170D WOBC Module G Practical 2A

Play Five-Card Draw

This exam emulates a hand of five-card draw poker.

The play is as follows: Prompt for # players (2-5) - YOU ARE PLAYER_1. The program does not require players provide an ante.

There are three rounds. The first round is a *betting round*. Each player is dealt five cards. PLAYER_1 decides to bet or fold. If PLAYER_1 folds, the game is over; no player wins.

If PLAYER_1 decides to bet, PLAYER_1 enters a bet amount and the program displays PLAYER_1 cards, PLAYER_1 bet amount, the pot amount and the amount PLAYER_1 has left.

After PLAYER_1 bets, the program determines if the other players want to bet or fold. If another player folds, they are taken out of the game. A player that folds is called *inactive*. An inactive player does not participate in any subsequent rounds If the other players decide to bet, they can bet only the amount PLAYER_1 bet (no raises allowed). If the other players bet, the program displays their cards, bet amount, the pot amount and the amount they have left. The first round concludes after every player bets or folds.

The second round is *not a betting round*. The second round is when all players have the option of discarding up to 3 cards and receiving an equal number of replacement cards. A player may elect to receive NO cards (standing pat). No player may fold in the second round. Each player after discarding/replacing cards has a hand of FIVE cards. The second round concludes after all *active* players have discarded and replaced their cards.

The third round is a *betting round*. The third round proceeds exactly as round one *for active players only*. The third round concludes after every player bets or folds.

After the third (betting) round, all active player hands are displayed and ranked.

The program determines the winning hand and prints the player name, the winning hand and the hand's rank.

- Poker hand rankings

Each hand contains five cards and is assigned a **rank** according to the following list of rules:

```
A straight flush is a series of five consecutive cards of the same suit.
The Ace may be the highest and the lowest value.
Example - Ace low: ('5', 'Spades'), ('Ace', 'Spades'), ('2', 'Spades'), ('3',
'Spades'), ('4', 'Spades')
Example - Ace high: ('King', 'Spades'), ('Ace', 'Spades'), ('Queen', 'Spades'),
('Jack', 'Spades'), ('10', 'Spades')
Example - no Ace: ('5', 'Spades'), ('6', 'Spades'), ('2', 'Spades'), ('3', 'Spades'),
('4', 'Spades')
```

```
A four of a kind is a hand with four cards of the same value
Example: ('3', 'Clubs'), ('2', 'Clubs'), ('3', 'Hearts'), ('3', 'Spades'), ('3',
'Diamonds')
```

```
A full house is a hand that contains three cards of one value and two of a different
value
Example: ('Queen', 'Hearts'), ('Queen', 'Diamonds'), ('7', 'Diamonds'), ('Queen',
'Clubs'), ('7', 'Hearts')
```

```
A flush is a hand with all cards of the same suit
Example: ('King', 'Spades'), ('10', 'Spades'), ('2', 'Spades'), ('3', 'Spades'), ('6',
'Spades')
```

```
A straight is a hand with five consecutive values.
As with the straight flush, the Ace may be a high or low value
Example - Ace low: ('3', 'Clubs'), ('2', 'Clubs'), ('4', 'Hearts'), ('5', 'Spades'),
('Ace', 'Spades')
Example - Ace high: ('King', 'Clubs'), ('Queen', 'Clubs'), ('Jack', 'Hearts'), ('10',
'Spades'), ('Ace', 'Spades')
Example: No Ace: ('3', 'Clubs'), ('2', 'Clubs'), ('4', 'Hearts'), ('5', 'Spades'),
('6', 'Spades')
```

```
A three of a kind is a hand with three cards of the same value.
Example: ('3', 'Clubs'), ('2', 'Clubs'), ('3', 'Hearts'), ('3', 'Spades'), ('6',
'Spades')
```

```
A two-pair is a hand with two sets of cards, each set is two cards of the same value.
Example: ('3', 'Clubs'), ('2', 'Clubs'), ('3', 'Hearts'), ('2', 'Spades'), ('10',
'Diamonds')
```

```
A pair is a hand with two cards of the same value.
Example: ('3', 'Clubs'), ('8', 'Clubs'), ('3', 'Hearts'), ('2', 'Spades'), ('10',
'Diamonds')
```

```
Any hand that is not included in the above ranks is a 'nothing' hand.
```

- Determining the winning hand

Each of the **active player's hands are *ranked** according to the rules stated above. If a player folds, they are inactive and their hand is not ranked.

The hand ranks and their order are as follows:

```
Straight flush beats a four of a kind,
four of a kind beats a full house
full house beats a flush
flush beats a straight
straight beats three of a kind
three of a kind beats two pair
two pair beats one pair
one pair beats none of the above hands (a 'nothing' hand)
```

If several players contains hands of the same rank, the program **may select any of these hands as the best hand**. For example, if two players have hands that are both flushes, the program does not determine which flush has the higher card (which is how the best hand would be determined in a real game of Poker).

Starter code

The following files **are coded for you** and included in the `starter/` folder:

- `validator.py` - Code that checks for a valid numeric entry (for a bet) and a valid B(et) or F(old) option.
- `RankTheHand.py` - Code that determines is a hand of five cards is a *pair, two pair, three of a kind, straight, flush, full house, four of a kind or straight flush.*

The work of the lab is in **completing the program** `Play5CardDraw.py`.

Sample output - Player_1 does not fold

Several outputs are shown. The first is when you (Player_1) do not fold:

```
Enter a number between 2 and 5, inclusive, for # players ==> 5
Do you bet or fold (B or F)?==> b
```

```
What's your bet Player_1? (between 1 and 1000)==> 100
Player_1 cards: [('7', 'Hearts'), ('5', 'Clubs'), ('9', 'Spades'), ('7', 'Spades'),
('Queen', 'Spades')]
Player_1 bets 100. The pot is at 100 Player_1 has 900 left
Player_2 folds
Player_3 cards: [('3', 'Diamonds'), ('8', 'Diamonds'), ('Queen', 'Hearts'), ('4',
'Diamonds'), ('9', 'Clubs')]
Player_3 bets 100. The pot is at 200 Player_3 has 900 left
Player_4 cards: [('10', 'Spades'), ('King', 'Clubs'), ('Jack', 'Clubs'), ('5',
'Spades'), ('2', 'Clubs')]
Player_4 bets 100. The pot is at 300 Player_4 has 900 left
Player_5 cards: [('Jack', 'Diamonds'), ('3', 'Hearts'), ('10', 'Clubs'), ('4',
'Spades'), ('4', 'Hearts')]
Player_5 bets 100. The pot is at 400 Player_5 has 900 left

Player Player_1 will exchange 2 cards....
Player Player_1 hand is now [('9', 'Spades'), ('7', 'Spades'), ('Queen', 'Spades'),
('7', 'Diamonds'), ('10', 'Hearts')]

Player Player_3 will exchange 3 cards....
Player Player_3 hand is now [('3', 'Diamonds'), ('Queen', 'Hearts'), ('Queen',
'Clubs'), ('Ace', 'Clubs'), ('6', 'Diamonds')]

Player Player_4 will exchange 2 cards....
Player Player_4 hand is now [('10', 'Spades'), ('King', 'Clubs'), ('Jack', 'Clubs'),
('7', 'Clubs'), ('8', 'Spades')]

Player Player_5 has opted not to exchange any cards (standing pat)
Player Player_5 hand is now [('Jack', 'Diamonds'), ('3', 'Hearts'), ('10', 'Clubs'),
('4', 'Spades'), ('4', 'Hearts')]
Do you bet or fold (B or F)?==> b
What's your bet Player_1? (between 1 and 900)==> 200
Player_1 cards: [('9', 'Spades'), ('7', 'Spades'), ('Queen', 'Spades'), ('7',
'Diamonds'), ('10', 'Hearts')]
Player_1 bets 200. The pot is at 600 Player_1 has 700 left
Player_3 cards: [('3', 'Diamonds'), ('Queen', 'Hearts'), ('Queen', 'Clubs'), ('Ace',
'Clubs'), ('6', 'Diamonds')]
Player_3 bets 200. The pot is at 800 Player_3 has 700 left
Player_4 cards: [('10', 'Spades'), ('King', 'Clubs'), ('Jack', 'Clubs'), ('7',
'Clubs'), ('8', 'Spades')]
Player_4 bets 200. The pot is at 1000 Player_4 has 700 left
Player_5 cards: [('Jack', 'Diamonds'), ('3', 'Hearts'), ('10', 'Clubs'), ('4',
'Spades'), ('4', 'Hearts')]
Player_5 bets 200. The pot is at 1200 Player_5 has 700 left

Active players - display the HAND and RANK for each player
Player Player_1 is holding [('9', 'Spades'), ('7', 'Spades'), ('Queen', 'Spades'),
('7', 'Diamonds'), ('10', 'Hearts')] which is a pair
Player Player_3 is holding [('3', 'Diamonds'), ('Queen', 'Hearts'), ('Queen',
'Clubs'), ('Ace', 'Clubs'), ('6', 'Diamonds')] which is a pair
Player Player_4 is holding [('10', 'Spades'), ('King', 'Clubs'), ('Jack', 'Clubs'),
```

```
('7', 'Clubs'), ('8', 'Spades')] which is a Nothing
Player Player_5 is holding [('Jack', 'Diamonds'), ('3', 'Hearts'), ('10', 'Clubs'),
('4', 'Spades'), ('4', 'Hearts')] which is a pair

The winner is Player_1 which drew a pair with the hand [[('9', 'Spades'), ('7',
'Spades'), ('Queen', 'Spades'), ('7', 'Diamonds'), ('10', 'Hearts')]]
```

- Output details - Player_1 does not fold:

The top two lines:

```
Enter a number between 2 and 5, inclusive, for # players ==> 5
Do you bet or fold (B or F)?==> b
```

is from a call to the functions in `validator.py`. There are no changes required for the file `validator.py`.

The lines:

```
Player Player_1 will exchange 2 cards....
Player Player_1 hand is now [('9', 'Spades'), ('7', 'Spades'), ('Queen', 'Spades'),
('7', 'Diamonds'), ('10', 'Hearts')]
```

are repeated for each active player. Check that the new hand for each player contains different (number of exchanged cards) cards than the originally dealt hand.

If a player bets, the player bet is what Player_1 bets. Players cannot raise; they either *match your bet or fold*.

Once a player folds, their information (hands, how much they bet) is not shown.

Lastly, the program **ranks the hands** and picks a winner. Note in the above example, players Player_1, Player_2, and Player_3 all drew pairs. The program does not determine **which player has the higher pair**. The solution picks out the first player with the pair encountered; Plyer_1 in this case.

- Sample output - Player_1 folds and invalid data is supplied to prompts

Here is output when you (Player_1) folds and invalid data is supplied to prompts:

```
Enter a number between 2 and 5, inclusive, for # players ==> 9
9 not between 2 and 5. Try again
Enter a number between 2 and 5, inclusive, for # players ==> 5
Do you bet or fold (B or F)?==> e
e not one of the valid characters {'B', 'f', 'b', 'F'}. Try again
Do you bet or fold (B or F)?==> b
What's your bet Player_1? (between 1 and 1000)==> 1600
1600 not between 1 and 1000. Try again
What's your bet Player_1? (between 1 and 1000)==> 200
Player_1 cards: [('9', 'Spades'), ('2', 'Clubs'), ('King', 'Spades'), ('5', 'Clubs'),
('6', 'Clubs')]
Player_1 bets 200. The pot is at 200 Player_1 has 800 left
Player_2 cards: [('6', 'Spades'), ('2', 'Spades'), ('8', 'Clubs'), ('Jack',
'Diamonds'), ('Jack', 'Spades')]
Player_2 bets 200. The pot is at 400 Player_2 has 800 left
Player_3 cards: [('Ace', 'Hearts'), ('10', 'Diamonds'), ('6', 'Hearts'), ('6',
'Diamonds'), ('5', 'Hearts')]
Player_3 bets 200. The pot is at 600 Player_3 has 800 left
Player_4 cards: [('8', 'Spades'), ('Jack', 'Clubs'), ('King', 'Hearts'), ('3',
'Hearts'), ('4', 'Diamonds')]
Player_4 bets 200. The pot is at 800 Player_4 has 800 left
Player_5 cards: [('5', 'Spades'), ('8', 'Diamonds'), ('10', 'Spades'), ('4',
'Spades'), ('Ace', 'Clubs')]
Player_5 bets 200. The pot is at 1000 Player_5 has 800 left

Player Player_1 will exchange 2 cards....
Player Player_1 hand is now [('2', 'Clubs'), ('King', 'Spades'), ('6', 'Clubs'), ('5',
'Diamonds'), ('7', 'Hearts')]

Player Player_2 will exchange 3 cards....
Player Player_2 hand is now [('6', 'Spades'), ('Jack', 'Diamonds'), ('3', 'Diamonds'),
('9', 'Clubs'), ('King', 'Clubs')]

Player Player_3 will exchange 1 card....
Player Player_3 hand is now [('Ace', 'Hearts'), ('6', 'Hearts'), ('6', 'Diamonds'),
('5', 'Hearts'), ('King', 'Diamonds')]

Player Player_4 will exchange 1 card....
Player Player_4 hand is now [('Jack', 'Clubs'), ('King', 'Hearts'), ('3', 'Hearts'),
('4', 'Diamonds'), ('4', 'Clubs')]

Player Player_5 will exchange 2 cards....
Player Player_5 hand is now [('5', 'Spades'), ('8', 'Diamonds'), ('Ace', 'Clubs'),
('7', 'Diamonds'), ('7', 'Clubs')]
Do you bet or fold (B or F)?==> f
You folded! You have 800 left
```

- Output details - Player_1 folds and invalid data is supplied to prompts

The responses to the invalid entries for the prompts is self-explanatory. Recall that the code that

checks numerical entries (number of players, bet amount) is contained in `validator.py`. The program calls the functions in `validator.py`, passing appropriate parameters.

When Player_1 folds, the program lists Player_1's remaining amount and stops. Also, when Player_1 folds, the **program does not compute a winner**.

DIcE Rubric

Document	Design Plan	Does the design plan provide a clear general overview of the project?	3%
		Is the design plan easy to understand?	2%
	Project Writeup	Does the writeup document challenges and successes encountered?	2%
		Does the writeup document any lessons learned?	3%
	Writing	Is the project free of grammatical and spelling errors?	4%
		Is non-code formatting consistent?	1%
	Code Formatting	Does `pycodestyle` produce no warnings or errors?	4%
		Are appropriate names chosen to enable code readability?	2%
		Are comments added where appropriate and aid understanding of the logic?	2%
		Is any outside code cited appropriately?	2%
	Total		**25%**
Implement	Version Control	Does the project have the correct name and default branch?	1%
		Were commits broken down into appropriate scopes?	3%
		Are commit messages simple and informative?	1%
	Architecture	Are effective and efficient data structures used?	5%
		Was the code designed and constructed in a modular fashion?	5%
		Were generally sound decisions made with regard to architecture?	15%
	Total		**30%**

Execute	Safety	Does the program avoid crashing or infinite loops, even on invalid input?	10%
	Parsing	Does the program pass `python3 compileall .` with no warnings?	5%
	Requirements	Were all inputs parsed correctly and yield the correct output?	10%
		Are all other requirements met?	20%
	Total		**45%**

Requirements

Area	Requirement
Document	All documentation must be in PDF format unless otherwise specified.
Document	All documentation must be located in a `doc/` folder at the top level of the project.
Document	The design document must be located in `doc/design.pdf`
Document	The project writeup must be located in `doc/writeup.pdf`
Document	All code must conform to PEP8 guidelines. `pycodestyle` must report no warnings or errors, except perhaps for line length.
Implement	Project must be stored in the assigned VCS account, under the project name `fivecarddraw`.
Implement	No third-party header files/libraries may be used unless signed off by the Program Manager or Instructor.
Implement	Project must use appropriate data types or structures.
Implement	Any automated tests and test code must be located in a `test/` folder at the top level.
Execute	Project must be invoked as `Play5CardDraw.py` from the top level directory of the repository.
Execute	Project must run on the class machine.
Execute	Project must not crash or get stuck in an infinite loop, even on invalid input.

Suggested Extra Credit Features

Area	Feature	+
Document	Write a `man(1)` page (not covered in class) to document the program.	+2
Document	Write a Test Plan document (`doc/testplan.pdf`) that covers how to test the program effectively.	+5
Implement	Write automated unit test cases that exercise the code. These should be stored under a `test/` folder at the top level of the project.	+5