Input Validation Iteration

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

- Continues the discussion of the BMI example presented in the previous module.
- Cover at the same time that iteration is introduced. Material provides a nice way to motivate while loops.
- Material assumes selection has been covered.
- Optional material assumes knowledge of functions.

Summary

Use conditional logic and iteration 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 (Python imperative)

This material continues the BMI example presented in the previous module. The questions listed below may be used to lead an in-class discussion.

Recalling the solution from the previous module:

User enters in height and weight and the program outputs their BMI category.

Our input validation strategy used the deny-by-default principal: if a user typed in an incorrect height or weight the program terminates.

Why might a user who accidentally enters in an incorrect input find the above approach annoying?

Consider a trustworthy user who types in correct height but incorrect weight (e.g. hits enter before typing anything in). The program will terminate and the user must restart again from the beginning. Consider a health care worker who has to calculate the BMI for a group or team. Every innocent error means restarting the program!

How could we improve the user's experience while still being consistent with deny-by default?

Rather than terminating the program, allow the user to reenter the input (i.e. correct their mistake).

How can we allow the user to correct their input errors?

Using iteration in validation logic allows the code to detect the invalid input while allowing the user an opportunity to correct the input. The general strategy is to ask for each input data inside a loop that terminates only when a valid value is entered.

This provides a nicer experience for the user when compared to simply terminating the program. Note that we are still abiding by the fail safe and deny-by-default design principals.

Updated BMI Solution without functions

```
#pre:  User ready to enter weight and height.
#post: Displays the BMI category corresponding to inputs
def main():
    weight = float(input("Enter your weight in pounds: "))

    while weight<=0 or weight>1000:
        print("Weight must be a number in range (0,1000].")
        weight = float(input("Re-enter your weight in pounds: "))

    height = float(input("Enter your height in inches: "))
    while height<=0 or height>100:
        print("Height must be a number in range (0, 100]")
        height = float(input("Re-enter your height in inches: "))
```

Optional Material (assumes functions have been discussed)

Updated BMI Solution using functions

```
def main():
    weight = getNumber("Enter your weight in pounds: ")
    while not isInputValid(weight, 0, 1000):
        print("Weight must be a number in range (0,1000]")
        weight = getNumber("Re-enter your weight in pounds: ")

    height = getNumber("Enter your height in inches: ")
    if not isInputValid(height, 0, 100):
        print("Height must be a number in range (0, 100]")
        height = getNumber("Re-enter your height in inches: ")

    result = getBMICategory(weight, height)
    print(" Your BMI category is " , result)
```

The other Python functions mentioned in the code above are defined in Input-Validation-2-Selection module.

Assessment Methods

Programming Problems

1. Write a program that asks the user for how many numbers they intend to enter. If the value entered is less than zero, display an appropriate error message. If the value entered is valid, use iteration (i.e., looping) to obtain that many numeric values from the user. These numeric values may be any value. After obtaining the correct number of numeric values from the user, compute and display the average of these numbers.
   For example, the user may indicate that they will enter 5 numbers. The user is prompted to enter each number, resulting in -10, 5.5, 100, -1.1, and 13 being entered. Your program would then display 21.48 as the average.
   Note that students would need to check for a divide-by-zero to avoid an arithmetic error.

   Input Validation 1 rubric can be used to grade students.

References

None.