Introduction to Computer Programming
Chapter 7: Files and Exceptions

Michael Scherger

Department of Computer Science

Kent State University

Contents

- Read from text files
- Write to text files
- Read and write complex data with files
- File examples
- Exceptions and error handling
- Exception and error handling examples

Trivia Challenge Game

- Example: Trivia Challenge Program
- Data File: trivia.txt

Reading From Text Files

- Secondary storage is the computer’s hardware unit for the long term storage of data.
- Hard disks are the most popular, but tapes, floppy disks, CF cards, and USB drives are other hardware devices for storing data persistently.
- A file is a sequence of characters stored on a disk drive. Files have a name and an optional file extension.

Reading From Text Files

- Opening and closing a text file
  - Files have to be opened before a program can read (write) data from it
  - Syntax

```
any_file = open( “data.dat”, “r” )
```

  - Files also have to be closed when a program is finished with it
  - Syntax

```
any_file.close()
```

Reading From Text Files

- Opening and closing a text file
  - The file subdirectory path may have to be included in the file name

    ```
    any_file = open( “c:\ICP10061\exams\test1.txt”, “r” )
    ```

  - The “r” is called the “access mode”
    - r is for reading
      - if the file does not exist, an error is raised
    - w is for writing
      - If the file exists, the contents are overwritten. If the file does not exist, it will be created
    - a is for appending
      - If the file exists new data is appended to the end of the file. If the file does not exist, it will be created
    - others listed in the text book

Reading From Text Files

- Reading characters from a text file
  - There are several functions and methods for reading data from a file
  - The read() method allows a program to read a specified number of characters from a file and returns them as a string
  - Example

```
any_file = open( “test.txt”, “r” )
print any_file.read(5)
print any_file.read(4)
any_file.close()
```

  - Python remembers where the file last read data by using a “file pointer” or “bookmark”

Reading From Text Files

- Reading characters from a text file
  - All files have a “end of file” indicator (EOF) that signals to your program that there is no more data to read in a file
  - Trying to read past the end of the file will return an empty string
  - Example

```
any_file = open( “test.txt”, “r” )
all_the_data = any_file.read()
any_file.close()
```

Reading From Text Files

- Reading characters from a line
  - Text files are often line oriented and your program may have to read and process one line at a time
  - The method read_line() is used to read characters *<u>from the current line only</u>*
  - Example

```
anystring = any_file.readline(1)
anystring = any_file.readline(5)
anystring = any_file.readline()
```

Reading From Text Files

- Reading all lines into a list
  - Another way to work with lines from a file is to read each line (string) into a list
  - The method read_lines() is used to read all the lines from a text file and store them into a list of lines (strings)
  - Example

```
any_list = any_file.readlines()
```

Reading From Text Files

- Looping through a text files
  - Text files are a type of sequence delimited by lines
  - Python programs can also read and process lines from text files by using iteration
  - Example

```
for line in any_file:
    print line
```

Reading From Text Files

- Example: Read It Program
- Data File: read_it.txt

Writing To A Text File

- Program must also be able to write data to files for other programs (or humans) to read
- Many text files are created (written to) automatically and as needed
  - Automatic creation of web pages
  - Database and program log files

Writing To A Text File

- Writing strings to a file
  - There are several functions for writing data to a file
  - To write a single string to a text file use the write() method
  - Example

```
any_file.write(“This is a test…\n”)
any_file.write(“This is only a test\n” )
```

    - Note both strings could have been concatenated together into one string and have the same result using one write statement

Writing To A Text File

- Writing a list of strings to a file
  - The writelines() method is the complement function to readlines()
  - It takes a list of strings and prints them to a file
  - Example

```
any_file.writelines( any_list )
```

    - The newline characters must be embedded in each string for proper formatting (as needed)

Writing To A Text File

- Example: Write It Program

Storing Complex Data in Files

- Text files are convenient because humans can read and manipulate the data (strings)

- Reading and writing more complex data structures such as dictionaries may require more parsing on the part of the programmer

- Python provides a method of storing complex data using “pickling” which is a form of “serialization”

Storing Complex Data in Files

- Pickling data and writing it to a file
  - To use the pickling functions your program must include the cPickle module

    ```
    import cPickle
    ```

  - Next open a file exactly the same way for text files

    ```
    car_file = open( “cars.dat”, “w” )
    ```

  - Store your data by “dumping it”

    ```
    car_list = [“Chevy”, “Ford”, “Dodge”, “V W”,
      “Honda”, “Toyota”]
    cPickle.dump( car_list, car_file )
    ```

  - Close the file

    ```
    car_file.close()
    ```

Storing Complex Data in Files

- Reading data from a file and unpickling it
  - Open the data (pickle) file
  - Read the information using the load() method

    ```
    car_list = cPickle.load( car_file )
    ```

  - Close the file

Storing Complex Data in Files

- Using a shelf to store pickled data
  - A shelf can be thought of as a dictionary in a disk file
  - When you add key/value pairs to the shelf (dictionary) they are written to disk
  - Periodically the program must use the sync() method to copy the shelf changes to disk

Storing Complex Data in Files

- Using a shelf to store pickled data
  - To use a shelf, the program must import shelve (note the name change!!!)

    ```
    import shelve
    ```

  - Next open a shelve file

    ```
    stuff = shelve.open( “data.dat” )
    ```

  - Write data to the shelve

    ```
    stuff[‘cars’] = [‘Chevy’, ‘Ford’, ‘Dodge’]
    ```

  - Synchronize the data

    ```
    stuff.sync()
    ```

Storing Complex Data in Files

- Using a shelf to retrieve pickled data
  - To read information out of the shelf treat it as a dictionary and supply a key

```
for key in stuff.keys():
    print key, stuff[key]
```

Storing Complex Data in Files

- Example: Pickle It Program

Handling Exceptions

- When Python (or any programming language) has an error, it stops the current execution and displays an error message
- “It raises an exception”
- Example:

```
>>> 1/0

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in -toplevel-
    1/0
ZeroDivisionError: integer division or modulo by zero
```

Handling Exceptions

- Example: Handle It Program

Handling Exceptions

- Using a try statement with an except clause
  - The most basic way to “handle” (or trap) an exception is to use the Python “try” and “except” clause
  - Example

```
try:
    num = int( raw_input( “Enter a number” ))
except:
    print “Something went wrong”
```

Handling Exceptions

- Specifying an exception type
  - There are different type of exceptions
    - IOError
      - Raised when an I/O operation fails, such as opening a non-existent file for reading
    - IndexError
      - Raised when a sequence is indexed with a number out of range
    - KeyError
      - Raised when a dictionary key is not found
    - NameError
      - Raised when a name of variable or function is not found
    - SyntaxError
      - Raised when a syntax error is found
    - TypeError
      - Raised when a built-in operation or function is applied to an object with the wrong type
    - ValueError
      - Raised when a built-in operation or function received an argument that has the right type but inappropriate value
    - ZeroDivisionError
      - Raised when the second argument of a division or modulo operation is zero

Handling Exceptions

- Specifying an exception type
  - Example

```
try:
   num = int( raw_input( “Enter a number” ))
except(ValueError):
   print “Something went wrong”
```

  - The print statement is only executed if the ValueError exception is raised

Handling Exceptions

- When should you trap exceptions?
  - Any point of external interaction with your program
    - Opening a file for reading
    - Converting data from an outside source such as a user
  - If you do not know what exception to trap, test it in interactive mode with Python

Handling Exceptions

- Handling multiple exception types
  - The except clauses can trap multiple types of exceptions
  - Example:

```
for value in (None, “Hello”):
    try:
        print “Attempting to convert”, value, “->”,
        print float( value )
    except(TypeError, ValueError):
        print “An error occurred”
```

Handling Exceptions

- Handling multiple exception types
  - Example:

```
for value in (None, “Hello”):
    try:
        print “Attempting to convert”, value, “->”,
        print float( value )
    except(TypeError):
        print “Can only convert a string or a number”
    except(ValueError):
        print “Problem converting a string of digits”
```

Handling Exceptions

- Getting an exception’s argument
  - Python allows the program to get the actual error message…(useful for debugging)
  - Example:

```
try:
   num = int( raw_input( “Enter a number” ))
except(ValueError), e:
   print “Data entered was not a number”, e
```

Handling Exceptions

- Adding an else clause
  - Example:

```
try:
   num = int( raw_input( “Enter a number” ))
except(ValueError), e:
   print “Data entered was not a number”, e
else:
   print “The value of num is”, num
```

Trivia Challenge Game (Again)

- Example: Trivia Challenge Program
- Data File: trivia.txt

- Data file format

  <category>

  <question>

  <answer 1>

  <answer 1>

  <answer 1>

  <answer 1>

  <correct answer>

  <explanation>