Write a program to store names

Overview
In this chapter you will make a program in Python. The program can be used by anyone who manages a team or group of people. The manager might be a team coach, an orchestra leader or a teacher. The program will help the manager make a team list. A teacher or team leader often needs to pick a name at random. For example, a teacher might pick a student to answer a question. A team manager might pick a player to take a practice shot. You will use a random number generator to pick a random name. You will learn how to save the team information as a text file. The data will not be lost when the program is closed.

Learning outcomes
By the end of this chapter you will know how to:

- store a series of names as a list variable
- append an item to a list
- use a condition-controlled and a counter-controlled loop
- explain how an index number is used to identify list elements
- find the length of a list and store this value as a variable
- use code modules written by other programmers
- make and use random numbers
- store blocks of code as procedures
- define and call a procedure with a parameter
- use local variables in procedures
- write and save data as a text file.
Talk about...
You have made programs with App Inventor and with Python.
- What are the advantages and disadvantages of the two programming languages?
- Which do you like best?
- What skills have you learned from studying these languages?

Make a guide
You have experience in writing programs in Python from *Matrix 1* and *Matrix 2*, Chapter 4, *Introducing Python*.
- Make a guide to Python for students.
- List the different Python commands that you have already learned. You can do this as a group activity.

List
- Element
- Append
- Procedure
- Module
- Define
- Call
- Main memory
- Secondary storage
- Index
- Run-time error
- Code library
- Import
- Local variable
- Parameter
In this chapter you will make a Python program. You will work with a list of names. In this lesson you will enter names and store them as a list. When you have completed this lesson you will be able to:

- store a series of names as a list variable
- append items to a list
- use a condition-controlled loop.

Learn about...

Look at Matrix 1 and Matrix 2, Chapter 4, Introducing Python. You have learned that a variable is a named area of storage. You have learned how to initialise a variable by using it to store a value. A variable usually stores just one item of data.

A list is a special kind of variable. A list can store several items of data. Each item in a list is called an element.

A list is shown in square brackets. Here is a command that initialises a list:

```
ColourList = ["Black", "White", "Grey", "Red"]
```

This list has four elements.

Append

Append means add to the end. The `append` command adds an element to the end of a list. For example:

```
ColourList.append("Pink")
```

This command will add a new element to `ColourList`.

Syntax error

A syntax error is an error that breaks the rules of a language. When you run a Python program with a syntax error, the program will stop. Python will show you an error message. The error message will explain what went wrong.

How to...

In this lesson you will make a program that appends names to a team list. You will use a condition-controlled loop. In Python, condition-controlled loops start with the word `while`.

Here is an example program. The program uses a condition-controlled loop. Each time the program loops, the user appends a name to the list. Then the program asks whether the user wants to add another name.
### add elements to list

TeamList = []
Another = "Y"

```python
while Another == "Y" or Another == "y":  
    TeamMember = input("enter a name: ")  
    TeamList.append(TeamMember)  
    Another = input("enter another name? (Y/N): ")
```

print(TeamList)

How does the user stop the loop? Why is the logical operator **or** used at the top of this loop?

**Variables**

This program has three variables.

1. **TeamList**: this is a list variable that stores a list of names.
2. **TeamMember**: this is a string variable that stores a single name before it is added to the list.
3. **Another**: this is a string variable. If the user enters "y" or "Y" then the loop repeats.

Can you identify the line where each variable is initialised?

**Simplify the loop**

A student decided to make the loop simpler. He decided that he would use only two variables. He would use the variable **TeamMember** to control the loop. If the user entered "x" then the loop would stop. Here is the program he wrote.

```python
### add elements to list
### THIS PROGRAM HAS AN ERROR

TeamList = []

while TeamMember !="x":  
    TeamMember = input("enter a name: ")  
    TeamList.append(TeamMember)
```

print(TeamList)

The **while** loop starts with a logical test. It checks whether the variable **TeamMember** has the value "x". The loop will continue while the variable does NOT have the value "x".

The student’s idea was good, but his program isn’t quite right yet. The program has a syntax error.
When he ran the program the student saw this error message.

    Traceback (most recent call last):
    File "C:\Python34\Py3 lesson1.py", line 6, in <module>
      while TeamMember != "x":
    NameError: name 'TeamMember' is not defined

The error message tells you that there is a mistake in this line.

        while TeamMember != "x":

The mistake is:

        name 'TeamMember' is not defined

Why is there an error?

Here is the answer. When the loop starts, the user hasn’t entered even one name yet. The variable `TeamMember` hasn’t been initialised. You can’t use a variable before it has been initialised.

**Correct the syntax error**

The variable `TeamMember` must be initialised. This line will initialise the variable.

        TeamMember = input("enter a name")

This command must come before the `while` loop begins. Inside the `while` loop there are commands to:

- append `TeamMember` to `TeamList`
- enter a new value for `TeamMember`

The prompt is "enter a name". You can make the prompt more useful. Add a message telling the user to type "x" to stop the loop. Here is the completed code.

        ## add elements to list
        TeamList = []
        TeamMember = input("enter a name (x to stop): ")

        while TeamMember != "x":
          TeamList.append(TeamMember)
          TeamMember = input("enter a name (x to stop): ")

        print(TeamList)

**Now you do it...**

1. Make a program that lets the user append names to make a list.
2. Save the program using a suitable name such as `MakeList`.
3. Run the program and correct any syntax errors that you find.
4. Use the program to create a list of names of your own choice. The list could be a fantasy sports team, your favourite actors, or students in your class.
If you have time...

1. Use the empty value "" (quote marks with nothing inside) instead of "x". The program will stop if the user presses Enter without typing anything.
2. Test the program by entering different values. Record your test results.

Test yourself...

1. What command can we use to make a list longer?
2. A while loop includes a logical test. What happens if the result of the test is ‘true’? What happens if the result of the test is ‘false’?
3. The two programs in this lesson use relational operators to make logical tests. What relational operators were used in these programs? What do these operators mean?
4. In this lesson you looked at two programs. Both programs add names to a list. Explain why the second program is easier to use.

Key words

Append: append is a Python command that adds a new element to the end of a list.

Element: An element is a single item in a list.

List: A list is a Python variable that stores several data items.
4.2 List elements

Learning outcomes
You have made a program that lets the user input a list. In this lesson you will work with the elements of the list. The elements are the different values that are stored in the list.

When you have completed this lesson you will be able to:
- explain how an index number is used to identify list elements
- find the length of a list and store this value as a variable
- print every element in a list by using a for loop.

Learn about...
A list is made of elements. For example, this list has four elements:

ColourList = ["Black", "White", "Grey", "Red"]

Each element in the list has its own name. The name of the element is the list name, plus a number. The number is called the index number. The index number tells you the position of the element in the list. The index number is an integer.

There is one unusual thing you need to remember. The index numbers start at 0, so the first element in the list is:

ColourList[0]

You use the elements of a list like normal variables. For example, you can print one element.

print(Colourlist[0])

How to...
The command len() tells you the length of a list. It tells you how many elements there are in a list. The command includes brackets. Put the name of the list inside the brackets.

In your program the list is called TeamList, so you can write the command:

len(TeamList)

The result of this command is a number. You can save the number as a variable. In this example we have called the variable HowMany.

HowMany = len(TeamList)

Now you can print out the length of the list using a print command.
 Enter a command to print a message saying how many elements there are in the list.

Here are the extra lines to add to the program.

```python
# how many elements in the list

HowMany = len(TeamList)
print("The team list has ", HowMany, " elements")
```

**Print one element**

You can print one element from `TeamList`. Just give the index number of the element. For example:

```python
print(TeamList[0])
```

Or:

```python
print(TeamList[6])
```

Now you can add a command to let the user input an index number. Remember to convert the string input to integer data type.

```python
index = input("choose an index number:")
index = int(index)
```

Now you can print out the element the user has chosen.

```python
print(TeamList[index])
```

**Add lines to the program to print a single element, with the index number chosen by the user**

Here are the extra lines to add to the program.

```python
# see one element of the list

index = input("Which element do you want to see?: ")
index = int(index)

print(TeamList[index])
```

**Print all elements**

In *Matrix 2*, Chapter 4, Introducing Python, you learned to use loops in programming. There are two types of loop.

- **Condition-controlled loop:** in Python this is a `while` loop.
- **Counter-controlled loop:** in Python this is a `for` loop.

You have used a `for` loop to count through a number range. For example:

```python
for i in range(10):
```
We have used a `for` loop to count through a number range. We can also use a `for` loop to count through a list. The `for` loop starts with the first element in the list. The next time round the loop, it goes to the second element in the list. When it reaches the last element in the list the loop stops.

Here is an example of a `for` loop. This `for` loop will count through the elements of `TeamList`. The command inside the loop will print the element.

```python
for element in TeamList:
    print(element)
```

The variable in this example is called `element`. It is good to use a name like that. It reminds us that we are counting through a list.

🎉 **Add lines to the program to print every element in the list using a `for` loop**

Here are all the extra lines you have added to the program so far.

```python
### how many elements in the list
HowMany = len(TeamList)
print("The team list has ",HowMany, " elements")

### print one element of the list

index = input("Which element do you want to see?: ")
index = int(index)
print(TeamList[index])

### print every element of the list

print("\n")
print("Here is the whole list")
for element in TeamList:
    print(element)
```

😊 **Now you do it...**

Add commands to the program to:

- show the number of elements in the list
- let the user choose an index number and print the element with that index number
- print all the elements in the list using a `for` loop.

Remember to run the program code to make sure it works as it should.

🎯 **If you have time...**

To do this extension activity you must apply programming skills to solve an extra problem.
The program you have made does not have any syntax errors in it. You can run the program code and it will work. However, the program can still go wrong. If users enter an index number that is too big they will get a **run-time error**.

Here is an example of an error message you might see:

```
This list has 9 elements
Which element do you want to see?: 9
Traceback (most recent call last):
  File "C:\Python34\Py3 lesson.py", line 22, in <module>
    print(TeamList[index])
IndexError: list index out of range
```

The largest number the user can enter is the list length minus 1. Use an **if... else** program structure to test the index number entered by the user. Only print the list element if the index number is smaller than the list length. Otherwise give a warning message: **the number is too big**.

**Test yourself...**

Look at this list and then answer the questions.

```python
CityList = ["Lima", "Athens", "Cairo", "Madrid", "Tokyo"]
```

1. How many elements are there in this list?
2. Write the command to print out the first element in the list.
3. If you gave this command what would be printed out?
   ```python
   print(CityList[3])
   ```
4. Write the commands to print out each element in this list using a **for** loop.
5. If you gave this command, there would be an error. Why?
   ```python
   print(CityList[5])
   ```

**Key words**

**Index**: Each element in a list can be identified using the list name plus an index number. The index shows the position of the element in the list. Index numbers start at 0.

**Run-time error**: A run-time error happens when you run the program. Typically, the user has entered data that cause a problem.
**Learning outcomes**

You have made a program that lets the user input a list. The program prints out the elements of the list. Now you will add code that picks out an element at random. A team manager or teacher could use this to pick a random person to do a task. For example, the random person might be asked to answer a question or to take a free kick.

When you have completed this lesson you will be able to:
- use code modules written by other programmers
- make and use random numbers.

**Learn about…**

A **module** means any file that stores Python code. Some blocks of Python code are very useful. They can be used in many different programs.

Many programmers store the most useful modules of code. They might use this code in future tasks. A collection of useful modules is called a **code library**.

**Python modules**

Anybody can download Python and put it on a computer for free. When you get Python you also get a library of useful modules. These extra modules were written by Python programmers. The programmers have agreed to share their code with anyone who uses Python.

**Import**

If you **import** a module to your program then you can use the extra Python commands made by these programmers. In this lesson you will import a module called **random**. This module has code that lets you make a random number.

**Random number**

A random number is a number that you cannot predict. You don’t know what the number will be.

When you tell the computer to make a random number you have to tell it the number range. You have to tell it the smallest and the biggest numbers that are allowed. The computer will pick at random anywhere from the smallest to the biggest.
How to...

In this lesson you will add code to pick a random element from TeamList. Remember, every element in a list has its own index number. The index number is an integer (whole number).

This is what you will do.

1. Find the number range.
2. Make a random integer in the number range.
3. Use the random integer as the index number of an element.

Find the number range

You want to pick an element at random from TeamList. What is the smallest and biggest number it can be?

- The smallest number is the index number of the first element in TeamList. That is always the number 0. This line of code will store the smallest number as a variable.
  ```python
  smallest = 0
  ```

- The biggest number is the index number of the last element in TeamList. The biggest index number in a list is always the length of the list, minus 1. This line of code will store the biggest number as a variable.
  ```python
  biggest = len(TeamList) - 1
  ```

Make a random integer

First you must import the module. The module is called random.

```python
import random
```

The command to make a random integer is:

```python
random.randint()
```

This command ends with two brackets. Inside the brackets you have to put the smallest and biggest numbers. This sets the number range.

```python
random.randint(smallest, biggest)
```

This command makes a random number. To complete the code you have to save the random number as a variable. Then you can use it in your code. We have called the variable random_index.

```python
random_index = random.randint(smallest, biggest)
```

Finally, we have added a line to print out the variable so we can check that the code is working.

```python
print(random_index)
```
Here is the complete code to make and show a random number.

```python
### pick an element at random

import random

smallest = 0
biggest = len(TeamList) - 1
random_index = random.randint(smallest, biggest)
print(random_index)
```

**Print out an element**

You have made a random number. Now you can use it to pick a random element. Remember each element of a list has an index number. For example:

```python
TeamList[5]
```

You can print an element. For example:

```python
print(TeamList[5])
```

Or use the random number as the index number.

```python
print(TeamList[random_index])
```

Here is the complete code from this lesson.

```python
### pick an element at random

import random

smallest = 0
biggest = len(TeamList) - 1
random_index = random.randint(smallest, biggest)

print("Here is a random name")
print(TeamList[random_index])
```

**Now you do it...**

1. Add code to the program to generate a random number and pick a random name from your team list.

2. Run the program to check that it works.

**If you have time...**

Here is a programming challenge. Make a program that generates 20 random numbers between 1 and 10. Use a `for` loop.

Print out your results. Count how many times each number from 1 to 10 has come up. Is it really a random distribution? Try running this program several times. The numbers that come up will be different each time. The numbers are different each time because they are random.
Test yourself...

1. What does the `randint` command do?
2. The `randint` command uses two numbers. In this lesson we have called them `smallest` and `biggest`. What are these numbers for?
3. Write the command that will make a random number between 1 and 100.
4. Explain how a code library can help programmers in their work.
5. Your program includes the command `import random`. What does that mean?

**FACT**

**Not perfectly random**

The numbers made by the random module are not perfectly random. They are random enough for general purposes such as picking a name from a list. A specialist mathematician would use more technical methods.

**Key words**

- **Code library**: A code library is a collection of useful code. Programmers save useful code. They can reuse it in different programs.
- **Import**: Importing a module brings all the code from the module into your program, so you can reuse the code.
- **Module**: A module is a file of Python code. A module might include helpful code that can be reused.