# Table of Contents

# Lesson 18 - Files

## Text files

A text file is a type of computer file that only contains plain text and doesn't contain any formatting or images. Text files have the extension **.txt**. Text files are very useful for storing data that will persist after the end of your program.

## Creating, reading from and writing to a text file

Creating a text file in Python is very easy and only requires a few lines of code. The first line below will try to open a file called "myFirstTextFile.txt". If there is no file with that name, the code will create a new file.

The second line of code will write to the opened file. In this case it will write "Test text". The final line will close the file. It is very important that this line is added and that the file is closed, otherwise the file may not update correctly.

```
file = open("myFirstTextFile.txt", 'w')
file.write("Test text")
file.close()
```

The next snippet of code will open the file in read only mode. The second argument in the open function is what controls this. The data in the file is then read and stored in a variable called dataIn. Finally the file is closed and the data is printed.

```
file = open('myFirstTextFile.txt', 'r')
dataIn = file.read()
file.close()
print(dataIn)
```

The different options for how to open a file are shown and explained in the table below.

| Command | Name | Explanation |
|---|---|---|
| “r” | Read | Opens a file for reading, error if the file does not exist. |
| “w” | Write | Opens a file for writing and will write over any data already there. Creates the file if it does not exist. |
| “a” | Append | Opens a file for appending and will add to any data already there. Creates the file if it does not exist |

### Challenge 1

1. Create a new file.
2. Write your name to the file.
3. Close the file.
4. Open the same file.
5. Add your date of birth to the file on a new line. Note: “\n”
6. Open the same file.
7. Read the information from the file.
8. Print the information.
9. Use the split function to break the information into two parts.
10. Print the information out nicely formatted.

## Challenge 2

1. Create a new file called login_details.txt
2. Prompt the user to enter a username.
3. Write the username to the login_details.txt file.
4. Write a newline to the login_details.txt file.
5. Prompt the user to enter a password.
6. Write the password to the login_details.txt file.
7. Close the file.
8. Open the file.
9. Read the data.
10. Split the data into two parts.
11. Ask the user to enter their username.
12. If it is incorrect print out “Wrong username” and exit() the program.
13. Ask the user to enter their password.
14. If it is incorrect print out “Wrong password” and exit() the program.
15. Welcome the user to their account.
16. (Extra) Change the program to continuously ask for the username and password until the correct details are entered.

## CSV files

Text files are very useful for storing small amounts of text information, but they aren't as good at storing numerical data, such as multiple temperature values.

To store this sort of information we will use CSV files. CSV stands for "comma separated variables". CSV files are essentially text files, but they have a specific structure for storing and separating data. CSV files use the ".csv" extension and can be opened by Excel, because they have a specific structure for storing multiple data items.

## Creating, reading from and writing to a CSV file

Interacting with a CSV is very similar to interacting with a text file. The first snippet of code below is writing five numbers to the CSV file. The only difference is that these numbers need to be separated by a comma.

```
file = open("myFirstCSVFile.csv", "w")
file.write("1,2,3,4,5")
file.close()
```

The next code snippet reads the information from the CSV file.

```
file = open("myFirstCSVFile.csv",'r')
dataIn = file.read()
file.close()
print(dataIn)
```

This next code snippet converts the values into a list

```
myList = dataIn.split(",")
myList = [int(item) for item in myList]
print(myList)
```

The code snippet above is a simple method of interacting with data in a csv file and is very similar to how we interact with a txt file. In the next section we will look at interacting with the csv file using the csv library.

## Using the csv library

The first step when we want to use any library is to import the library. In this case the library is simply called csv.

```
import csv
```

The next step is learning how to write to a file and then read from the file. The code snippet below will write a set of headings to a file called patients.csv.

```
header = ["firstName","lastName","phoneNum","dob","age"]
file = open("patients.csv","w")
db = csv.writer(file)
db.writerow(header)
file.close()
```

The first line creates a list of the items that we want to add to the file. Using the csv library requires that we enter a list of items. The order of these items is important and needs to be in the same order as any other data entered into the file.

Opening the file is the same as before. Note that I used “w” to open the file, removing any text that was there before.

The following two lines are new. The first line creates a connection to the file that we will be using. db stands for database or where we are going to store our information and this is just a variable name.

The second line is writing our data to the patients.csv file. In this case we are adding all of the headers that we need.

```
db = csv.writer(file)
db.writerow(header)
```

Finally, we close our file, when we are finished with it.

Adding more records to the file is very similar to the previous example. Note that I am opening the file with “a” and if we want to add multiple records we will just use the writerow command multiple times.

The order of these attributes is also very important and need to be in the same order as the headers that we entered previously.

```
record1 = ["Joan", "Byrne", "0981 45877","2/2/75","45"]
record2 = ["Gideon","Jones","098376800","4/7/59","61"]
file = open("patients.csv", "a")
db = csv.writer(file)
db.writerow(record1)
db.writerow(record2)
file.close()
```

The code snippet below shows how to read from our file. This code will read all the information in the file and convert it into a list.

```
file = open("patients.csv","r")
records = list(csv.reader(file))
file.close()
print(records)
```

The output of this will look like below. Note the “[[“ at the beginning of the output. This tells us that this is a 2D list or a nested list. We have seen something like this when we were learning about dictionaries.

```
[['firstName', 'lastName', 'phoneNum', 'dob', 'age'], ['Joan', 'By
rne', '0981 45877', '2/2/75', '45'], ['Gideon', 'Jones', '09837680
0', '4/7/59', '61']]
```

To convert this output into 1D list we can write the code below.

```
file = open("patients.csv","r")
records = list(csv.reader(file))
file.close()
for record in records:
  print(record)
```

The output will look like this.

```
['firstName', 'lastName', 'phoneNum', 'dob', 'age']
['Joan', 'Byrne', '0981 45877', '2/2/75', '45']
['Gideon', 'Jones', '098376800', '4/7/59', '61']
```

If we want to avoid displaying the header row we can change the for loop.

```
file = open("patients.csv","r")
records = list(csv.reader(file))
file.close()
for record in records[1:]:
  print(record)
```

We can also print out individual attributes from each row by adding an index. The code below will print out the age for each patient.

```
file = open("patients.csv","r")
records = list(csv.reader(file))
file.close()
for record in records[1:]:
  print(record[4])
```

Finally, we can print all of the information out in a user friendly way.

```
print("First Name\t Last Name\t Phone
Number\t Date of Birth\t Age")
print
("-------------------------------------
----------------------")
for record in records[1:]:
  print(record[0],"\t\t", record[1],
  "\t\t", record[2], "\t", record[3],
  "\t\t", record[4])
```

The output is below.

```
First Name    Last Name    Phone Number     Date of Birth    Age
-------------------------------------------------------------
Joan          Byrne        0981 45877       2/2/75           45
Gideon        Jones        098376800    4/7/59           61
```

## Challenge 3

1. Create a new file called patients.csv
2. Add a header row to the file.
3. Add three patient records to the file.
4. Print out all the information from the file.
5. Print out each row of information separately.
6. Print out just the first and last name of the patients.
7. Print out all the information formatted in a user friendly way.

## Challenge 4

1. Create a new file called students.csv
2. Write Python code to add the name, grade and age of three students to the file.
3. Ask the user to input a new student record.
4. Display all the information in the students.csv file in a user-friendly way.
5. Calculate the average age of the students.
6. Calculate the average grade of the students.
7. Print out the students names in alphabetical order.

## Challenge 5

1. Go to repl teams and start Lesson 18 - Part 3.
2. Read the code from each of the two files.
3. Print the code from each file in a user friendly way.
4. Calculate the average grade for the computer science students.
5. Calculate the average grade for the French students.
6. Calculate the average grade for the German students.
7. Calculate the letter grade for the computer science students and add this grade to their file.
8. Print out the students names in alphabetical order.