Input Validation Review

Target Course

CS1 (imperative)

Learning Goals

A student shall be able to:

1. Describe security design principles and identify security issues associated with common threats and attacks.
2. 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	1. Explain why input validation and data sanitization is necessary in the face of adversarial control of the input channel. [Familiarity] 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

- Material assumes knowledge of selection, iteration, exception handling, and (Python) lists.

Summary

Describes an assignment to write input validation and error handling code.

Estimated Time

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

Materials

Use materials from Input Validation modules 1 through 4.

Assessment Methods

Programming Problem

The program code listed below grades multiple choice quizzes as follows:

a) The user is asked how many questions are in the quiz.
b) The user is asked to enter the key of correct answers on a single line. There should be one answer for each question in the quiz, and each answer should be a letter from a-e. For example, a b e b d c a b c d might be the key for a 10-question quiz. The key is stored in a list.
c) The user is asked to enter the answers for the quiz to be graded on a single line. Again there should be one answer for each question, and each answer should be a letter between a-e.
d) After the user has entered all of the answers to be graded, the program prints the number correct and the percent correct.
e) The user is given the choice to grade another quiz. When the user wants to grade another quiz, the algorithm is repeated starting with step c).

```
1   #pre:  User ready to enter a list of strings
2   #post: Returns a list of the words entered by user.
3   def getList(prompt):
4       line = input(prompt)
5       return line.split()
6
7   #pre:  key is a list of correct answers and qAns is a list of quiz responses
8   #post: Returns the number of correct answers in qAns and the percentage correct
9   def gradeQuiz(key, qAns):
10      correct = 0;
11      for i in range(0, len(key)):
12          if qAns[i] == key[i]:
13              correct = correct + 1
14      return (correct, correct/len(key)*100)
15
16  #pre:user ready to grade a quiz
17  #post:user is able to enter key and quiz answers and have the quiz graded
18  def main():
19      key = getList("Enter the key for the quiz:")
20      repeat = "y"
21      while repeat == "y":
22          qAns = getList("Enter " + str(len(key)) + " quiz answers.")
23          (c, p) = gradeQuiz(key, qAns)
24          print("Num. Correct:", c, "Percentage Correct: ", p)
25          repeat = input("Grade another ?( y or n)")
```

Student Tasks:

1. Review the code given below to make sure you follow the logic.
2. Input validation
   a. Make a list of the input errors which may occur in the current program.
3. Exceptions
   a. Identify each line of the program where an exception may occur and briefly describe how each might occur.
4. Update code to handle input errors gracefully. Use exception handling for at least one error.
5. Describe a specific way that fail-safe and deny-by-default design principals were applied in your error handling code.
6. Reflection
   a. How do input channels pose risks to privacy and the security of applications?
   b. How does input validation and proper handling of errors help?

Programming Problem Solution

- Task 2, List of input validations
  - Check that the key contains only letters between a and e
  - Check that the quiz answers are all letters between a and e
  - Check that the key and quiz answers are of the same length and that the length is not 0
  - Check that the response to Grade another is y or n
- Task 3, List of exceptions
  - IndexError can occur on line 12 if there are less quiz answers than key answers
  - ZeroDivisionError can occur on line 14 if the length of key is 0
- Task 4
  - The sample solution below uses exception handling to handle IndexError and input validation to handle the ZeroDivisionError.

- Task 5
  - o In accordance to deny-by-default, the program ignores key and quiz answer values outside the range a-e. In each case the user is asked to re-enter a valid input. In accordance to the fail-safe principal the program exits or continues safely even when presented with in-correct inputs.
- Task 6
  - o Malicious users can use input channels to launch attacks against software by constructing inputs which are designed to reveal private information or perform unauthorized commands. Input validation and error handling provides a defense against the common input error types to ensure that the input data is valid before it is used.

```
#pre:  User ready to enter a list of strings
#post: Returns a list of the words entered by user.
def getList(prompt):
    line = input(prompt)
    return line.split()

#pre:  key is a list of the correct answers and qAns is a list of quiz 
responses
#post: Returns the count of correct answers and the percentage correct
#      or None if qAns has too few answers
def gradeQuiz(key, qAns):
    try:
        correct = 0
        percentage = 0
        for i in range(0, len(key)):
            if qAns[i] == key[i]:
                correct = correct + 1
        percentage = correct/len(key)*100
    except IndexError:
        print("Too few quiz answers supplied")
        correct = None
        percentage = None
    finally:
        return (correct, percentage)

#pre:vals and validRange are lists
#post:Returns True if each entry of vals is an entry of validRange
def valuesInRange(vals, valueRange):
    for i in range(0, len(vals)):
        if vals[i] not in valueRange:
            return False
    return True


#pre:user ready to grade a quiz
#post:user is able to enter key and quiz answers and have the quiz graded
def main():
    validRange = ["a", "b", "c", "d", "e"]
    key = getList("Enter the key for the quiz: ")
    while not valuesInRange(key, validRange) or len(key)<=0:
        print("Key must have values from:", validRange)
        key = getList("Enter the key for the quiz: ")

    repeat = "y"
    while repeat == "y":
        qAns = getList("Enter " + str(len(key)) + " quiz answers ")
        while not valuesInRange(qAns, validRange):
            print("Please supply", len(key) , "values from:", validRange)
            qAns = getList("Enter " + str(len(key)) + " quiz answers ")

        (correct, percentage) = gradeQuiz(key, qAns)
        if correct!= None:
            print("Number Correct:", correct, " Percentage Correct:", \
                        percentage)

        repeat = input("Grade another ?( y or n)")
        while not valuesInRange(repeat, ["y", "n"]):
            repeat = input("Enter y to grade another or n to quit: ")
```

References

None.