CompSci 101 - Assignment 02 Mastermind

Due: 4:30pm, 4th September 2020.

Worth: This assignment is marked out of 30 and is worth 3% of your final mark.

Topics covered:

- Functions, `if` statements, string manipulation, converting types

The work done on this assignment **must** be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone else for help. Under no circumstances should you use code written by another person in your assignment solution or give your code to another student.

Very Important:

This assignment has two sections.

Section A is marked using Coderunner (22 marks). For this section you need to define 17 functions. The 17 functions, when inserted into the Section B skeleton program, create the game of Mastermind (see the game description below). Each function is described in Section A of this document and there are four Python skeleton programs which you **MUST** use to develop the 17 functions.

Section B (8 marks).
Insert the 17 functions from Section A into the Assignment 2 program (see Page 14 of this document). The program should execute without error allowing the user to play the Mastermind game.

Submission

Section A - Functions 1 - 17 are submitted using CodeRunner3.

Section B - Submit your completed Assignment 2 Python program (just one Python file named `"YourUsernameA2.py"`, e.g., `afer023A2.py`) using the Assignment Dropbox:

https://adb.auckland.ac.nz/Home/

Notes:

- This assignment is marked out of 30 and is worth 3% of your final mark. Five marks out of 30 are assigned for the style of your program (program docstring, variable names, etc.). Three marks out of 30 are assigned if your final Mastermind program executes without error.
- You **MUST** only use the features taught in CompSci 101.
- An example output using the completed program is available at the end of this document:

Assignment 2: The game of Mastermind

For this assignment you will develop the Mastermind game. In this game four letters (A, B, C and D) are randomly combined (the hidden letters) and the player tries to guess the exact combination of the letters. Note that the hidden letters may contain repeated letters. The player is allowed eight guesses and, if they are unable to correctly guess the hidden letters, they lose the game. After each guess the player is given the following feedback:

- how many letters of their guess are correct, i.e. the letter is correct and is in the correct position.
- how many letters of their guess are one of the hidden letters but are not in the correct position.

Assignment 2 Section A (22 marks)

For this section you need to develop 17 functions described in Section A of this document. Each of the functions are to be done in IDLE and then submitted using CodeRunner3. Download the four Python skeleton programs from the CompSci 101 assignments website:

https://www.cs.auckland.ac.nz/courses/compsci101s2c/assignments/

The following four programs must be used to develop the 17 functions used in the Mastermind game.

- Use the A2Functions1To4.py program to complete and test functions 1, 2, 3 and 4.
- Use the A2Functions5To8.py program to complete and test functions 5, 6, 7 and 8.
- Use the A2Functions9To12.py program to complete and test functions 9, 10, 11 and 12.
- Use the A2Functions13To17.py program to complete and test functions 13, 14, 15, 16 and 17.

Once you are happy that a function executes correctly, submit it to CodeRunner:

https://coderunner3.auckland.ac.nz/moodle/

Section A

The skeleton code for each of the Section A programs is provided. Each program requires you to complete the function. As you complete each function you can test that the function is correct by pasting the whole function (including the header) into CodeRunner3 and pressing the CHECK button.

After the description of each of the 17 functions there is a textbox with some example code and below it a textbox containing the expected output for that code.

1.

Define the `get_player_name()` function. This function prompts the user for their name using the prompt, `"Enter name: "`, and the function returns the name entered by the user.

Example code:

```
name = get_player_name()
print(name)
```

Output (after the user enters the name `"Mika"` when prompted):

```
Enter name: Mika
Mika
```

<----------------------->

2.

Define the `display_game_welcome(player_name)` function. This function is passed a string parameter: the player's name. The function prints 5 lines of output:

- Line 1 is a blank line,
- Line 2 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3.
- Line 3 prints the three spaces, the string, `"MASTERMIND. Welcome "`, followed by the player's name and finally a full stop,
- Line 4 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3.
- Line 5 is a blank line.

Example code:

```
display_game_welcome("David")
display_game_welcome("Jian")
```

Output:

```
******************************
   MASTERMIND.  Welcome David.
******************************

******************************
   MASTERMIND.  Welcome Jian.
******************************
```

<----------------------->

3.

Define the `get_main_menu_selection()` function. This function prints four lines of output and returns the integer entered by the user:

- Line 1 prints the string `'1. Play a game'`,
- Line 2 prints the string `'2. See instructions'`,
- Line 3 prints the string `'0. Exit'`,
- Then the code stores the integer value entered by the user in response to the prompt made up of 3 spaces and `'Enter selection: '`,
- The last line is a blank line.

The function returns the integer value entered by the user.

Example code:

```
selection = get_main_menu_selection()
print(selection, selection - 1)
print()
selection = get_main_menu_selection()
print(selection, selection + 4)
```

Output:

```
1. Play a game
2. See instructions
0. Exit
   Enter selection: 1

1 0

1. Play a game
2. See instructions
0. Exit
   Enter selection: 2

2 6
```

<---------->

4.

Define the `display_game_instructions()` function. The function prints 8 lines of output:

- Line 1 is a blank line,
- Line 2 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3.
- Line 3 prints three spaces and `"The letters A, B, C and D are randomly combined and hidden from view."`,
- Line 4 prints three spaces and `"The player has to guess the four letters in the correct order."`,
- Line 5 prints three spaces and `"The maximum number of guesses allowed is eight."`,

- Line 6 prints three spaces and "Note: letters can be repeated.",
- Line 7 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3.
- Line 8 is a blank line.

Example code:

```
display_game_instructions()
```

Output:

```
**************************************************************************
   The letters A, B, C and D are randomly combined and hidden from view.
   The player has to guess the four letters in the correct order.
   The maximum number of guesses allowed is eight.
   Note: letters can be repeated.
**************************************************************************
```

<---------------------------->

5.

Define the `display_game_end_message(player_name)` function. This function is passed one string parameter: the player's name. The function prints 3 lines of output:

- Line 1 is a blank line,
- Line 2 prints "Bye ", the player's name followed by "! Thank you for playing.",
- Line 3 is a blank line.

Example code:

```
display_game_end_message("David")
display_game_end_message("Jacob")
```

Output:

```
Bye David! Thank you for playing.


Bye Jacob! Thank you for playing.
```

<---------------------------->

6.

Define the `get_game_menu_selection()` function. This function prints five lines of output and returns the integer entered by the user:

- Line 1 prints the string `'1. Have a guess'`,
- Line 2 prints the string `'2. See list of guesses so far'`,
- Line 3 prints the string `'3. Take a peek'`,
- Line 4 prints the string `'0. Return to the main menu'`,
- Then the code stores the integer value entered by the user in response to the prompt made up of 3 spaces and `'Enter selection: '`,
- The last line is a blank line.

The function returns the integer value entered by the user.

Example code:

```
selection = get_game_menu_selection()
print(selection, selection * 5)
```

Output (the user input is shown in bold):

```
1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 6

6 30
```

<---------->

7.

Define the `display_guess_history(user_guesses_so_far)` function. This function is passed one string parameter: a string made up of the player's previous guesses and results. If the parameter string is made up of only space characters the function prints: `"No guesses have been entered yet."`. Otherwise the function first prints `"Guesses so far:"` on one line followed by the parameter string on the following lines.

Note: the `user_guesses_so_far` parameter is a string which may contain several newline characters.

Hint: the `strip()` string method is useful for obtaining a string without leading and trailing spaces.

Example code:

```
guess_string = "   1. AAAA: correct 2, in wrong position 0\n" + \
"   2. BBBB: correct 1, in wrong position 0\n" + \
"   3. AABC: correct 1, in wrong position 2"
display_guess_history(guess_string)
print()
display_guess_history("       ")
```

Output:

```
Guesses so far:
   1. AAAA: correct 2, in wrong position 0
   2. BBBB: correct 1, in wrong position 0
   3. AABC: correct 1, in wrong position 2

No guesses have been entered yet.
```

<---------->

8.

Define the `get_random_letters()` function. The first line of this function is:

```
all_letters = "ABCD"
```

This function returns a string comprising 4 random letters obtained from the `all_letters` string above.

Note: the letters in the returned string can be repeated.

VERY IMPORTANT: In order to pass the CodeRunner tests for this function, you need to generate four random numbers which are valid indices of the `"ABCD"` string using the `random.randrange()` function.

Example code:

```
print("1.", get_random_letters())
print("2.", get_random_letters())
```

Two possible outputs:

```
1. CDDC
2. ACAD
```

<---------->

9.

Define the `display_cheating(letters)` function. This function is passed a string parameter: a string of 4 letters. The function prints 3 lines of output:

- Line 1 prints 8 `">"` symbols,
- Line 2 prints the parameter string with single spaces between the letters,
- Line 3 prints 8 `">"` symbols.

Example code:

```
display_cheating("CADB")
print()
display_cheating("DDCA")
```

Output:

```
>>>>>>>>
C A D B
>>>>>>>>

>>>>>>>>
D D C A
>>>>>>>>
```

<----------->

10.

Define the `get_user_guess(guess_number, request)` function which is passed a whole number (`guess_number`) and a string (`request`) as parameters. The function first creates a prompt made up of the request parameter, a single space, followed by the `guess_number` parameter and finally the string `": "`. The function asks the player to enter their guess using the created prompt and returns the string entered by the user in uppercase characters.

Example code:

```
guess = get_user_guess(3,"Enter guess")
print(guess)
guess = get_user_guess(4,"Enter guess number")
print(guess)
```

Output (the user input is shown in bold):

```
Enter guess 3: aabc
AABC
Enter guess number 4: DdAb
DDAB
```

<----------->

11.

Define the `get_number_which_match(hidden_letters, user_letters)` function. This function is passed two string parameters of length 4. The function returns a count of the number of letters in the two strings which are exactly the same and **in the same position** in both parameter strings.

Example code:

```
print(get_number_which_match("CADB", "CDDB"))
print(get_number_which_match("CABC", "CABC"))
```

Output:

```
3
4
```

<---------->

12.

Define the `get_feedback_string(guess_number, user_guess, number_matching, number_correct_not_in_place)` function. This function is passed four parameters: an integer, a string and two integers. The function returns a string made up of:

- the `guess_number` integer,
- followed by `". "`,
- the `user_guess` string,
- followed by `": correct "`,
- followed by the number_matching integer,
- followed by `", in wrong position "`

and finally,

- the `number_correct_not_in_place` integer.

Example code:

```
print(get_feedback_string(3,"ABCC", 2, 1))
print(get_feedback_string(1,"CABC", 0, 3))
```

Output:

```
3. ABCC: correct 2, in wrong position 1
1. CABC: correct 0, in wrong position 3
```

<---------->

13.

Define the `get_win_loss(hidden_letters, user_letters, guess_number)` function. This function is passed three parameters: two strings and an integer. The function returns the integer:

- 1 if the first two parameter strings are identical,
- -1 if the first two parameter strings are not identical and the third parameter integer is 8 or greater,
- 0 in all other cases.

Example code:

```
print(get_win_loss("ABCC", "ABCC", 7))
print(get_win_loss("CABA", "CABC", 8))
print(get_win_loss("CABA", "CBBC", 6))
```

Output:

```
1
-1
0
```

<---------->

14.

Define the `display_has_won_feedback(player_name, number_of_guesses)` function. This function is passed two parameters: the name of the player and the number of guesses made. The function prints 4 lines of output:

- Line 1 is a blank line,
- Line 2 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3,
- Line 3 prints 3 spaces followed by "Congratulations ", the player's name, followed by ", You guessed the letters in ", the number of guesses taken and, finally, either the string " guess." if the second parameter has the value 1 or the " guesses." if the second parameter is not 1,
- Line 4 prints a line of stars. The number of stars is 6 greater than the length of the string in line 3.
-

Example code:

```
display_has_won_feedback("David", 8)
print()
display_has_won_feedback("Lia", 1)
```

Output:

```
************************************************************
   Congratulations David, You guessed the letters in 8 guesses.
************************************************************
```

```
************************************************************
   Congratulations Lia, You guessed the letters in 1 guess.
************************************************************
```

<---------------------------->

15.

Define the `display_has_lost_feedback(player_name, hidden_letters, max_number_of_guesses)` function. This function is passed three parameters: the name of the player, a string of hidden letters and the maximum number of guesses allowed. The function prints 2 lines of output:

- Line 1 prints 3 spaces, the string `"Unfortunately "`, the player's name, the string `", You did not guess the letters within "` followed by the maximum number of guesses allowed and finally the string `" guesses."`.
- Line 2 prints 3 spaces, followed by `"The letters were: "` and finally, the `hidden_letters` parameter.

Example code:

```
display_has_lost_feedback("David", "BCAA", 8)
print()
display_has_lost_feedback("Jia", "BADA", 6)
```

Output:

```
   Unfortunately David, You did not guess the letters within 8 guesses.
   The letters were: BCAA

   Unfortunately Jia, You did not guess the letters within 6 guesses.
   The letters were: BADA
```

<---------------------------->

16.

Define the `get_string_without_matches(letters_to_check, letters)` function which is passed two strings of length 4 as parameters. The function returns a new string where the characters of the first parameter string which match exactly with the characters of the second parameter string are replaced by spaces, e.g. the function looks at each letter of the first parameter, letters, and returns a new string made up of:
blanks for any characters which match and are in the same position in both parameter strings, and the actual character for any characters which do not match or which are not in the same position.

Example code:

```
print("***" + get_string_without_matches("CADB", "CDDB") + "***")
print("***" + get_string_without_matches("AADD", "CCCA") + "***")
print("***" + get_string_without_matches("CBBA", "CCCA") + "***")
```

Output:

```
***  A  ***
***AADD***
***  BB ***
```

<---------->

17.

Define the `get_number_correct_not_in_place(hidden_letters, user_letters)` function. This function is passed two string parameters of length 4. The function compares the two strings and returns the number of letters in the `user_letters` parameter string which are also in the `hidden_letters` parameter string but are not in the same position.

IMPORTANT NOTE: because repeat letters are allowed, you have to be careful to count the letters which are in an incorrect position only once, e.g. if the hidden letters are `"ACCC"` and the user letters are `"BAAA"` the function should return 1 because only one of the "A"s is present in the hidden letters string and is in a different index.

Note: for this question, you may find it useful to use the `get_string_without_matches()` function developed in the previous question. This function can be used (twice) to replace (with a space) the letters which match and are in the same position in both parameter strings, e.g. if the strings are `"ABBC"` and `"BBAC"`, calling the `get_string_without_matches()` twice:

`get_string_without_matches("ADBC", "BDAC")` gives `"A B "`
`get_string_without_matches("BDAC", "ADBC")` gives `"B A "`

This will help you write the code which gives the number of letters which are correct but not in the correct place (i.e. 2 in the example above).

When testing this function in CodeRunner, you can assume that the `get_string_without_matches()` function is available and you should not include it in your answer.

Example code:

```
print(get_number_correct_not_in_place("CACC", "CACC"))
print(get_number_correct_not_in_place("AACC", "CAAA"))
print(get_number_correct_not_in_place("BABC", "BBDB"))
```

Output:

```
0
2
1
```

<---------->

Assignment 2 Section B (8 marks)

Once you have completed the 17 functions from Section A, download the Assignment 2 skeleton program from the CompSci 101 assignments website:

`https://www.cs.auckland.ac.nz/courses/compsci101s2c/assignments/`

Rename the file: "`YourUsernameA2.py`", e.g., `afer023A2.py`. Add your 17 functions from Section A to the program and check that the program executes correctly. Submit your completed Python program using the Assignment Dropbox:

https://adb.auckland.ac.nz/Home/

IMPORTANT: Your program MUST include a docstring at the top of the file (containing your name, your username and a correct description of the program) and your program MUST be named correctly (i.e. as stated above).

Some parts of the skeleton program:

```
import random

def main():
    player_name = get_player_name() # 1
    display_game_welcome(player_name) # 2

    user_choice = -1
    while user_choice != 0:
        user_choice = get_main_menu_selection() # 3
        if user_choice == 1:
            play_one_game(player_name)
        elif user_choice == 2:
            display_game_instructions() # 4

    display_game_end_message(player_name) # 5


def play_a_game(player_name):
    a_win = 1
    a_loss = -1
    win_loss = 0
    max_number_of_guesses = 8
    guess_number = 0
    user_guess_results = "    "
    user_guess = ""

    hidden_letters = get_random_letters() # 6

    user_choice = -1
```

```
    while user_choice != 0:
        user_choice = get_game_menu_selection() # 7
        if user_choice == 2:
            display_guess_history(user_guess_results) # 8
        elif user_choice == 3:
            display_cheating(hidden_letters) # 9
        elif user_choice == 1:
            if user_guess == hidden_letters or guess_number >
                                                        max_number_of_guesses:
                print("This game has finished.")
            else:
                guess_number += 1
                user_guess = get_user_guess(guess_number, "Enter guess") # 10
                number_in_correct_place = get_number_which_match
                                            (hidden_letters, user_guess) # 11
                number_correct_not_in_place =
          get_number_correct_not_in_place(hidden_letters, user_guess) # 16 and 17
                feedback = get_feedback_string(guess_number, user_guess,
                      number_in_correct_place, number_correct_not_in_place) # 12
                print("   Results  ", feedback, "\n")
                user_guess_results += feedback + "\n    "
                win_loss = get_win_loss(hidden_letters, user_guess,
                                                        guess_number) # 13

                if win_loss == a_win:
                    display_has_won_feedback(player_name, guess_number) # 14
                    print()
                    return a_win
                elif win_loss == a_loss:
                    display_has_lost_feedback(player_name,
                                    hidden_letters,max_number_of_guesses) # 15
                    print()
                    return a_loss
```

Example Output

Below is some output produced by the completed program. Your program must execute in the way described and the output should have the same format as the output below. The player input is shown in a larger bold magenta coloured font.

```
Enter name: Adriana

**********************************
   MASTERMIND.  Welcome Adriana.
**********************************

1. Play a game
2. See instructions
0. Exit
   Enter selection: 1
```

```
1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 3

>>>>>>>>
A A D B
>>>>>>>>

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 1: aadd
   Results   1. AADD: correct 3, in wrong position 0

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 2: aadb
   Results   2. AADB: correct 4, in wrong position 0


****************************************************************
   Congratulations Adriana, You guessed the letters in 2 guesses.
****************************************************************

1. Play a game
2. See instructions
0. Exit
   Enter selection: 2


********************************************************************************
   The letters A, B, C and D are randomly combined and hidden from view.
   The player has to guess the four letters in the correct order.
   The maximum number of guesses allowed is eight.
   Note: letters can be repeated.
********************************************************************************
```

```
1. Play a game
2. See instructions
0. Exit
   Enter selection: 1

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 1: aabb
   Results   1. AABB: correct 2, in wrong position 0

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 2: ccbb
   Results   2. CCBB: correct 1, in wrong position 1

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 3: cabd
   Results   3. CABD: correct 1, in wrong position 3

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1

Enter guess 4: adcb
   Results   4. ADCB: correct 2, in wrong position 2

1. Have a guess
2. See list of guesses so far
3. Take a peek
0. Return to the main menu
   Enter selection: 1
```

```
Enter guess 5: dacb
   Results   5. DACB: correct 4, in wrong position 0


**********************************************************************
   Congratulations Adriana, You guessed the letters in 5 guesses.
**********************************************************************

1. Play a game
2. See instructions
0. Exit
   Enter selection: 0


Bye Adriana! Thank you for playing.
```