# Chapter 13
# Files and Exception Handling

## Motivations

Data stored in the program are temporary; they are lost when the program terminates. To permanently store the data created in a program, you need to save them in a file on a disk or other permanent storage. The file can be transported and can be read later by other programs. There are two types of files: text and binary. Text files are essentially strings on disk. This chapter introduces how to read/write data from/to a text file.

When a program runs into a runtime error, the program terminates abnormally. How can you handle the runtime error so that the program can continue to run or terminate gracefully? This is the subject we will introduce in this chapter.

# Objectives

- To open a file, read/write data from/to a file (§13.2)
- To use file dialogs for opening and saving data (§13.3).
- To develop applications with files (§13.4)
- To read data from a Web resource (§13.5).
- To handle exceptions using the **try**/**except**/**finally** clauses (§13.6)
- To raise exceptions using the raise statements (§13.7)
- To become familiar with Python's built-in exception classes (§13.8)
- To access exception object in the handler (§13.8)
- To define custom exception classes (§13.9)
- To perform binary IO using the pickle module (§13.10)

# Open a File

How do you write data to a file and read the data back from a file? You need to create a file object that is associated with a physical file. This is called *opening a file*. The syntax for opening a file is as follows:

```
file = open(filename, mode)
```

| Mode | Description |
|---|---|
| 'r' | Open a file for reading only. |
| 'w' | Open a file for writing only. |
| 'a' | Open a file for appending data. Data are written to the end of the file. |
| 'rb' | Open a file for reading binary data. |
| 'wb' | Open a file for writing binary data. |

# Write to a File

```
outfile = open("test.txt", "w")
outfile.write("Welcome to Python")
```

| file | |
|---|---|
| read([number: int]): str | Returns the specified number of characters from the file. If the argument is omitted, the entire remaining contents are read. |
| readline(): str | Returns the next line of file as a string. |
| readlines(): list | Returns a list of the remaining lines in the file. |
| write(s: str): None | Writes the string to the file. |
| close(): None | Closes the file. |

WriteDemo

# Testing File Existence

```
import os.path
if os.path.isfile("Presidents.txt"):
    print("Presidents.txt exists")
```

## Read from a File

After a file is opened for reading data, you can use the read method to read a specified number of characters or all characters, the readline() method to read the next line, and the readlines() method to read all lines into a list.

ReadDemo

## Append Data to a File

You can use the 'a' mode to open a file for appending data to an existing file.

AppendDemo

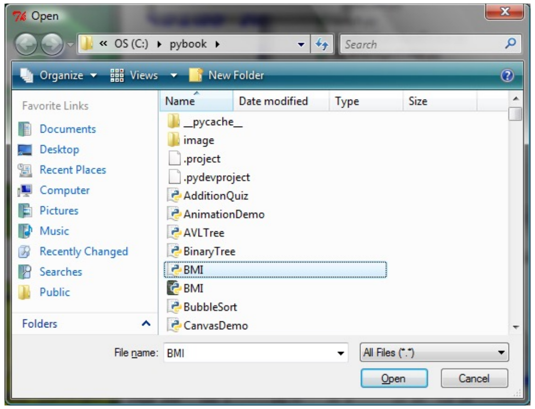
## Writing/Reading Numeric Data

To write numbers, convert them into strings, and then use the write method to write them to a file. In order to read the numbers back correctly, you should separate the numbers with a whitespace character such as ' ', '\n'.

WriteReadNumbers

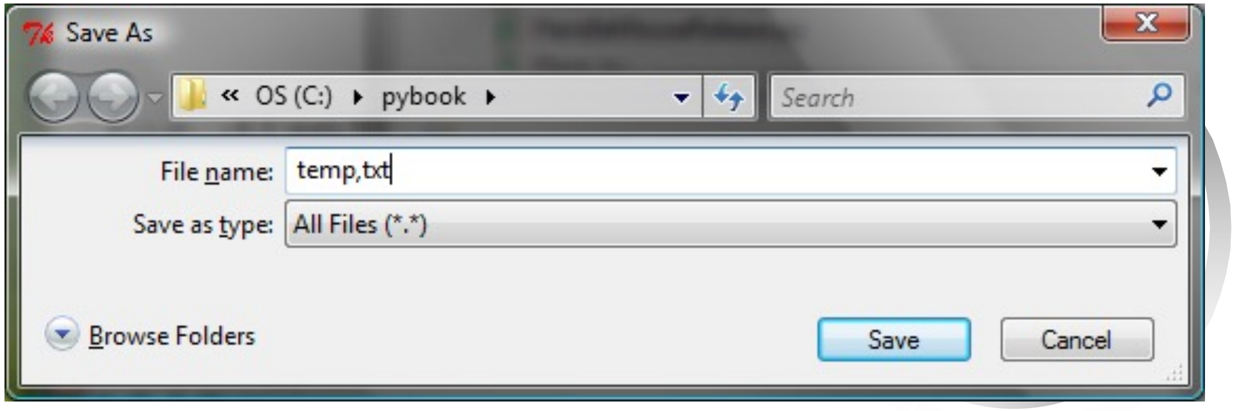
# File Dialogs

```
from tkinter.filedialog import askopenfilename
from tkinter.filedialog import asksaveasfilename

filenameforReading = askopenfilename()
print("You can read from from " + filenameforReading)

filenameforWriting = asksaveasfilename()
print("You can write data to " + filenameforWriting)
```



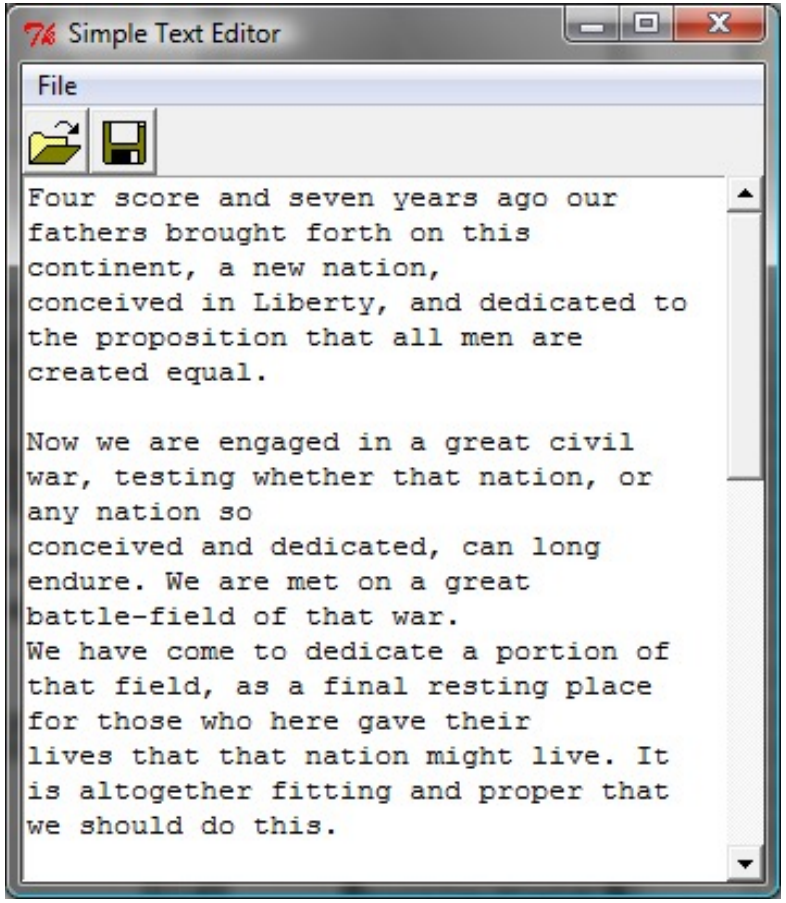
__pycache__
image
.project
.pydevproject
AdditionQuiz
AnimationDemo
AVLTree
BinaryTree
BMI
BMI
BubbleSort
CanvasDemo

File name: BMI
All Files (*.*)

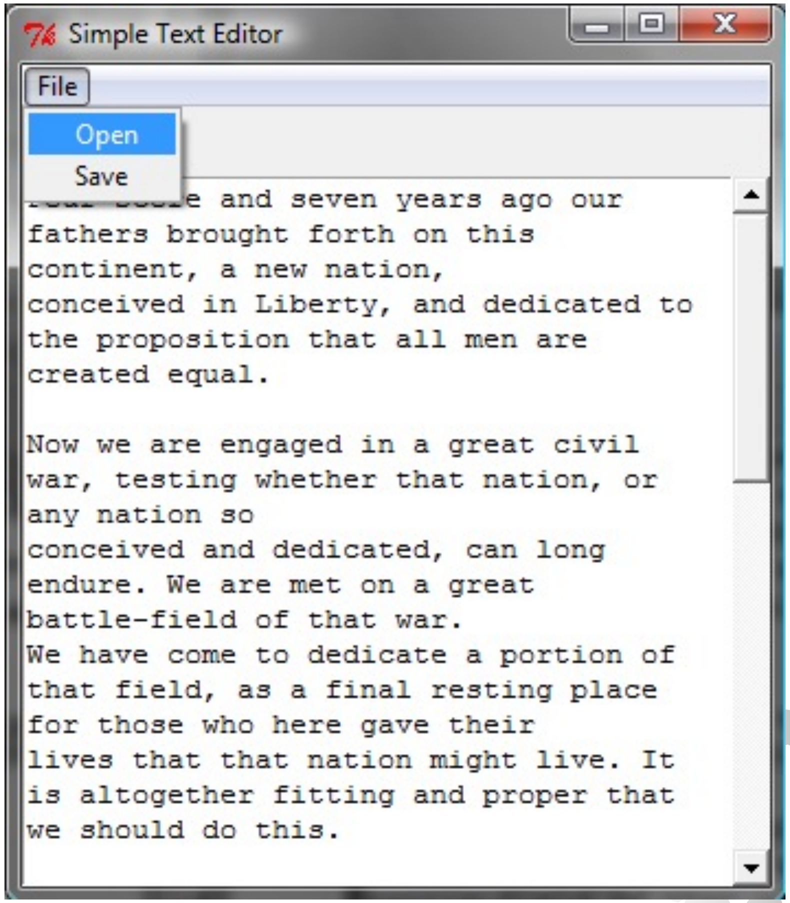


File name: temp,txt
Save as type: All Files (*.*)

# File Editor



```
Four score and seven years ago our
fathers brought forth on this
continent, a new nation,
conceived in Liberty, and dedicated to
the proposition that all men are
created equal.

Now we are engaged in a great civil
war, testing whether that nation, or
any nation so
conceived and dedicated, can long
endure. We are met on a great
battle-field of that war.
We have come to dedicate a portion of
that field, as a final resting place
for those who here gave their
lives that that nation might live. It
is altogether fitting and proper that
we should do this.
```



```
Four score and seven years ago our
fathers brought forth on this
continent, a new nation,
conceived in Liberty, and dedicated to
the proposition that all men are
created equal.

Now we are engaged in a great civil
war, testing whether that nation, or
any nation so
conceived and dedicated, can long
endure. We are met on a great
battle-field of that war.
We have come to dedicate a portion of
that field, as a final resting place
for those who here gave their
lives that that nation might live. It
is altogether fitting and proper that
we should do this.
```

FileEditor

## Problem: Counting Each Letter in a File

The problem is to write a program that prompts the user to enter a file and counts the number of occurrences of each letter in the file regardless of case.

CountEachLetter

## Retrieving Data from the Web

Using Python, you can write simple code to read data from a Website. All you need to do is to open a URL link using the urlopen function as follows:

```
infile = urllib.request.urlopen('http://www.yahoo.com')
```

```
import urllib.request
infile = urllib.request.urlopen('http://www.yahoo.com/index.html')
print(infile.read().decode())
```

CountEachLetterURL

# Exception Handling

When you run the program in Listing 11.3 or Listing 11.4, what happens if the user enters a file or an URL that does not exist? The program would be aborted and raises an error. For example, if you run Listing 11.3 with an incorrect input, the program reports an IO error as shown below:

```
c:\pybook\python CountEachLetter.py
Enter a filename: newinput.txt
Traceback (most recent call last):
  File "CountEachLetter.py", line 23, in <module>
main()
  File "CountEachLetter.py", line 4, in main
Infile = open(filename, "r"> # Open the file
IOError: [Errno 2] No such file or directory: 'newinput.txt'
```

## The `try ... except` Clause

```
try:
    <body>
except <ExceptionType>:
    <handler>
```

## The `try ... except` Clause

```
try:
<body>
except <ExceptionType1>:
    <handler1>
...
except <ExceptionTypeN>:
    <handlerN>
except:
    <handlerExcept>
else:
    <process_else>
finally:
    <process_finally>
```

TestException

# Raising Exceptions

You learned how to write the code to handle exceptions in the preceding section. Where does an exception come from? How is an exception created? Exceptions are objects and objects are created from classes. An exception is raised from a function. When a function detects an error, it can create an object of an appropriate exception class and raise the object, using the following syntax:

```
raise ExceptionClass("Something is wrong")
```

RaiseException

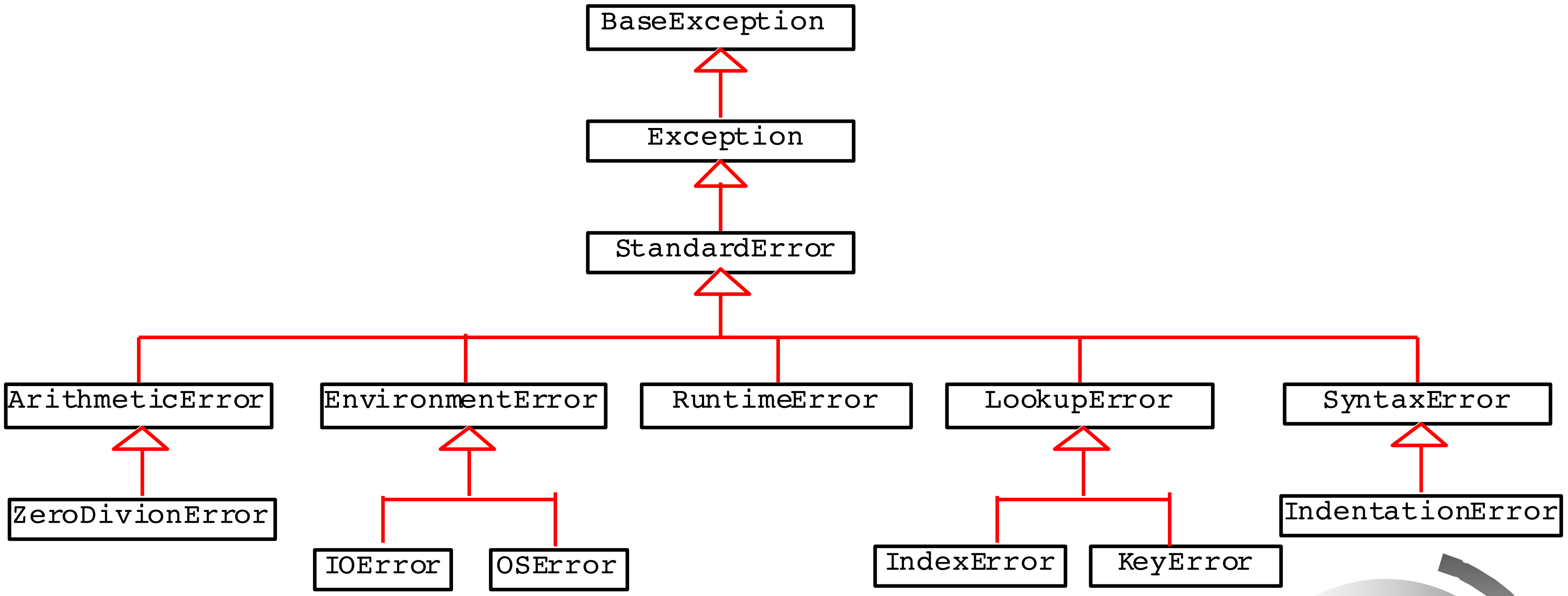
# Processing Exceptions Using Exception Objects

You can access the exception object in the except clause.

```
try
    <body>
except ExceptionType as ex:
    <handler>
```

ProcessExceptionObject

# Defining Custom Exception Classes



TestCircleWithCustomException

InvalidRadiusException

# Binary IO Using Pickling

To perform binary IO using pickling, open a file using the mode 'rb' or 'wb' for reading binary or writing binary and invoke pickle module’s dump and load functions to write and read data.

BinaryIODemo

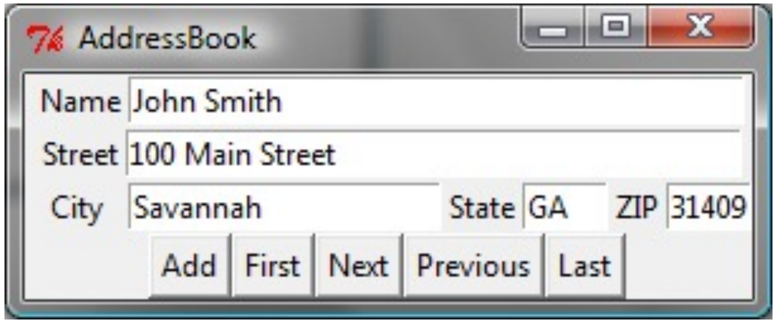
## Detecting End of File

If you don't know how many objects are in the file, how do you read all the objects? You can repeatedly read an object using the load function until it throws an EOFError exception. When this exception is raised, catch it and process it to end the file reading process.

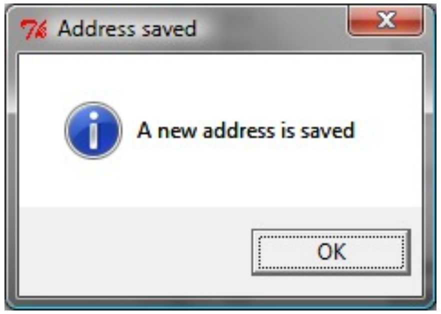
DetectEndOfFile

# Case Study: Address Book

Now let us use object IO to create a useful project for storing and viewing an address book. The user interface of the program is shown below. The *Add* button stores a new address at the end of the file. The *First*, *Next*, *Previous*, and *Last* buttons retrieve the first, next, previous, and last addresses from the file, respectively.



Name John Smith
Street 100 Main Street
City Savannah State GA ZIP 31409



A new address is saved

AddressBook