Activity 1.3.8 While Loops

Introduction

A `while` loop is another way to iterate code, as you discovered while using Scratch™ programming language. Unlike `for` loops, which are used to iterate across a collection of a known length, a `while` loop is controlled by a conditional expression that might be less predictable. The conditional expression is evaluated once before an iteration through the `while` block and again before each additional iteration. If the conditional expression is `False` when evaluated (always at the beginning/end of an iteration), execution jumps to the code after the `while` block.

Procedure

1. Form pairs as directed by your teacher. Meet or greet each other to practice professional skills and establish norms.

2. Launch Canopy and open an editor window.

3. Start a new program in the code editor by choosing **File** > **New** > **Python file**. Save the file as `JDoe_BSmith_1_3_8.py`

4. Any Boolean operator, like `==`, `!=`, or `<=`, can be used in the conditional expression for a `while` loop. One useful Boolean expression for validating input is the `in` operator. Validating input means checking that the input is what you expected. Try the following code:

```
x1  def validate():
x2      guess = '1' # initialization with a bad guess
x3      answer = 'hangman word'
x4      while guess not in answer:
x5          guess = raw_input('Name a letter in \''+answer+'\': ')
x6      print('Thank you!')
```

Why is line x2 necessary? Try changing `'1'` to `'a'`. Flip the '1' and 'a' along with 'not in' to 'in'. How does the function change?

5. The following code is poorly annotated. Read and discuss it with your partner. Replace the comments on lines 5-8, 13, 16, 18, and 21 with your own. Then try it and modify your comments if needed.

```
1  from __future__ import print_function
2  import random
3
4  def guess_winner(players=('Amy', 'Bill', 'Cathy', 'Dale')):
5      '''Summarize the function in this docstring.
```

```
    Provide descriptions for the arguments and say what type each one is.
    Describe the type and meaning of the value returned.
    '''
    winner = random.choice(players)

    ####
    # Summarize the following section of code here
    ####
    print('Guess which of these people won the lottery: ',end='')
    for p in players[:len(players)-1]: # explain index here
        print(p+', ', end='')
    print(players[len(players)-1]) # explain this line here

    ####
    # Summarize the following section of code here
    ####
    guesses = 1
    while raw_input() != winner:
        print('Guess again!')
        guesses += 1
    print('You guessed in', guesses, 'guesses!')
    return guesses
```

6. It is often useful to consider how many times a `while` loop will execute. If `guess_game(options)` is called with a list of 100 options, how many times might the user have to guess?

7. Define a function `go_guess()` that implements a number guessing game as described below. Strategize as pair programmers. Then iteratively code and test. The function should work for any range of integers.

```
In []: go_guess()
I have a number between 1 and 20 inclusive.
Guess: 6
6 is too low.
Guess: 10
10 is too high.
Guess: 9
Right! My number is 9! You guessed in 3 guesses!
```

8. If you change `between 1 and 20` from the previous program to `between 1 and 6000`, how many guesses will you need to guarantee that you have the right answer? Explain.

Conclusion

1. Describe the difference between a `while` loop and a `for` loop.

2. Reflect on how effectively you and your partner worked together. Individually, describe the strengths of your process and style, and describe what could be improved and how.