**Lesson 8 External Data**

**Taking It Outside**

So far every time you’ve wanted to deal with some data you’ve had to type it in using the input() function. But at the end of the program that all disappears and you have to start all over again. What we need is a way of storing data so that the next time we use our program we can read the data in ready to carry on with it.

We can do that by using an external data file. A data file is just a text file and we can use some Python commands to read the lines of text in and some others to send text out. For the moment we’ll just focus on reading in data.

**Some Data To Read**

Before we start we’ll need a text file to read. That’s easily done using Leafpad. This is the Raspberry Pi equivalent of Notepad. So open Leafpad (You’ll find it in the Accessories section of the Pi’s “Start” button) and type in this list:

Sheep

Horse

Goat

Horse

Goat

Goat

Sheep

Save the text file. Call it data.txt and make sure that you have saved it in the same folder where you will be saving your programs.

**Getting Organised**

When we read the data in we’ll need somewhere to keep it. There are several lines in this text data so we can’t save it in a variable. That can only hold one value so we’ll need to use an ***array***, a special kind of variable that holds lots of individual pieces of data.

An array is a bit like a chest of drawers where each drawer can hold one piece of data. The array we’ll use will need 7 “drawers” but we don’t need to worry about that because we can add and take away spaces in the array as we go along.

**And Finally We’ll Start Programming …**

So once you’ve got data stored as a file, you need to be able to bring it into the program. We do this using the open() function to open the file. Once we’ve opened the file we can then use a loop to read in each line of data in the file into our array.

Type in this program and run it to see what happens. You obviously don’t need to type in the explanations on the right! (Did I need to tell you that?)

|  |  |
| --- | --- |
| names = []File = open(“data.txt”,“r”)nextline = File.readline().strip()while nextline != “”: names.append(nextline) nextline = File.readline().strip()File.close()print(“List of names”)num\_names = len(names)for counter in range(0,num\_names): print(names[counter])print(“There are”,num\_names,“names”) | Make a new empty arrayOpens the file with a “file handle”Read in the first line from the file“” means the end of the fileAdd to the array….. and read the next line from the fileFinished with the file so close itlen tells you how many in the arrayUse a for loop because we know how many |

**So, Let’s Explain**

Before you can use an array you have to *initialise* it. This means “start it up” and say what it has in it. In this case we made an array called names and we wanted it empty to start with.

We then need to open the text file and create a *file handle*. This is done using the open() function. In this you say which file you are going to open and what you are going to do with it (“w” means “write” i.e. send data to the file; “r” means “read” i.e. get data from the file).

**New Commands**

**open( , )**
opens a file, setting the connection to the text file and explaining what you are going to do:

* “w” means “write”
* “r” means “read”

The result of the open() function is sent to a variable and this is used throughout to refer to the file. In this case we called this variable File.

Next we need to read in the first line. This is done using the readline() function which reads in one line at a time. We must use the File variable so it knows which file to read from. The result of the readline() function is assigned to a variable. I’ve called that variable nextline.

**New Commands**

**readline()**
reads in a line of text from the data file

**strip()**
removes all the extra bits in a line except for the text

The bit on the end, strip(), is there to remove everything except the text. You can’t see but there all sorts of control characters used in text files and we don’t want them interfering.

Now that we’ve got one line, we need to check it. If it doesn’t have text in it, we’ve reached the end of the file. Otherwise we’ll read on…

**New Commands**

**append()**
adds another element to an array

**close()**
closes a file

Each time we go around the while loop our program adds the line it just read in to the array. To do that we use the append() command.

Eventually the loop finishes because we reach the end of the text file and so we close the file.

You now have an array stored called names which has 7 elements which have been brought in from the text file. It’s a bit like table that looks like this:

|  |  |  |
| --- | --- | --- |
| names | [0] | Sheep |
|  | [1] | Horse |
|  | [2] | Goat |
|  | [3] | Horse |
|  | [4] | Goat |
|  | [5] | Goat |
|  | [6] | Sheep |

If we want the program to work its way through the array, we are best to use a for loop but to do that we need to know how many elements there are in the array. To do that we use the len() function. len is short for “length”. We assign the result to a new variable and we now know how many rows are in the array!

**New Commands**

**len()**
tells the program how many elements are in an array

After that the for loop works its way through the array using the variable counter to decide which element in the array to show. The range starts from 0 because Python arrays start at element 0 (not 1).

**So let’s add some more stuff…**

Edit the program so that it looks like this. The bits to add are in italics. Be careful about indenting. Notice the comments which have been included. Add your own comments to explain the new part.

# New empty array to hold names

names = []

# Open data.txt file

File = open(“data.txt”,“r”)

# Read in lines of text and add to names array

nextline = File.readline().strip()

while nextline != “”:

 names.append(nextline)

 nextline = File.readline().strip()

File.close()

# Find out how many elements in array

num\_names = len(names)

***lookingfor = “Sheep”***

***foundnumber = 0***

for counter in range(0,num\_names):

 ***if names[counter] == lookingfor:***

 ***foundnumber = foundnumber + 1***

***print(“There are”,foundnumber,“sheep”)***

** Over to You**

Add some more code to the program so that you can type in which animal you want it to tell you about. So you could for example input “Goat” and it will tell you that there are “3 Goats”.

Change the program again so that you can type in a number (don’t forget int!) and the program will tell you what is at that position in the array e.g. if you typed in 3 it would tell you Horse is at position 3.

**Extension**

Look back at your notes on using while to validate input and add some code so that if the person types in a number that is too large, the program will ask for a new number.

**Problems Problems**

By now you will have learnt about *run-time errors*. These are errors caused because there are resources missing when the program runs. A typical problem is when your problem asks for an external text file and the file is not there. If you left it like that, the program would just crash which is not very user-friendly.

**New Commands**

**try:**
creates a section of code which is checked for errors as it is run

**except:**
the section that is run if an error happens in a try: section

We can prevent this by using Python’s *error-trapping* code. This is included using the try … except keywords. Any program code after a try: will be checked for run-time errors as it runs; if an error occurs, the code jumps to the except: section where you can put an error message of some sort.

Adapt your code from previous programs or just write this out completely new:

names = []

try:

 File = open(“data.txt”,“r”)

 nextline = File.readline().strip()

 while nextline != “”:

 names.append(nextline)

 nextline = File.readline().strip()

 File.close()

 print(“File opened”)

 fileopened = True

except:

 print(“There was a problem opening the file.”)

 fileopened = False

Run the program and check that it tells you that the file was opened. Change the name of the text file data.txt or delete it. Run he program again and see what happens.