

Topic 8: Files and Exceptions

“A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.”

-Douglas Adams

Textbook

- Strongly Recommended Exercises
 - The Python Workbook: 141, 142, 143 and 148
- Recommended Exercises
 - The Python Workbook: 150, 154, 155, 160 and 162
- Recommended Readings
 - Starting Out with Python
 - Chapter 7 (2nd Ed.) / Chapter 6 (3rd Ed. and 4th Ed.)

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: Most 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
 - ...

Opening a Text File

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

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")
```

Reading from a File

- Once a file has been opened, we can invoke methods on the file handle 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

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

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

Example: Computing a Sum

Special Characters

- Inside of text files we have characters
 - letters
 - numbers
 - punctuation marks
- Also have characters that control spacing
 - newline
 - tab
 - other control characters

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

Escape Sequences

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

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`

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

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`

Command Line Parameters

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

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

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 `input()` function is equivalent to `sys.stdin.readline().rstrip()`

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

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

Example: Count Words

- Key Idea:
 - Since standard output is a file, the same statements can be used to write output to the screen or a file
 - Since standard input is a file, the same statements can be used to read input from the keyboard or a file
 - Once the files are set up, the rest of the program is identical for files / screen / keyboard

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 (especially 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

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 = float(input("Enter a number: "))
b = float(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?

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

Exceptions

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

Exceptions

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

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

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

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 can our program throw an exception if it realizes something is wrong?

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

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

Example

- A table to store data about people

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?

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

Primary Keys and Foreign Keys

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

Primary Keys and Foreign Keys

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

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
 - Results returned, often as lists
 - Program uses values retrieved from database to compute and present the desired result

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

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

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

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