Metropolitan State University, Saint Paul, Minnesota

ICS 140 Computational Thinking with Programming

Lab 6

# Common While Loop Patterns

While loops can be used to execute a block of code as long as a specified condition is met. Some common uses of while loops include input validation, running totals and sentinel values.

## Input Validation Loops

The following code is an example of an input validation loop.

**Text

Description automatically generated**

The score value is input on line 1 giving a base qualification for the while loop. If the input is valid (meaning it is within 0 to 100, the while condition is false and the indented code never runs. If however, the score is outside that range, the user will be prompted until they provide data within the appropriate range.

Let’s do some practice coding with input validation loops.

**Example:** Write a program that checks for a “y” or “n” input and has an input validation loop that rejects all other answers.

user\_answer = input(“Are you sure you want to delete this file? y/n”)  
**#What would go here???**

print(“The input you entered was invalid.”)  
 user\_answer = input(“Are you sure you want to delete this file? y/n”)  
print(“The program would start running now and could process the validated input….”)

# more stuff would happen here in a real program.

**Answer:** while user\_answer != “y” and user\_answer != “n”:

Notice that a while loop in this circumstance uses the and statement to cover both negative use cases. You may find it helpful to read the while statement out loud or consider potential good and bad values and how they would be treated by the Boolean expression.

**Question 1:** Write a program that checks for a value between 1 and 6. The program will prompt for the result of a dice roll. If an invalid number is given, then the program will give an error and try again.

dice\_role = int(input(“Enter the result of your dice roll: “))

**# What would go here???**

print(“You have given an invalid number”)

dice\_role = int(input(“Enter the result of your dice roll: “))

print(“You have rolled a”, dice\_role)

**Answer: while dice\_role < 1 and dice\_role > 6:**

**Question 2:** Write a program that checks for a value of “red”, “blue” or “yellow”. The program will prompt for a primary color. If an invalid color is given, the program will give an error and ask again.

primary\_color = input(“Enter a primary color: “)

**# What would go here???**

print(“You have given an invalid color.”)

primary\_color = input(“Enter a primary color: “)

print(“Wow! My favorite color is”, primary\_color)  
  
**Answer:** while primary\_color != ‘red’ or primary\_color != ‘blue’ or primary\_color != ‘yellow’:

**Question 3:** Write a program that asks the user to choose between coke and pepsi. The program will prompt for these values and keep looping until it gets one of the answers it is expecting.

beverage = input(“Would you like a coke or pepsi: “)

while beverage != “coke” and beverage != “pepsi”:

print(“Please enter either ‘coke’ or ‘pepsi’.”)

**#What would go here???**

print(“I like”, beverage , “too!”)

**Answer:** beverage = input(“Would you like a coke or pepsi: “)

**Question 4:** Write a program that asks the user to enter an even number. The program will give an error if an odd number is given and ask again for an even number.

even\_number = int(input(“Enter an even number: “)

while even\_number % 2 == 1:

print(even\_number, “is odd!”)

**#What would go here???**

print(even\_number, “is divisible by 2.”)

**Answer:** even\_number = int(input(“Enter an even number: “)

**Question 5:** Now that we have solved these problems by creating the appropriate while statement and the appropriate input statement, let’s put it together to create an input validation loop for the following simple program that prompts for a coin flip.

coin\_flip = input(“Enter the results of a coin flip. (heads/tails): “)

**# write the input validation loop for this program.**

**Answer:**

**while coin\_flip != ‘heads’ or coin\_flip != ‘tails’:**

**Print(‘Enter either heads or tails.’)**

coin\_flip = input(“Enter the results of a coin flip. (heads/tails): “)

print(‘Your coin flip is’, coin\_flip)

## Running Totals and Sentinels

Here is a simple example of code that keeps a running total.  
Text

Description automatically generated

On line one, we create an accumulator “total” that is set to a default value of 0. On line 2, we have a while loop that will continue to run as long as the total is less than 100. On line 3, the first statement within the loop gets a purchase amount from the user. On line 4, the purchase amount is added to the total. If the total is over 100, the loop stops, otherwise, it will prompt again and keep adding the amount to the total until it is over 100. Finally, once the total is over 100, line 5 will execute and print the total.

Let’s do some practice coding with running totals.

**Example**: Write a program that asks uses to guess a number and uses an accumulator to track the number of guesses.

secret = 7

count = 0

while guess != secret:  
 guess = int(input(“Guess a number”))

**# What goes here to update the accumulator???**

print(“You guessed correctly on attempt”, count)

**Answer:** count += 1

**Question 6**: A sentinel is a value that can be entered as input that will tell your program to exit the loop. Write a program that allows users to deposit checks into their account.

total\_deposit = 0

check\_amount = int(input(“Enter the check amount (or enter ‘0’ to end): “))

while check\_amount != 0:

**#What would go here to update the accumulator???**

check\_amount = int(input(“Enter the check amount (or enter ‘0’ to end): “))

print(“Your total deposit is”, total\_deposit)

**Answer**: total\_deposit += check\_amount

**Question 7**: Write a calorie tracker that asks the user to enter the calories for the meals they ate in the day and have them enter 0 when they are finished entering meals.

total\_calories = 0

meal\_calories = int(input(“Enter the calories for this meal ( or enter ‘0’ to end): “))

while meal\_calories !=0

total\_calories += meal\_calories

**# What would go here???**

print(“Your total calories is”, total\_calories)

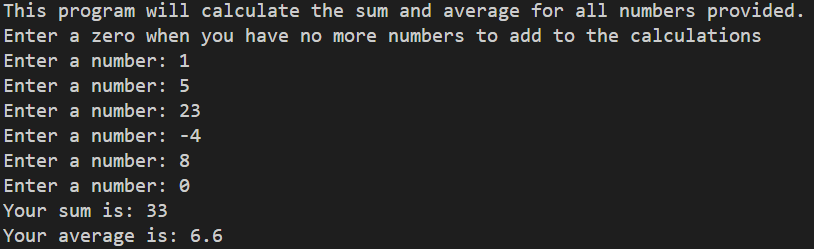
**Answer**: meal\_calories = int(input(“Enter the calories for this meal ( or enter ‘0’ to end): “))

The lab challenge for this week uses a similar concept called a sentinel where a while loop runs until a specific entry is provided. See section 4.5 on page 182 of the text for more information on using sentinels.

**Finding Sums and Averages**

Write a program with a loop that asks the user to enter a series of numbers. The user should enter a zero to signal the end of the series. After all the numbers have been entered, the program should display their sum and average.

When the program is run it should look something like this:



Copy the python code in the section below.

**Python Code**

A screenshot of a computer program

Description automatically generated

Take a screenshot of an example run of the program and paste it below.

**Example Output**

**A computer screen with white text

Description automatically generated**

Run a few more examples with different inputs to make sure it always adds as expected.

**Test Results**

A computer screen shot of a computer screen

Description automatically generated

A computer screen shot of a program

Description automatically generated