Chapter 7: Working with Files

What we will learn:
- ✓ How to open files in Python
- ✓ Syntax for manipulating files
- ✓ Creating a database
- ✓ Making queries

What you need to know before:
- ✓ Data types
- ✓ Lists

Keywords: File, Absolute path, Relative path, Database, Query, SQL

Files are used to store data on a secondary storage device. All the data that we have been using so far is data that disappears when we stop running the program. In order to have persistent data, we have to store it on a secondary storage medium. This data can be used after the running program is closed or whenever we wish to do so.

In Python we use the *open()* function, which is a built-in function, to open a file.

Syntax:

<variable_name> = open(<filename>, <mode>)

<variable_name> is a file object and will be used to refer to the opened file.

<filename> is the name of the file and extension. If the file is stored in the same directory as the running program then the name of the file is enough, otherwise the full file path is required which can be relative or absolute, we will discuss these later.

<mode> refers to the possible operations on the opened file. The possible modes are summarised in the table below.

Mode	Meaning
'r'	Open for reading (default)
'w'	Open for writing. If the file exists, Python will delete all the contents first; if the file does not exist, Python will create a new file
'x'	Create a new file and open it for writing
'a'	Open for writing, appending to the end of the file if it exists
'b'	Binary mode
't'	Text mode (default)
'+'	Open a disk file for updating (reading and writing)
'U'	Universal newline mode (for backward compatibility)

Table 4

Example:

```
#Open the file in write mode
text_file = open("olympic_games.txt", "w")
```

This will create a text file called *Olympic_games.txt* in the current working directory, or if the file exists it will truncate (delete all the contents) the file and open it ready to be written to. *text_file* is the file handler stream created in the program, that we will use to refer to and access the file on disk.

7.1 Writing to a File

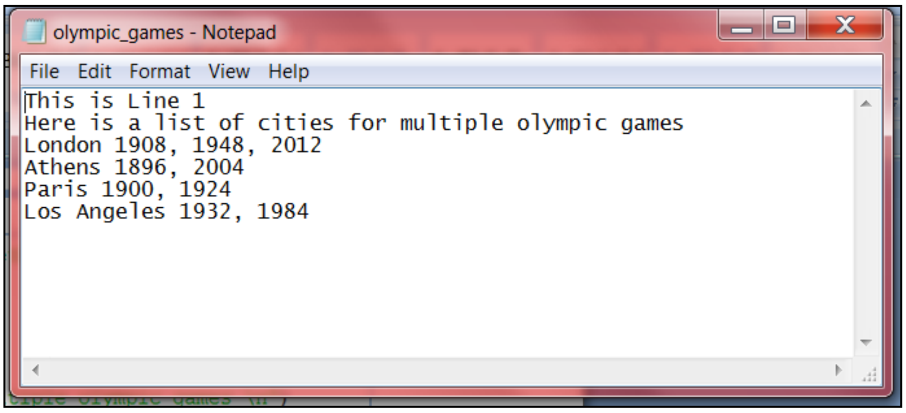
We use the write() function to write contents to a file.

```
print("Writing to a text file with the write() method")

#Open the file in write mode
text_file = open("olympic_games.txt", "w")

text_file.write("This is Line 1\n")
text_file.write("Here is a list of cities for multiple olympic games \n")
text_file.write("London 1908, 1948, 2012 \n")
text_file.write("Athens 1896, 2004 \n")
text_file.write("Paris 1900, 1924 \n")
text_file.write("Los Angeles 1932, 1984 \n")

text_file.close()
```

The code snippet above will create a new file, or open an existing file, called *Olympic_games.txt*; write some information to the file then close the file. A few things are worth noting here.

- `"w"` mode will either create a new file or open an existing one and delete all the contents
- `write()` invoked on the file handler will write the file
- `\n` is an escape sequence used to create new lines in text files
- `close()` invoked on the file handler will close the current opened file. It is always a good idea to close files when you are no longer using them

The following text file can be viewed in the current working directory.



```
This is Line 1
Here is a list of cities for multiple olympic games
London 1908, 1948, 2012
Athens 1896, 2004
Paris 1900, 1924
Los Angeles 1932, 1984
```

Notice that the name of the file is *olympic_games*, which is the name supplied in the program code. The *\n* inserts a new line in the document.

7.2 Printing a File to the Screen

We use the *read()* function to read data from a file that is open in read mode and the *print()* function to print data in the file handler to the screen.

```
#now open the file and print to the screen
text_file = open("olympic_games.txt", "r")
print(text_file.read())

text_file.close()
```

The snippet above will first close the open stream, then reopen the stream, this time for reading. It will then use the read() function to read the stream and print the contents to the screen using the print() function. Finally, we close the stream.

The entire script and output is repeated below.

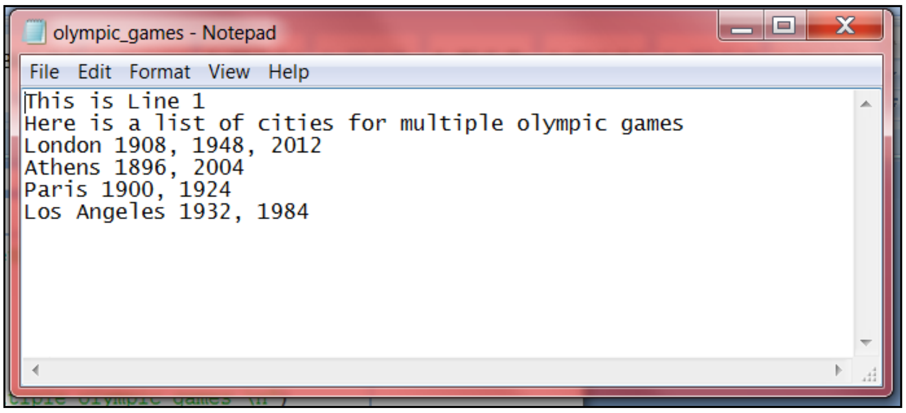
The Script

```
# Creating a file and writing to it using the write method

print("Writing to a text file with the write() method")

#Open the file in write mode
text_file = open("olympic_games.txt", "w")

text_file.write("This is Line 1\n")
text_file.write("Here is a list of cities for multiple olympic games \n")
text_file.write("London 1908, 1948, 2012 \n")
text_file.write("Athens 1896, 2004 \n")
text_file.write("Paris 1900, 1924 \n")
text_file.write("Los Angeles 1932, 1984 \n")

text_file.close()

#now open the file and print to the screen
text_file = open("olympic_games.txt", "r")
print(text_file.read())

text_file.close()
```

The Output

```
>>> ================================ RESTART ================================
>>>
Writing to a text file with the write() method
This is Line 1
Here is a list of cities for multiple olympic games
London 1908, 1948, 2012
Athens 1896, 2004
Paris 1900, 1924
Los Angeles 1932, 1984
```

The Text File
This text file will be created and saved in the current working directory.


olympic_games - Notepad

File Edit Format View Help

```
This is Line 1
Here is a list of cities for multiple olympic games
London 1908, 1948, 2012
Athens 1896, 2004
Paris 1900, 1924
Los Angeles 1932, 1984
```

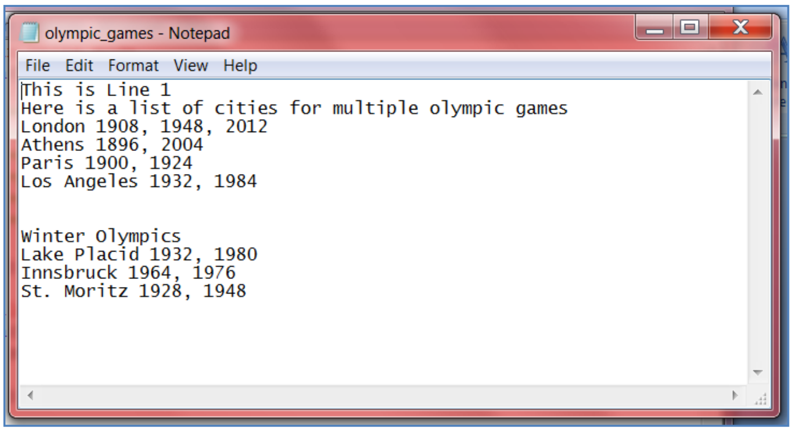
Note that text files take text (string) as input and, therefore, in order to write other data types to a text file they will first have to be converted to string. The *str()* function becomes very handy.

7.3 Appending a File

To add data to a file without deleting what is already there, we open the file in append mode.

```
#First open the file in append mode
text_file = open("olympic_games.txt", "a")

# Now we add the winter games host to the file
text_file.write("\n\nWinter Olympics\n")
text_file.write("Lake Placid 1932, 1980\n")
text_file.write("Innsbruck 1964, 1976\n")
text_file.write("St. Moritz 1928, 1948\n")

#Now close the file
text_file.close()
```

The code above will open the file in append mode and add the winter Olympic cities that have hosted the Games more than once. Again, we close the file after using it. We can then reopen the file and print the contents to the screen. We should see both sets of information in the file, and if we open the text file on our computer we should see the updated text file as seen below.


olympic_games - Notepad

File Edit Format View Help

```
This is Line 1
Here is a list of cities for multiple olympic games
London 1908, 1948, 2012
Athens 1896, 2004
Paris 1900, 1924
Los Angeles 1932, 1984


Winter Olympics
Lake Placid 1932, 1980
Innsbruck 1964, 1976
St. Moritz 1928, 1948
```

7.4 File Path

A **directory structure (folder structure)** is the way in which an operating system displays its files to the user. The Windows operating system uses drive names (typically identified by a letter) to denote the root directory or folder. The directory separator is the "\" (backslash), while the Unix-based operating system uses the "/" (forward slash) as its directory separator. A file name is therefore the unique address given to identify a file. There are two ways of stating a file name using the address: 1) relative path and 2) absolute path.

In the previous section, we used a relative file path to open files. Relative file paths start by using the current directory; therefore by running the code, Python will create or open a file called *sample.txt* from the current working directory.

```
>>> new_file = open("sample.txt", "w")
```

The current working directory is the folder that the running program is operating in. We can find out the current working directory by using the function `getcwd()`. The function is a part of the OS module and therefore we will have to import this module before we can use the function. The following code will print the path of the current working directory:

```
>>> import os
>>> print(os.getcwd())
C:\Users\Devz\PythonExercise
>>>
```

To create the file at some other location, we could use the absolute file path, where we give the full address, for example:

```
>>> new_file = open("C:\\Users\\Devz\\PythonExercise\\sample.txt", "w")
>>>
```

This will open or create a file called sample.txt in the specified location *C:\Users\Devz\PythonExercise\* It is worth mentioning that in the command above we use double backslash to denote one backslash; the reason for this is that "\" denotes an escape sequence in Python, therefore we use two backslashes for a backslash. Alternatively, we could proceed the file path with "r". File paths such as the one below are called **absolute** file paths; the difference is that the address is given from the topmost folder on the drive.

```
>>> new_file = open(r"C:\Users\Devz\PythonExercise\sample.txt", "w")
>>>
```

7.5 Working with Databases

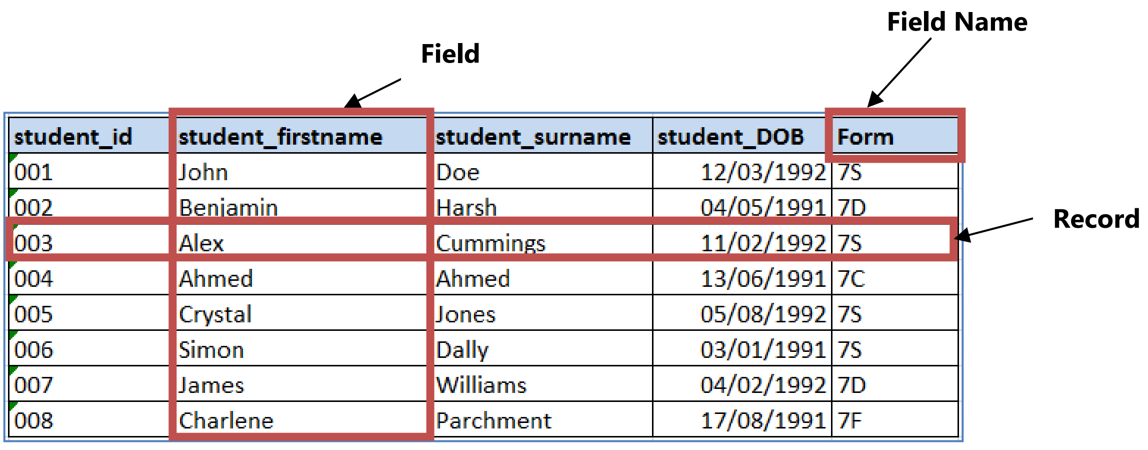
A database is a collection of tables that store data. The software that operates on this data is called a database management system (DBMS). Most databases will contain more than tables, but also queries, triggers and views; each table is identified by a unique name. Almost all organisations will store data and use a database at some point. It is therefore important for us to write programs that can use the data in a database and update the database. The most common operation carried out on a database is a query. Query in its strictest sense is asking the database a question and the database responds by providing all the data items that meet the criteria. Before we discuss queries, it is important to explain some key database terminologies. For the illustration in the table below, we will use a database with a table of students similar to what is used in schools.

Name	Meaning	Examples
Table	A collection of data relevant to one entity	A student's table
Records	Collection of all the data on a single entity	001, John, Doe, 12/03/1992, 7S
Field	Single column of data in a table	John Benjamin Alex Ahmed Crystal Simon
Field Name	The name given to the field	First Name

Table 5

Here is an example of a table within a database with the sections labelled.

Name of Table: Students


Field Name
Field
Record

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

Each field in the data has a data type. The data types are similar to the ones discussed in Chapter 4. The universal language used to create and manipulate databases is called structured query language (SQL pronounced sequel).

Some DBMS, such as Microsoft Access, will generate the SQL codes for the user. In order to use databases in Python, it is worth understanding some SQL commands, we will turn our attention to this.

7.6 SQL

We will discuss the following statements available in SQL:

- CREATE TABLE – *this will create a table in a database*
- SELECT – *display data that is in a table in a database*
- UPDATE – *change data in a table in a database*
- DELETE – *remove data from a table in a database*
- INSERT INTO – *insert new data into a table in a database*

7.6.1 The CREATE TABLE Statement

We first use the CREATE TABLE statement to create the table that we will be using as a running example.

Syntax:

```
CREATE TABLE <table_name>
(
<field_name> <data_type>,
<field_name> <data_type>,
<field_name> <data_type>
)
```

`<table_name>` is the name of the table to be created in the database.

`<field_name>` is the name of the column to be created in the table.

`<data_type>` is the data type of the column.

Example:

```
CREATE TABLE Students
(
student_id text,
student_firstname text,
student_surname text,
student_DOB date,
Form text
)
```

This will create the following table called "Students" in the database

student_id	student_firstname	student_surname	student_DOB	Form

For the rest of the SQL section, we will use the following table as our running example.

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

7.6.2 The SELECT Statement

The SELECT statement is used to display data in a table. We can view all the data, or we can specify a criterion, in which case only data that meets the criteria will be displayed.

Syntax:

```
SELECT <field_name> FROM <table_name>
```

`<field_name>` is the name of the column to be selected.
`<table_name>` is the name of the table in the database.

Example:

```
SELECT student_firstname FROM Students
```

This will produce the following output:

student_firstname
John
Benjamin
Alex
Ahmed
Crystal
Simon
James
Charlene

```
SELECT student_firstname, student_surname FROM Students
```

This will produce the following output:

student_firstname	student_surname
John	Doe
Benjamin	Harsh
Alex	Cummings
Ahmed	Ahmed
Crystal	Jones
Simon	Dally
James	Williams
Charlene	Parchment

The wildcard (*) is a quick way of selecting all columns in the table and, therefore:

```
SELECT * FROM Students
```

will produce the entire table

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

7.6.3 The WHERE Clause

The "Where" clause is used in conjunction with the select statement; this is used to specify a criteria in the selection.

Syntax:

```
SELECT <field_name> FROM <table_name>
WHERE <field_name> operator <value>
```

`<field_name>` is the name of the column to be selected.
`<table_name>` is the name of the table in the database.
`operator` is any operator from the table below.
`<value>` is any legal value that can be stored in the field name selected.

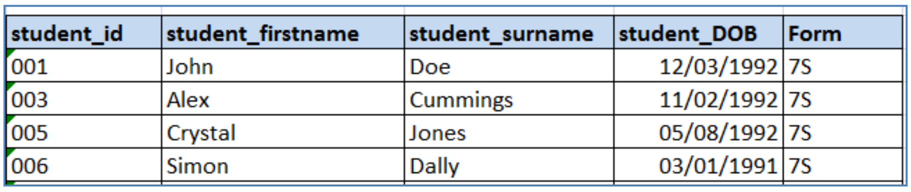
Operator	Meaning
=	Equal to
<> or !=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
BETWEEN	Value between a given range including the bounds
LIKE	Matches a given pattern
IN	To find more than one value in a column

Table 6

Example:

```
SELECT * FROM Students
WHERE Form='7S'
```

This will produce



student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
003	Alex	Cummings	11/02/1992	7S
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S

Notice that '7S' requires single quotation marks as the data type is text; there is no need for quotation marks if the data type is numeric.

7.6.4 The UPDATE Statement

This statement is used to change records by updating them in a table in a database.

Syntax:

```
UPDATE <table_name>
SET <column1> = value1, <column2>=value2, ...
WHERE <field_name> = value
```

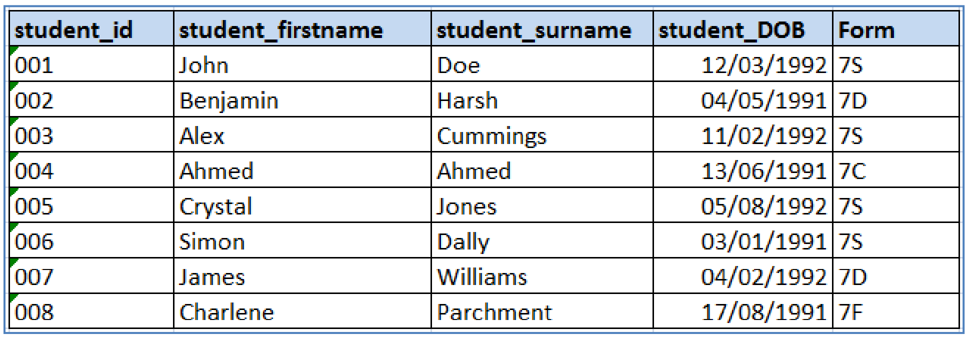
`<table_name>` is the name of the table in the database.
`<column1>` is the first column to be updated.
`<column2>` is the second column to be updated.
`<field_name>` specifies the record to be updated.

Note that the WHERE clause specifies the particular record to be updated. If this is left blank, then all the records will be updated.

Example:

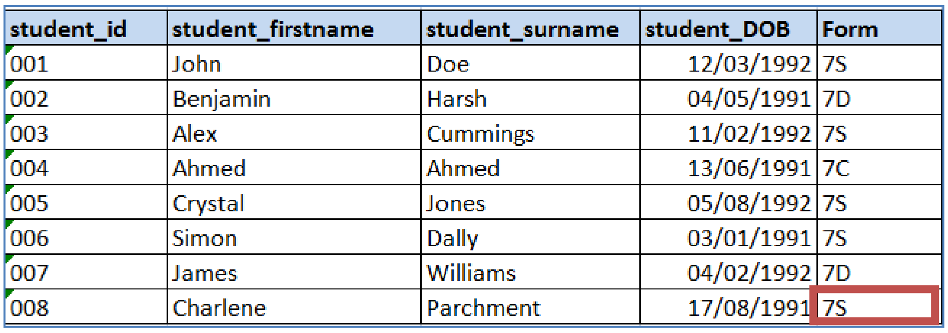
```
UPDATE Students
SET Form = '7S'
WHERE student_firstname='Charlene'
```

Before the update:



student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

After the update:



student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7S

Notice that the Charlene is now in form 7S.

7.6.5 The DELETE Statement

This statement is used to remove records from a database.

Syntax:

```
DELETE FROM <table_name>
WHERE <field_name> = value
```

`<table_name>` is the name of the table in the database.

`<field_name> = value` specifies the record to be deleted.

Example:

```
DELETE FROM Students
WHERE student_firstname='Charlene'
```

Before the delete statement:

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

After the delete statement:

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D

7.6.6 The INSERT INTO Statement

The insert into statement is used to insert a new record into a database.

Syntax:

```
INSERT INTO <table_name> (column1, coulmn2, column3, ...)
VALUES (value1, value2, value3, ...)
```

`<table_name>` is the name of the table in the database.
`column` is the column into which the value should be inserted.
`value` is the value to be inserted into the matching column.

Note that the column is optional, but SQL will insert the values into consecutive columns if they are omitted. Specifying the column is a good way to insert data into non-consecutive columns.

Example:

```
INSERT INTO Students
VALUES ('009', 'Paul', 'Piggot', '15/07/1992', '7C')
```

Before the insert into statement:

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7F

After the insert into statement:

student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S
002	Benjamin	Harsh	04/05/1991	7D
003	Alex	Cummings	11/02/1992	7S
004	Ahmed	Ahmed	13/06/1991	7C
005	Crystal	Jones	05/08/1992	7S
006	Simon	Dally	03/01/1991	7S
007	James	Williams	04/02/1992	7D
008	Charlene	Parchment	17/08/1991	7S
009	Paul	Piggot	15/07/1992	7C

We could have used the field name to specify the columns where we want the data to be inserted, but in this case we want to insert it into all columns and therefore it is not necessary.

7.7 SQL and Python

Python provides the SQL*ite3* library to manipulate lightweight disk databases. Before we try to create or use a database, we first have to import the SQLite3 library. We use the keyword import for this. This is demonstrated below.

```
#we first import the sqlite3 library

import sqlite3
```

Connect

Now that the SQLite3 library is imported, we can connect to and use a database. Similar to files, before we use a database we need to have a handler in our program that interacts with the physical database on a disk or in the RAM. We use the `connect()` function, as defined in the SQLite3 library. This function will open a connection to the SQLite database file. The following code snippet will connect to a database called "School".

```
#we first import the sqlite3 library

import sqlite3

#Create a handler called new_db that connects to db on disk
new_db = sqlite3.connect('C:\\Users\\Devz\\PythonExercise\\school.db')
```

The argument to the function is the file path for the database; notice that we use double backslash '\\' in our file path as single backslash denotes an escape sequence.

Create Cursor Object

In Python, the cursor object allows us to work with and manipulate databases. To create a cursor object we call the cursor method on the connection object. In our example, the connection object is new_db, we will create a new cursor called c.

```
#we first import the sqlite3 library

import sqlite3

#Create a handler called new_db that connects to db on disk
new_db = sqlite3.connect('C:\\Users\\Devz\\PythonExercise\\school.db')

#Create a new cursor object to manipulate the database
c=new_db.cursor()
```

Now that we have a connection to our database and a cursor object, we can execute SQL statements to create our table and insert data into our tables. We will create a table called "Students" with the same properties as before.

Create Table

We first create a table.

```
#Now create a table called Students
c.execute('''CREATE TABLE Students
(student_id text,
student_firstname text,
student_surname text,
student_DOB date,
Form text)
''')
```

Notice that we are using SQL commands to create the table.

Use the INSERT INTO statement to populate the table.

Next, we use the INSERT INTO statement to insert values into our table.

```
#insert data into our table
c.execute('''INSERT INTO Students
          VALUES ('001', 'John', 'Doe', '12/03/1992', '7S')''')
```

Notice that we are using the version where we do not specify the field names as we will be inserting information into all columns.

Commit and Close

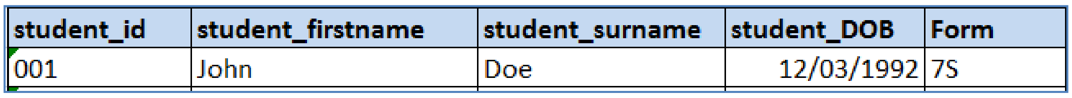
We use the `commit()` statement to save/commit the final transaction and finally close the database.

```
#Save changes using the commit() function
new_db.commit()

#Close the connection to the database
new_db.close()
```

Putting all the codes together:

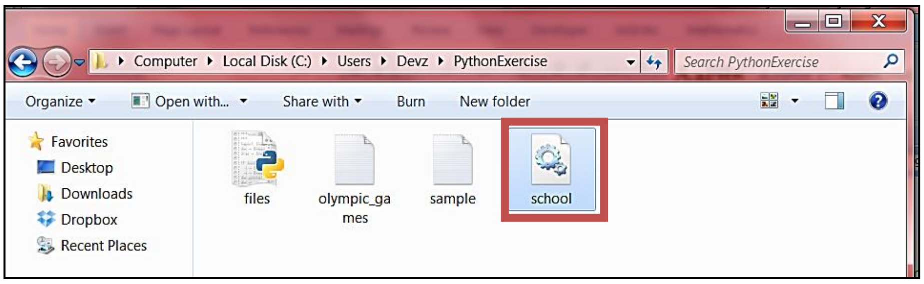
```
#we first import the sqlite3 library

import sqlite3

#Create a handler called new_db that connects to db on disk
new_db = sqlite3.connect('C:\\Users\\Devz\\PythonExercise\\school.db')

#Create a new cursor object to manipulate the database
c=new_db.cursor()

#Now create a table called Students
c.execute('''CREATE TABLE Students
(student_id text,
student_firstname text,
student_surname text,
student_DOB date,
Form text)
''')

#insert data into our table
c.execute('''INSERT INTO Students
          VALUES ('001', 'John', 'Doe', '12/03/1992', '7S')''')

#Save changes using the commit() function
new_db.commit()

#Close the connection to the database
new_db.close()
```

After executing this code, the database called school.db will be created in the specified location (*C:\Users\Devz\PythonExercise\school.db*), with a table called "Students" and the record inserted.



student_id	student_firstname	student_surname	student_DOB	Form
001	John	Doe	12/03/1992	7S

The physical files are shown below. Note that this file is not readable by normal programs on the computer, and therefore if we open this file we will only see symbols.



We can, however, use Python to read the data from this file and print this data to the screen. The following code snippet reconnects to the database, reads the file and prints the data to the screen.

```
#Re-connect to the database
new_db=sqlite3.connect('C:\\Users\\Devz\\PythonExercise\\school.db')

#Create a cursor object
c=new_db.cursor()

#Use the SELECT statement with wildcard to select all columns
c.execute("SELECT * FROM Students")

#use the fetchone function to fetch one row of data
row=c.fetchone()

#print the row to the screen
print(row)

#Finally close the connection
new_db.close()
```

The result of executing this code is:

```
>>> ================================ RESTART ================================
>>>
('001', 'John', 'Doe', '12/03/1992', '7S')
>>>
```

It is always good practice to close the database connection once you have finished using the database as other processes will not be able to access the opened database.

Here are some other functions defined on the object cursor and a description of what they do.

executemany(...)
    Repeatedly execute a SQL statement

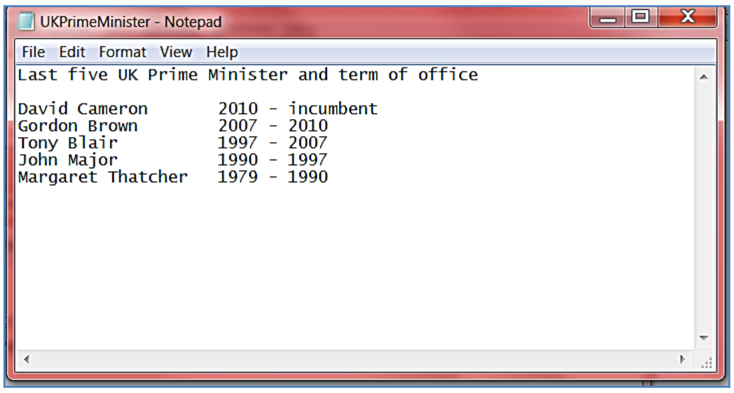
executescript(...)
    Executes multiple SQL statements at once

fethchall(...)
    Fetches all rows from the result set

fethchmany(...)
    Fetches several rows from the result set

Practice Exercise 7

1. a. Create a text file in the current working directory that creates and stores the text file that shows a list of the last five (5) Prime Ministers of the United Kingdom as shown below. ❑



```
Last five UK Prime Minister and term of office

David Cameron       2010 - incumbent
Gordon Brown        2007 - 2010
Tony Blair          1997 - 2007
John Major          1990 - 1997
Margaret Thatcher   1979 - 1990
```

b. Now write a Python program that will read the file called "`UKPrimeMinister.txt`", that was created above, and print the content to the screen. The output is shown below. ❑

```
>>> ================================ RESTART ================================
>>>
Writing text file
Last five UK Prime Minister and term of office

David Cameron       2010 - incumbent
Gordon Brown        2007 - 2010
Tony Blair          1997 - 2007
John Major          1990 - 1997
Margaret Thatcher   1979 - 1990

>>>
```

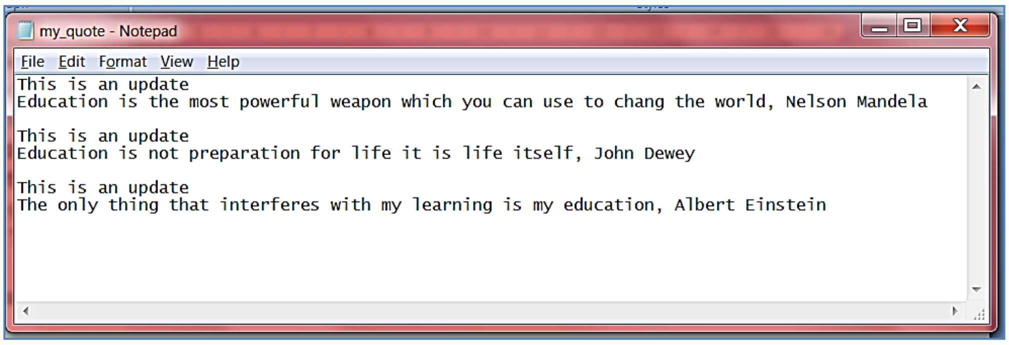
2. The code below should create a text file called "`my_quote.txt`". The function `update_file()` is defined to take a file name and a string containing a quote. The function should open the file called "my_quote.txt" and update it with a new quote. The program should prompt the user to enter a quote three times. Then print the contents of the file to the screen. At the moment there are two things wrong with the code.
   a. Only the last quote is saved in the file
   b. The user is being prompted four times instead of three

   Examine the code and correct it to get the desired outcome. ❑

Desired Outcome:

```
>>> ================================ RESTART ================================
>>>
Enter your favourite quote: Education is the most powerful weapon which you can use to chang the world, Nelson Mandela
Enter your favourite quote: Education is not preparation for life it is life itself, John Dewey
Enter your favourite quote: The only thing that interferes with my learning is my education, Albert Einstein
This is an update
Education is the most powerful weapon which you can use to chang the world, Nelson Mandela

This is an update
Education is not preparation for life it is life itself, John Dewey

This is an update
The only thing that interferes with my learning is my education, Albert Einstein

>>>
```

Text File:



```
This is an update
Education is the most powerful weapon which you can use to chang the world, Nelson Mandela

This is an update
Education is not preparation for life it is life itself, John Dewey

This is an update
The only thing that interferes with my learning is my education, Albert Einstein
```

Code to be Corrected:

```python
# update three quotes to a file

file_name = "my_quote.txt"
#create a file called my_quote.txt
new_file = open(file_name, 'w')
new_file.close()



def update_file(file_name, quote):
    #First open the file
    new_file = open(file_name, 'w')
    new_file.write("This is an update\n")
    new_file.write(quote)
    new_file.write("\n\n")

    #now close the file
    new_file.close()

for index in range(1,3):
    quote = input("Enter your favourite quote: ")
    update_file(file_name, quote)


# Now print the contents to the screen
new_file = open(file_name, 'r')
print(new_file.read())

# And finally close the file
new_file.close()
```

3. Complete the following code to create a database called “Library.db” that has a table called “Books”. The code should then populate the table with the following information. ❑

book_isbn	book_title	book_type	book_author	publisher
978-0-340-88851-3	A2 Pure Mathematics	Non fictional	Catherine Berry	Hodder Education
978-1-118-10227-5	Android 4 Application Development	Non fictional	Reto Meier	Wiley
0-596-00699-3	Programming C#	Non fictional	Jesse Liberty	O Reilly

It should then print the snippet overleaf back to the screen.

Incomplete Code

```
#we first import the sqlite3 library
import sqlite3

#insert the correct path here
new_db = sqlite3.connect('C:')

#Create a new cursor object to manipulate the database
c=new_db.cursor()

#Now create a table called Students
c.execute('''CREATE TABLE Books
(book_isbn text,
book_title text,
book_type text,
book_author text,
publisher text)
''')
#insert Statements here to add data to the table


new_db.commit()

#Close the connection to the database
new_db.close()


#Re-connect to the database - insert the correct path
new_db=sqlite3.connect('C:')

#Create a cursor object
c=new_db.cursor()

#Use the SELECT statement to select all the data

#use the fetchone function to fetch one row of data
book_library=c.fetchall()

#print the row to the screen
for book in book_library:
    print(book)

#Finally close the connection
new_db.close()
```

4. Explain the effect of running the following code on a database.

   a. *`SELECT Product.Name, Product.Quantity, Product.Price`*

   ....

   ....

   b. *`FROM Product`*

   ....

   ....

   c. *`WHERE Product.Quantity > 20`*

   ....

   ....