2A. I created a program using the Python programming language that adds up your total pins knocked down during a round of bowling. The purpose of the program is to add bowling pin totals frame by frame to a list as well as a counter that adds up the total pin fall. The video depicts many of the program's features. The user enters their name and then a menu bar gives the user options if they would like to enter a score, see their pin total frame by frame, or see their updated pin total. The video shows the user enter their score in different frames, and shows what happens if the user bowls a spare or strike. Frame ten also has special features to allow a third and fourth roll, similar to actual bowling. At the end of the program, typing "exit game" ends the program.

2B. I independently developed this program over the course of eight programs that built off of one another. I started with an original, basic code, and expanded the code with new functions and elements to improve its complexion. I brainstormed different features that people use at bowling alleys, and I identified that I wanted a total frame counter, a frame by frame list, and features for spares and strikes. The largest problem I encountered while writing this program revolved around the counter that adds up the total pins frame by frame. I was able to get the user to input their scores, but unable to get the counter, "real_total", to add together the scores. After experimenting with different ways to solve the issue, I realized I was not returning the value of "real_total" and "total" at the conclusion of various functions. This solution permitted the scores to be added together and used across a few of the functions. I also encountered an opportunity with my code. Once I was satisfied with the functionality of my code, I realized it did not have a great aesthetic look in the terminal. I researched how to add color to Python and from termcolor, I imported colored and cprint. These two imports allowed me to add color to specific lines of code as well as add specific attributes such as bolding font.

2C. An algorithm that I developed individually in my code is **play_game()**. play_game() is a loop that is comprised of if-else statements and three algorithms. The loops iterates over three possible choices that all call different algorithms. play_game() asks the user if they would like to enter a new score, see their current pin total frame by frame, and see the current overall pin total. The loop uses logic with if-else statements. If the user inputs "1" and activates **enter_score(real_total),** a new algorithm is called. enter_score(real_total) asks the user which frame they would like to enter scores for and is made up of an if-statement that requires logic. The if-statement leads to various courses of action if the frame is between 1-9, equal to 10, or not in the range of 0 to 10. If the user inputs "2" and activates **pins_by_frame()**, a new algorithm that uses logic is called. pins_by_frame() is a selection statement that checks to see if you have entered fram scores into frame_total. If scores are entered, then your pin totals frame-by-frame are printed. If no scores are entered, the selection statement will produce an alternative message. All three algorithms that make-up play_game() are extremely pivotal to the functionality of the program and provide the user with options to view or add their pin totals.

```
68 def play_game(action, real_total):
69     cprint("\n1. Enter a new score", 'cyan', attrs=['bold'])
70     cprint("2. To see your current pin total frame by frame.", 'cyan', attrs=['bold'])
71     cprint("3. To see your current total pins knocked down.", 'cyan', attrs=['bold'])
72     cprint("4. Enter 'exit game' if you are done bowling.", 'cyan', attrs=['bold'])
73     action = input("What do you want to do %s? " % name)
74     if action == "1":
75         # pass and return real_total
76         real_total = enter_score(real_total)
77     elif action == "2":
78         pins_by_frame()
79     elif action == "3":
80         #pass in real total
81         total_pins(real_total)
82     else:
83         if action != 'exit game':
84             cprint("\nThis is not one of your options. Try again. ", 'red', attrs=["bold"])
85     return (action, real_total)
86
```

```
12 def enter_score(real_total):
13     cprint("\n1. Enter a score for frame 1", 'magenta', attrs=['bold'])
14     cprint("2. Enter a score for frame 2", 'yellow', attrs=['bold'])
15     cprint("3. Enter a score for frame 3", 'magenta', attrs=['bold'])
16     cprint("4. Enter a score for frame 4", 'yellow', attrs=['bold'])
17     cprint("5. Enter a score for frame 5", 'magenta', attrs=['bold'])
18     cprint("6. Enter a score for frame 6", 'yellow', attrs=['bold'])
19     cprint("7. Enter a score for frame 7", 'magenta', attrs=['bold'])
20     cprint("8. Enter a score for frame 8", 'yellow', attrs=['bold'])
21     cprint("9. Enter a score for frame 9", 'magenta', attrs=['bold'])
22     cprint("10. Enter a score for frame 10", 'yellow', attrs=['bold'])
23     cprint("11. enter '11' if you want to do something new.", 'blue', attrs=['bold'])
24     frame = int(input("What do you want to do? "))
25     if frame > 0 and frame < 10:
26         total, real_total  = two_rolls(real_total)
27         print_message(frame, total)
28         #include a return of the real_total
29         return (real_total)
30     elif frame == 10:
31         total, real_total  = frame_ten(real_total)
32         print_message(frame, total)
33         return (real_total)
34     elif frame == 11:
35         return 1
36     else:
37         print("Input ERROR. Try again.")
38         frame = input("What do you want to do? ")
```

```
110 def pins_by_frame():
111     if len(frame_total) > 0:
112         print("\nHere is your current pin totals:\n")
113         for frame in frame_total:
114             print(frame)
115     else:
116         print("\nYou have no scores currently entered.")
```

2D. An abstraction that I individually developed in my code is the function **enter_score(real_total).** This abstraction greatly helps to manage the complexity of my program. With enter_score(real_total), I did not have to continue entering a large amount of code each time I called this function. enter_score(real_total) serves as a main hub for accessing many other functions in my program and without it, many other functions would serve no purpose. The name of my abstraction as well as the names of the functions within my abstraction are worded specifically to lower the complexity and inform any reader of the function's purpose. enter_score(real_total) uses a logical if-else statement to operate and integrates math with its imbedded algorithms. First, the code segment, prints eleven lines of code that present options for the user to input their desired frame. The user is asked for their frame and then logic is used with if-else statements. If the frame entered is greater than 0 and less than 10, the functions two_rolls(real_total) and print_message(frame, total) are called. two_rolls(real_total) uses math to add together the user entered scores for "roll1" and "roll2". If the frame entered is equal to 10, the functions frame_ten(real_total) and print_message(frame, total) are called. frame_ten(real_total) uses math to calculate the pin total for specific spares and strikes rolls in the final frame.

```
12 def enter_score(real_total):
13     cprint("\n1. Enter a score for frame 1", 'magenta', attrs=['bold'])
14     cprint("2. Enter a score for frame 2", 'yellow', attrs=['bold'])
15     cprint("3. Enter a score for frame 3", 'magenta', attrs=['bold'])
16     cprint("4. Enter a score for frame 4", 'yellow', attrs=['bold'])
17     cprint("5. Enter a score for frame 5", 'magenta', attrs=['bold'])
18     cprint("6. Enter a score for frame 6", 'yellow', attrs=['bold'])
19     cprint("7. Enter a score for frame 7", 'magenta', attrs=['bold'])
20     cprint("8. Enter a score for frame 8", 'yellow', attrs=['bold'])
21     cprint("9. Enter a score for frame 9", 'magenta', attrs=['bold'])
22     cprint("10. Enter a score for frame 10", 'yellow', attrs=['bold'])
23     cprint("11. enter '11' if you want to do something new.", 'blue', attrs=['bold'])
24     frame = int(input("What do you want to do? "))
25     if frame > 0 and frame < 10:
26         total, real_total  = two_rolls(real_total)
27         print_message(frame, total)
28         #include a return of the real_total
29         return (real_total)
30     elif frame == 10:
31         total, real_total  = frame_ten(real_total)
32         print_message(frame, total)
33         return (real_total)
34     elif frame == 11:
35         return 1
36     else:
37         print("Input ERROR. Try again.")
38         frame = input("What do you want to do? ")
```