Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

3.2 Mastermind

This lab will be available until April 8th, 11:59 PM EDT

In this assignment we will write a version of the Mastermind game using Python. Please visit the following site to familiarize yourself with the rules of the game:

https://howdoyouplayit.com/mastermind-board-game-rules-how-do-you-play-mastermind/

In our version of the game the user will play against the computer. Here is the set of modified rules to play against the computer:

1. The computer will generate a secret color code. The computer can use any combination of colors, including using two or more of the same color.
2. The player will try to guess the secret code by trying to enter the exact colors in the same positions as in the secret code.
3. The computer will give feedback to the user by indicating:
    - The number of right colors in the right position (e.g. 2 black)
    - The number of right colors but not in the right position (e.g. 1 white)

Here are a few examples:

Example #1

Computer's secret code: Blue-Red-Purple-Green

User's attempt: Red-Yellow-Purple-White

Computer's response: 1 white, 1 black.

The response above indicates one correct color in the right place (Purple), one correct color in the wrong place (Red) and two wrong colors.

Example #2

Computer's secret code: Purple-Blue-Green-Red

User's attempt: White-Yellow-Green-Green

Computer's response: 1 black

The response (black) shows one correct color in the correct place (Green). Note that even though the user entered Green twice, the feedback shows only one black. That's because the secret code has only one Green.

Example#3

Computer's secret code: Purple-Purple-Yellow-Green

User's attempt: Red-Yellow-Blue-White

Computer's response: 1 white

Just one white in the feedback for the Yellow entered by the user in the wrong place.

Example #4

Computer's secret code: Blue-Purple-Blue-Yellow

User's attempt: Green-Blue-Blue-White

Computer's response: 1 white, 1 black

This response (1 white, 1 black) is for the two Blue entered by the user: one in the right place and one in the wrong place.

Besides the rules, here are some additional specifications:

1. You will use list to store the following colors: blue, red, white, green, yellow, purple.
2. Your program will generate a secret code made of 4 colors from the list above and provide feedback for each guess. The secret code should be randomly generated for each new game. You should import the module `random`. You will be using the following two functions:
    - `random.seed(seed_value)` seeds the random number generator using the given `seed_value`. For testing purposes, use the seed value *350*, which will cause the computer to choose the same random numbers every time the program runs.
    - `random.randint(a, b)` returns a random number between a and b (inclusive).
3. Print out the number of white pegs and the number of black pegs as the feedback for each guess, e.g., 2 whites 1 black Note that the feedback only says how many exact and partial matches, and nothing about which slots are matching.
4. The program should perform error-handling and let the user know when they have entered an invalid color. The user should be allowed to try again until they enter a valid color.

5. Your program must include a loop so that the player can repeatedly enter guesses and received feedback from the system. When the player enters a correct guess (all exact matches), the system prints out a congratulation and tells the user how many guesses were required.
6. Your solution must include at the very least the following functions:
    - `generateSecretCode(colorList)`: a function that received a list of colors then generates and returns a random secret color code (as a list). Use `random.randint(0, 5)` to get a number between 0 and 5 inclusive, which you can then map to one of the colors of in `colorList`.
    - `computeExactMatches(computerColorList, playerColorList)`: a function that compares the secret color code with player's guess and returns the number of exact matches (e.g.: correct colors in the right position).
    - `computePartialMatches(computerColorList, playerColorList)`: a function that compares the secret color code with player's guess and returns the number of partial matches (e.g.: correct colors that are in the wrong position).

Please make sure you include a header comment at the top of your file, as well as comments before each function.

Below there's a sample run:

```
Mastermind Game
The computer has generated a secret color code.
Can you guess the code?
These are the valid colors you can use: blue red white green yellow
purple

Please enter color 1 : orange
That is not a valid color. Try again.
Please enter color 1 : BLUE
Please enter color 2 : red
Please enter color 3 : Green
Please enter color 4 : white
1 black
1 white


Please enter color 1 : blue
Please enter color 2 : white
Please enter color 3 : purple
Please enter color 4 : yellow
2 black
2 white
```

```
Please enter color 1 : blue
Please enter color 2 : yellow
Please enter color 3 : purple
Please enter color 4 : white
1 black
3 white


Please enter color 1 : blue
Please enter color 2 : white
Please enter color 3 : yellow
Please enter color 4 : purple
1 black
3 white


Please enter color 1 : blue
Please enter color 2 : purple
Please enter color 3 : yellow
Please enter color 4 : white
2 black
2 white


Please enter color 1 : blue
Please enter color 2 : purple
Please enter color 3 : white
Please enter color 4 : yellow
4 black
0 white


Congratulations, you guessed the secret code!
It took you 6 turn(s).
```

Notes:

- You may find the following site useful to compare your output against the expected program output: Diffchecker
- The purpose of this problem is to practice writing functions and using modules.
- Please make sure to submit a well-written program. Good identifier names, useful comments, and spacing will be some of the criteria that will be used when grading this

assignment.
- This assignment can be and must be solved using only the materials that have been discussed in class. Do not look for alternative methods that have not been covered as part of this course.

How your program will be graded:

- correctness: the program works and performs all tasks correctly: 60% (autograded by Zybooks)
- complies with requirements (it properly implements required functions, input validation, loops/conditionals, random module): 25% (TAs)
- code style: good variable names, comments, proper indentation and spacing: 15% (TAs)

312782.1712434.qx3zqy7

LAB ACTIVITY | 3.2.1: Mastermind | 0 / 100

Downloadable files

`main.py` **Download**

main.py — Load default template...

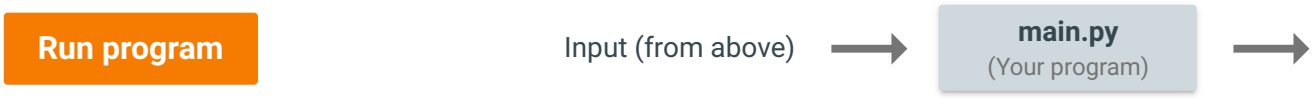
```
1 # TO DO: ADD YOUR HEADER COMMENT HERE
2
3 # TO DO: IMPORT NEEDED MODULE(S)
4
5 # TO DO: DEFINE REQUIRED FUNCTIONS - EACH FUNCTION MUST HAVE A COMMENT
6
7 if __name__ == "__main__":
8     # TO DO: WRITE THE MAIN ROUTINE HERE
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.



Program output displayed here

Signature of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

Trouble with lab?