Chapter 8

How to handle exceptions

Objectives

Applied

1. Add the proper level of exception handling to your programs.
2. Use raise statements to test the exception handling in your programs.

Knowledge

1. Describe the types of exceptions that need to be handled by a program.
2. Describe the operation of a try statement with one or more except clauses.
3. Describe the use of an exception object in your exception handling routines.
4. Describe the use of the exit() function in the sys module.

Objectives (cont.)

5. Describe the use of a finally clause in a try statement.
6. Explain why you may need to raise exceptions when testing your exception handling routines.

Code that can cause a ValueError exception

```
number = int(input("Enter an integer: "))
print("You entered a valid integer of " + str(number) + ".")
print("Thanks!")
```

The console for a valid integer

```
Enter an integer: 5
You entered a valid integer of 5.
Thanks!
```

The console for an invalid integer

```
Enter an integer: five
Traceback (most recent call last):
  File "C:\murach\python\book_figs\ch07\fig1.py", line
1, in <module>
    number = int(input("Enter a valid integer: "))
ValueError: invalid literal for int() with base 10:
'five'
```

Two functions that can cause a ValueError exception

Function	Reason for exception
`int(data)`	Can't convert the data argument to an int value.
`float(data)`	Can't convert the data argument to a float value.

The syntax for a try statement that catches an exception

```
try:
    statements
except [ExceptionName]:
    statements
```

How to handle a ValueError exception

```
try:
    number = int(input("Enter an integer: "))
    print("You entered a valid integer of " + str(number) + ".")
except ValueError:
    print("You entered an invalid integer. Please try again.")
print("Thanks!")
```

The console for a valid integer

```
Enter an integer: 5
You entered a valid integer of 5.
Thanks!
```

The console for an invalid integer

```
Enter an integer: five
You entered an invalid integer. Please try again.
Thanks!
```

The console for a valid integer

```
Enter an integer: 5
You entered a valid integer of 5.
Thanks!
```

The console for an invalid integer

```
Enter an integer: five
You entered an invalid integer. Please try again.
Thanks!
```

The user interface for the Total Calculator

```
try:
    number = int(input("Enter an integer: "))
    print("You entered a valid integer of " + str(number) + ".")
except:
    print("You entered an invalid integer. Please try again.")
print("Thanks!")
```

The user interface for the Total Calculator (cont.)

```
The Total Calculator program

Enter price: ten
Invalid decimal number. Please try again.
Enter price: 9.99
Enter quantity: 2.5
Invalid integer. Please try again.
Enter quantity: 3

PRICE:    9.99
QUANTITY: 3
TOTAL:    29.97
```

The code

```
def get_price():
    while True:
        try:
            price = float(input("Enter price: "))
            return price
        except ValueError:
            print("Invalid decimal number. Please try again.")

def get_quantity():
    while True:
        try:
            quantity = int(input("Enter quantity: "))
            return quantity
        except ValueError:
            print("Invalid integer. Please try again.")
```

The code (cont.)

```
def main():
    print("The Total Calculator program\n")

    # get the price and quantity
    price = get_price()
    quantity = get_quantity()

    # calculate the total
    total = price * quantity

    # display the results
    print()
    print("PRICE:    ", price)
    print("QUANTITY: ", quantity)
    print("TOTAL:    ", total)

if __name__ == "__main__":
    main()
```

The hierarchy for five common exceptions

```
Exception
    OSError
        FileExistsError
        FileNotFoundError
    ValueError
```

The syntax for a try statement with multiple except blocks

```
try:
    statements
except ExceptionName:
    statements
[except ExceptionName:
    statements] ...
```

Code that handles multiple exceptions

```
filename = input("Enter filename: ")
movies = []
try:
    with open(filename) as file:
        for line in file:
            line = line.replace("\n", "")
            movies.append(line)
except FileNotFoundError:
    print("Could not find the file named " + filename)
except OSError:
    print("File found - error reading file")
except Exception:
    print("An unexpected error occurred")
```

The console when a FileNotFoundError occurs

```
Could not find the file named films.txt
```

The console when an OSError occurs

```
File found - error reading file
```

The console when any other Exception occurs

```
An unexpected error occurred.
```

The built-in type() function

```
type(object)
```

The exit() function of the sys module

```
exit()
```

The complete syntax for the except clause

```
except [ExceptionName] [as name]:
    statements
```

Code that handles multiple exceptions

```
import sys

filename = input("Enter filename: ")
movies = []
try:
    with open(filename) as file:
        for line in file:
            line = line.replace("\n", "")
            movies.append(line)
except FileNotFoundError as e:
    print("FileNotFoundError:", e)
    sys.exit()
except OSError as e:
    print("OSError:", e)
    sys.exit()
except Exception as e:
    print(type(e), e)
    sys.exit()
```

The console when a FileNotFoundError occurs

```
FileNotFoundError: [Errno 2] No such file or directory:
'films'
```

The console when an OSError occurs

```
OSError: [Errno 13] Permission denied: 'movies.csv'
```

The user interface for the Movie List 2.0 program with exception handling

```
COMMAND MENU
list - List all movies
add -  Add a movie
del -  Delete a movie
exit - Exit program

Command: list
1. Monty Python and the Holy Grail (1975)
2. Cat on a Hot Tin Roof (1958)
3. On the Waterfront (1954)
4. Gone with the Wind (1939)
5. Wizard of Oz (1939)

Command: del
Number: x
Invalid integer. Please try again.
Number: 6
There is no movie with that number. Please try again.
Number: 4
Gone with the Wind was deleted.
```

A console that handles a file I/O exception

```
COMMAND MENU
list - List all movies
add -  Add a movie
del -  Delete a movie
exit - Exit program

Could not find movies.csv file.
Terminating program.
```

The code

```
import csv
import sys

FILENAME = "movies.csv"

def read_movies():
    try:
        movies = []
        with open(FILENAME, newline="") as file:
            reader = csv.reader(file)
            for row in reader:
                movies.append(row)
        return movies
    except FileNotFoundError:
        print("Could not find " + FILENAME + " file.")
        exit_program()
    except Exception as e:
        print(type(e), e)
        exit_program()
```

The code (cont.)

```
def write_movies(movies):
    try:
        with open(FILENAME, "w", newline="") as file:
            writer = csv.writer(file)
            writer.writerows(movies)
    except Exception as e:
        print(type(e), e)
        exit_program()

def exit_program():
    print("Terminating program.")
    sys.exit()
```

The code (cont.)

```
def delete_movie(movies):
    while True:
        try:
            number = int(input("Number: "))
        except ValueError:
            print("Invalid integer. Please try again.")
            continue

        if number < 1 or number > len(movies):
            print("There is no movie with that number. " +
                  "Please try again.")
        else:
            break

    movie = movies.pop(number - 1)
    write_movies(movies)
    print(movie[0] + " was deleted.\n")
```

The code (cont.)

```
def main():
    display_menu()
    movies = read_movies()
    while True:
        command = input("Command: ")
        if command == "list":
            list_movies(movies)
        elif command == "add":
            add_movie(movies)
        elif command == "del":
            delete_movie(movies)
        elif command == "exit":
            break
        else:
            print("Not a valid command. Please try again.\n")
    print("Bye!")
```

The complete syntax for a try statement

```
try:
    statements
 except [ExceptionName] [as name]:
    statements
[except [ExceptionName] [as name]:
    statements] ...
[finally:
    statements]
```

A function that uses a with statement to clean up resources

```
def read_movies(filename):
    try:
        with open(filename, newline="") as file:
            movies = []
            reader = csv.reader(file)
            for row in reader:
                movies.append(row)
        return movies
    except Exception as e:
        print(e)
```

A function that uses a finally clause to clean up resources

```
def read_movies(filename):
    try:
        file = open(filename, newline="")
        try:
            movies = []
            reader = csv.reader(file)
            for row in reader:
                movies.append(row)
            return movies
        except Exception as e:
            print(type(e), e)
        finally:
            file.close()
    except FileNotFoundError as e:
        print(e)
```

The syntax for the raise statement

```
raise ExceptionName("Error message")
```

Raising a ValueError exception

```
raise ValueError("Invalid value")
```

Raising an exception for testing an exception handler

```
def get_movies(filename):
    try:
        with open(filename, newline="") as file:
            raise OSError("OSError")     # for testing
            movies = []
            reader = csv.reader(file)
            for row in reader:
                movies.append(row)
        return movies
    except Exception as e:
        print(type(e), e)
```

Raising an exception that should be handled by the calling function

```
def get_movies(filename):
    if len(filename) == 0:
        raise ValueError("The filename argument is required.")
    with open(filename, newline="") as file:
        movies = []
        reader = csv.reader(file)
        for row in reader:
            movies.append(row)
    return movies
```

Logging an exception and raising it for the calling function

```
def get_movies(filename):
    try:
        with open(filename, newline="") as file:
            movies = []
            reader = csv.reader(file)
            for row in reader:
                movies.append(row)
        return movies
    except Exception as e:
        log_exception(e)
        raise e
```