

# Reading and writing files

## Chapter 13

# This chapter covers

- Opening files and file objects
- Closing files
- Opening files in different modes
- Reading and writing text or binary data

# Opening files and file objects

- The traditional way that file paths and filesystem operations have been handled in Python is by using functions included in the `os` and `os.path` modules.
- These functions have worked well enough but often resulted in more verbose code than necessary.
- Since Python 3.5, a new library, `pathlib`, has been added; it offers a more object-oriented and more unified way of doing the same operations.

# Read text file: KORD.TXT

```
1  CHICAGO O'HARE INTERNATIONAL, IL, United States (KORD) 41-59N 087-55W 200M
2  Oct 10, 2017 - 10:44 PM EDT / 2017.10.11 0244 UTC
3  Wind: from the NE (040 degrees) at 14 MPH (12 KT):0
4  Visibility: 9 mile(s):0
5  Sky conditions: overcast
6  Weather: light rain
7  Precipitation last hour: 0.05 inches
8  Temperature: 59.0 F (15.0 C)
9  Dew Point: 55.9 F (13.3 C)
10 Relative Humidity: 89%
11 Pressure (altimeter): 30.03 in. Hg (1016 hPa)
12 ob: KORD 110244Z 04012KT 9SM -RA BKN013 BKN060 OVC100 15/13 A3003 RMK AO2 P0005 T01500133
13 cycle: 2
```

# Program to read text file

```
1  myfile = "../data/KORD.TXT"
2
3  ✓ with open(myfile, 'r') as file_object:
4      # read the first line
5      line = file_object.readline()
6      print(line)
7
8      # keep reading
9  ✓ while file_object.readline() != "" :
10     line = file_object.readline()
11     print(line)
12
13  # close
14  file_object.close()
```

# Opening files in write or other modes

- The second argument of the open command is a string denoting how the file should be opened.
  - 'r' means "Open the file for reading,"
  - 'w' means "Open the file for writing" (any data already in the file will be erased),
  - and 'a' means "Open the file for appending" (new data will be appended to the end of any data already in the file).
  - If you want to open the file for reading, you can leave out the second argument; 'r' is the default.

# Reading and writing with pathlib

- In addition to its path-manipulation powers discussed in chapter 12, a Path object can be used to read and write text and binary files.
- This capability can be convenient because no open or close is required, and separate methods are used for text and binary operations.
- One limitation, however, is that you have no way to append by using Path methods, because writing replaces any existing content.

```
1  from pathlib import Path
2
3  myfile = "mytextfile.txt"
4  mybinfile = "mybinfile"
5
6  # open text file
7  p_text = Path(myfile)
8
9  # write to file
10 p_text.write_text('Text file contents')
11
12 # read file
13 line = p_text.read_text()
14 print(line)
15
16 # open binary file
17 p_binary = Path(mybinfile)
18 p_binary.write_bytes(b'Binary file contents')
19
20 # read bin file
21 binline = p_binary.read_bytes()
22 print(binline)
```



# Screen input/output and redirection

- Use the built-in input method to prompt for and read an input string.

```
1  # get string input
2  x = input("Enter file name to use: ")
3  print(x)
4
5  # get number
6  x = int(input("Enter your number: "))
7  print(x)
```

```
1  import sys
2
3  print("Write to the standard output.")
4
5  sys.stdout.write("Write to the standard output.\n")
6
7  # get user input
8  s = sys.stdin.readline()
9  print(s)
```

# Summary

- File input and output in Python uses various built-in functions to open, read, write, and close files.
- In addition to reading and writing text, the struct module gives you the ability to read or write packed binary data.
- The pickle and shelve modules provide simple, safe, and powerful ways of saving and accessing arbitrarily complex Python data structures.