# **Lab 8: Input Validation**

This lab accompanies Chapter 7 of *Starting Out with Programming Logic & Design*.

Name: David White

**Lab 8.1 – Input Validation**

|  |
| --- |
| Critical Review  If a computer reads bad data as input, it will produce bad data as output. Programs should be designed to reject bad data that is given as input.  Garbage in, garbage out (GIGO), refers to the fact that computers cannot tell the difference between good data and bad date.  Both numbers and strings can be validated. |

The goal of this lab is to identify potential errors with algorithms and programs.

**Step 1:** Imagine a program that calls for the user to enter a password of at least 8 alphanumeric characters. Identify at least two potential input errors.

A special character (i.e !, @, #, etc.) or a password of less than 8 charachters

**Step 2:** Imagine a program that calls for the user to enter patients’ blood pressure. Blood pressure ranges are between 50 and 230. Identify at least two potential input errors.

A letter or special charater or a number greater than 230

**Step 3:** Open either your Lab 5-3 flowchart or your Lab 5-4.py Python code. This program allowed the user to enter in 7 days worth of bottle returns and then calculated the average. Examine the program and identify at least two potential input errors.

When asked to end program someone could enter a number. When asked to enter the number of bottles, someone could enter a letter

**Step 4:** Open either your week 6 homework flowchart or your Python code (Teacher’s Calculator assignment). This program allowed a teacher to enter any number of test scores and then calculated the average score. Examine the program and identify at least two potential input errors.

When asked to enter how many students took the test someone could enter a letter. When asked to enter a score someone could enter a number greater than 100.

**Lab 8.2 – Input Validation and Pseudocode**

|  |
| --- |
| Critical Review  Input validation is commonly done with a loop that iterates as long as an input variable contains bad data. Either a posttest or a pretest loop will work. If you want to also display an error message, use a pretest loop, otherwise, a posttest loop will work.  Functions are often used for complex validation code. |

The goal of this lab is to write input validation pseudocode.

**Step 1:** Examine the following main module from Lab 5.2. Notice that if the user enters a capital ‘Y’ the program will end since the while loop only checks for a lower case ‘y’.

Module main ()

**//Step 1: Declare variables below**

Declare Integer totalBottles = 0

Declare Integer counter = 1

Declare Integer todayBottles = 0

Declare Real totalPayout

Declare String keepGoing = ‘y’

**//Step 3: Loop to run program again**

While keepGoing == ‘y’

**//Step 2: Call functions**

getBottles(totalBottles, todayBottles, counter)

calcPayout(totalBottles, totalPayout)

printInfo(totalBottles, totalPayout)

Display “Do you want to run the program again? (Enter y for yes or n for no).”

Input keepGoing

End While

End Module

**Step 2:** Write a line of code that will convert the input value to a lower case value. (See Validating String Input, Page 340).

toLower(keepGoing)

**Step 3:** Examine the getBottles module from the same program. Notice the potential input error of the user entering a negative value into todayBottles. Rewrite the module with an input validation loop inside the existing while loop that will verify that the entry into todayBottles is greater than 0. If they enter a 0 or negative value, display an error message. (Reference: Input Validation Loop, Page 335).

**Previous Code**

//getBottles module

Module getBottles(Integer totalBottles, Integer todayBottles, Integer counter)

While counter <=7

Display “Enter number of bottles returned for the day:”

Input todayBottles

totalBottles = totalBottles + todayBottles

counter = counter + 1

End While

End Module

**Validation Code**

//getBottles module

Module getBottles(Integer totalBottles, Integer todayBottles, Integer counter)

While counter <=7

Display “Enter number of bottles returned for the day:”

Input todayBottles

While todayBottles <= 0

Display “Please enter a number greater than 0

Display “Enter the number of bottles returned for the day: “

Input todayBottles

totalBottles = totalBottles + todayBottles

counter = counter + 1

End While

End Module

**Step 4:** Examine the following pseudocode from Teacher’s Calculator. Rewrite the module with a validation loop so that no less than 2 students and no more than 30 students take the test.

**Previous Code**

Module getNumber(Integer Ref number)

Display “How many students took the test: ”

Input number

End Module

**Validated Code**

Module getNumber(Integer Ref number)

Display “How many students took the test: ”

Input number

While number < 2 AND number > 30

Display “Please enter a number between 2 and 30”

Display “How many students took the test: ”

Input number

End Module

**Step 5:** Examine the following pseudocode from Teacher’s Calculator. Rewrite the module with a validation loop so that the test score must be between 0 and 100.

**Previous Code**

Module getScores(Real Ref totalScores, Integer number, Real score, Integer counter)

For counter = 1 to number

Display “Enter their score:”

Input score

Set totalScores = totalScores + score

End For

End Module

**Validated Code**

Module getScores(Real Ref totalScores, Integer number, Real score, Integer counter)

For counter = 1 to number

Display “Enter their score:”

Input score

Set totalScores = totalScores + score

While score > 0 AND score < 100

Display “Please enter a number between 1 and 100”

Display “Enter their score:”

Input score

End For

End Module

**Lab 8.3 – Functions and Flowcharts**

|  |
| --- |
| Critical Review  Based on the type of loop used for validation, you may have noticed the concept of a priming read. This is this the first input before the validation loop.  The purpose of this is to get the first input value that will be tested by the validation loop.  A priming read is used with a while loop, rather than a do-while loop.  **Note:**  If the programmer is asking for a particular type of input (either numeric or string), the user is free to enter something else.  This will normally cause a fatal error at some point of program execution.  Avoiding these fatal errors is beyond the scope of basic Raptor programming. What this means is that all errors cannot be resolved using Raptor. |

This lab requires you to modify the flowchart from your week 7 homework (Teacher’s Calculator) to incorporate validation loops. Use an application such Visio, Draw.io (must convert to a PDF file before submitting), or simply draw the flowchart by hand and upload a picture of it in JPG format.



 **Lab 8.4 – Python Code and Input Validation**

Critical Review

Numeric validation loops in Python are done by writing the types of loops you already are familiar with. Code using a prime read might look as follows:

number = int(input('Enter a number between 1 and 10: '))

while number < 1 or number > 10:

print 'Please enter a number between 1 and 10'

number = int(input('Enter a number between 1 and 10: '))

Validation code for string input uses the not keyword to check the opposite of something. Code using a prime read might look as follows:

endProgram = input('Do you want to end program? (Enter no or yes): ')

while not (endProgram == 'yes' or endProgram == 'no'):

print 'Please enter a yes or no'

endProgram = input('Do you want to end program? (Enter no or yes): ')

The goal of this lab is to convert the Teacher’s Calculator program in Lab 8.3 to Python code.

**Step 1:** Start the IDLE Environment for Python. Open your Teacher’s Calculator program and click on File and then Save As. Select your location and save this file as *Lab*8-4*.py*. Be sure to include the .py extension.

**Step 2:** Modify the documentation in the first few lines of your program to include your name, the date, and a brief description of what the program does to include validation.

**Step** **3:** Modify the main function so that the user must enter either a ‘yes’ or ‘no’ value in order for the loop to continue. Use a prime read and a while loop with an error message if a bad number is entered.

**Step 4:** Modify the getNumber function so that the user must enter a number between 2 and 30. Use a prime read and a while loop with an error message if a bad number is entered.

**Step 5:** Modify the getScores function so that the user must enter a number between 0 and 100. Use a prime read and a while loop with an error message if a bad number is entered.

**Step 6:** Execute your program so that all error code works and paste final code below:

#######################################################

# Name: David White

# Class: CIS-1400

# Assignment: Lab 8-4

# File: lab8-4.py

# Purpose: test average calculator with input validation

#######################################################

print('\n\*\*\*David White\*\*\*\n') # Display author's name

def main():

# declare vars

endProgram = "no"

totalScores = int(0)

averageScores = int(0)

score = int(0)

counter = int(1)

# loop to run again

while endProgram == "no":

# reset vars

totalScores = 0

averageScores = 0

counter = 1

score = 0

# call functions

num = getNum()

totalScores = getScores(totalScores, num, score, counter)

averageScores = getAverage(totalScores, num)

printAverage(averageScores)

endProgram = input("Do you want to end the program? (yes/no) ")

# end while

while endProgram != "yes" or endProgram != "no":

print("ERROR: Please enter yes or no")

endProgram = input("Do you want to end the program? (yes/no)")

# end while

# get number of students

def getNum():

num = int(input("How many students took the test: "))

while num > 30 and num < 2:

print("ERROR: Please enter a number between 2 and 30")

num = int(input("How many students took the test: "))

# end while

return num

# get scores

def getScores(totalScores, num, score, counter):

for counter in range(0, num):

score = int(input("Enter their score: "))

while score > 100 and score < 0:

print("ERROR: please enter a number between 0 and 100")

score = int(input("Enter their score:"))

totalScores = int(totalScores + score)

# end for

return totalScores

# calculate avg

def getAverage(totalScores, num):

averageScores = totalScores / num

return averageScores

# show avg

def printAverage(averageScores):

print("The average test score is: " + format(averageScores, '>.2f'))

# end mod

# call main

main()