Better Input Validation

Input Validation Using Functions

1. Write a function, *is_within_range*, that takes three integer arguments, *a*, *b* and *c*, and determines whether *a* lies in the range *b*-*c*, inclusive. Note that it could be the case that $b<c$, $b>c$ or $b=c$. In the body of your program, ask the user to enter two values, both between *m* and *n*. Ensure that the user does this by calling your function and using a loop. Once you have verified the two values, display their sum.
2. Write a function, t*ype_of_character*, that takes a single-character string, *s*, are determines whether it is a letter, a number, or a special character. Your function should return the strings "letter", "number" or "special" for each case. In the body of your program, ask the user to enter a letter and a number, and use a loop to ensure that the user does so, ignoring any special characters or additional letters/numbers. Print the letter this many times.
3. *DIY GPS*: Write a function, *get_direction*, that asks the user to enter a single letter representing a cardinal direction (*N*, *E*, *S*, *W*), or an *X* to stop. Your function should ensure that only one of these five options (either upper- or lowercase) is returned. Your main program should start the user at a fixed location, (0, 0). Use a loop to repeatedly obtain a direction from the user, then update their position. After each update, you should output the user's current location in the format "your location is X blocks east and Y blocks north of start." As some additional touches, you may want to modify your output to handle these specific cases: the user is at the starting position, or the user is directly north/south/east/west of start (print only the one direction).
4. *Avoid the Number*: A game is played where the computer randomly generates an integer between 1 and 5 inclusive. The player guesses an integer between 1 and 5. If the player guesses the same number as the computer, the game ends. Otherwise, the player scores one point and a new round is played. Write a program that plays *Avoid the Number*. Ensure that the user chooses a value between 1 and 5 each time (use your *is_within_range* function from Q1?), and display the final score once the game has ended.
5. *Factor Subtractor*: In this game, a random integer, *i*, between 20 and 100 is generated. The first player subtracts a factor of *i* (but not *i* itself) from *i*. This creates a new value for *i*. The second player then subtracts a factor from this new value, etc. This is done until one of the players makes the new value 1. This player is the winner. Write a function, *can_subtract*, that takes two arguments, *k* and *n*, and determines whether *k* is a factor of *n*, but not *n* itself. Your main program should repeatedly prompt a player to enter a valid factor by calling your function within a loop. Once a valid factor has been entered, subtract it from the current value, then switch players. Be sure to display the winning player!