Input Validation Exceptions

Target Course

CS1 (imperative)

Learning Goals

A student shall be able to:

1. Apply principles of secure design and defensive programming techniques when developing software.

IAS Outcomes

The CS2013 Information Assurance and Security outcomes addressed by this module are:

IAS Knowledge Topic	Outcome
Defensive Programming	3. Classify common input validation errors, and write correct input validation code. [Usage] 5. Demonstrate the identification and graceful handling of error conditions. [Usage]
Principles of Secure Design	2. Summarize the principle of fail-safe and deny-by-default. [Familiarity]

Dependencies

- Cover at the same time that exceptions are introduced.
- Material assumes knowledge of selection and iteration.
- Material continues the BMI example from modules 2 and 3.

Summary

Use exceptions to validate input data.

Estimated Time

[Provide the estimated amount of lecture time to cover this module, using the notion of time as defined in CS2013.]

Materials

How can exception-handling help with information assurance and security?

- Exception handling is an additional way to catch input errors and deal with them. When exceptions occur and are not handled they cause the program to display an error message and stop.
- Exception handling allows us to separate error handling code from normal operational code. The main flow of the algorithm can be written separately from error handling code, which often results in more maintainable code. [1]

What are some best practices when using exception handling?

- Detailed internal error messages such as stack traces, database dumps, and error codes should not be displayed in the final software product as these messages can provide clues about the inner workings of the application and could be used to fine tune attacks. These types of messages should only be used when you are debugging a program. In accordance with the fail-safe design principle, exceptions should be caught and sensitive messages should not be displayed. Generic error message should be used instead.
- In the try-except block you have the option of using a finally clause. The finally clause should be used to close all connections to opened resources such as files, databases or to networks. A finally clause should also rollback any processing which is only partially

complete. According to the fail-safe principle the state should remain stable and fail gracefully even after an error.

Example 1: Update BMI program to use exception handling

The existing `getNumber` function introduced in the previous selection module throws a `ValueError` exception if the user's input cannot be converted into a float causing the program to crash. The `getNumber` function is updated below to ask the user to re-enter the number when a `ValueError` is encountered. This is done by placing a try/except block inside a while loop which only breaks when a number has been entered.

```
def getNumber(prompt):
    while True:
        try:
            num = float(input(prompt))
            break
        except ValueError, KeyboardInterrupt:
            print("Please enter a number.")
        except:
            print("Unanticipated Error.")
    return num
```

Example 2: Using finally clause

Problem: Suppose the file quizgrades.txt contains a list of student quiz scores, where each line in the file starts with the student's name followed by their quiz grades. For example:

joe 10 15 20 30 40

bill 23 16 19 22

sue 8 22 17 14 32 17 24 21 2 9 11 17

grace 12 28 21 45 26 10

john 14 32 25 16 89

Write a program that prints out the average quiz grade for each student.

Discussion of Solution:

- The program may encounter an `IOError` if the file is not found or is not readable. This is handled with a `try/except/finally` block where the `finally` block is a used to close the file reference if it was opened.
- The program might encounter a `TypeError` if one of the quiz scores is non-numeric. This is handled by the inner `try/except block` by skipping the line with the non-numeric value and moving on to the next student.

A possible solution is given below.

```
try:
    file = None
    file = open("quizgrades.txt", "r")

    for aline in file:
        items = aline.split()
        try:
            sum = 0
            for i in range(1,len(items)):
                print(items[i])
                sum = int(items[i]) + sum
            avg = sum / (len(items)-1)
            print(items[0], " quiz average is " , avg)
        except TypeError:
           print("Non-numeric quiz score for student ", items[0])
except IOError:
       print("Error: can\'t find file or read data.")
except:
       print("Unanticipated error.")
finally:
    if file != None:
        file.close()
```

Assessment Methods

Write a program that reads the file `Baa Baa Black Sheep.txt`, and displays the following information:

1. The total number of characters in this text file, where every character whether it is a letter, digit, punctuation, or whitespace is counted.
2. The total number of words in this text file. A word cannot contain any punctuation symbols, starts and ends with a letter, and only contains lowercase letters.
3. For each unique word, count the number of times (i.e., frequency) that the word appears in this text file. For this step use a dictionary where the key is a unique word value and the associated value is a frequency count indicating how many times the word was found in the text file.

Here are the contents of the file `Baa Baa Black Sheep.txt`

```
Baa, baa, black sheep,
Have you any wool?
Yes sir, yes sir,
Three bags full.

One for the Master,
One for the Dame,
One for the little boy
Who lives down the lane.

Baa, baa, black sheep,
Have you any wool?
Yes sir, yes sir,
Three bags full.
```

Here are the results that the program should display:

```
Total number of characters:  247
Total number of words:  48
Word       Count
----       -----
one        3
```

```
down       1
have       2
yes        4
any        2
sheep      2
sir        4
little     1
for        3
three      2
black      2
master     1
you        2
wool       2
dame       1
full       2
bags       2
who        1
lives      1
boy        1
lane       1
baa        4
the        4
```

References

[1] The Java Tutorials. Retrieved August, 2015 from https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html.

[2] Towson Security Injection Modules. Retrieved August, 2015 from

http://cis1.towson.edu/~cssecinj/modules/other-modules/build-the-lab/build-the-lab-lab-module-2/.

[3] OWASP- Code review and error handling. Retrieved August, 2015 from https://www.owasp.org/index.php/Codereview-Error-Handling#Failing_Securely.