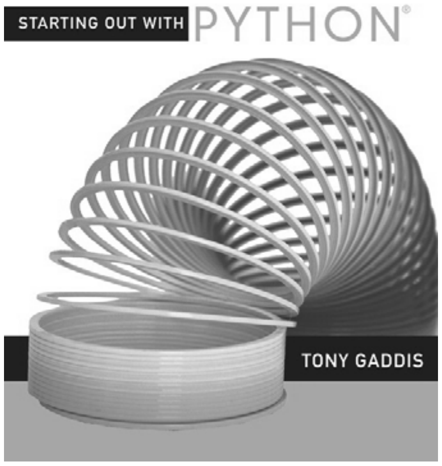
Topic 8: Files and Exceptions

1

Recommended Readings



- Chapter 7

2

Files

- Variables are temporary
  - Value is lost when program ends
  - Value is lost if computer loses power
- Files provide a less volatile form of storages
  - Values are retained after the program ends
  - Values are retained when the computer loses power

Types of Files

- Two types of files
  - Text files
    - Encoded using ASCII or Unicode
    - Can be viewed with editors such as Emacs and Notepad
    - Examples: Python source files, web pages, …
  - Binary files
    - Contain arbitrary sequences of bits which do not conform to ASCII or Unicode characters
    - Examples: Images, word processor files, …

File Access

- Two different ways to access data
  - Sequential Access
    - Start at the beginning of the file
    - Read data from the file in the order that it occurs
  - Random Access File
    - Jump to an arbitrary location in the file
    - Read some data
    - Jump to a new location
    - Read more data
    - …

5

Opening a Text File

- Text files are opened in one of three possible modes
  - Read
  - Write
  - Append

6

Opening a Text File

- Files must be opened before they can be used
  - Use the open function
    - Parameter 1: The name of the file to open
    - Parameter 2: The one character abbreviation for the mode
    - Returns a file handle
  - Example:

```
inf = open(“input_file.txt”, “r”)
```

7

Reading from a File

- Once a file has been opened, we can invoke methods on the file variable to read data from the file
  - read(): reads the entire file as a string
  - readline(): reads one line as a string
- Once we have read some data we often have to process it before it is ready for use by the rest of our program

8

Closing Files

- Files should be closed after we are done with them
  - Some operating systems limit the number of files that can be open at one time
  - Failing to close the file can result in a loss of data when writing to a file
  - Use the close method
    - Doesn’t require any parameters

9

Example: Computing a Sum

- Write a program that reads numbers from a file and computes their sum
  - User will enter the name of the file
  - Numbers will be in the file one per line

10

Example: Computing a Sum

11

Special Characters

- Inside of text files we have characters
  - letters
  - numbers
  - punctuation marks
- Also have characters that control spacing
  - newline (sometimes a 2 char sequence!)
  - tab
  - other control characters

12

Escape Sequences

- Escape sequences provide a mechanism for placing a character that controls spacing inside a string
  - What if we want to have a string that includes a newline character
  - What about a double quote character?
  - Use an escape sequence
    - Begins with \ (backslash)
    - Followed by one character describing the character that should be inserted

13

Escape Sequences

- Common Escape Sequences:
  - \n – linefeed
  - \t – tab
  - \” – double quote
  - \\ – backslash

14

Newline Headaches

- Representation of newline varies by operating system
  - Unix and MacOS X – newline is represented by the linefeed character, \n
  - DOS and Windows – newline is represented by two characters: a carriage return followed by a linefeed, \r\n
  - On MacOS 9 newline is represented by a carriage return, \r

15

Command Line Parameters

- Most programs require input to run
  - Can be read from the keyboard
  - Can be read from a file
  - Can come from parameters provided when the program is executed

16

Command Line Parameters

- Command line parameters are stored in the variable sys.argv
  - A list with one element for each parameter
  - The element at index 0 is the name of the program
  - All parameters are handled as strings
  - Don’t forget to import sys

17

Command Line Parameters

18

Command Line Parameters

- Update our program for summing numbers so that the name of the file is passed as a command line parameter

Example: Counting Words

- Write a program that computes some statistics about a text file
  - Number of lines
  - Number of words
  - Number of characters
- Filename will be provided as a command line parameter

Example: Counting Words

21

Writing to Files

- Data can be saved for future use by writing it to a file
  - Two writing options:
    - Create a new empty file
      - CAUTION: If the file already exists its contents will be lost without warning
    - Append data to the end of an existing file
  - Use the write method once the file is open
    - Takes one string as a parameter
      - Variables of other types must be cast to strings
    - Does not automatically include a newline character

22

Example: Counting Words

- Extend our word counting program so that it writes the report to a file
  - Name of file will be provided as a second command line parameter

Standard Input, Standard Output and Standard Error

- We have been using files since the first program that we wrote
  - Standard output is a file
    - Values written go to screen
    - Opened automatically when the program starts
    - Closes Automatically when the program ends
    - File variable is sys.stdout
    - We can write to stdout using the write method, just like any other file

Standard Input, Standard Output and Standard Error

- Standard Input
  - Also a file
  - The raw_input() function is equivalent to sys.stdin.readline().rstrip()
  - The input function does additional work to determine what type of value to return

25

Standard Input, Standard Output and Standard Error

- Standard Error
  - Another output file
  - Values written go to screen
  - Intended for displaying error messages instead of program output
  - Allows us to redirect program output separately from error messages
  - Useful for debugging

26

Example: Counting Words

- Extend our program so that it is more flexible
  - If no parameters are provided input is read from the keyboard and output is sent to the screen
  - If one parameter is provided input is read from the file named as a parameter and output is sent to the screen
  - If two parameters are provided, input is read from a file and output is written to a file

27

Files

- Text files provide an easy mechanism for loading and storing small amounts of data
  - Impractical for larger amounts of data that is frequently accessed (esp. in random order)
  - Accessing disks is slow
    - Important for programs that read and write frequently to optimize access to files
    - Common solution: Use existing database package

28

Exceptions

- What kinds of errors can occur?

Exceptions

- Most runtime errors are exceptions
  - If the exception isn't caught it causes the program to crash
    - Error messages say what exception was thrown and what line it was thrown from
  - Exceptions can be caught
    - Once the exception is caught, the program can take necessary actions to recover from the exception and then continue executing

Exceptions

- Consider the following program:

```
a = input("Enter a number: ")
b = input("Enter another number: ")

print a,"+",b,"=",a+b
print a,"-",b,"=",a-b
print a,"*",b,"=",a*b
print a,"/",b,"=",a/float(b)
```

- What can go wrong?

31

Exceptions

- Dividing by zero gives a ZeroDivisionError exception
  - We can catch this exception and provide different behavior
    - Create a try block which contains the code that might throw an exception
    - Create an except block to catch the exception and provide more desirable behaviour

32

Exceptions

- Rewrite the arithmetic program so that divide by zero exceptions are caught

33

Exceptions

- What happens if a string is entered instead of a number?
  - Extend our program so that this situation is handled more nicely

34

Exceptions

- Most file operations can throw exceptions
  - Try to open a file that doesn't exist
  - Try to read from a file that you don't have permission to read
  - Someone removes memory stick / CD while you are reading from it
  - These exceptions should be caught, even if the exception handler simply displays a meaningful message and quits the program

35

Multiple Exceptions

- Each try block can have several except blocks
  - Each kind of exception needs to be handled differently
  - May include a default except block that doesn't specify the type of exception to catch
    - Catches all exceptions that aren't caught by another except block
    - Typically displays an error message and exits

36

Exceptions

- Exceptions:
  - are thrown when an error occurs
  - can be caught to recover from the error
- We have only scratched the surface:
  - What happens if an exception is thrown inside a function?
  - How do we throw an exception ourselves?

37

Databases

- A structured collection of records organized for ease of search and retrieval
  - Many commercial packages exist
    - Oracle, Microsoft Access, SQL Server, DB2, …
  - Some free options too
    - MySQL, …
  - Packages often provide a GUI so that a user can access the data

38

Databases

- Most current databases use the relational model
  - Database consists of two parts
    - Schema: Describes the structure of the data
    - Data: The actual records being stored
  - Data is organized into tables
    - Each table consists of one or more (almost always) columns

39

Example

- A table to store data about people

40

Relationships

- How do we efficiently model the relationship that a person lives at an address?
  - Can more than one person live at an address?
  - Can a person have more than one address?

41

Primary Keys and Foreign Keys

- Primary Key:
  - A unique value associated with each row in a table
  - Typically an integer
- Foreign Key:
  - A primary key value from another table residing in the current table

42

Primary Keys and Foreign Keys

- How do we model the possibility that many people may reside at one address?

43

Primary Keys and Foreign Keys

- How do we efficiently model the possibility that people may have several addresses?

44

Accessing a Database

- Database runs as a separate process
  - Commands sent to database
    - Use a language known as SQL (Structured Query Language)
    - Human readable, reasonably intuitive
    - Much like working with QuickDraw
  - Results returned, often as lists
    - Program uses values retrieved from database to compute and present the desired result

45

Databases

- Provide a ready-made solution for dealing with larger amounts of data
  - Careful database design is important
    - Avoid data duplication
    - Queries on large databases may need to be optimized
  - Tools are readily available
    - MySQL is free to download
    - Python libraries available for interacting with many different database packages

46

Wrapping Up - Files

- Files provide longer term storage of data
  - Types
    - Text files
    - Binary files
  - Can be opened for
    - Reading
    - Writing
    - Appending
  - Separate databases are commonly used to manage larger amounts of data

47

Wrapping Up - Exceptions

- Exceptions
  - Many runtime errors are exceptions
  - Default behaviour: crash program
  - Exceptions can be caught
    - Put code that might cause an exception in a try block
    - Use except blocks to catch exceptions that can be recovered from

48

Where Are We Going?

- Now you have a large set of tools:
  - – Input, output, variables
  - – If statements
  - – For loops and while loops
  - – Functions
  - – Lists, dictionaries and strings
  - – Files and exceptions
- These tools are sufficient to solve many interesting problems

49