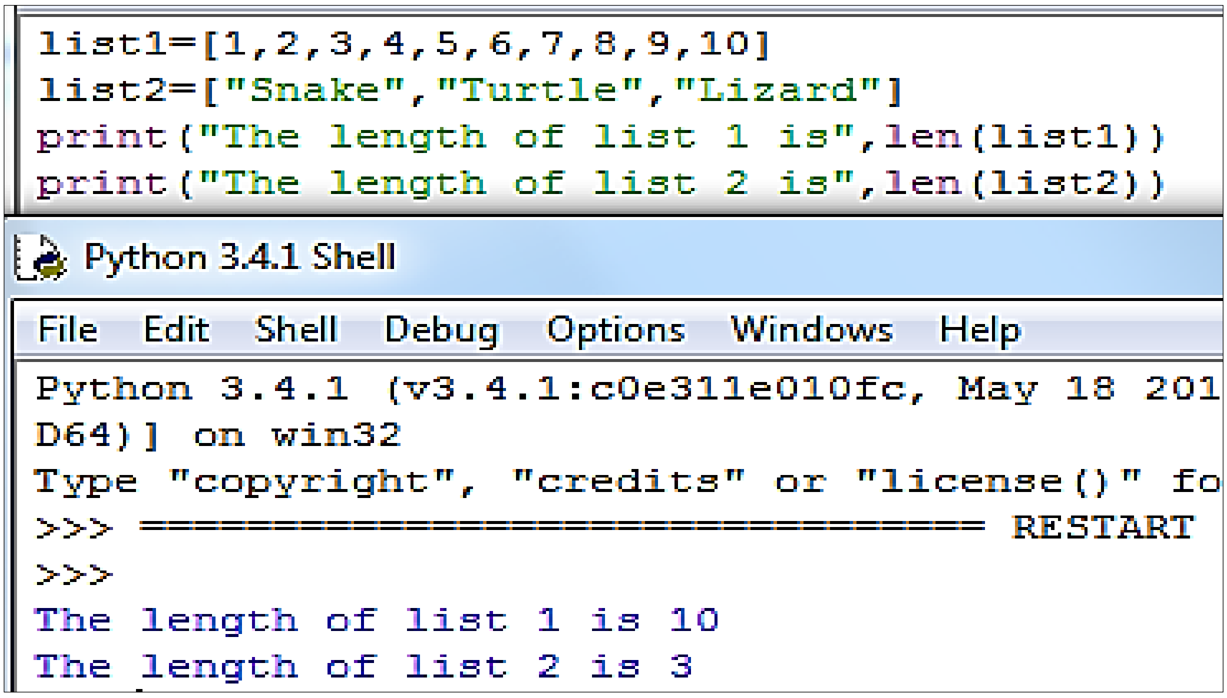
Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC
D64)] on win32
Type "copyright", "credits" or "license()" for more information
>>> ================================ RESTART ==================
>>>
List before inserting ['Theresa', 'Alana', 'Tayybah']
List after inserting ['Theresa', 'Joanne', 'Alana', 'Tayybah']
>>>
```

How long is the list?

The method you have used to find the length of a string is similar to the method required to find the length of a list.



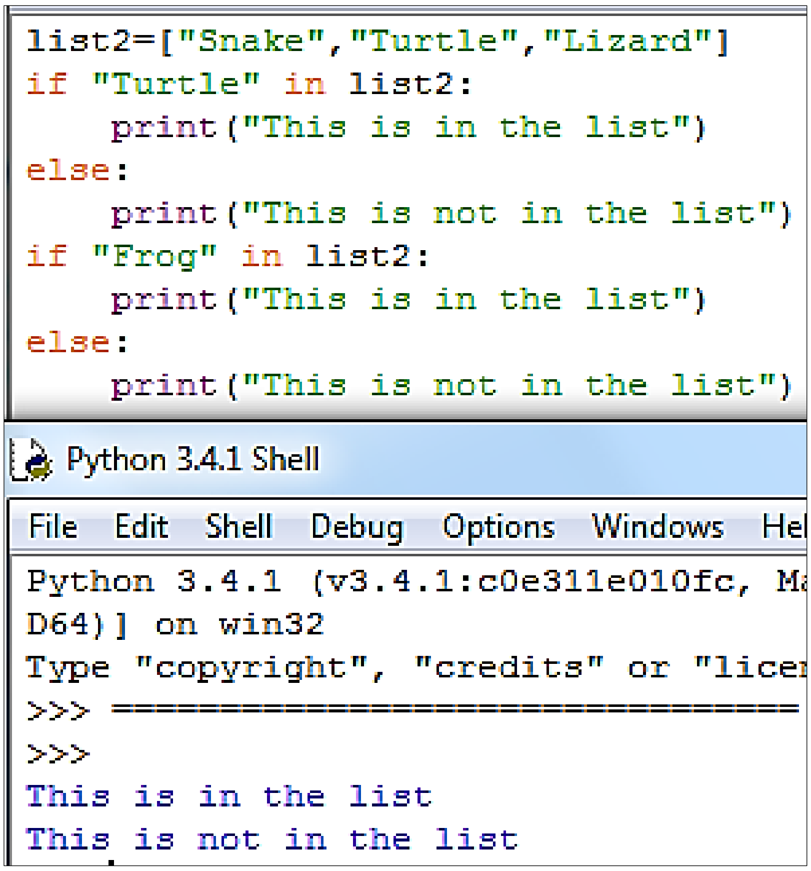
```
list1=[1,2,3,4,5,6,7,8,9,10]
list2=["Snake","Turtle","Lizard"]
print("The length of list 1 is",len(list1))
print("The length of list 2 is",len(list2))
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 201
D64)] on win32
Type "copyright", "credits" or "license()" fo
>>> ================================== RESTART
>>>
The length of list 1 is 10
The length of list 2 is 3
```

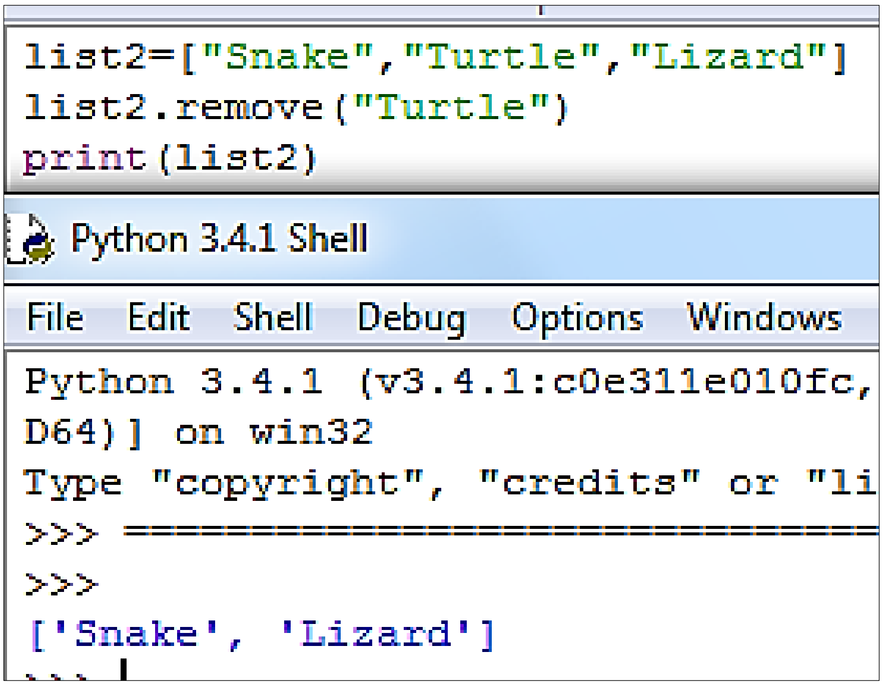
Is an item in the list?
It is easy to check whether an item is in a list using **in**.



```
list2=["Snake","Turtle","Lizard"]
if "Turtle" in list2:
    print("This is in the list")
else:
    print("This is not in the list")
if "Frog" in list2:
    print("This is in the list")
else:
    print("This is not in the list")
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, M
D64)] on win32
Type "copyright", "credits" or "licer
>>> ================================
>>>
This is in the list
This is not in the list
```

Removing an item from a list
As well as appending items in a list you can also delete items from a list.

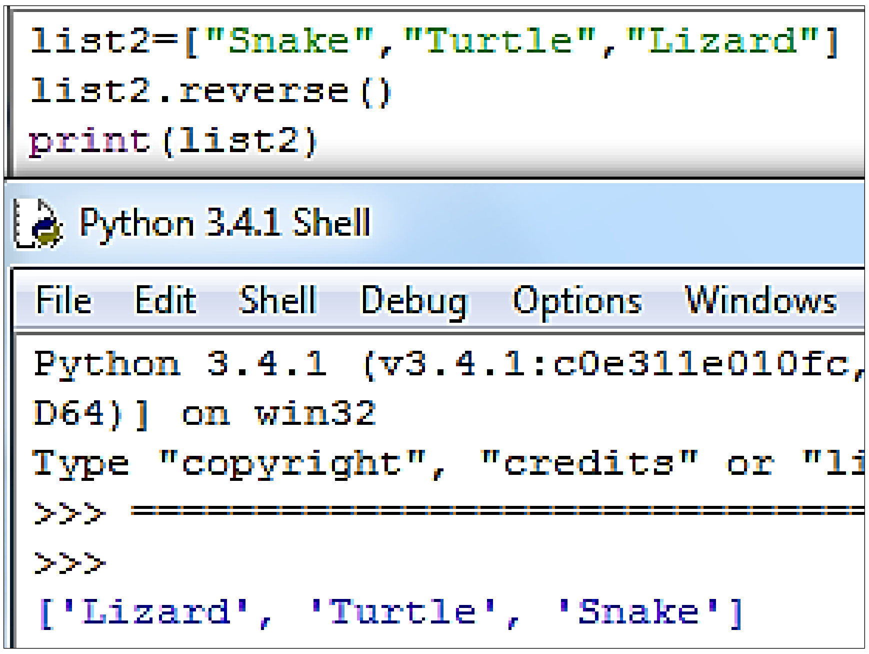


```
list2=["Snake","Turtle","Lizard"]
list2.remove("Turtle")
print(list2)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc,
D64)] on win32
Type "copyright", "credits" or "li
>>> ================================
>>>
['Snake', 'Lizard']
```

Reversing the order of a list

It is sometimes useful to reverse the order of a list.


```
list2=["Snake","Turtle","Lizard"]
list2.reverse()
print(list2)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc,
D64)] on win32
Type "copyright", "credits" or "li
>>> ==============================
>>>
['Lizard', 'Turtle', 'Snake']
```

Activity 7.3

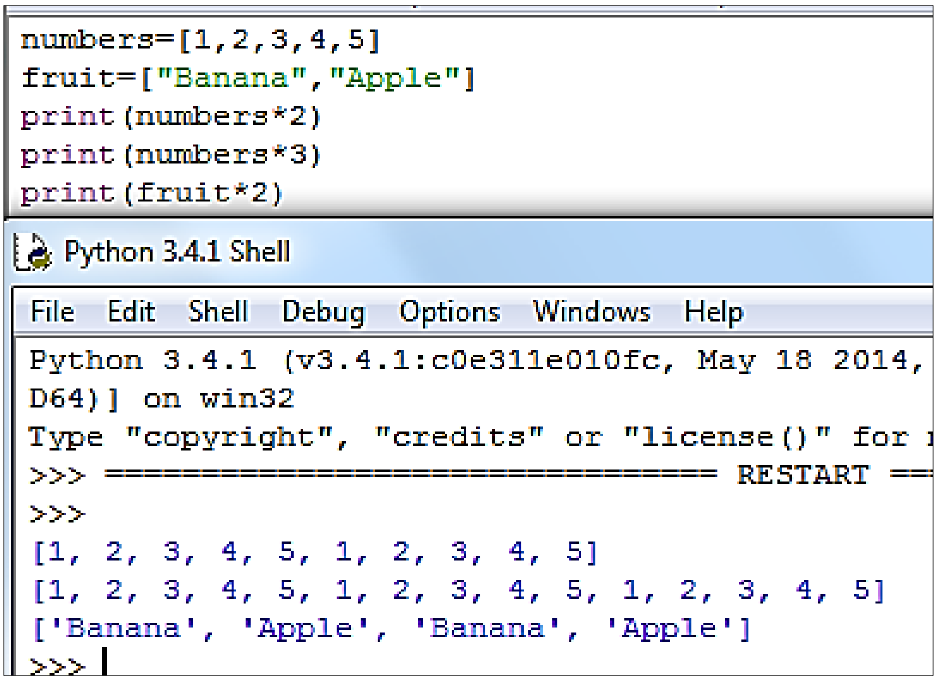
1 You work for a UK Bank in their IT department. The bank is creating a list of customers. The bank already has these customers on their list:
Charlotte Brown, Joe Thompson, Aisha Khan, Dipak Patel, Carla Peterson
a. Create this list in Python.
b. Add another line of code to this program that asks the customer their name.
c. Develop your program so that this name is added to the list.
d. Develop the program so that it continually asks if you have another name. If the answer is yes then the program asks the customer's name. If the answer is no then the program stops and displays the list of customers.
e. Create a menu system for your program. The menu offers the following choices:

1. View the names on the list.
2. Add a name to the list.
3. Search for a name in the list.
4. Remove a name from the list.
5. Sort the list into alphabetical order.
6. Quit the program

2 a. Create this list in Python:
numbers=[1,11,56,89,54,32,54,78,90,25,56,78,43,23,23,45,65,76,78]
b. Sort the list in numerical order from low to high.
c. Sort the list in numerical order from high to low (Reverse!)

Multiply lists

It can be useful sometimes to multiply lists. Here are some examples.

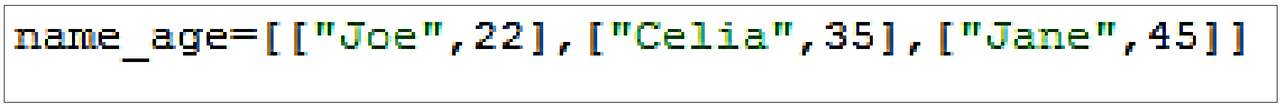


```
numbers=[1,2,3,4,5]
fruit=["Banana","Apple"]
print(numbers*2)
print(numbers*3)
print(fruit*2)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014,
D64)] on win32
Type "copyright", "credits" or "license()" for i
>>> ================================ RESTART ==
>>>
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
['Banana', 'Apple', 'Banana', 'Apple']
>>> 
```

Two dimensional arrays (Lists)

So far we have looked at simple lists. These are called one dimensional arrays. Usually the information we need to store is more complex. In this case we need to use two dimensional arrays (2D Lists). These are sometimes called lists of lists.
Here is an example of a two dimensional list that stores names and ages:



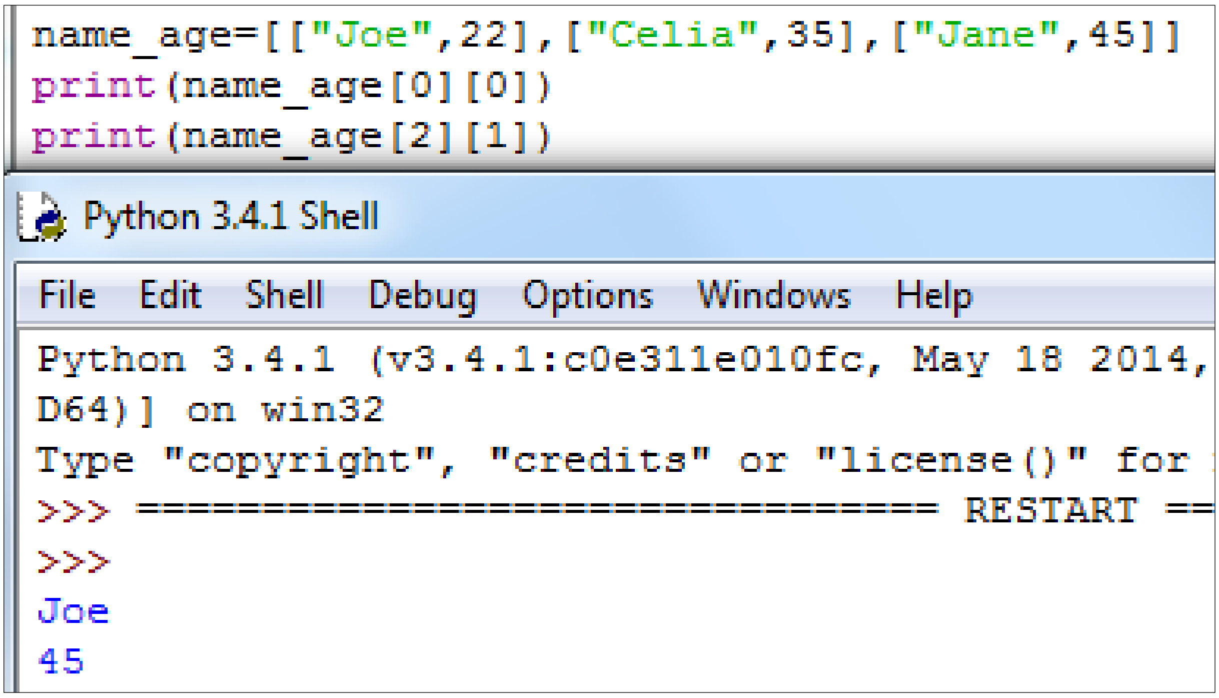
```
name_age=[["Joe",22],["Celia",35],["Jane",45]]
```

It is important to know the location of an item in a list using the index.

For example Joe is at 0, 0 and Jane's age is at 2, 1

Index position within outer list	**0**		**1**		**2**	
Index position within inner list	**0**	**1**	**0**	**1**	**0**	**1**
Item	**Joe**	**22**	**Celia**	**35**	**Jane**	**45**

You can use the **index position** to locate an item in a 2D list. Here is an example.



```
name_age=[["Joe",22],["Celia",35],["Jane",45]]
print(name_age[0][0])
print(name_age[2][1])
```

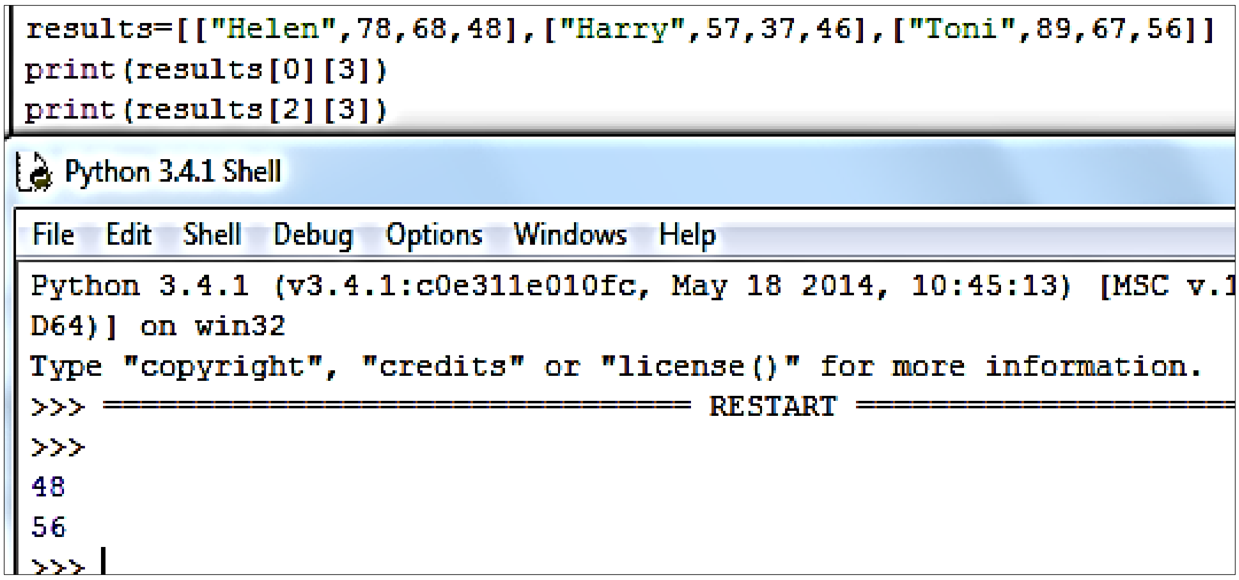
Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014,
D64)] on win32
Type "copyright", "credits" or "license()" for
>>> ================================ RESTART ==
>>>
Joe
45
```

Here is another example. In this case we are using the index location to access Helen's last test result (0, 3) and Toni's last test result (2, 3).

Index position within outer list	**0**				**1**				**2**			
Index position within inner list	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Item	**Helen**	**78**	**68**	**48**	**Harry**	**57**	**37**	**46**	**Toni**	**89**	**67**	**56**



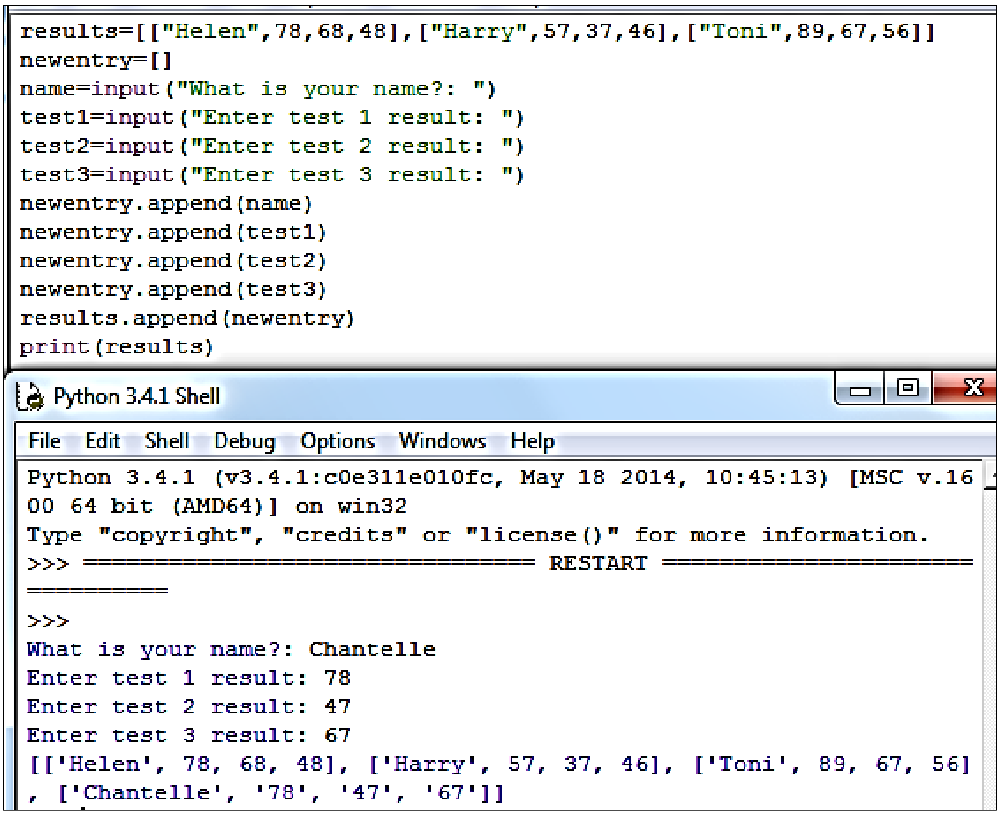
```
results=[["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
print(results[0][3])
print(results[2][3])
```

Python 3.4.1 Shell

File Edit Shell Debug Options Windows Help

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ====================
>>>
48
56
>>>
```

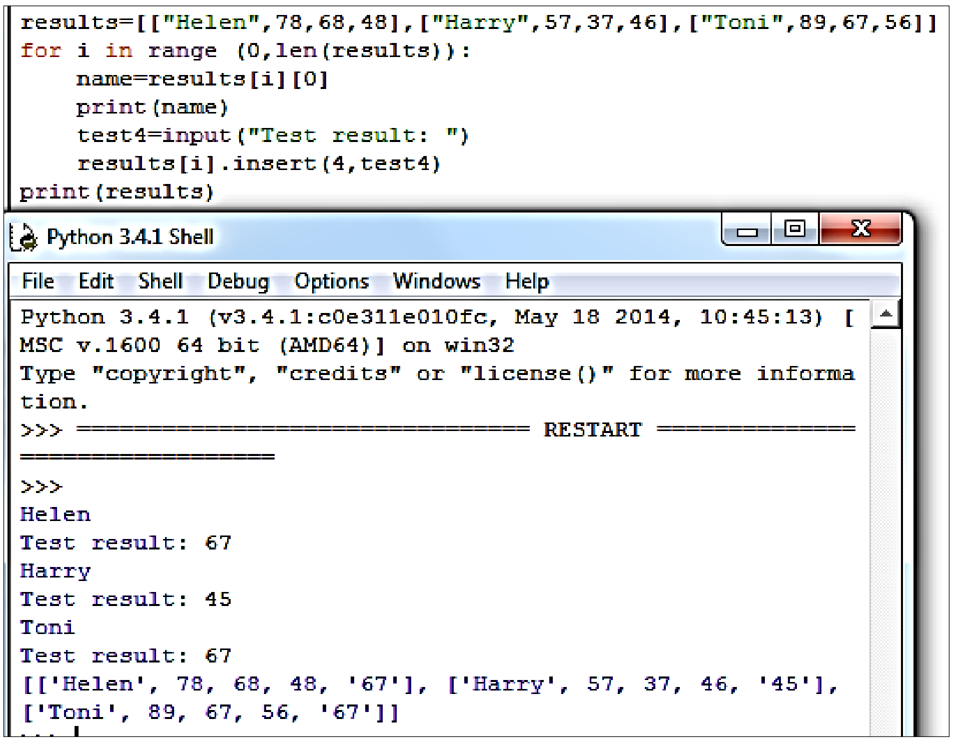
Adding a list to an existing 2D list using append.
Sometimes you need to be able to add a list to a 2D list. For example I have the results of another student called Chantelle who got 78, 47 and 67 in her tests. I wish to add her results to the existing list. This is one way that this could be done:



```
results=[["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
newentry=[]
name=input("What is your name?: ")
test1=input("Enter test 1 result: ")
test2=input("Enter test 2 result: ")
test3=input("Enter test 3 result: ")
newentry.append(name)
newentry.append(test1)
newentry.append(test2)
newentry.append(test3)
results.append(newentry)
print(results)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.16
00 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ======================
==========
>>>
What is your name?: Chantelle
Enter test 1 result: 78
Enter test 2 result: 47
Enter test 3 result: 67
[['Helen', 78, 68, 48], ['Harry', 57, 37, 46], ['Toni', 89, 67, 56]
, ['Chantelle', '78', '47', '67']]
```

Adding a single item to each list.
Sometimes you want to be able to add an individual item to each list. This can be done using a FOR loop that asks the result for each person in the list.

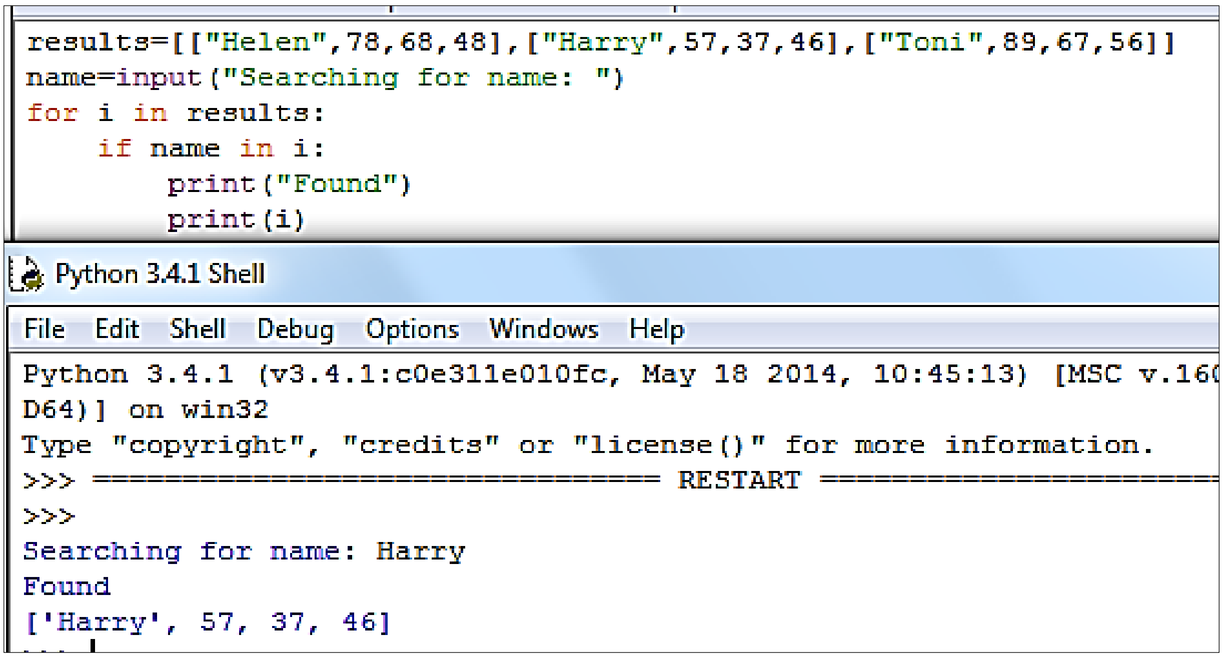


```
results=[["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
for i in range (0,len(results)):
    name=results[i][0]
    print(name)
    test4=input("Test result: ")
    results[i].insert(4,test4)
print(results)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [
MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more informa
tion.
>>> ================================ RESTART ==============
==================
>>> 
Helen
Test result: 67
Harry
Test result: 45
Toni
Test result: 67
[['Helen', 78, 68, 48, '67'], ['Harry', 57, 37, 46, '45'],
['Toni', 89, 67, 56, '67']]
```

Searching in a 2D list

It is also possible to search for an item or group of items in a 2D List. The program below looks though the list of lists and if it finds a match it will then display the information for that particular person.

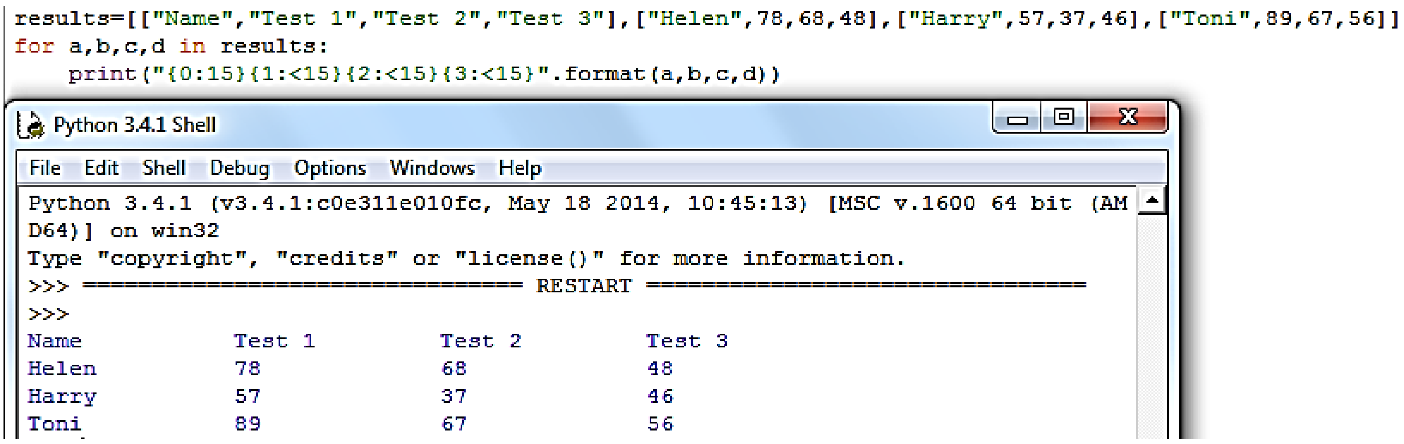


```
results=[["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
name=input("Searching for name: ")
for i in results:
    if name in i:
        print("Found")
        print(i)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.160
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ========================
>>> 
Searching for name: Harry
Found
['Harry', 57, 37, 46]
```

Displaying a 2D list as a table

Once you have created a 2D list you may wish to display the list in the form of a table. You can use formatting to make sure the list is organised into a table.



```
results=[["Name","Test 1","Test 2","Test 3"],["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
for a,b,c,d in results:
    print("{0:15}{1:<15}{2:<15}{3:<15}".format(a,b,c,d))
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Name           Test 1         Test 2         Test 3
Helen          78             68             48
Harry          57             37             46
Toni           89             67             56
```

Activity 7.4

1 a. Create a 2D list (Array) of the five planets in our solar system that are nearest to the Sun. Include the distance of each planet from the Sun (in millions of Kilometres) and the number of moons each one has. The planet nearest to the Sun must be located at index 0 and as the index number increases each planet is further from the Sun.

b. Develop a program that can search for planet Earth and then display the planets distance from the Sun and its number of moons.

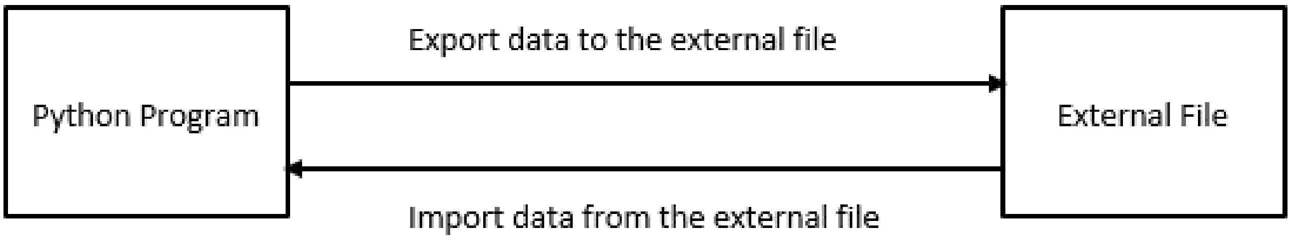
c. A new planet called “Alderaan” has been discovered at 100 million km from the Sun. It has 1 moon. Add this to your 2D array.

d. Now add to the 2D array how long it takes for each planet to rotate on its axis. Alderaan rotates at the same speed as Earth.

e. Develop a program that displays all of your data in a table.

8. File handing operations (open, read, write and close)

When creating programs it is sometime useful to be able to store and retrieve data from an external file. If you don't export data to an external file, the data is only available whilst the program is running.
Unless you say otherwise, the file is always saved to the same folder as the Python program you have produced.


Export data to the external file
Python Program
External File
Import data from the external file

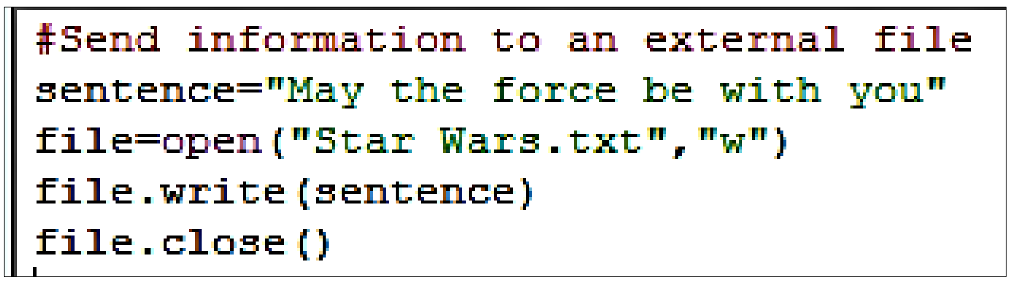
Modes for Reading and Writing Files

Different modes are used according to what you want the program to do.

Mode	Meaning
r	Read mode used when the file is being read into the program
w	Write mode is used to edit and write new information to a file. Any existing file with the same name will have their information deleted and replaced with the new information.
a	Append mode is used to add (append) new information to an existing file.
r+	This is a special mode used to allow both read and write actions to be performed.

How do I write information to a file?

You have a line of text that you would like to send to an external file. In this case the sentence is exported an external file called **Star Wars.txt**. The mode used is **w** as we are writing the sentence to the file. You must use close at the end to make sure the file is closed.
When you run the program nothing appears to happen, but if you look in the same folder as the Python program you will find a text file called **Star Wars** and if you open this file you will find the sentence.


```
#Send information to an external file
sentence="May the force be with you"
file=open("Star Wars.txt","w")
file.write(sentence)
file.close()
```

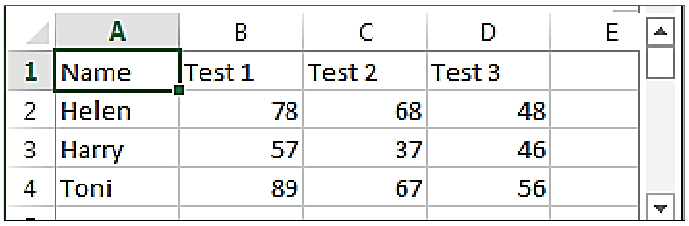
Sometimes you want to write a program that automatically closes the file.

```
sentence="May the force be with you"
with open("Star Wars.txt","w")as file:
    file.write(sentence)
```

How do I write a list to an external file?

You can also write a list (array) to an external file. In this case it is a good idea to write the file as a csv file as the table can then be viewed within a spreadsheet. To do this you must import the csv module.
The program below will take the list and export it to a csv file which can then be viewed from within a spreadsheet. This will appear in the same folder as the Python program.

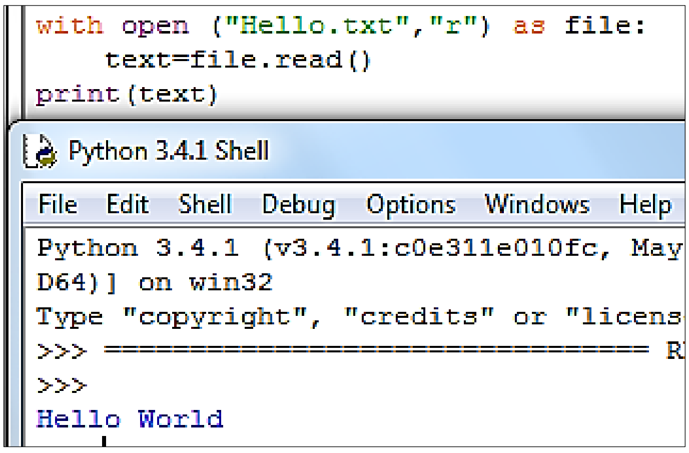
```
import csv
results=[["Name","Test 1","Test 2","Test 3"],["Helen",78,68,48],["Harry",57,37,46],["Toni",89,67,56]]
with open("Test Table.csv","w",newline="") as file:
    writer=csv.writer(file)
    for n in results:
        writer.writerow(n)
```



	A	B	C	D	E
1	Name	Test 1	Test 2	Test 3	
2	Helen	78	68	48	
3	Harry	57	37	46	
4	Toni	89	67	56	

How do I read information from a file back into my program?

If you have already created a text file it is possible to read the text from this file back into your program. In the example shown below a notepad file has been created containing the text Hello World. This has then been saved as Hello.txt. The program below has been written to take the text from the external file and assign it to the variable **text**.

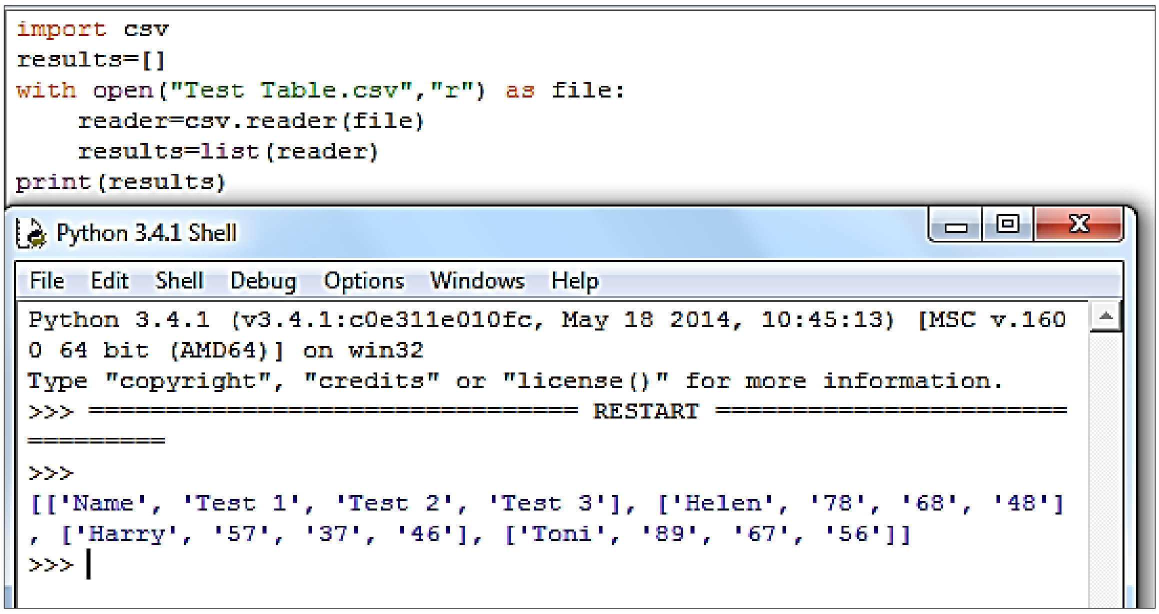


```
with open ("Hello.txt","r") as file:
    text=file.read()
print(text)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May
D64)] on win32
Type "copyright", "credits" or "licens
>>> ================================ R
>>>
Hello World
```

How do I read a list from a file back into my program?

If you have created the list using a csv file then by performing the reverse of the operation shown earlier it is possible to read the csv file back into an empty list.



```
import csv
results=[]
with open("Test Table.csv","r") as file:
    reader=csv.reader(file)
    results=list(reader)
print(results)
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13) [MSC v.160
0 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART =======================
=========
>>>
[['Name', 'Test 1', 'Test 2', 'Test 3'], ['Helen', '78', '68', '48']
, ['Harry', '57', '37', '46'], ['Toni', '89', '67', '56']]
>>> 
```

Activity 8.1

1. Here are the instructions for a game.

First player 1 must throw the dice. If the player throws a 6 they have won the game. If not then player 2 throws the dice. This continues until one of the players throws a 6 and wins the game.

(a) Develop a program that writes the instructions shown above to an external file called Instructions.txt

(b) Develop the program so that the contents of the external file can be read back into the program and stored using a variable called instructions.

2. (a) You have been asked to create a program using Python that asks the user their name, their favourite band, their favourite singer and their favourite song. The program stores this information in a list and exports the information to an external file.

(b) You have been asked to create another program that can take the information stored in the external file produced in part a, and import it back into a list. The program then displays the list as a formatted table.