

Create – Applications From Ideas

Written Response Submission Template

Please see [Assessment Overview and Performance Task Directions for Student](#) for the task directions and recommended word counts.

Program Purpose and Development

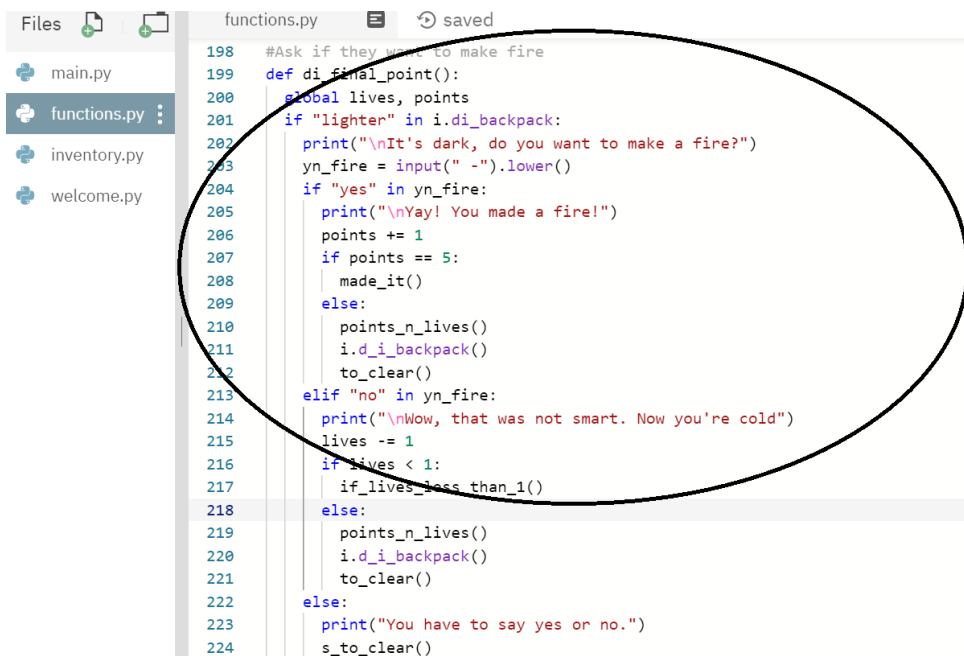
2a)

The programming language used is Python. The purpose of my program is to create game for the user to play. They answer a series of questions and then depending if they answer right they will earn points or lose lives. If they earn 5 points before losing all 3 lives, then they win the game. In the video, it asks the user if they want to play, they answer yes and the game starts. Once the game starts, the user's answer decides they want to go to the swamp. After, they start to answer a series of questions, when they answered correctly they earned a point, when they answered incorrectly, they lost a point. Once they finally got all 5 points, without losing all their three lives, they won the game.

2b)

I created the entire program independently. I wanted to give the option for the player to decide if they want to play the game. So, first I wrote that it would ask for the user's input, yes they want to play or no they don't. Then I wrote a logic concept to determine what to do based off the input. After testing the code I realized I should add `lower()` to the input of the user in order to make it simpler for the logic concept to accept the input. I tested it again and decided to change the way the logic concept was written, now, as long as yes or no was in the input, then the boolean was true. A difficulty I had was I couldn't change the value of the lives and points variables in a function. I solved this by making the variables global in the function, allowing them to be changed. Another difficulty was when creating a function, the code wouldn't recognize the function I'd written. I solved this by examining the code and realized that I'd forgotten to add `:` when defining the function, I added it and then the function worked.

2c)



```

198 #Ask if they want to make fire
199 def di_final_point():
200     global lives, points
201     if "lighter" in i.di_backpack:
202         print("\nIt's dark, do you want to make a fire?")
203         yn_fire = input(" -").lower()
204         if "yes" in yn_fire:
205             print("\nYay! You made a fire!")
206             points += 1
207             if points == 5:
208                 made_it()
209             else:
210                 points_n_lives()
211                 i.d_i_backpack()
212                 to_clear()
213         elif "no" in yn_fire:
214             print("\nWow, that was not smart. Now you're cold")
215             lives -= 1
216             if lives < 1:
217                 if_lives_less_than_1()
218             else:
219                 points_n_lives()
220                 i.d_i_backpack()
221                 to_clear()
222         else:
223             print("You have to say yes or no.")
224             s_to_clear()

```

In this algorithm the first sub algorithm functions depending on the what the user says. The first sub algorithm works as a logic concept, if the user says yes, then the algorithm will add a point to the points variable, if they say no then the algorithm will print a statement, and subtract a life along with other lines of code. The other subalgorithm works idependly also as a logic concept, if there are 5 points, then it will call on the made_it() function, if not then it will call on points_n_lives(), i.d_i_backpack(), and to_clear() functions. When all the subalgorithms work together they help the original and whole algorithm, which is another logic concept, and are able to determine if the player earned a point or a lost a life and what the effects of that are. Depending on those variables, points and lives, the whole algorithm will determine whether or not the user has a enough points to win the game or lost all their lives and lost the game if they indeed had the "lighter" tool in their backpack.

2d)

```

25
26 #####
27 def if_lives_less_than_1():
28     global points
29     count = 0
30     t.sleep(4)
31     o.system("clear")
32     print("--Lives: " + str(lives))
33     print("--Points: " + str(points))
34     if points != 5:
35         while points != 5:
36             points += 1
37             count += 1
38     if count != 1:
39         print("\nAll you needed were " + str(count) + " more points!!")
40     else:
41         print("\nAll you needed was " + str(count) + " more point!! ")
42     print("GAME OVER!")
43
44 #####
45 def made_it():
46     t.sleep(4)

```

I wrote this code so that every time the player lost, by loosing all three lives, the screen will display how many live and points they last had. It will also show how many more points or point the user needed in order to win the game. The abstraction helped manage the complexity of the program because everytime the user lost the game, I would put this function. This function gets used more than once in the program and so instead of having to write these lines of codes multiple times, I write just the function. This also makes it more organized and error free because if you were to constantly be writing the same complicated instructions, there is a chance that you will misspell or forget something.