# Lesson 7 - Filing

|  |  |
| --- | --- |
| **Lesson Outcomes**  In this lesson you will learn:   * the difference between text and byte files; * how to create, read and write files; * to write and read date into variables; * how to use direct-access using the SEEK command. | **C:\Users\Graham\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\X6CHINOH\MC900441498[1].png** |

## Introduction to Files

Files are very important and allow us to store data from a program to backing storage e.g. a hard drive, USB or disk. In most programming languages files come as two main types: **binary** and **text.** Binary files are used to store **raw data** such as images, sounds or other non-text information.

Text files are different, as the data is stored as **CHARACTERS** with each block of data either stored as a complete line of text or individual characters on the page. The **TEXT** file is made up of ASCII characters and saved with the extension “.txt”. You may also include basic text formatting such as bold, italic, carriage return, line spaces and tabs using **special control characters**.

In Python you can create a text file using the file object as follows:

file object = open(file\_name [, access\_mode][, buffering])

**filename**: the name of the file you are going to create

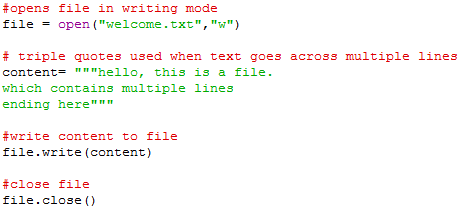
**access mode:** determines what mode the file is going to be open in e.g. reading, writing or append.

**Buffering:** allows the computer to read data into a buffer in order to speed up access. In this exercise we will be ignoring this option.

Here is a list of the different access modes:

|  |  |
| --- | --- |
| **Modes** | **Description** |
| r | Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| rb | Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| r+ | Opens a file for both reading and writing. The file pointer will be at the beginning of the file. |
| rb+ | Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file. |
| w | Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| wb | Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| w+ | Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| wb+ | Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| a | Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| ab | Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| a+ | Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| ab+ | Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

## Writing File

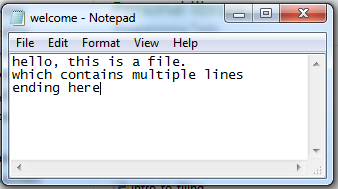
To create a basic file you need to open the file, write the text content and then close the file:

The above program will create a file called “welcome.txt” in write mode. The file should appear in your default directory (where you saved the program), however you can specify the directory e.g. “F:\my\_python\_work\welcome.txt”.

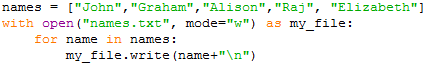
The “content” variable contains the text to be saved to the file. You will notice **treble quotes** have been used to tell Python that this string is across multiple lines.

The file.write(content) command takes the contents of variable content and writes to the opened file.

The last line closes the file. It is important you close the file otherwise the data won’t be written.

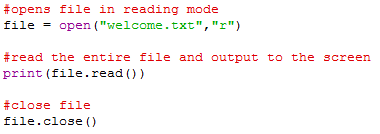
If you open the file in note or word pad you should see the following:

Alternatively, rather than writing entire pieces of text you might want to store a list of names. For example, the following program writes a List to a file:



Notice here the “with” command has been used (this is to provide error-trapping) and the open command in one line. The file is written to using a FOR loop, and each line is written with **“\n”** attached at the end. The **“\n”** is a control character, which tells the text editor to display a NEW LINE.

## Reading a file

Having created a text file, reading it is simply the reverse. An example of how to read from the above file:

This program opens the file we create earlier in **read mode.** The file.read() function reads the entire file and returns as a string which is then printed out to the screen.

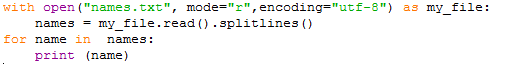
Alternatively, rather than reading the whole file you might want to just read the first line:

Note, when reading a file you start from the beginning and work your way through (serial access). Using the above program, if I inputted the command file.readline() again it would give me the next line (line 2). You must also remember that the readline also includes the carriage return (last character) within the string.

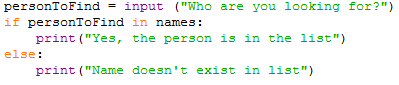
You may also just want to read the first 10 characters, in which case you would enter:

It is also possible to include the file reading in a for…loop:

**You could simplify** the whole process by using just a few commands. Here is a program which reads the names text file we created earlier:



This opens the file, and splits each element into a List (covered in a previous lesson) using the .splitlines() method. This works by creating a new item when a carriage return is detected (newline).

The beauty of storing the data in a **List**, is then you can search for a specific element:

### Tasks

7.1 Write a program which asks the user for 5 names; each time the name will be saved to a text file. The program should output the contents of the file after the names have been entered.

7.2 Create a file (in notepad) with a series of usernames and passwords. Format the text so that there is a line for the username and a line for the password. Repeat this process for each user.

Write a program which will prompt the user for their username and password and display appropriate error messages if incorrect username or password provided. The program should read the above file to check:

1. The username exists
2. The password entered equals the password stored for the username given

If the username or password is incorrect an appropriate error message should be displayed.

7.3 Change the program in 7.2 so the user has options at the beginning of the program:

1) Enter username and password  
 2) Change username and password

Please enter option 1 or 2.

You will need to add function to the program to allow the user to change their username and password. These changes should be saved to the file

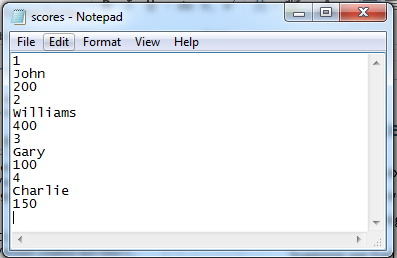
**HINT:** you will need their old username first to so that you can find and replace the data in the file.

7.4 Change the program in 7.3 so that the password is encrypted. Use a simple Caesar cipher method to encrypt the password (e.g. shift each character by 1 so that A becomes B and B becomes C etc. Z should become A.). The encrypted password is saved to the file. To compare entered password encrypt the user entry before comparing with password stored on the file.

## Direct Access Filing

So far in these exercises you have dealt with serial access files, that is, to read a file you start at the beginning and work your way through. In filing it is more efficient to go directly to the data you need rather than going through each line of text in turn.

Suppose we had the following text file which stores player’s scores in a game:



If we wanted to find player number 3 score we would have to go through each line to compare the player number read with the number we wanted. Once we have that position then we can simply go to the next line for the Name, and the next line for Score.

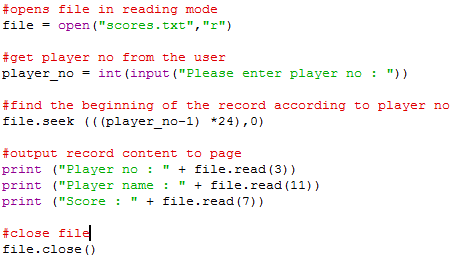
The above method is reasonable when dealing with a few entries, but if we had 1000s then this would be very inefficient and time consuming.

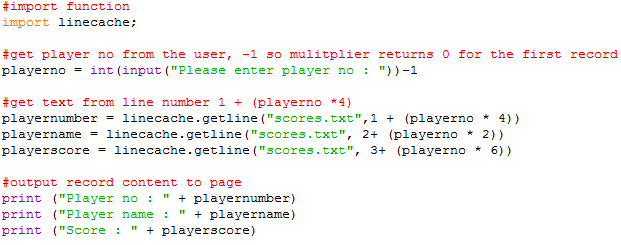
Python allows direct-access filing using the seek command. The seek command allows us to specify a position in the file to read from. However, there are drawbacks, as you have to specify the exact character position. In order to calculate the position of N th record we would need to fix the character length for each player entry. In the above example the structure is as follows:

|  |  |
| --- | --- |
| Player No | 3 characters + 1 (carriage return) |
| Name | 11 Characters +1 |
| Score | 7 characters + 1 |

**Note:** Carriage return is a control character which tells the text editor (notepad) to take the text to the next line.

Using the above we calculate that each record is 24 characters long. Knowing the length of the record we can calculate the position of any record in the file using the basic formula player no \*24.

For example, if we want to read player number 4, we could us a basic formula to find the correct start position in the file e.g. (4-1) \* 24 for the character no. In Python:

Although this works, it is a little untidy and complicated. There are other methods which we can use to make the process easier. One method is to use the linecache import. Using this import library we can specify the line number we want to read. We can adapt the other program to refer to lines as opposed to character positions:

You will notice, using the linecache import, we have to load the text file with the line we want each time, thus removing the whole open and close text file approach.

### Tasks

7.5 Write a program which takes a series of marks for a set of students and stores them in a file. You should store the file in the following format:

|  |  |
| --- | --- |
| **Record No** | 2 characters |
| **Firstname** | 10 characters |
| **Surname** | 10 characters |
| **Mark (%)** | 3 characters |

Test the program with 4 records.

7.6 Add to the program so the user can specify a record number to view on the screen.

7.7 **Extension**

Change the program from exercise 7.6 so the user can specify a record to delete.

**HINT**: load all the data in, delete the record, and save it back.



The above reads the text into the variable lines. Lines is written back starting from line 3 to -1 (from the end of the file)