MASTERMIND

THE ORIGINAL CODE-BREAKING GAME

NOW COLORBLIND-ACCESSIBLE!

Outsmart a computer with MasterMind, a text-based game based on the original boardgame from 1971! Play as the codebreaker and guess the hidden code within 10 guesses. The codemaker (the computer) sets a hidden code consisting of numbers 0 through 9. With each guess, the codemaker reveals how many numbers are correct and in the correct position or correct but in the wrong position. Compete to guess the hidden code in as few guesses as possible!

GAME SET-UP

Select a difficulty level by typing one of these keywords, then pressing the Enter key:

- random - the computer selects at random from 4 to 8:
  - 4 - code is 4 numbers long. Great for quick games
  - 5 - code is 5 numbers long. For a bit of a challenge
  - 6 - code is 6 numbers long. A more intense challenge
  - 7 - code is 7 numbers long. Got lots of time to spare?
  - 8 - code is 8 numbers long. For TRUE MASTERMINDS

TURN-BY-TURN INSTRUCTIONS

1. Type your guess, then press the Enter key

2. The computer gives feedback on your guess:
   - R - correct number, correct position
   - W - correct number, wrong position

   Sample feedback: "2R - 1W" means the guess contains 2 correct numbers in the correct positions and 1 correct number in the wrong position. The challenge is deciphering which numbers!

3. This cycle repeats until the 10th valid guess or until the correct code is guessed (and the game is won!)

REFERENCE: HTTPS://WEBGAMESONLINE.COM/MASTERMIND/RULES.PHP

MASTERMIND

THE ORIGINAL CODE-BREAKING GAME

NOW COLORBLIND-ACCESSIBLE!

WINNING THE GAME

Win the game by correctly guessing the hidden code (all numbers are correct and in their correct positions) by Guess #10. If the code is not guessed by then, the hidden code is revealed and the computer gives the option to play again or quit.
The scoreboard is also shown when a game is won. If you beat the current score (lower is better) for that difficulty level, the computer will ask for your name to add to the scoreboard.
(Note: name must be 1-7 characters long)

CLARIFICATIONS & BONUS FEATURES

>>> The game interface displays which Guess Number out of 10 the player is currently on. Note: Using lifelines (See NEED a HINT? below) "uses up" guesses or skips Guess # forward!
>>> Guesses of the wrong length or with characters other than numbers 0 through 9 result in "Invalid input". The computer will then automatically ask for a new guess.
>>> The computer will only give feedback on the overall guess, i.e. not indicating which numbers correspond to "R" or "W".

NEED A HINT? Type ONE of these keywords followed by the Enter key:
lifeline#1 - reveals 1 correct number anywhere in the code, uses up 1 guess
lifeline#2 - reveals 1 correct number in the correct position, uses up 2 guesses
>>> IMPORTANT: ONLY ONE LIFELINE ALLOWED PER GAME!
>>> Lifelines won't work if there are too few guesses left.

BONUS FEATURES! Type the following keywords at any time after difficulty selection, followed by pressing the Enter key:
reset - starts a new game
quit - exits the game

REFERENCE: HTTPS://WEBGAMESONLINE.COM/MASTERMIND/RULES.PHP

MASTERMIND

IMPLEMENTATION NOTES

GENERAL IMPLEMENTATION

main()

- Initialize and later call the following functions:
  - SequenceGenerator() - selects difficulty and generates the hidden code
  - GameInput() - parses user input, determines what to do next
  - CheckGuess() - validates a user-inputted guess and gives feedback
  - highscore() - displays and adds a player name to leaderboard
- Prevent accidentally invoking the script when user doesn't intend to
- Set playingGame to True (for reset and quit functions)
- while playingGame (is True):
  - Set playingRound to True (for reset and quit functions)
  - Set masterSeq to the return value of SequenceGenerator() (the hidden code)
  - Display the length of masterSeq (len(masterSeq)) and number of guesses
  - Initialize variables:
    - attempts - set value: 10; decremented as game progresses
    - markedAnswer - a masterSeq list copy; "marked" as game progresses
    - lineUsed - set to False; for lifeline function
  - KeepPlaying - string variable controlling game's state
  - guess - user input consisting of numbers 0 through 9
  - While attempts > 0 and playingRound (is True):
    - Initialize and set answer to a list copy of masterSeq
    - Set KeepPlaying, attempts, guess, markedAnswer, lineUsed to return and input values of GameInput(attempts,answer,markedAnswer,lineUsed)
    - Set KeepPlaying, markedAnswer to return and input values of CheckGuess(guess, answer, markedAnswer)
    - If KeepPlaying = "continue", continue (return to beginning of loop)
    - If KeepPlaying = "win":
      - Display win message and score (= 10 minus attempts)
      - Display the previously selected difficulty (= len(answer))
      - Call highscore(difficulty, score)
      - Set playingRound to False
    - If KeepPlaying = "reset" or "quit", set playingRound to False
  - If attempts = 0 or playingRound is False: (round ended state)
    - If attempts = 0 and KeepPlaying is not "win": (game lost)
      - Display loss message and reveal the hidden code (masterSeq)
    - While loop asks for user input until reset or quit is indicated:
      - Initialize and set resetQuery to string user input
        - if resetQuery = "reset":
          - Display status message and set playingRound to False
          - break (exit the loop)
        - if resetQuery = "quit":
          - Display message thanking user for playing
          - Set both playingRound and playingGame to False
          - break (exit the loop)
- system("pause") prevents shutdown if main code is accessed outside interpreter

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

codeGenerator.py

- Specification: Asks user for desired difficulty level and returns a randomly generated list of string values (0-9).
- User selects the game difficulty (length of the hidden code to be guessed) via:
    - entering a specific integer from 4 to 8, or
    - the string "random"
- Function then continuously asks for input until valid.
    - isnumeric() validates that the input is numeric as desired
    - randint() generates random integers from a set range
- Returns the master sequence (the hidden code) generated from list comprehension
    - List comprehension more quickly creates new lists vs. for-loops.

SequenceGenerator()

- Import random from randint library
- Initialize and set invalidInput to True
- While invalidInput (is True):
    - Initialize and set level to user input, lowercase and with spaces removed
        - input() displays a message specifying valid inputs
    - If level = "random":
        - Set level to a random integer from 4 to 8 (randint(4,8))
        - Intialize and set masterSeq to a list of length level, whose elements are random integers from 0 to 9 (randint(0,9))
        - Set invalidInput to False
    - If level consists of numeric characters (isnumeric()) from 4 to 8:
        - Set masterSeq to a list of length level, whose elements are random integers from 0 to 9 (randint(0,9))
        - Set invalidInput to False
    - Otherwise (else), display a prompt specifying valid inputs
- Return masterSeq

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

inputs.py GameInput()

- Specification: Function continuously takes user input, validates it, and processes it appropriately. Returns these values: game state determiner (quit, continue, etc.), attempts remaining, the user's guess, markedAnswer, lineUsed.
- Multiple conditional statements determine:
    - What the input tells the game to do (e.g. quit, reset, continue the game) to give appropriate feedback to the user
        - For lifelines, LifelineRequest() is called.
    - Whether the input is valid or invalid
        - Checks if input is all numbers and if the same length as answer
        - Then checks if the input has alphanumeric characters that would result in ValueError.
        - Then a final else block catches any unexpected inputs, blank inputs, other characters, that would result in an error.
        - If input results in an error, it would loop back to ask for input again until valid input is given.

GameInput()

- Import random
- Define the function LifelineRequest()
- Takes attempts, answer, markedAnswer, lineUsed as input
- While GameInput receives valid input (while True):
    - Display which guess the player is currently on (= 11 - attempts)
    - Initialize and set guess to take user input, lowercase and without spaces
    - If guess = "lifeline#1" or guess = "lifeline#2" calls LifelineRequest()
        - Set attempts, markedAnswer, lineUsed to the return values of LifelineRequest(guess, attempts, answer, markedAnswer, lineReqs)
    - If guess = "quit":
        - Return "quit", attempts, guess, markedAnswer
    - If guess = "reset":
        - Return "reset", attempts, guess, markedAnswer
    - If guess string length = answer string length and guess isnumeric():
        - Decrement attempts by 1
        - Return "continue", attempts, list(guess), markedAnswer, lineUsed
    - Otherwise (else), guess is invalid and computer displays guide prompts
        - If guess string length and answer string length are not equal and guess isnumeric():
            - Display guide message with the correct string length of answer (len(answer))
        - If guess contains alphanumeric characters:
            - Display message stating guess only takes integers and specific strings: lifeline#1, lifeline#2, reset, quit
        - Otherwise (else):
            - Display message stating input contains invalid characters

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

inputs.py LifeLineRequest()

- Specification: Takes kind of lifeline, attempts remaining, master sequence, marked Answer, and lineUsed. Outputs requested lifeline and returns attempts remaining, new marked answer, and lineUsed.
    - kind - either "lifeline#1" or "lifeline#2"
    - lineUsed - Boolean indicating if a lifeline was already used
- lifeline#1
    - Create a list of the unmarked numbers from markedAnswer and choose 1 number at random from this new list.
    - The first instance of that number is then marked "W" in markedAnswer.
    - If there are duplicate numbers in the hidden code, no indication is given if the number is found more than once in the code (They are marked as "W")
- lifeline#2
    - Create a list of the unmarked numbers' positions.
    - A number is randomly chosen from this list, and then that number's value is retrieved from the unmarked "answer" list.

LifelineRequest()

- Takes kind, attempts, answer, markedAnswer, lineUsed as input
- If lineUsed (is True; exists), display a message stating lifeline already used
- If kind = 'lifeline#1' and attempts > 1:
    - Initialize and set choices to a list copy of markedAnswer's unmarked elements (not marked "R" or "W")
        - If not choices: (if all elements in markedAnswer have been marked)
            - Display a message stating the lifeline will be redundant
        - Otherwise (else):
            - Initialize and set choice to a random element from choices
            - Set lineUsed to True; decrement attempts by 1
            - Display choice and a message stating 1 attempt was used
            - Mark the equivalent number choice in markedAnswer as "W"
- If kind = 'lifeline#2' and attempts > 2:
    - Initialize and set positions to a list copy of markedAnswer's elements not marked "R" (includes unmarked numbers and those marked "W")
    - If not positions: (if all elements in markedAnswer have been marked "R")
        - Display a message stating the lifeline will be redundant
    - Otherwise (else):
        - Initialize and set position to a random index in positions
        - Set choice to the number in answer corresponding to that index
        - Set lineUsed to True; decrement attempts by 2
        - Display choice, (position + 1), its index, and a message stating 2 attempts were used
        - Mark the equivalent number choice in markedAnswer as "R"
- Otherwise (else), display attempts and a not-enough-guesses message
- Return attempts, markedAnswer, lineUsed

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

feedback.py

- Specification: Parameters guess, unmarked, answer, and markedAnswer are used to check if the guess is correct (or if the user wants to reset or quit) then prints feedback on the guess. Returns Game State determiner and markedAnswer.
- r and w represent counts of correct values and positions in that guess.
- markedAnswer is a master list containing the correct "R"'s and "W"'s of the player across multiple guesses. It is used to give a relevant lifeline hint.
- answer is a temporary list containing the correct "R"'s and "W"'s for the current guess only. It is used to give accurate feedback during that turn. Marking correct guesses in answer prevents incorrect increments of "R" and "W".
- First for-loop marks "R"'s, preventing marking accidental "W"'s if the guess has multiple correct guesses. For example: Answer: 3859; Guess: 3333
    - After first for-loop, Answer is now: R859; Guess is now: R333.
    - On second for-loop, since "3" is no longer in Answer, the remaining "3"'s in the guess will not trigger "W".
- In cases where the answer has duplicate numbers: "W"'s are marked in the second for-loop, preventing accidental "W"'s. For example: Answer: 9266; Guess: 6600
    - Loop 1 does nothing (no "R"'s in this Guess)
    - After first run of second for-loop: Answer: 92W6; Guess: W600
    - After second run of second for-loop: Answer: 92WW; Guess: WW00
- Resetting and quitting the game calls the CheckGuess() function, because of the design of the functions and the order they are called.

CheckGuess()

- Takes guessList, answerList, and markedAnswer as input
- If guessList and answerList are equivalent strings, return "win", markedAnswer
- If guessList = "reset", return "reset", markedAnswer
- If guessList = "quit", return "quit", markedAnswer
- Initialize and set:
    - r to 0; w to 0
    - guess to a list copy of guessList; answer to a list copy of answerList
- For each character in guess: marks all "R"'s (correct value and position)
    - If that character in guess is the same in answer in that position:
        - Increment r by 1
        - Mark that character as "R" in answer, guess, and markedAnswer
- For each character in guess: marks all "W"'s (correct value)
    - If that character exists in answer and is not marked "R":
        - Increment w by 1
        - Initialize and set index to the position of that character in answer
        - Mark the character in that position in answer as "W"
        - If the character in that position in markedAnswer is not "W":
            - Mark it as "W"
- Display the feedback for that guess as "(r)R - (w)W"
- Return "continue", markedAnswer

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

score.py highscore()

- Specification: From parameters difficulty and score, create a text file that stores the difficulty, score, and name of player.
- Calls functions scoreBoard() and getName()

- scoreBoard()
  - Specification: Reads a .txt file that contains the Difficulty, Score, and Name, then checks if the player won the game with lower attempts (higher score). If True, it will ask for the player's name and rewrite the .txt file. Lastly, it would print out the new scoreboard.

- getName()
  - Specification: Player would be repeatedly asked for his name if the input does not consist of 1-7 characters only.
  - This function also checks if a scoreboard.txt exists in the directory. If there is no such file, it would create a new file and write a specific text on the file.
    - os.path.isfile() - checks if a file exists in a specific directory.
    - with open() as file - opens the file and ensures that the file gets closed after use. It shortens the code by removing the file.close() in the code.

highscore()

- Define the functions scoreBoard() and getName()
- Initialize and set fileName to "scoreboard.txt"
- Initialize and set fileExists to the file's location (path)
- If the file does not exist: (create one)
  - Open and write to the file fileName:
    - Initialize and set text to a string, the starting line of the file
    - Write text to the file
    - Display a message stating file not found, a new file will be created
- Call scoreboard(fileName, difficulty, score)

MASTERMIND

IMPLEMENTATION NOTES

SPECIFIC FUNCTION IMPLEMENTATION

highscore(): scoreBoard() and getName()

scoreBoard()

- Takes path, difficulty, and score as input
- Change score and difficulty to string type variables
- Initialize an empty list board
- Open and read the file given path:
    - Initialize and set reader to a list of each line in the file
    - For every item (line) in reader:
        - Initialize and set scores to that line split into a list of strings
        - Append that list in scores to board
- For each item (line) in board:
    - If difficulty is the same as that line's first item (difficulty level):
        - If that line's third item (a score) is "-" or is greater than score:
            - Display difficulty and congratulatory message (new high score)
            - Initialize and set name to return value of getName()
            - Set that line's second item to name
            - Set that line's third item to score
            - Exit the loop (break)
        - If that line's third item and score both equal 1: (highest score)
            - Display that item's second item (a player name) and a message stating that that player got that highest score first
        - If score is greater than or equal to that line's third item:
            - If score is equal to that line's third item: (same score)
                - Display that item's second item (a player name) and a message stating that that player got that score first
            - If score is greater than that line's third item: (higher score)
                - Display that item's second item (a player name) and a message stating that that player got a higher score
        - Exit the loop (break)
- Initialize and set an empty string newBoard
- Display scoreboard headers ("Scoreboard" and "Level Score Name")
- For each item in board:
    - Initialize and set winners to a string displaying level, score, name
    - Display winners
    - Append winners as a formatted string (new line) to newBoard
- Write newBoard to the file

getName()

- while True: (until input is valid)
    - Initialize and set name to user input, uppercase and with spaces removed
    - If name is 1 to 7 characters long and contains no spaces:
        - Return name
    - (Otherwise,) display a message specifying valid input

MASTERMIND

REFERENCES

REFERENCES

- f-strings (`print(f"stringhere"`)
  - https://realpython.com/python-f-strings/

- List Comprehension
  - https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions

- open()
  - https://docs.python.org/3/library/functions.html#open
  - https://book.pythontips.com/en/latest/open_function.html

- os.path library
  - https://docs.python.org/3/library/os.html
  - https://docs.python.org/3/library/os.path.html

- random library
  - random – Generate pseudo-random numbers – Python 3.10.1 documentation
  - https://docs.python.org/3/library/random.html#random.choice

- Mastermind Reference Game and Rules
  - https://webgamesonline.com/mastermind/
  - https://webgamesonline.com/mastermind/rules.php