CSI 1401 Week 6 Resources

by Allen Yan

This weeks, CSI 1401 will begin chapter 5, which introduces the concept of looping. Looping is extremely useful for programs that need to run for an indeterminate number of repetitions. The class will be spending 2 weeks on chapter 5.

As a reminder, I will be leading a group tutoring session on CSI 1401 every Wednesday from 7:00 pm to 8:00 pm (central time). The session will be conducted online via Microsoft Meetings, where I will be available to provide interactive help to students. If the above time window does not work for you, or if you need additional help, Baylor's Tutoring Center also provides 1-on-1 online tutoring appointments. For more information on signing up for group tutoring or individual tutoring, please visit https://www.baylor.edu/tutoring.

Week 6 Important Topics

Chapter 5 – Looping

- While loops
- While loop exercises

1. While loops

While loops will be the first type of loop you learn. It will repeatedly execute the block of code after the loop expression (called the ***loop body***) as long as the loop expression evaluates to a Boolean True. Each execution of the loop body is called an ***iteration***. The loop expression is evaluated before each iteration of the loop.

Loops should always have a ***reachable end condition*** that causes the loop to terminate when met. There are a number of ways to implement these conditions, and different textbooks may name them differently. The textbook for this course uses the name ***sentinel value***. See the example code from the textbook below from the textbook:

```python
nose = '0'  # Looks a little like a nose
user_value = '-'

while user_value != 'q':
    print(' {} {} '.format(user_value, user_value))  # Print eyes
    print('  {}  '.format(nose))  # Print nose
    print(user_value*5)  # Print mouth
    print('\n')

    # Get new character for eyes and mouth
    user_input = input("Enter a character ('q' for quit): \n")
    user_value = user_input[0]

print('Goodbye.\n')
```

The code above prints out faces composed of the input made by user until the user enters the character ‘q’, which then causes the program to end. In this case, the character ‘q’ will be considered the sentinel value, and the variable `user_value` assessed is called the ***loop variable***.

One important thing to note is that not only does the sentinel value need to be set, the programmer also needs to make sure that the program is able to reach it. Whether it be requesting input from the user every iteration like the code above, or in the case where the loop variable is a counter, update the counter in every iteration of the loop. See below for an example program from the textbook that uses a counter for loop variable.

```
'''Program that calculates savings and interest'''

initial_savings = 10000
interest_rate = 0.05

print('Initial savings of ${}'.format(initial_savings))
print('at {:.0f}% yearly interest.\n'.format(interest_rate*100))

years = int(input('Enter years: '))
print()

savings = initial_savings
i = 1  # Loop variable
while i <= years:  # Loop condition
    print(' Savings at beginning of year {}: ${:.2f}'.format(i,
savings))
    savings = savings + (savings*interest_rate)
    i = i + 1  # Increment loop variable

print('\n')
```

As shown, the variable `i` is used as the loop variable and has the behavior of a counter. The last line of the loop body increments `i` for every iteration of the loop. So, the loop variable is guaranteed to reach the sentinel value, which in this case is the user-made variable `years`, after a finite number of iterations. If the loop expression is never able to evaluate to False, the loop will execute forever and result in an error called ***infinite loop***, which is common but deadly. Therefore, always make sure that the loop can reach its end condition.

2. While loop exercises

The topic of while loop will likely cover the majority of week 6. Since looping may be a completely new concept for some of the students, it could take a while (sorry, no pun intended) to familiarize the looping structure and the types of problems looping can solve. I have included a few exercise programs below (with answers provided) to help you understand looping. Some of the problems may be more concise when solved using for loops, but while loops can do and more. Make sure to try the problems yourself before looking at the answers!

a) Write a Python program that asks for a word from the user and prints it in reverse.

```
word = input("Give me a word: ")
index = len(word) - 1

while index >= 0:
    print(word[index], end="")
    index = index-1
print("\n")
```

b) Write a Python program to print out the following pattern with loops.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

```
n = 5

i = 1
while i <= n:
    j = 0
    while j < i:
        print('*', end=' ')
        j = j+1
    print('')
    i = i+1

i = 4
while i > 0:
    j = 0
    while j < i:
        print('*', end=' ')
        j = j + 1
    print('')
    i = i - 1
```

Useful Resources

- Basic Python Tutorial on GeeksforGeeks:
  https://www.geeksforgeeks.org/python-programming-language/
    - This page provides links to detailed explanations to many entry-level Python concepts along with examples. I encourage taking a look at it if the examples in your textbook were not clear enough.
- Official Python Documentation:
  https://docs.python.org/3/
    - This may be a bit heavy-handed for a beginner-level programmer since the official Python documentation is very thorough and technical. However, learning how to read official documentations is crucial to becoming a good programmer, because the official documentation contains information on everything you can find about Python elsewhere and more. Therefore, I encourage slowly easing yourself into learning how to read the official documentation.