

Homework 4

Moodle Submission Deadline: 2024/05/08 (Wednesday) 23:59

Problem 1: Minesweeper

[**hw4_1.py**]

You must ever play minesweeper game, right? No matter whether you have ever been familiar with minesweeper, you will be an expert of minesweeper player and programmer. 😊 Yes, in this homework, you are asked to write a program to implement the command-line version minesweeper. The rules of minesweeper can be found via Wikipedia:

https://en.wikipedia.org/wiki/Microsoft_Minesweeper

You can also try to play online minesweeper here:

<http://minesweeperonline.com/#beginner>



Here we provide a list of basic settings of the required text-version minesweeper program.

- (1) The board for minesweeper is 9×9 . Totally there are 81 cells.
- (2) The board needs to be exactly printed using characters '-', '+', and '|'.
- (3) Columns are labeled from a to i, and rows are labeled from 1 to 9.
- (4) Each cell is labeled by the column followed by the row. For example, cell label 'a5' is the cell of column 'a' and row '5'. Each cell label is used to allow users to enter for unfolding the cell, and adding/removing flags.
- (5) The number of mines is 10.
- (6) Use random functions to randomly determine the cells with 10 mines.
- (7) Use characters to display the statuses of cells in the board. The possible characters with their meanings are described as below.
 - ' ': (Empty, 空白): the cell is neither unfolded nor flagged yet.
 - '0': the nearby 8 cells contain no mines
 - '1'/'2'/'3'/'4'...'8': the nearby 8 cells contain 1/2/3/4...8 mines.
 - 'F': the cell is flagged as mine.
- (8) The first unfolding is surely safe. Therefore, at the first round, when the user chooses to unfold a certain cell, you need to ensure such cell is '0', and use random functions to randomly determine the 10 cells with mines in other 80 cells.

Here we also provide two examples of expected minesweeper program. Please visit this video.

https://www.dropbox.com/s/3shl00yw48wnid6/hw4_ex.mp4?dl=0

You are asked to implement the following features in the minesweeper game. Please carefully refer to the description with the corresponding snapshot.

You are asked to write comments to describe the meaning of each part in hw4_1.py.

Sample input & output

- (a) First of all, display the board, present the instructions, and allow users to input. Also display the number of mines left at each round.

```
C:\Python35\workspace\2018計算機概論>python hw4.py
  a  b  c  d  e  f  g  h  i
1 | | | | | | | | |
2 | | | | | | | | |
3 | | | | | | | | |
4 | | | | | | | | |
5 | | | | | | | | |
6 | | | | | | | | |
7 | | | | | | | | |
8 | | | | | | | | |
9 | | | | | | | | |
+---+---+---+---+---+---+---+---+
Enter the column followed by the row (ex: a5). To add or remove a flag,
add 'f' to the cell (ex: a5f).Type 'help' th show this message again
Enter the cell (10 mines left):
```

- (b) Allow users to do possible actions for every cell at each round. When the user does some action (unfold a cell, or add/remove a flag into/from a cell), you program need to instantly display the corresponding status for all cells on the board. At the first round, when the user chooses to unfold a certain cell, you need to ensure such cell is '0', and randomly determine 10 mines in other 80 cells using any random functions. Then you program continuously unfold the all of the connected '0' cells and their nearby cells.

```
Enter the cell (10 mines left): d6
  a  b  c  d  e  f  g  h  i
1 | 0 | 0 | 0 | 1 | | | | |
2 | 0 | 0 | 0 | 1 | 2 | | | |
3 | 2 | 2 | 1 | 0 | 1 | | | |
4 | | | 1 | 0 | 1 | 2 | 2 | 1 |
5 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
7 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |
8 | 1 | 1 | 0 | 1 | | | | |
9 | | 1 | 0 | 1 | | | | |
+---+---+---+---+---+---+---+---+
Enter the cell (10 mines left):
```

(c) Allow users to type 'help' at each round during the game to show the instructions again.

```

Enter the cell (10 mines left): help
  a  b  c  d  e  f  g  h  i
+---+---+---+---+---+---+---+---+
1 | 0 | 0 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
2 | 0 | 0 | 0 | 1 | 2 |  |  |  |  |
+---+---+---+---+---+---+---+---+
3 | 2 | 2 | 1 | 0 | 1 |  |  |  |  |
+---+---+---+---+---+---+---+---+
4 |  |  | 1 | 0 | 1 | 2 | 2 | 2 | 1 |
+---+---+---+---+---+---+---+---+
5 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
+---+---+---+---+---+---+---+---+
7 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |  |
+---+---+---+---+---+---+---+---+
8 | 1 | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
9 |  | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+

Enter the column followed by the row (ex: a5). To add or remove a flag,
add 'f' to the cell (ex: a5f).

Enter the cell (10 mines left):

```

(d) Allow users to do possible actions for every cell at each round. When the user does some action (unfold a cell, or add/remove a flag into/from a cell), you program need to instantly display the corresponding status for all cells on the board.

```

Enter the cell (10 mines left): f3f
  a  b  c  d  e  f  g  h  i
+---+---+---+---+---+---+---+---+
1 | 0 | 0 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
2 | 0 | 0 | 0 | 1 | 2 |  |  |  |  |
+---+---+---+---+---+---+---+---+
3 | 2 | 2 | 1 | 0 | 1 | F |  |  |  |
+---+---+---+---+---+---+---+---+
4 |  |  | 1 | 0 | 1 | 2 | 2 | 2 | 1 |
+---+---+---+---+---+---+---+---+
5 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
+---+---+---+---+---+---+---+---+
7 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |  |
+---+---+---+---+---+---+---+---+
8 | 1 | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
9 |  | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+

Enter the cell (9 mines left): f3f
  a  b  c  d  e  f  g  h  i
+---+---+---+---+---+---+---+---+
1 | 0 | 0 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
2 | 0 | 0 | 0 | 1 | 2 |  |  |  |  |
+---+---+---+---+---+---+---+---+
3 | 2 | 2 | 1 | 0 | 1 |  |  |  |  |
+---+---+---+---+---+---+---+---+
4 |  |  | 1 | 0 | 1 | 2 | 2 | 2 | 1 |
+---+---+---+---+---+---+---+---+
5 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
+---+---+---+---+---+---+---+---+
7 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |  |
+---+---+---+---+---+---+---+---+
8 | 1 | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
9 |  | 1 | 0 | 1 |  |  |  |  |  |
+---+---+---+---+---+---+---+---+

Enter the cell (10 mines left):

```

- (e) Examine whether there are invalid input and show: 'Invalid Cell', illegal unfolding and show: 'That cell is already shown', and illegal flagging and show: 'Cannot put a flag there', or 'There is a flag there'.

<pre> Enter the cell (9 mines left): e7 a b c d e f g h i +---+---+---+---+---+---+---+---+ 1 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 2 0 0 0 1 2 +---+---+---+---+---+---+---+---+ 3 2 2 1 1 0 1 F +---+---+---+---+---+---+---+---+ 4 1 1 0 1 2 2 2 1 +---+---+---+---+---+---+---+---+ 5 2 2 1 1 0 0 0 0 0 0 +---+---+---+---+---+---+---+---+ 6 0 0 0 0 0 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 7 0 0 0 1 1 2 2 1 1 +---+---+---+---+---+---+---+---+ 8 1 1 1 0 1 +---+---+---+---+---+---+---+---+ 9 1 0 1 +---+---+---+---+---+---+---+---+ That cell is already shown Enter the cell (9 mines left): </pre>	<pre> Enter the cell (9 mines left): f3 a b c d e f g h i +---+---+---+---+---+---+---+---+ 1 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 2 0 0 0 1 2 +---+---+---+---+---+---+---+---+ 3 2 2 1 1 0 1 F +---+---+---+---+---+---+---+---+ 4 1 1 0 1 2 2 2 1 +---+---+---+---+---+---+---+---+ 5 2 2 1 1 0 0 0 0 0 0 +---+---+---+---+---+---+---+---+ 6 0 0 0 0 0 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 7 0 0 0 1 1 2 2 1 1 +---+---+---+---+---+---+---+---+ 8 1 1 1 0 1 +---+---+---+---+---+---+---+---+ 9 1 0 1 +---+---+---+---+---+---+---+---+ There is a flag there Enter the cell (9 mines left): </pre>
<pre> Enter the cell (9 mines left): k9 a b c d e f g h i +---+---+---+---+---+---+---+---+ 1 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 2 0 0 0 1 2 +---+---+---+---+---+---+---+---+ 3 2 2 1 1 0 1 F +---+---+---+---+---+---+---+---+ 4 1 1 0 1 2 2 2 1 +---+---+---+---+---+---+---+---+ 5 2 2 1 1 0 0 0 0 0 0 +---+---+---+---+---+---+---+---+ 6 0 0 0 0 0 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 7 0 0 0 1 1 2 2 1 1 +---+---+---+---+---+---+---+---+ 8 1 1 1 0 1 +---+---+---+---+---+---+---+---+ 9 1 0 1 +---+---+---+---+---+---+---+---+ Invalid cell. Enter the column followed by the row (ex: a5). To add or remove a flag, add 'f' to the cell (ex: a5f). Enter the cell (9 mines left): </pre>	<pre> Enter the cell (9 mines left): g4f a b c d e f g h i +---+---+---+---+---+---+---+---+ 1 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 2 0 0 0 1 2 +---+---+---+---+---+---+---+---+ 3 2 2 1 1 0 1 F +---+---+---+---+---+---+---+---+ 4 1 1 0 1 2 2 2 1 +---+---+---+---+---+---+---+---+ 5 2 2 1 1 0 0 0 0 0 0 +---+---+---+---+---+---+---+---+ 6 0 0 0 0 0 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 7 0 0 0 1 1 2 2 1 1 +---+---+---+---+---+---+---+---+ 8 1 1 1 0 1 +---+---+---+---+---+---+---+---+ 9 1 0 1 +---+---+---+---+---+---+---+---+ Cannot put a flag there Enter the cell (9 mines left): </pre>

- (f) Show 'Game over' if the user unfolds a cell with mine, then show all of the true mine cells with 'X', and ask the user to play again.

<pre> Enter the cell (9 mines left): e1 Game Over a b c d e f g h i +---+---+---+---+---+---+---+---+ 1 0 0 0 1 1 X 1 0 0 0 +---+---+---+---+---+---+---+---+ 2 0 0 0 1 2 3 2 2 1 +---+---+---+---+---+---+---+---+ 3 2 2 1 1 0 1 X X 2 X +---+---+---+---+---+---+---+---+ 4 X X 1 1 0 1 2 2 2 1 +---+---+---+---+---+---+---+---+ 5 2 2 1 1 0 0 0 0 0 0 +---+---+---+---+---+---+---+---+ 6 0 0 0 0 0 0 0 0 1 1 +---+---+---+---+---+---+---+---+ 7 0 0 0 1 1 2 2 1 1 X +---+---+---+---+---+---+---+---+ 8 1 1 1 0 1 X X 1 1 1 +---+---+---+---+---+---+---+---+ 9 X 1 1 0 1 2 2 1 0 0 +---+---+---+---+---+---+---+---+ Play again? (y/n): </pre>

- (g) Show 'Win' if all mines are swept, display the time taken using time functions, and ask the user to play again.

```

Enter the cell (1 mines left): d8f
You Win. It took you 1 minutes and 54 seconds.

  a  b  c  d  e  f  g  h  i
+---+---+---+---+---+---+---+---+
1 | 1 | 1 | 1 | X | 2 | 2 | X | 1 | 0 |
+---+---+---+---+---+---+---+---+
2 | X | 1 | 1 | 1 | 2 | X | 2 | 1 | 1 | 0 |
+---+---+---+---+---+---+---+---+
3 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 0 |
+---+---+---+---+---+---+---+---+
4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 1 | 0 |
+---+---+---+---+---+---+---+---+
5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
+---+---+---+---+---+---+---+---+
6 | 0 | 1 | 1 | 2 | 2 | 1 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
7 | 0 | 2 | X | X | 2 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
8 | 0 | 2 | X | X | 3 | 1 | 1 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
9 | 0 | 1 | 2 | 3 | X | 1 | 1 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+

Play again? (y/n):

```

- (h) Re-generate the board, and determine all mine cells using random functions in a new board if the user chooses to play again and select a certain cell.

```

Play again? (y/n): y
  a  b  c  d  e  f  g  h  i
+---+---+---+---+---+---+---+---+
1 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
2 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
3 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
4 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
5 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
6 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
7 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
8 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+
9 |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+

Enter the column followed by the row (ex: a5). To add or remove a flag,
add 'f' to the cell (ex: a5f).Type 'help' th show this message again
Enter the cell (10 mines left):

```

Problem 2: The Blackjack Game

[[hw4_2.py](#)]



One approach to strengthen your programming skills is the game development. In this problem, therefore, you are asked to develop a Blackjack (二十一點) game, which allows a user to play against a computer dealer (莊家). First of all, let me remind you the rules of Blackjack as follows:

- First, both player and dealer receive two cards from a shuffled deck of 52 cards (一副洗好的牌).
- After the first two cards are delivered to dealer and player, the player is asked if they want another card (called "**hitting**" (再抽)), or if he/she are happy with the cards they have already (called "**staying**" (停止)). **The goal is to make the sum of the player's card values as close to 21 as possible, without going over. If the player makes 21 exactly, he/she has Blackjack, which cannot be beat. If the player goes over 21, he/she "busts" (爆掉) and lose the game. The player is allowed to stop hitting at any time point.**
- The number cards (2 through 10) are worth the number displayed, **face cards (JACK, QUEEN, and KING) are worth 10, and an ACE can be worth either 1 or 11.** For example, if the first two cards in the player's hand are a JACK and an ACE, you needs to count the ACE as 11 because $10 + 11 = 21$ and have Blackjack. But if the player has already had the cards worth 18, and decides to hit and gets an ACE, you should count such ACE as 1, because counting it as 11 would put the player at 29 and get busted.
- **Once the player's hand is finished, the dealer will try to do the same things. The dealer must keep hitting until he/she gets to 17. If the dealer gets above 17 without busting, then they can stay.**

Finally, the game is settling by simple rules: (1) if the player has blackjack, he/she wins the game, unless the dealer also has Blackjack, in which case the game is a tie. (2) If the dealer busts and the player does not, the player wins. (3) If the player busts, the dealer wins. (4) If the player and the dealer both do not bust, whoever is closest to 21 wins.

To help you implement the Blackjack game, we divide the Blackjack game into six of the division of labor in terms of the program logic, as described in the following. The program logic forms the game engine, enforcing the rules of the game.

- (a) **Preprocessing.** Create the deck of cards by combing the 13 ranks (ACE, 2, 3, 4, 5, 6, 7, 8, 9, 10,

JACK, QUEEN, and KING) with 4 suits (SPADE, HEART, DIAMOND, and CLUB). There are 52 cards. Then before the game starts, you need to shuffle the cards.

- (b) **Settle the Stage.** Send the player's and dealer's first two cards by random. In other words, four cards that will be owned by the player and the dealer respectively are taken from the 52 cards.
- (c) **Compute the Total Value.** Your program has to reason the total value of cards in the hand. Two things must be done here. First, your program needs to reason the value of a card. For example, ACE = 1 or 11, JACK/QUEEN/KING = 10. Second, by default your program considers ACE as 11. Then your program needs to count up the number of ACEs in the hand, then checks to see that the total value in the hand is higher than 21. If it is, and there is an ACE in the hand, your program needs to consider the ACE as 1, and re-computes the total value. If not, your program obtains the current total value of the player. Otherwise, if there is a second ACE in the hand, your program again considers the second ACE as 1 and checks again. We continue to do this until we have exhausted all the ACEs in the player's hand. Finally, your program obtains the total value in the player's hand.
- (d) **Game Logic.** The next thing we code is the logic of gameplay. Your program has to ask the player whether he/she would like to hit or stay, and continue to ask them until they bust, or they decide to stay. One piece of information that's crucial to the player's decision is their current cards in the hands. Therefore, we need to print the player's cards in the hand and current total value each time before asking for their response and input. As long as the player's hand isn't a bust, we ask for the player's input: hit (再抽) = 1, stay (停止) = 0. If the player wants to hit, we again deliver him/her a new card by randomly drawn a card from the **remaining** cards in the deck, and immediately print the newly drawn card. If they ask to stay, the action of the player will be terminated, and your program moves on to the dealer. In the meanwhile, your program needs to calculate the player's value, and the dealer's current value (on his/her first two cards). If the player's hand isn't a bust, we print the dealer's total value and current cards. Then, while the dealer's hand is worth less than 17, the dealer is made to hit. Each time each dealer hits, you need to print the new drawn card. By design, this loop halts when the dealer exceeds 17.
- (e) **Determine the Winner.** At this point, the game is nearly finished. All that remains is to check the player's total value and the dealer's total value against the list of scoring rules we outlined above. To start, we obtain the label, and number for the score of the dealer's hand. Next, we simply enumerate the possible end states, and ask which of them our game satisfies. Based on the outcome, we print to the screen declaring the winner.
- (f) **Ask the Player to play or not.** If the player does not want to play again, quit the Blackjack game. If the player wants to play again, your program needs to rearrange the deck of cards to let the player play one more time.

You are asked to write comments to describe the meaning of each part in hw4_2.py.

Sample Input and Output

Example 1	Example 2
<pre> c:\Python35-32\workspace>python hw4_p2.py Your current value is 19 with the hand: 10-HEART, 9-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 11 with the hand: 3-CLUB, 8-DIAMOND Dealer draws QUEEN-HEART Dealer's current value is Blackjack! (21) with the hand: 3-CLUB, 8-DIAMOND, QUEEN-HEART *** Dealer wins! *** Want to play again? (y/n): y ----- Your current value is 11 with the hand: 8-CLUB, 3-SPADE Hit or stay? (Hit = 1, Stay = 0): 1 You draw 6-DIAMOND Your current value is 17 with the hand: 8-CLUB, 3-SPADE, 6-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 16 with the hand: JACK-HEART, 6-CLUB Dealer draws 3-HEART Dealer's current value is 19 with the hand: JACK-HEART, 6-CLUB, 3-HEART *** Dealer wins! *** Want to play again? (y/n): y ----- Your current value is 8 with the hand: 5-HEART, 3-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 1 You draw ACE-CLUB Your current value is 19 with the hand: 5-HEART, 3-DIAMOND, ACE-CLUB Hit or stay? (Hit = 1, Stay = 0): 1 You draw 8-SPADE Your current value is 17 with the hand: 5-HEART, 3-DIAMOND, ACE-CLUB, 8-SPADE Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 15 with the hand: KING-HEART, 5-SPADE Dealer draws KING-CLUB Dealer's current value is Bust! (>21) with the hand: KING-HEART, 5-SPADE, KING-CLUB *** You beat the dealer! *** Want to play again? (y/n): n </pre>	<pre> c:\Python35-32\workspace>python hw4_p2.py Your current value is 11 with the hand: 5-DIAMOND, 6-HEART Hit or stay? (Hit = 1, Stay = 0): 1 You draw 10-HEART Your current value is Blackjack! (21) with the hand: 5-DIAMOND, 6-HEART, 10-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 13 with the hand: 9-HEART, 4-SPADE Dealer draws QUEEN-DIAMOND Dealer's current value is Bust! (>21) with the hand: 9-HEART, 4-SPADE, QUEEN-DIAMOND *** You beat the dealer! *** Want to play again? (y/n): y ----- Your current value is 11 with the hand: 9-DIAMOND, 2-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 1 You draw 9-SPADE Your current value is 20 with the hand: 9-DIAMOND, 2-DIAMOND, 9-SPADE Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 16 with the hand: JACK-HEART, 6-CLUB Dealer draws 2-HEART Dealer's current value is 18 with the hand: JACK-HEART, 6-CLUB, 2-HEART *** You beat the dealer! *** Want to play again? (y/n): y ----- Your current value is 20 with the hand: QUEEN-SPADE, 10-SPADE Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 10 with the hand: 8-SPADE, 2-DIAMOND Dealer draws JACK-DIAMOND Dealer's current value is 20 with the hand: 8-SPADE, 2-DIAMOND, JACK-DIAMOND *** You tied the dealer, nobody wins. *** Want to play again? (y/n): n </pre>

Example 3	Example 4
<pre> c:\Python35-32\workspace>python hw4_p2.py Your current value is 8 with the hand: 5-CLUB, 3-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 1 You draw 4-DIAMOND Your current value is 12 with the hand: 5-CLUB, 3-DIAMOND, 4-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 1 You draw 8-DIAMOND Your current value is 20 with the hand: 5-CLUB, 3-DIAMOND, 4-DIAMOND, 8-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 12 with the hand: JACK-CLUB, 2-CLUB Dealer draws QUEEN-HEART Dealer's current value is Bust! (>21) with the hand: JACK-CLUB, 2-CLUB, QUEEN-HEART *** You beat the dealer! *** Want to play again? (y/n): y ----- Your current value is 14 with the hand: 3-HEART, ACE-CLUB Hit or stay? (Hit = 1, Stay = 0): 1 You draw 4-CLUB Your current value is 18 with the hand: 3-HEART, ACE-CLUB, 4-CLUB Hit or stay? (Hit = 1, Stay = 0): 1 You draw ACE-SPADE Your current value is 19 with the hand: 3-HEART, ACE-CLUB, 4-CLUB, ACE-SPADE Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 20 with the hand: 10-SPADE, QUEEN-CLUB *** Dealer wins! *** Want to play again? (y/n): y ----- Your current value is Blackjack! (21) with the hand: ACE-CLUB, QUEEN-DIAMOND Hit or stay? (Hit = 1, Stay = 0): 1 You draw 4-HEART Your current value is 15 with the hand: ACE-CLUB, QUEEN-DIAMOND, 4-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 14 with the hand: 4-SPADE, 10-SPADE Dealer draws JACK-SPADE Dealer's current value is Bust! (>21) with the hand: 4-SPADE, 10-SPADE, JACK-SPADE *** You beat the dealer! *** Want to play again? (y/n): n </pre>	<pre> c:\Python35-32\workspace>python hw4_p2.py Your current value is Blackjack! (21) with the hand: ACE-CLUB, JACK-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 8 with the hand: 2-CLUB, 6-CLUB Dealer draws 4-CLUB Dealer draws 2-DIAMOND Dealer draws 2-SPADE Dealer draws 3-HEART Dealer's current value is 19 with the hand: 2-CLUB, 6-CLUB, 4-CLUB, 2-DIAMOND, 2-SPADE, 3-HEART *** You beat the dealer! *** Want to play again? (y/n): y ----- Your current value is 20 with the hand: JACK-DIAMOND, QUEEN-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 20 with the hand: QUEEN-DIAMOND, KING-DIAMOND *** You tied the dealer, nobody wins. *** Want to play again? (y/n): y ----- Your current value is 12 with the hand: ACE-SPADE, ACE-HEART Hit or stay? (Hit = 1, Stay = 0): 1 You draw 9-HEART Your current value is Blackjack! (21) with the hand: ACE-SPADE, ACE-HEART, 9-HEART Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is 16 with the hand: QUEEN-HEART, 6-DIAMOND Dealer draws 4-SPADE Dealer's current value is 20 with the hand: QUEEN-HEART, 6-DIAMOND, 4-SPADE *** You beat the dealer! *** Want to play again? (y/n): y ----- Your current value is 18 with the hand: 8-HEART, 10-CLUB Hit or stay? (Hit = 1, Stay = 0): 0 Dealer's current value is Blackjack! (21) with the hand: ACE-DIAMOND, JACK-CLUB *** Dealer wins! *** Want to play again? (y/n): y ----- Your current value is 19 with the hand: 8-SPADE, ACE-HEART Hit or stay? (Hit = 1, Stay = 0): 1 You draw 4-SPADE Your current value is 13 with the hand: 8-SPADE, ACE-HEART, 4-SPADE Hit or stay? (Hit = 1, Stay = 0): 1 You draw QUEEN-HEART Your current value is Bust! (>21) with the hand: 8-SPADE, ACE-HEART, 4-SPADE, QUEEN-HEART *** Dealer wins! *** Want to play again? (y/n): n </pre>

Note

This is a homework for each **individual**. 必須於程式檔內註解註明系及姓名學號。

How to Submit Your Homework?

Submission in NCKU Moodle

Before submitting your homework, please zip the files in a zip file, and name the file as “學號_hw4.zip”. For example, if your 學號 is H12345678, then your file name is:

“H12345678_hw4.zip” or “H12345678_hw4.rar”

When you zip your files, please follow the instructions provided by TA’s slides to submit your file using NCKU Moodle platform <http://moodle.ncku.edu.tw> .

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you.