

CIS 40 Final project

Write the game Mastermind that interacts with the user in a command line interface and draws the moves of the game with turtle graphics.

This project is for a team of 2 - 3 students. Each team member will upload the same code on the due date and will receive the same score.









The score of 50 points is divided into 2 parts: 25 points for a working demo in class, 25 points for code that meets all requirements.

Mastermind game description

Mastermind is a game for 2 players: the computer and the user. The computer comes up with a random pattern which is made up of 3 different colors (for example: blue purple red). The user has 10 tries to guess the 3 colors, in the correct order, out of a list of 6 different colors (red, green, blue, yellow, purple, orange).

As the user guesses, the computer gives hints to the user. The hints are in 2 different colors: black and gray. Each black means that one color is correct *and* at the correct position. Each gray means that one color is correct but at the wrong position. Not getting a black or gray hint means the color is wrong.

Here's an example of a game being played. The game output starts at the bottom line of colored circles and goes up the page:

	User's guess	Computer's hint	
4th guess: swap order of orange and yellow			Got it!
3rd guess: keep yellow, put back green, add orange			1 color in correct pos (green), 2 correct colors in wrong pos.
2nd guess: keep red, guess 2 other colors			1 correct and 2 wrong colors. Purple or yellow is correct.
1st guess: red, green, blue			1 color in correct position, 2 wrong colors.

For further reading of the Mastermind game (history, math analysis, algorithms to solve, more facts than you ever want to know about the game): [https://en.wikipedia.org/wiki/Mastermind_\(board_game\)](https://en.wikipedia.org/wiki/Mastermind_(board_game))

[One of the algorithms is developed by Donald Knuth, also known as the father of computer algorithms. Take a quick glance at it if you are curious about a topic that CS senior level students or graduate students work with.]

For an online game that you can play and get to know how the game works:

<http://www.braininshape.com/mastermind/>

To play, click the colored balls and then click the slots of your choice to enter your guess. Then click Check. In this version, red means correct color in correct position, white means correct color in wrong position.

Demo requirements

1. Print a welcome message to the game
2. Prompt to let the user choose from:
 - 3 - beginner level with 3 color pattern
 - 4- expert level with 4 color patternIf the user did not type in 3 or 4, set the level to 3.
Print the level that the user will play (3 or 4).
3. Play one game:
 - choose a random pattern of 3 (or 4) different colors, depending on the user level.
 - loop as long as the user hasn't guessed the correct pattern or hasn't used up 10 tries:
 - print a list of available color: r = red, g = green, b = blue, y = yellow, p = purple, o = orange
 - ask the user for a guess of 3 (or 4) different colors
 - use turtle graphics to display the user's guess
 - determine if any of the colors is correct and if any of the correct colors is in the correct position
 - use turtle graphics to display the hints
 - when the loop stops, print:
 - "You win in <num> tries" where <num> is the number of tries the user has used up
or
 - "Sorry" and print the actual pattern
4. Loop to play again: ask the user if they want to play again, and either loop back to play one more game, or end the program.

Here is the user interaction of the game that produced the graphics output in the previous page:

```
Welcome to Mastermind, the CIS 40 version!
Your choices:
3. beginner level with 3 colors
4. expert level with 4 colors
Choose 3 or 4: 3
Playing at level 3
r = red, g = green, b = blue, y = yellow, p = purple, o = orange
Type 3 different letters in the order of your 3 choices: rgb
r = red, g = green, b = blue, y = yellow, p = purple, o = orange
Type 3 different letters in the order of your 3 choices: ryp
r = red, g = green, b = blue, y = yellow, p = purple, o = orange
Type 3 different letters in the order of your 3 choices: ygo
r = red, g = green, b = blue, y = yellow, p = purple, o = orange
Type 3 different letters in the order of your 3 choices: ogy
You win in 4 tries!
Play again? y/n: n
```

Code requirements

1. Write a MastermindOutput class in a file called mOutput.py.

MastermindOutput will handle the graphics output for the game. It should have the following methods:

- An **__init__** method to create the Turtle object and set it to the correct initial position such that the pattern is drawn in the left column and the hints are drawn in the right column.
- A **drawUserGuess** method that will draw a user's guess as 3 (or 4) filled circles of the user's choice of color. Within one game, each time drawUserGuess runs, it will draw the user's pattern above the previous run, so that the pattern forms a column that goes up from the bottom of the screen.
- A **drawHints** method that will draw the hint pattern of black and gray circles. The circle for the hints should be smaller than the circle for the user's guess. All black circles should appear before any gray circles. The hint pattern should be on the same line as the user guess pattern on screen.
- A **drawCircle** method that is called by both the drawUserGuess and drawHints methods. The drawCircle method will draw *one* filled circle with the appropriate size and color.
- Any other "support" method that you think is necessary.

2. Write a MasterMind class in a file called mm.py. The Mastermind class will handle all aspects of one game.

- An **__init__** method that prints the welcome message, initializes object attributes, and starts the Turtle object through MastermindOutput class.
- A method that lets the user chooses the level (3 or 4).
- A method that randomly chooses a pattern of 3 or 4 colors.
- A method that reads in the user input. You can assume that the user will give the correct number of letters (3 or 4) and the letter will not repeat (for example, not: ggr).
- A method that checks the user input against the computer generated pattern. This method needs to check, for each of the user's color choice: whether the color correct, and if correct, whether it is in the correct location. The method uses the result to create the hints for the user.
- A method that plays one game. This method uses the MastermindOutput object to draw the user's guess and the corresponding hints. This method has a loop for one game, as outlined in the Demo requirement step 3.
- A **playLoop** method that loops to keep calling the method to play one game above, until the user chooses to quit. You can decide how you let the user chooses to quit, but prompt clearly so they know.
- Write any other "support" method that you need.

3. At the end of the mm.py file, add the driver code that has the following 2 lines:

```
m = Mastermind()           # start the Mastermind game object
m.playLoop()               # loop to play Mastermind until user quits
```

These should be the only 2 lines in the driver code.

4. Important coding requirement: you must use only Python concepts and topics that are covered in class. Any new or advanced concepts will result in a point deduction. The purpose of this requirement is to encourage everyone to learn how to implement the "basics" of programming really well. You'll have plenty of opportunities to use advanced Python techniques and data structures in the next classes.

One additional Turtle method that's useful for this project: `t.ht()` to hide the turtle in the graphics output.

Testing

To play the Mastermind game, the user will interact with the script through the IDLE shell, and the pattern and hints will be drawn at the graphics output window. Both turtle graphics and IDLE use the Tcl module for graphics, which means that they cannot run concurrently (at the same time).

Therefore, you should do *unit testing* first. This means test the mOutput.py as one unit, and test mm.py as a different unit.

- At the end of the mOutput.py, add test driver code to test the all methods of the MastermindOutput object, making sure that it can draw the output correctly.
- In the mOutput object, comment out all the calls to the MastermindOutput object methods that draw the user's pattern and hints, and instead put in print commands to print the user's pattern and hints as strings of characters, ie. "rgb" for red, green, blue. This way the code doesn't interact with the MastermindOutput class. It will make the testing of the game go faster if you don't have to wait for the graphics screen.

After both the graphics output and the game output are as expected, then you can begin integration testing. To start *integration testing*: comment out the lines of code that prints the patterns and hints as characters, and bring back (uncomment) the calls to MastermindOutput's methods to draw the output. Then follow the instructions below to run your python program from a command window. (You might not be able to use IDLE at this point, depending on your system set up.)

Instructions to run the game for Windows system at school:

1. Put your mm.py and mOutput.py into one project folder and save it in your Z drive folder
2. Copy the project folder to the desktop
3. Start the command window: At the Start menu, type `cmd`
4. At the command window, go to the project folder by typing `cd Desktop\project`
5. At the command window, type the command to run python:
`"C:\Program Files (x86)\Python35-32\python" mm.py`

Instructions to run the game for your Mac or Windows system:

1. Put your mm.py and mOutput.py into one project folder on the desktop
2. Start the command window
3. At the command window, go to the project folder that's on the desktop
4. At the command window, run python by typing the command: `python mm.py`
(Depending on how python is set up on your computer, you might need a path in front of the "python" command, similar to the path for the Windows system at school shown above)