Chapter 18

File Handling

Learning Objectives

After completing this chapter, the reader will be able to:

- Create, read, write, and update files.
- Become familiar with sequential-access file processing.
- Understand text processing in Python.
- Learn about the binary files and Excel files

- Differentiate between a text file and a binary file.
- Open a text file for input/output and write strings or numbers to the file
- Learn about context managers

18.1 Introduction to Files

Most of the current file systems include three primary parts.

- **Data:** the file’s contents, as written by the author or editor.
- **End of file (EOF):** special character used to denote the end of a file.
- **Header:** the file’s header contains information about the file’s contents (file name, size, type, and so on)

File Access Modes

File access mode defines what operations can be performed. There are many access modes in Python, namely,

(i) Read Mode,

(ii) Write Mode and

(iii) Append Mode.

Read Access Modes

(i) Read only (‘r’):

(ii) Read and write (‘r+’)

Write Access Modes

(i) Write only (‘w’)

(ii) Write and read (‘w+’)

Append Access Modes

(i) Append only, without truncating a file (‘a’)

(ii) Append with read and write (‘a+’)

18.2 Types of Files

Text Files : Text files are :

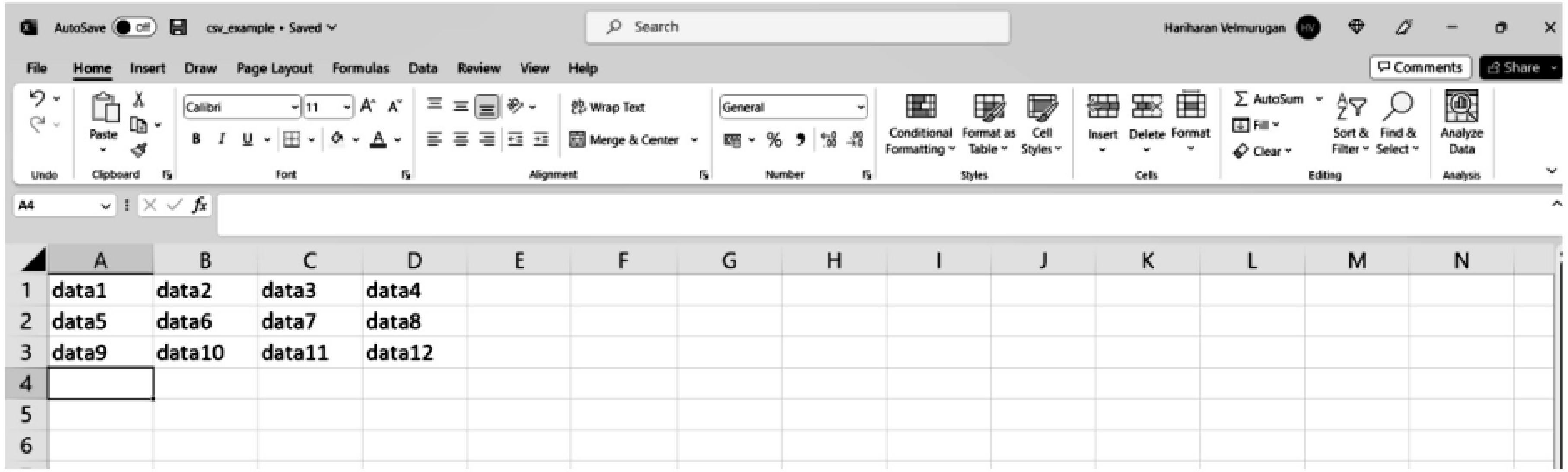
- configuration (like cfg, ini, and reg),
- documents (like txt, tex, and RTF),
- source code (like c, app, js, py, and java),
- tabular data (like CSV, and tsv), and
- web standards (like HTML, XML, CSS, and JSON).

Binary Files

binary files are

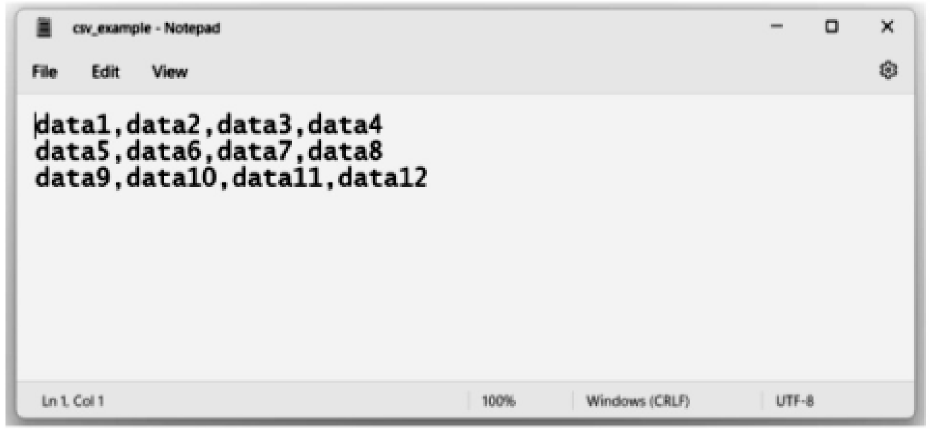
- archive files (like .zip, .rar, .iso, and .7z),
- audio files (like .mp3, .wav, .mka, and .aac),
- database files (like .mdb, .accde, .frm, and .sqlite),
- document files (like .pdf, .doc, and .xls),
- executable files (like .exe, .dll, and .class),
- image files (like .png, .jpg, .gif, and .bmp), and
- video files (like .mp4, .3gp, .mkv, and .avi).

CSV Files



	A	B	C	D
1	data1	data2	data3	data4
2	data5	data6	data7	data8
3	data9	data10	data11	data12
4				
5				
6				

#This is the screenshot of the output



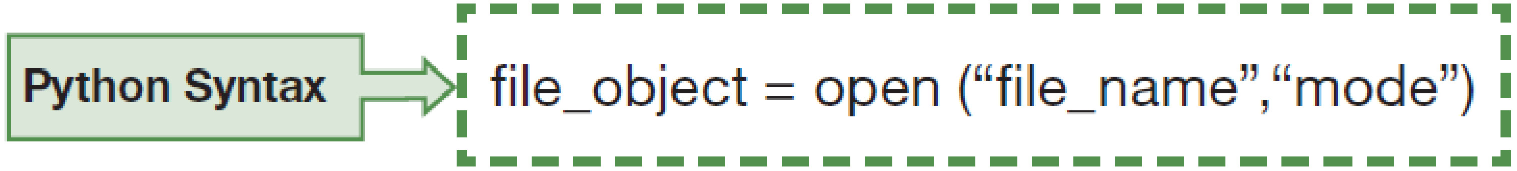
```
data1,data2,data3,data4
data5,data6,data7,data8
data9,data10,data11,data12
```

18.3 Input and Output Operations

To open a file: Use Python’s built-in open function or open() command to acquire and open a file object.

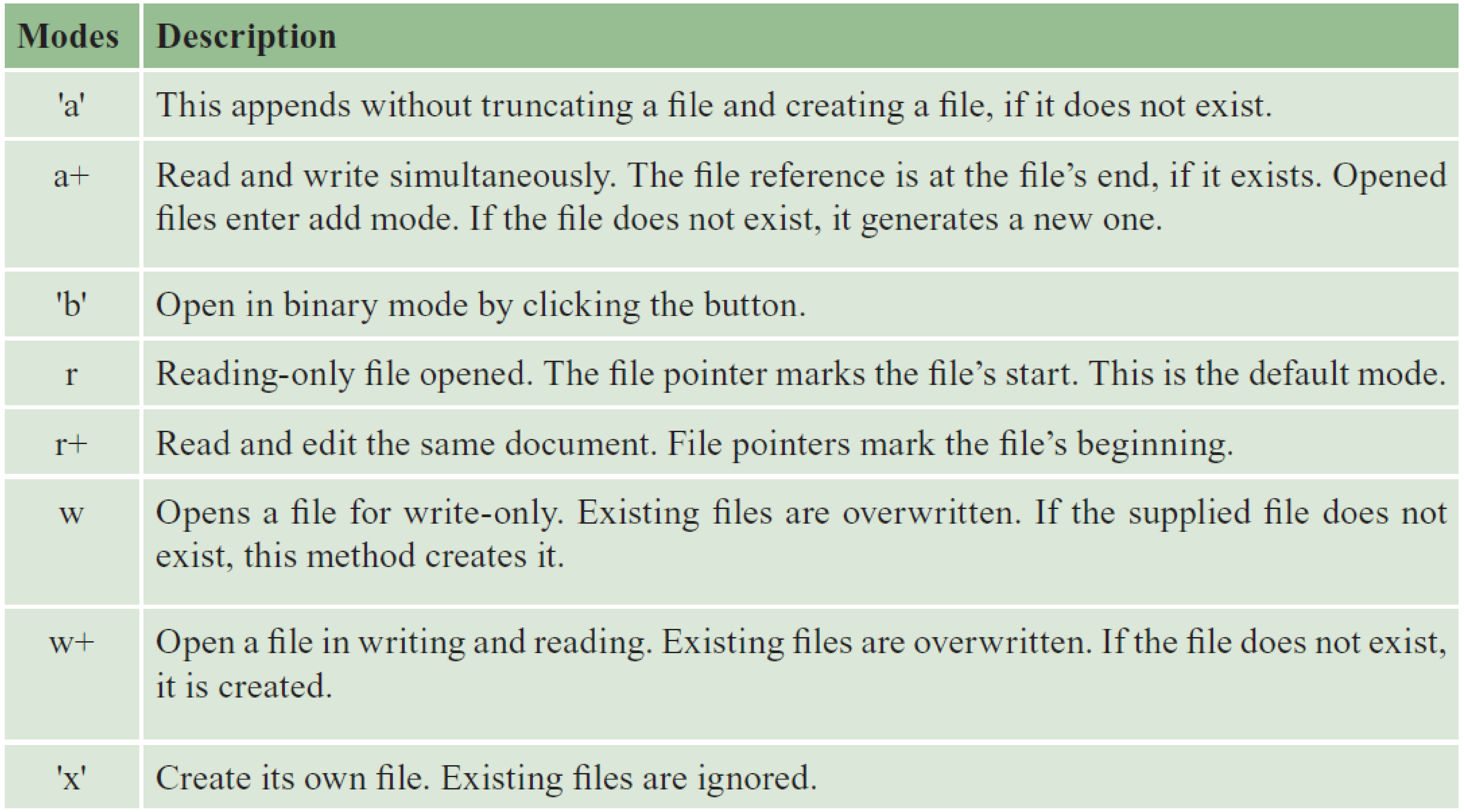
The open function returns a file object when used.

File objects have ***methods*** and ***attribute*** to collect/manipulate file information. This function returns a file object and is also called a ***handle*** (because it reads or modifies files).



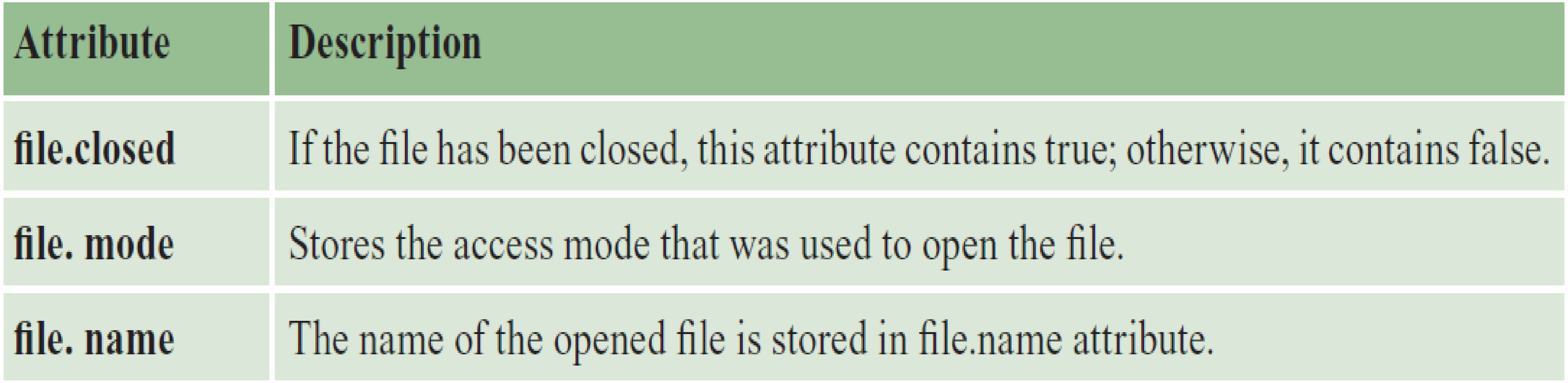
```
file_object = open (“file_name”,“mode”)
```

Table 18.1: Modes of opening the file and their Description



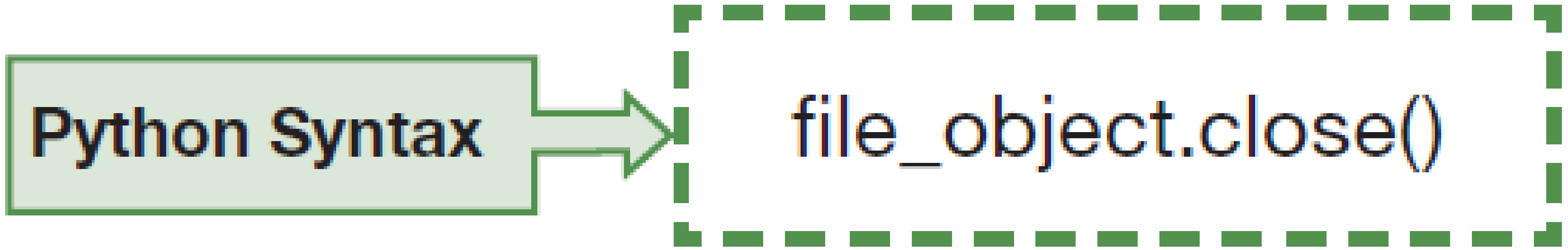
Modes	Description
'a'	This appends without truncating a file and creating a file, if it does not exist.
a+	Read and write simultaneously. The file reference is at the file's end, if it exists. Opened files enter add mode. If the file does not exist, it generates a new one.
'b'	Open in binary mode by clicking the button.
r	Reading-only file opened. The file pointer marks the file's start. This is the default mode.
r+	Read and edit the same document. File pointers mark the file's beginning.
w	Opens a file for write-only. Existing files are overwritten. If the supplied file does not exist, this method creates it.
w+	Open a file in writing and reading. Existing files are overwritten. If the file does not exist, it is created.
'x'	Create its own file. Existing files are ignored.

Table 18.2: Attributes associated with File Object and its Description



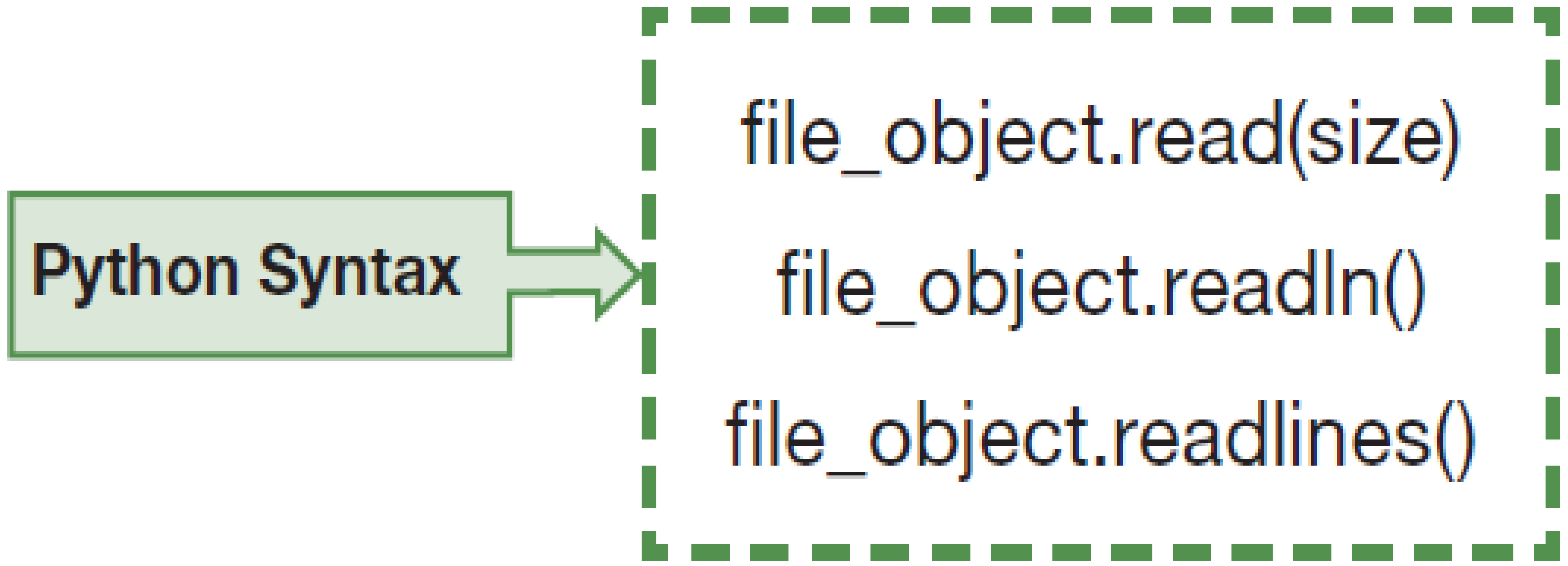
Attribute	Description
file.closed	If the file has been closed, this attribute contains true; otherwise, it contains false.
file. mode	Stores the access mode that was used to open the file.
file. name	The name of the opened file is stored in file.name attribute.

To close a file: A file must be properly closed when all activities have been completed. The Python close() frees up resources that a file object had previously held. The close() method does not delete the object (deletion of variable in Python environment) and I/O operations are not permitted on the closed file but basic information (Attributes and its values) is available with the File Object.



```
file_object.close()
```

To read from a file:



```
file_object.read(size)
file_object.readln()
file_object.readlines()
```

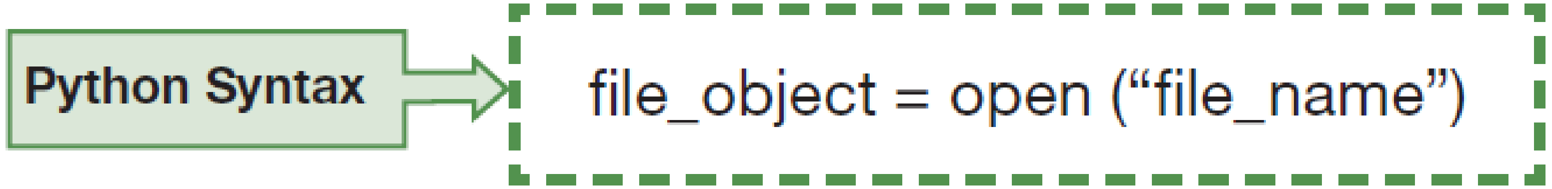
Table 18.3: File Methods in Python

Methods	Description
close()	Close a file that is currently open. If the file is already closed, it has no effect.
read(n)	The file can be read up to n characters at a time. The file is read to the end of the value and is negative or None.
readline(n=-1)	Take one line from the file and return it. Inputs no more than n bytes, if no limit is given.
readlines(n=-1)	A list of lines from the file is returned. Provides a limit of n bytes or characters to be read, if provided.
seek (offset)	The file's location should now be offset bytes.

Methods	**Description**
write(s)	Return the number of characters written to the file after the string is written.
writelines(lines)	Add lines to the document.

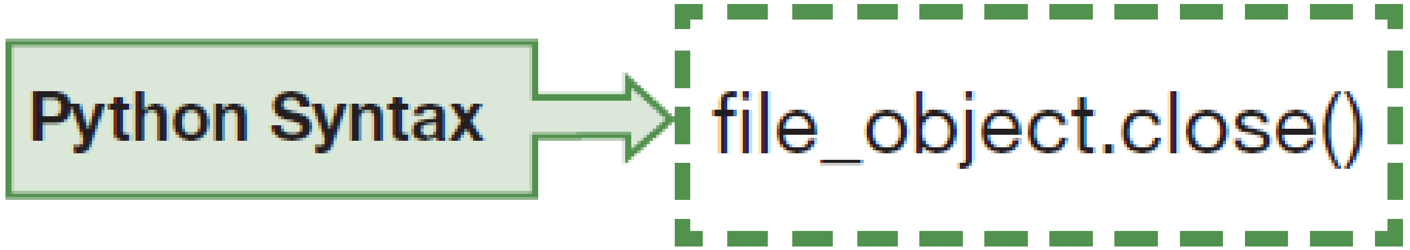
18.4. File handling operations in Text Files

To open a text file: open function(open ()) creates text file objects and opens the file



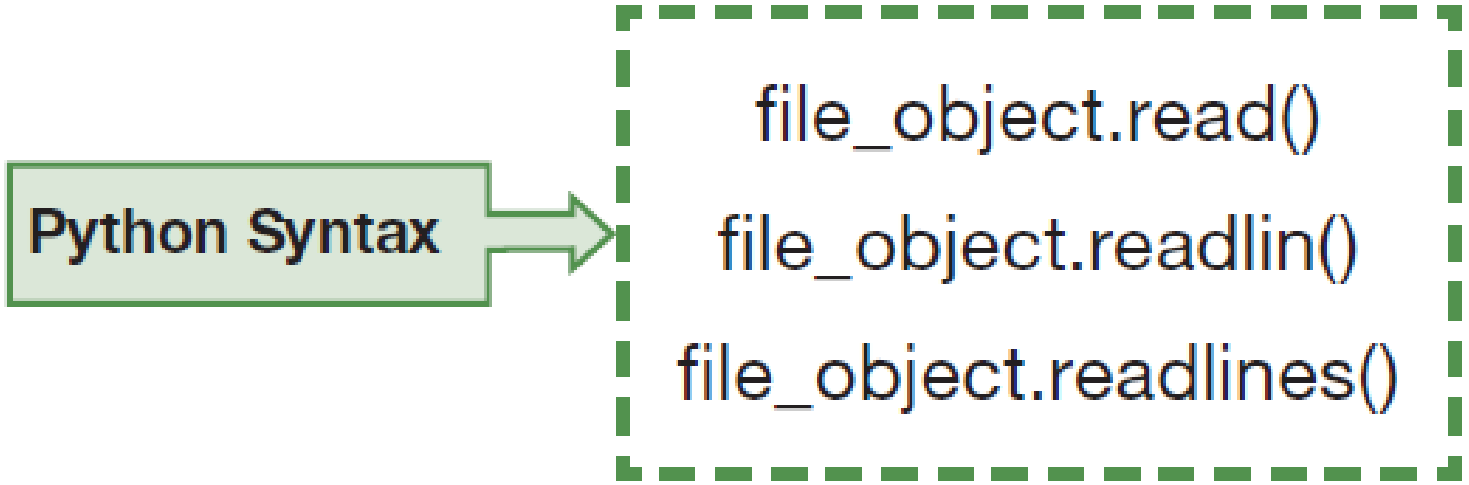
```
file_object = open (“file_name”)
```

To close a text file: after completing all operations to the file, it is necessary to close it correctly because the files’ resources will be freed up using the Python close() method



```
file_object.close()
```

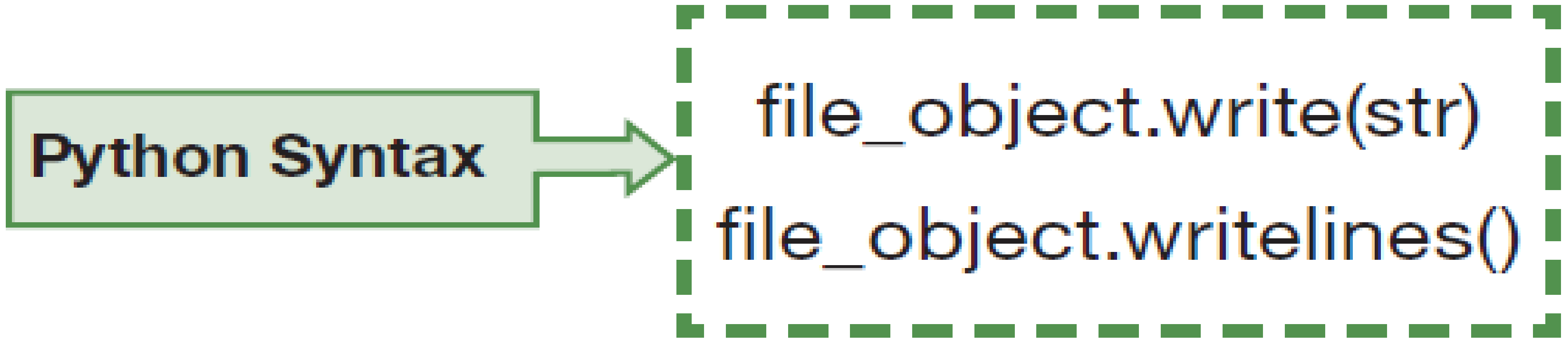
To read from the text file: The open function reads a Python file. The open() is used to return the file object called file handle.



There are three ways to read text from a text file using the file object.

- **read(size = n)**
- **readline()**
- **readlines()**

To write into the text f ile: writing to a file is easy. The file handle object’s write() method is used to write the data .



```
file_object.write(str)
file_object.writelines()
```

- **write()**: Text is written to a file using the write() method.

- **writelines()**: Write a string list to a file using the writelines() method. Rather than only accepting a list, the writelines() method also takes different iterable objects.

18.5. File handling operations in binary files.

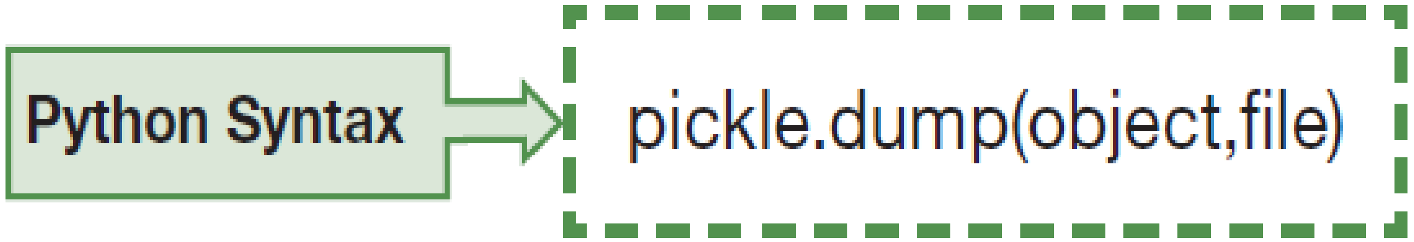
A binary file can be anything, including

- binary document formats (Word and PDF),
- binary image,and
- audio files, such as JPEG, GIF, or MP3.

The format of binary file format does not contain readable characters.

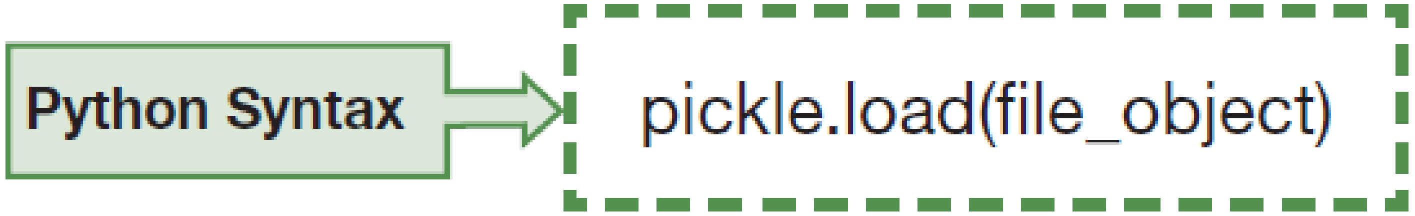
18.5.1. Pickle()

The pickle module serializes and deserializes Python objects.



```
pickle.dump(object,file)
```

To read data from a binary file or object, the pickle module uses the load() function. Load() de-serializes data streams.



```
pickle.load(file_object)
```

18.6. File handling operations in Excel files.

Programmer can do tasks like reading and writing to an Excel sheet file (i.e., Excel Files [xlsx, xlsm,xls]) or CSV Files using Python’s built-in support for this.

Programmers can carry out these activities owing to a variety of libraries that are accessible.

18.7 Python Context Manager

Python’s context managers help manage resources. It has two phases, namely the setup phase and a teardown phase.

General strategies usually adapted for resource management are:

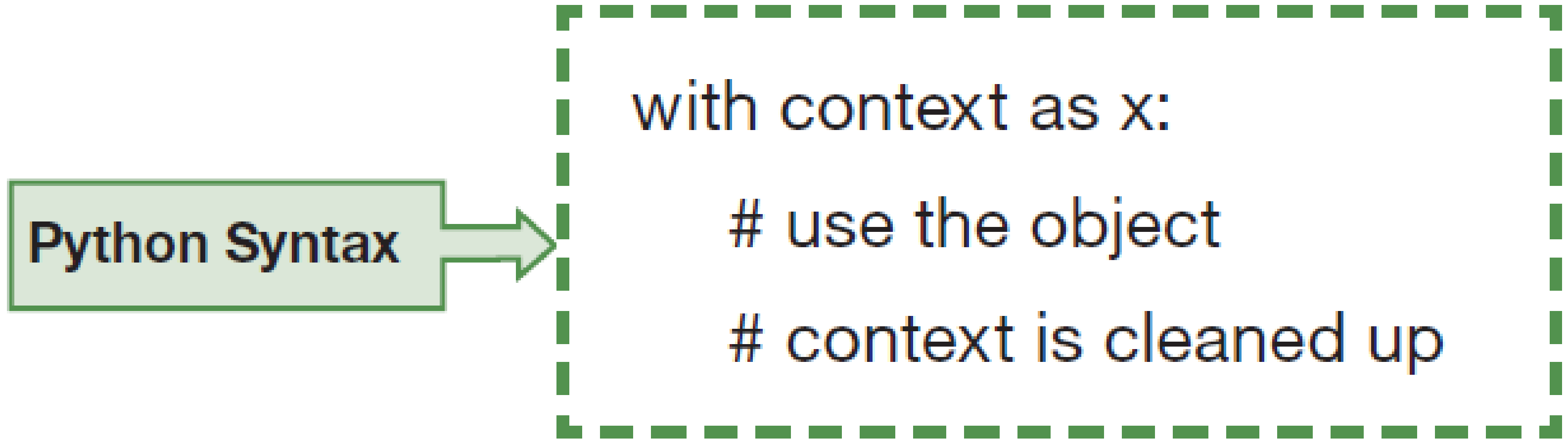
(i) A try-finally structure

(ii) A structure using with.

(iii) Using generators.

(iv) Using class.

- With the generator, class will be dealt with in this section.
- **Context manager using WITH:** a context manager in Python is often used as an expression in the "with" statement to aid in the automatic management of resources. The frequently used method to invoke a context manager is using a with statement. syntax is as follows.



```
with context as x:
    # use the object
    # context is cleaned up
```

Context manager using CLASS: a class File(object) is created with three functions, namely

- __init__ method to open the file.
- __enter__method to open the file and return it.
- __exit__method to close the file.