CSI 102: Lab 03

Loops

Anthony Cavallo

09.23.2025

I certify that this lab report is entirely my own work.

Introduction

This is the third lab of CSI 102, where we moved forward from conditionals into using loops in Python. Loops are an essential tool for repeating actions and checking conditions multiple times without rewriting code. This lab exercise was created to strengthen the ability to use while loops, if/elif/else statements in combination with loops, and iterative structures for controlling program flow. The primary goals were to (1) create a program to maintain the top three high scores for an arcade game and (2) generate a diamond pattern using loops and conditionals.

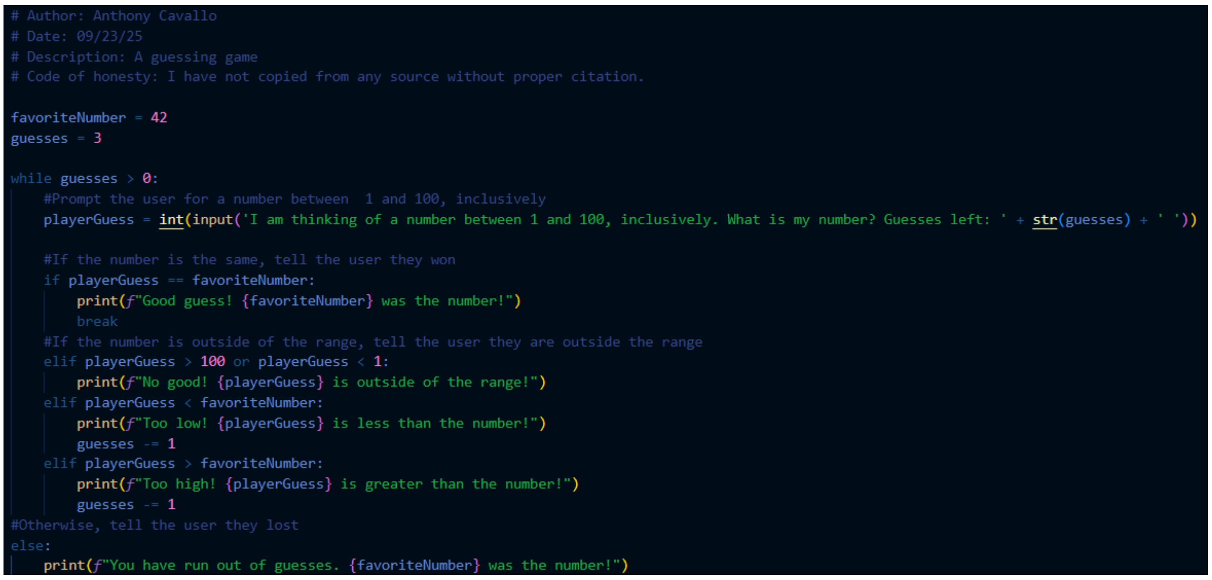
Methods:

Task 1: Guessing Game (Redux)

The first program required an enhancement to the old Guessing Game where loops are used to further boost the gameplay.

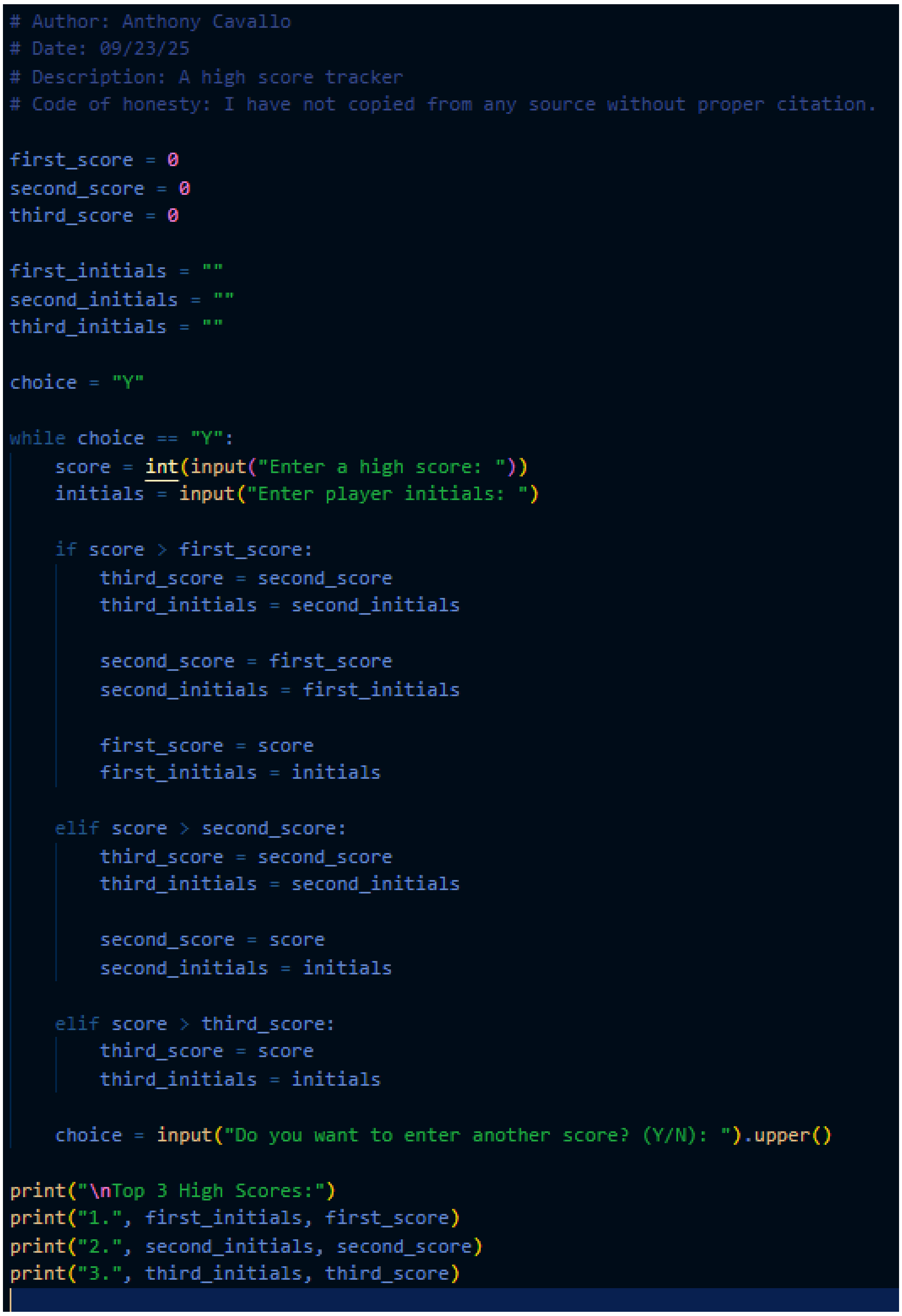
Steps included:

1. Downloading the file.
2. Making the necessary adjustments (while loops, extra conditions)
3. Changing the inputs to reflect the new changes.



```python
# Author: Anthony Cavallo
# Date: 09/23/25
# Description: A guessing game
# Code of honesty: I have not copied from any source without proper citation.

favoriteNumber = 42
guesses = 3

while guesses > 0:
    #Prompt the user for a number between  1 and 100, inclusively
    playerGuess = int(input('I am thinking of a number between 1 and 100, inclusively. What is my number? Guesses left: ' + str(guesses) + ' '))

    #If the number is the same, tell the user they won
    if playerGuess == favoriteNumber:
        print(f"Good guess! {favoriteNumber} was the number!")
        break
    #If the number is outside of the range, tell the user they are outside the range
    elif playerGuess > 100 or playerGuess < 1:
        print(f"No good! {playerGuess} is outside of the range!")
    elif playerGuess < favoriteNumber:
        print(f"Too low! {playerGuess} is less than the number!")
        guesses -= 1
    elif playerGuess > favoriteNumber:
        print(f"Too high! {playerGuess} is greater than the number!")
        guesses -= 1
#Otherwise, tell the user they lost
else:
    print(f"You have run out of guesses. {favoriteNumber} was the number!")
```

Task 2: High Scores



```
# Author: Anthony Cavallo
# Date: 09/23/25
# Description: A high score tracker
# Code of honesty: I have not copied from any source without proper citation.

first_score = 0
second_score = 0
third_score = 0

first_initials = ""
second_initials = ""
third_initials = ""

choice = "Y"

while choice == "Y":
    score = int(input("Enter a high score: "))
    initials = input("Enter player initials: ")

    if score > first_score:
        third_score = second_score
        third_initials = second_initials

        second_score = first_score
        second_initials = first_initials

        first_score = score
        first_initials = initials

    elif score > second_score:
        third_score = second_score
        third_initials = second_initials

        second_score = score
        second_initials = initials

    elif score > third_score:
        third_score = score
        third_initials = initials

    choice = input("Do you want to enter another score? (Y/N): ").upper()

print("\nTop 3 High Scores:")
print("1.", first_initials, first_score)
print("2.", second_initials, second_score)
print("3.", third_initials, third_score)
```

The second program required creating and maintaining the top three high scores for the arcade game Root Beer Tapper.

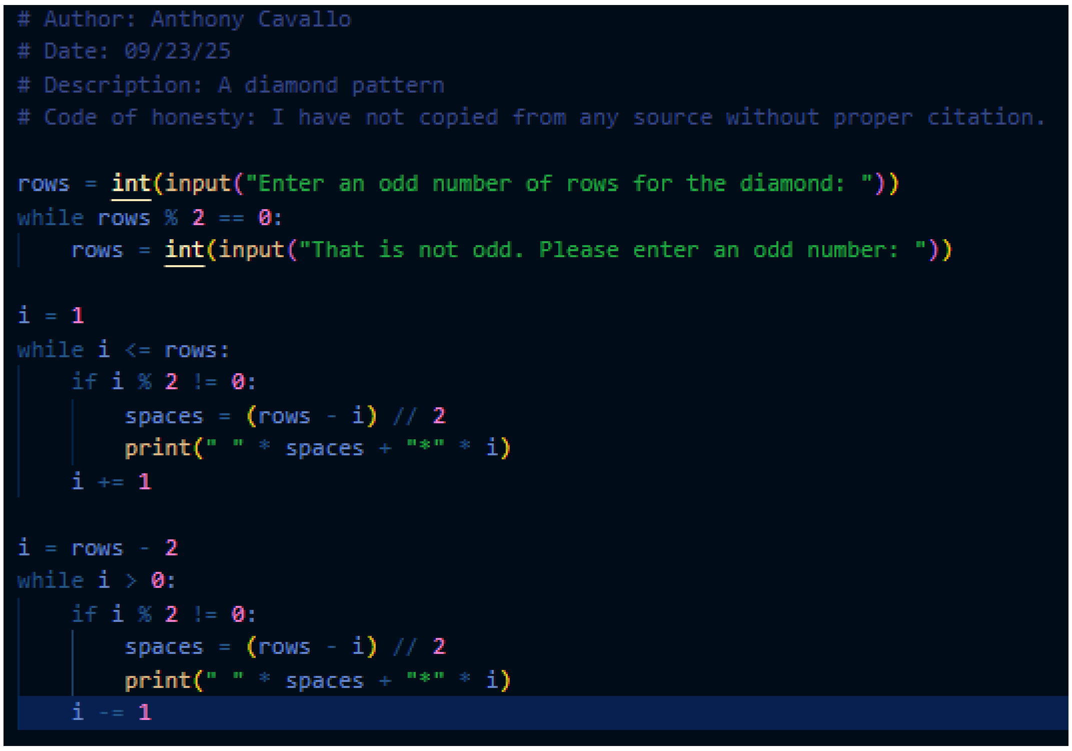
Steps included:

1. Initializing variables for the top three scores and initials.
2. Using a loop to repeatedly ask the user for scores and initials.

3. Using if/elif/else statements to compare the new score against the top three and shift scores down as necessary.
4. Asking the user if they want to continue inputting scores, where typing "Y" continues the loop.

Snippet:
```
if score > first_score:
   third_score = second_score
   second_score = first_score
   first_score = score
```

Task 3: Diamond Pattern

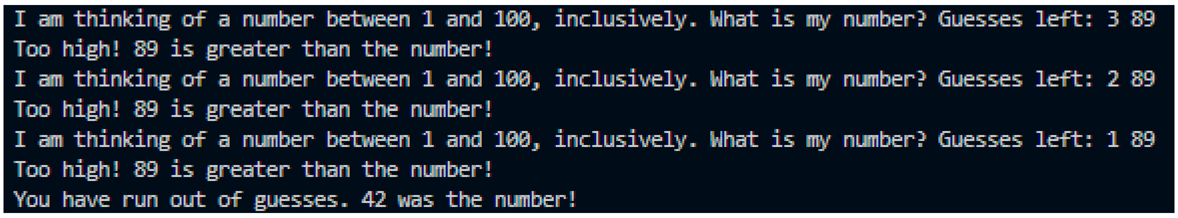


```
# Author: Anthony Cavallo
# Date: 09/23/25
# Description: A diamond pattern
# Code of honesty: I have not copied from any source without proper citation.

rows = int(input("Enter an odd number of rows for the diamond: "))
while rows % 2 == 0:
    rows = int(input("That is not odd. Please enter an odd number: "))

i = 1
while i <= rows:
    if i % 2 != 0:
        spaces = (rows - i) // 2
        print(" " * spaces + "*" * i)
    i += 1

i = rows - 2
while i > 0:
    if i % 2 != 0:
        spaces = (rows - i) // 2
        print(" " * spaces + "*" * i)
    i -= 1
```

The third program required drawing a diamond using only loops.
Steps included:
1. Asking the user to input the number of rows, ensuring that it was an odd number with a loop that re-prompts if an even number is entered.
2. Printing the top half of the diamond by combining spaces and stars, increasing each row until the widest row.
3. Printing the bottom half of the diamond by decreasing the number of stars symmetrically.

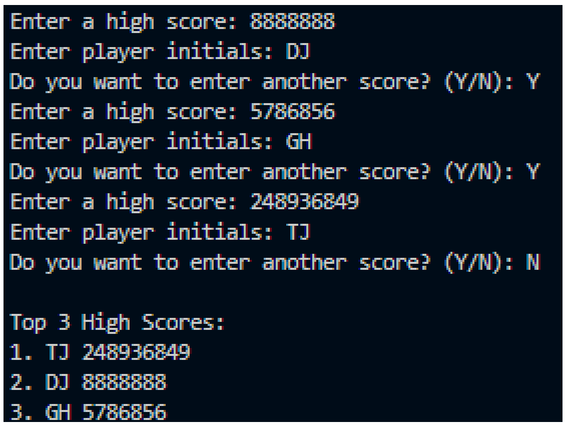
Results:

Task 1:



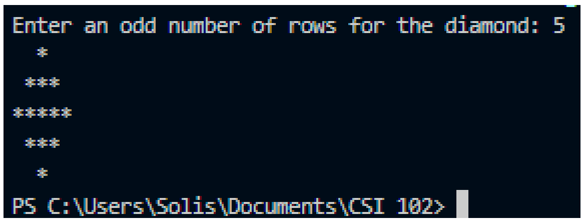
```
I am thinking of a number between 1 and 100, inclusively. What is my number? Guesses left: 3 89
Too high! 89 is greater than the number!
I am thinking of a number between 1 and 100, inclusively. What is my number? Guesses left: 2 89
Too high! 89 is greater than the number!
I am thinking of a number between 1 and 100, inclusively. What is my number? Guesses left: 1 89
Too high! 89 is greater than the number!
You have run out of guesses. 42 was the number!
```

Task 2:



```
Enter a high score: 8888888
Enter player initials: DJ
Do you want to enter another score? (Y/N): Y
Enter a high score: 5786856
Enter player initials: GH
Do you want to enter another score? (Y/N): Y
Enter a high score: 248936849
Enter player initials: TJ
Do you want to enter another score? (Y/N): N

Top 3 High Scores:
1. TJ 248936849
2. DJ 8888888
3. GH 5786856
```

Task 3:



```
Enter an odd number of rows for the diamond: 5
  *
 ***
*****
 ***
  *
PS C:\Users\Solis\Documents\CSI 102>
```

Discussion:

This lab helped me understand how loops and conditionals interact to create flexible programs. The high scores program demonstrated how looping can handle unpredictable amounts of data input, while the diamond program showed how loops and arithmetic can control alignment and shape in text output.

In future projects, I could apply these same concepts to build leaderboards, repeatable input menus, or graphical text-based designs. The use of loops ensures code efficiency, and understanding how to manipulate them with conditions makes it possible to model many real-world processes in programs.

While lists are not allowed in this lab, I could see how using them would simplify shifting scores in the high scores task. That insight will likely be useful later.

Challenges:

One challenge was keeping the shifting logic correct in the high scores task. It was easy to accidentally overwrite a score before moving it down to the next position. I solved this by carefully ordering the assignments (moving third to second, second to first, etc.) before assigning the new score.

Another challenge was formatting the diamond shape. Initially, it was uneven because of missing spaces, but after calculating (rows - i) // 2 spaces, the diamond printed correctly.

Conclusion:

From this lab, I learned how to:

- Use while loops to control program flow.
- Combine conditionals and loops to handle complex input scenarios.
- Apply loops to create text-based patterns.
- Maintain ordered data (like high scores) without advanced data structures.

These skills are foundational for programming and will be directly applicable in future labs and projects. I also now see more clearly how loops reduce redundancy and simplify repeated actions in code.