Learning Objectives

- Express algorithms in flowcharts to process data in files
- Write code that reads from, writes to and appends to text files
- Working with records and files

Text Files

The programs we have looked at so far have all used variables which are stored in memory.

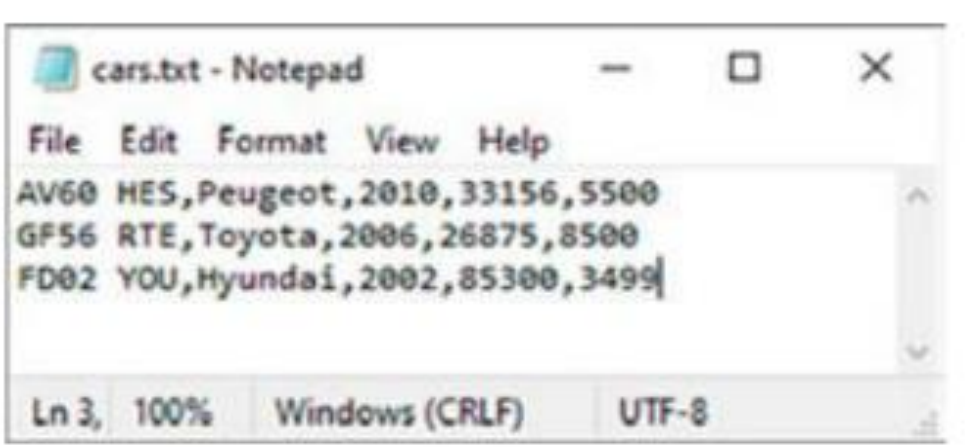
Often, however, the data needs to be **stored in a file** which can be held **permanently**, from where it can be read next time it is needed.

Text files contain text that is in lines.

There is no other structure, unlike files of records.

When you read from a text file you can only read a whole line at a time.

When you write to it, you can only write one line at a time.



Modes

There are three different modes that files can be opened in:

- **Read mode** – the contents of the file can be read, but no changes can be made to the file
- **Append mode** – new content can be added to the end of the file
- **Write mode** – data can be written to the file

	Code	Description
1	`theFile = open("ExamMarks.txt", "r")`	Opens a file for reading
2	`theFile.close()`	Closes a file
3	`theFile = open("ExamMarks.txt", "w")`	Opens a file for writing

Writing to a file

In this example we ask the user to input names and then write them to a text file.

```
nameFile = open("names.txt", "w") # w opens in write mode
moreNames = True
while moreNames:
    name = input("Please enter name: ")
    if name != "xxx":
        nameFile.write(name + "\n")
    else:
        moreNames = False
nameFile.close()
```

When using the write() function, the text is simply added into the file. The \n is known as an escape character for a new line. It is treated as one character, but won't display in the file.

Reading from a file

Now the names are stored in the file, we can read them back and print them out, or store them in
a list where they could then be processed, for example.

```
nameFile = open("names.txt", "r") # r opens in read mode
for line in nameFile:
    print(line)
nameFile.close()
```

Reading multiple lines from a file

It is possible to read all the lines from a file at once and put them into a list ready for later processing.

```
nameFile = open("names.txt", "r")
names = nameFile.readlines()
nameFile.close()
for i in range(len(names)):
    names[i] = names[i].strip("\n")
    print(names[i])
```

Be aware that the list will likely contain \n characters. The example shows how these can be removed by using the strip() function.

Appending to a file

This example allows an individual name to be appended to the end of an existing text file.

```
nameFile = open("names.txt", "a") # opens in append mode
name = input("Please enter name: ")
nameFile.write(name + "\n")
nameFile.close()
```

Appending a list of strings to a file

Sometimes, you may already have a list of strings which you want to append to a text file. In this case, there is no need to append each line separately. All the strings can be added using just one Python function.

Remember that new line characters \n and spaces will need to be in the strings if you want them to be in the text file.

```
names = ["Amy\n", "Brian\n", "Charles\n", "Dorothy\n"]
nameFile = open("names.txt", "a")
nameFile.writelines(names)
nameFile.close()
```

Worked Example: Animal Farm

Ask the user to enter a type of animal
Write the input to the text file 'animals'
Repeat asking the user for an animal until they input 'x'
Remember you are overwriting the text file

```
2  #Ask the user to enter a type of animal
3  animal = input("Type of animal")
4
5  #Write the input to the text file 'animals'
6  file = open('animals.txt', 'w')
7
8  #Repeat asking the user for an animal until they input 'x'
9
10 while animal !='x':
11
12   #finish code
```

Animal Farm

main.py animals.txt

```
1
2  #Ask the user to enter a type of animal
3  animal = input("Type of animal")
4
5  #Write the input to the text file 'animals'
6  file = open('animals.txt', 'w')
7
8  #Repeat asking the user for an animal until they input 'x'
9
10 while animal !='x':
11
12   file.write(animal+'\n')
13   animal = input("Type of animal")
14
15 file.close()
16 #Finally print the contents of the text file along with the total number of animals
17 #Remember you are overwriting the text file
18 file = open('animals.txt', 'r')
19 for line in file:
20   print(line)
21
22 file.close()
23
```

Appending another item to the text file

- This program starts by writing the items in the list to the text file.
- Then using "a" - append will add BMW to the end of the text file.

main.py cars.txt

```
1  #code used to set up the file for testing purposes
2  cars_list =["Ford", "Honda", "Mitsubishi","Toyota"]
3  myFile =open("cars.txt","w")
4  for item in cars_list:
5   myFile.write(item + '\n')
6  myFile.close()
7
8  myFile = open("cars.txt", "a")
9  myFile.write("BMW\n")
10 myFile.close()
11
12 myFile = open("cars.txt", "r")
13 for line in myFile:
14     print(line, end="")
15 myFile.close()
```

Reading a text file into a list

Here is a text file with some car details in it:

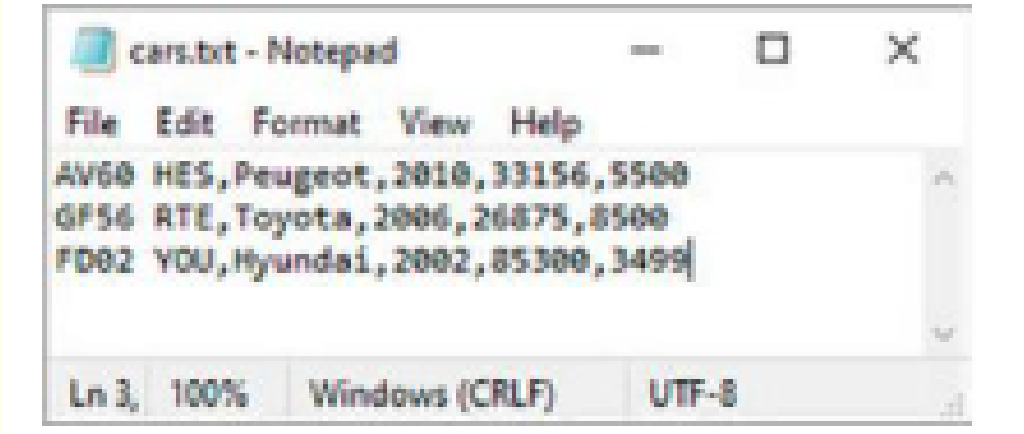
Table data is often stored in files with each field separated by a comma and each record on a different row.

These files are known as CSV (comma-separated values) files:

RE05 HSD, Ford, 2005, 97500, 650

SW12 SDF, Vauxhall, 2012, 59650, 2500

BN64 WJR, Nissan, 2014, 39900, 18000



- The following shows how the cars CSV file could be read and converted into a 2D list storing one car record on each row:

```
carTable = []
nameFile = open("cars.txt", "r")
for line in nameFile:
    line = line.strip("\n")
    carRecord = line.split(",")
    carTable.append(carRecord)
    nameFile.close()
print(carTable)
```

The CSV file is first opened. Each line of the file represents one car record.

So, for each line, first strip away any \n escape characters.

Then split the line by the "," character. This creates a list of all the fields in the line.

These are stored in the carRecord list. Now append this list to the carTable list.

Finally close the CSV file.

Worked Example: Searching for a name

- This program searches a file for a name
- It allows the user to input a name
- Search the lines in the file using a loop
- if the name is found - print “found”
- else print "not found”

```
Powered by trinket
 paul
Searching textfile jane
Searching textfile peter
Searching textfile paul
Searching textfile rachel
Found
```

```
1  name = input()
2
3  myfile = open("names.txt",'r')
4
5  found = False
6  for line in myfile:
7    print("Searching textfile",line, end="")
8    if line.rstrip() == name.lower():
9      #strip() is used to remove the new line
10     found = True
11
12 if found:
13   print("Found")
14 else:
15   print("Not Found")
16
17 myfile.close()
```