Chapter 13 Files and Exception Handling

1

To open a file for reading, use

```
infile = open("c:\\book\\test.txt", "r")
```

To open a file for writing, use

```
outfile = open("c:\\book\\test.txt", "w")
```

To open a file for appending, use

```
outfile = open("c:\\book\\test.txt", "a")
```

2

The file name is wrong. It should be

```
infile = open("c:\\book\\test.txt", "r")
```

3

When you open a file for reading, a runtime exception will be thrown if a file does not exist. When you open a file for writing, if a file already exists, the file will be destroyed.

4

You can use

os.path.isfile("test.txt")

to check if a file exists.

5

To read 30 characters from a file, use

s = file.read(30)

6.

To read the entire content from a file, use

s = file.read()

7.

To read one line from a file, use

s = file.readline()

8.

To read the entire content into a list of multiple lines from a file, use

```
list = file.readlines()
```

9.

The program will not have a runtime error if you invoke read() or readline() at the end of a file.

10.

If you invoke read() or readline() at the end of a file, the method will return ''. You can use this to detect the end of a file.

11.

To write data to a file, use

```
outfile.write(string)
```

12

You can use the notation r"string" to denote a raw string. For example, the following statement is fine.

```
input = open(r"c:\pybook\Scores.txt", "r")
```

13

To write a numeric value to a file, convert it to a string. To read a numeric value from a file, read it as a string and convert the string to a numeric value.

14

```
# Display a file dialog box for opening an existing file
filename = askopenfilename([options])
```

15

```
# Display a file dialog box for specifying a file for saving data
filename = asksaveasfilename([options])
```

16

To open a file for a Web page, use

```
infile = urllib.request.urlopen("http://www.yahoo.com/index.html")
```

17

To return a raw string from a string, use the decode function.

18

- Will statement3 be executed? No
- If the exception is not caught, will statement4 be executed? No
- If the exception is caught in the except block, will statement4 be executed? Yes
- If the exception is passed to the caller, will statement4 be executed? No

19

```
Index out of bound
```

20

```
Divided by zero!
```

21

```
Index out of bound
```

22

- Will statement5 be executed if the exception is not caught? No
- If the exception is of type Exception3, will statement4 be executed, and will statement5 be executed? Yes.

23

```
raise ExceptionClass()
```

24

It enables a function to throw an exception to its caller. The caller can handle this exception. Without this capability, the called function itself must handle the exception or terminate the program. Often the called function does not know what to do in case of error. This is typically the case for the library functions. The library function can detect the error, but only the caller knows what needs to be done when an error occurs. The essential benefit of exception handling is to separate the detection of an error (done in a called function) from the handling of an error (done in the calling method).

25

```
Done
Nothing is wrong
Finally we are here
Continue
```

26

```
Index out of bound
Finally we are here
Continue
```

27

```
except ZeroDivisionError:
```

should be placed before

```
except ArithmeticError:
```

28

Define a class that extends Exception or a subclass of Exception.

29

To open a file for writing binary data, use

```
outfile = open(file="filename.dat", "wb")
```

To open a file for reading binary data, use

```
infile = open(file="filename.dat", "rb")
```

30

To write binary data to a file, use the dump function from the pickle module as follows:

```
pickle.dump(data, outfile)
```

To read binary data from a file, use the load function from the pickle module as follows:

```
pickle.load(infile)
```

31

The file is closed in the finally clause after pickle.load(infile) is executed, even though the end of file is not reached.

32

Yes.