COMP 1039

Problem Solving and Programming

# Programming Assignment 2

Prepared by
Jo Zucco

Modified for SAIBT by Graham Winch and Jane Emmett

School of Computer and Information Science
The University of South Australia

## Contents

## INTRODUCTION

This document describes the second assignment for Problem Solving and Programming.

The assignment is intended to provide you with the opportunity to put into practice what you have learnt in the course by applying your knowledge and skills to the implementation of a game called **Blackjack dice** and **a program which will keep a record of the players who achieve high scores while playing Blackjack dice**.

This assignment is an **individual task** that will require an **individual submission**. **You will be required to present your work to your practical supervisor during your practical session held in week 12 of the study period.** Important: You must attend the practical session that you have been attending all study period in order to have your assignment marked.

This document is a kind of specification of the required end product that will be generated by implementing the assignment. Like many specifications, it is written in English and hence will contain some imperfectly specified parts. Please make sure you seek clarification if you are not clear on any aspect of this assignment.

## ASSIGNMENT OVERVIEW

There are **two (related) parts** to this assignment:

### Part I: Implementation of the game Blackjack dice

You are required to write a Python program that allows a player to play **a game of Blackjack dice** against the computer. The program will allow a player to play as many games of Blackjack dice as they wish. Please ensure that you read sections titled 'Part I specification' below for further details.

### Part II: High scores player information

You are required to write a Python program that will keep a record of the players who achieve high scores while playing Blackjack. The program will allow players to play Blackjack against the computer and will maintain information on high scoring players (using Lists). Player information will be stored in a text file that will be read in when the program commences. Once the initial high scores data has been read in from the file, the program should allow the user to interactively query and manipulate the high scores information as well as play Blackjack against the computer (using the program you developed in Part I). Please ensure that you read sections titled 'Part II specification' below for further details.

**Note: Although this assignment has two related parts, if you are not able to complete Part I of the assignment, you can still implement Part II. Likewise, if you are not able to complete Part II of the assignment, you can still implement Part I.**

***Please ensure that you read sections titled 'Part I Specification' and 'Part II Specification' below for further details.***

## GRADUATE QUALITIES

By undertaking this assessment, you will progress in developing the qualities of a University of South Australia graduate.

The Graduate qualities being assessed by this assignment are:

- The ability to demonstrate and apply a body of knowledge (GQ1) gained from the lectures, workshops, practicals and readings. This is demonstrated in your ability to apply problem solving and programming theory to a practical situation.

- The development of skills required for lifelong learning (GQ2), by searching for information and learning to use and understand the resources provided (Python standard library, lectures, workshops, practical exercises, etc); in order to complete a programming exercise.

- The ability to effectively problem solve (GQ3) using Python to complete the programming problem. Effective problem solving is demonstrated by the ability to understand what is required, utilise the relevant information from lectures, workshops and practical work, write Python code, and evaluate the effectiveness of the code by testing it.

- The ability to work autonomously (GQ4) in order to complete the task.

- The use of communication skills (GQ6) by producing code that has been properly formatted; and writing adequate, concise and clear comments.

## PART I SPECIFICATION – BLACKJACK DICE

You are required to write a Python program called `yourID_blackjack.py` that allows a player to play **a game of Blackjack dice** against a computer opponent. The program will allow a player to play as many games of Blackjack dice as they wish.

### Blackjack Dice Rules (Two-Dice)

Blackjack is a popular card game where the goal is to beat the dealer by having the higher hand which is not more than 21.

**For this assignment:**

We will change this card game into a dice game. Instead of a deck of 52 cards, we will use **two 10 sided dice** in order to have the higher total, or "hand" which is not more than 21.

- You are to simulate two 10 sided dice in order to play the game of Blackjack dice
  (the face value on the die are 1 - 10).
- Blackjack hands are scored by their point total.
- The hand with the highest total wins, as long as it isn't more than 21.
- A hand with a higher total than 21 is said to 'bust'.
- Rolling a one (1) counts as eleven (11) **unless** this would cause the player to bust, in which case it is worth one (1).
- All other values are worth their face value rolled.

**Game play:**

1. The dealer (in this case the computer) rolls one 10-sided dice (the face value on the die are 1 – 10). The dice value is displayed to the screen.
2. The player then plays out his/her hand and rolls two 10-sided dice (the face value on the die are 1 – 10).
3. During the player's turn, the player is faced with two options, either:
   a. Hit:
      After the initial roll of two dice, a player may choose to roll an additional die as many times as he/she wishes, adding the resulting number from each roll to his/her total.
   b. Stand:
      Do not roll again. The player's turn is over.
4. The player continues to roll one die until, the player chooses to stand, or the player busts (that is, exceeds a total of 21). The player's turn is over after deciding to stand or if he/she busts.
5. Once the player has finished, the dealer (computer) plays out the dealer's hand.
   **a.** The dealer (computer) rolls the second 10-sided dice and displays his/her initial hand to the screen.
   **b.** Following this, dealer repeatedly rolls one die u**ntil, the dealer's total is 17 or greater, or the dealer busts (that is, exceeds a total of 21)**.
   Note: the dealer **always** plays his/her hand regardless of what happens with the player's hand.
6. A winner is then determined based on the player and dealer's hand values according to the Blackjack rules listed below and displayed to the screen.

**Rules:**

- If the player busts, he/she loses even if the dealer also busts.
- If the player and the dealer have the same point value, it is called a 'push' and neither win the hand (draw).
- A die roll of one (1) counts as eleven (11) unless this would cause the player/dealer to bust, in which case it is worth one (1).
- The dealer must hit until he or she has a point value of 17 or greater.
- The player cannot stand on a point value less than 15.
- The player with the higher hand that does not exceed 21 wins the game!

## PRACTICAL REQUIREMENTS (PART I)

It is recommended that you develop this part of the assignment in the suggested stages.

**It is expected that your solution will include the use of:**

- Your solution in a file (module) called `yourID_`*`blackjack`*`.py`.
- Appropriate and well constructed `while` and/or `for` loops. (Marks will be lost if you use `break` statements in order to exit from loops).
- Appropriate `if, if-else, if-elif-else` statements (as necessary).
- The use of the `random.randint(1,10)` function in order to simulate the roll of the dice.
- Appropriate constants, i.e. PLAYER_MIN_SCORE = 15, DEALER_MIN_SCORE = 17, etc.
- The following functions:
    1. get_choice(): prompts for, reads and validates the user's choice of h or s (hit or stand).
        - Accepts no parameters and returns the choice entered by the user.
    2. deal_players_hand(): plays out the player's hand.
        - Accepts no parameters and returns the value (total) of the player's hand.
        - Calls function `get_choice().`
        - Rolls two 10-sided dice (value 1 – 10) and displays the initial hand to the screen
        - The player is faced with two options, either:
            - Hit: After the initial roll of two dice, a player may choose to roll an additional die as many times as he/she wishes, adding the number from each roll to his/her total.
            - Stand: Do not roll again.
        - The function repeatedly rolls one die until, the player chooses to stand, or the player busts (more than 21) – then the player's turn is over
    3. deal_dealers_hand(die_value): plays out the dealer's hand.
        - The function accepts the initial die roll value as a parameter and returns the value (total) of the dealer's hand.
        - The dealer (computer) rolls the second 10-sided dice and displays his/her initial hand to the screen.
        - The dealer repeatedly rolls one die until, the dealer's total is 17 or greater, or the dealer busts (more than 21).
        - Note: the dealer **always** plays his/her hand regardless of what happens with the player's hand.

4. determine_winner(): determines the winner based on the player and dealer's hand values and according to the Blackjack rules.
    - Accepts the player's and the dealer's total hand value as parameters and returns result (see below).
    - The hand values along with the winner is displayed to the screen (as seen in the sample output). play_blackjack()
    - The function returns the zero (0) if the dealer wins, one (1) if the game is a draw (push) and three (3) if the player wins.

5. `play_blackjack():` allows a player to play Blackjack against the computer until he/she does not wish to continue playing
    - Returns the total score resulting from all games played and does not accept any parameters.
    - Calls: `deal_players_hand(), deal_dealers_hand(), determine_winner()`
    - After every game, the player's total game score is updated.
        - 3 points are awarded if the player wins
        - 0 points are awarded if the player loses and
        - 1 point is awarded if the player draws with the dealer.
    - This function keeps a cumulative total of the score returned from function `determine_winner().`
    - Prompts if wants to play 'Play again [y|n]?'

- Output that **strictly** adheres to the assignment specifications. If you are not sure about these details, you should check with the 'Sample Output – Part I' provided at the end of this document.

- Good programming practice:
    1. Consistent commenting, layout and indentation. You are to provide comments to describe: your details, program description, all variable definitions, and every significant section of code.
    2. Meaningful variable names.

NOTE: **Do not** use `break`, or `continue` statements (or any other technique to break out of loops) in your solution – doing so will result in a significant mark deduction.

**PLEASE NOTE: You are reminded that you should ensure that all input and output conform to the specifications listed here; if you are not sure about these details you should check with the sample output provided at the end of this document or post a message to the discussion forum in order to seek clarification.** Please ensure that you use Python 3.4.3 (or later) in order to complete your assignments.

## STAGES (PART I)

It is recommended that you develop this part of the assignment in the suggested stages. Many problems in later stages are due to errors in early stages. **Make sure you have finished and thoroughly tested each stage before continuing.**

The following stages of development are recommended:

**Stage 1:** implement part of the `play_blackjack()` function.

Create a file called `yourID_blackjack.py` and enter the following code which defines and calls function `play_blackjack()`:

```
def play_blackjack():
    total_score = 0

    print("In function play_blackjack!")

    return total_score


# Call function play_blackjack() - plays blackJack against the computer
score = play_blackjack()

# Display score to the screen
print('\n\nYour score is:', score)
```

**Stage 2:** Use the `random.randint()` function to simulate the dealer rolling one 10-sided dice (1 – 10).

Display the value of the die as seen below. Don't forget to import the random module.

*Sample output:*
```
| Dealer hand: 8
```

This code should be placed in the `play_blackjack()` function.

**Stage 3: I**mplement the function that plays out the player's hand (called `deal_players_hand()`).

This function does not accept any parameters, and returns the player's hand total. This function is called from function `play_blackjack()`.

In function `deal_players_hand()`:

- The player rolls two 10-sided dice (the face value on the die are 1 – 10). Use the `random.randint()` function to create a simulation of rolling the player's hand. Display the value of die 1, die 2 and the total roll value as seen below.

  *Sample output:*
  ```
  | Player hand:  6 + 7 = 13
  ```

- Include code that checks the player's hand for a die roll of one (1):
    - o If 1 is rolled, change the value to 11.
    - o If the values of both dice are 1, only change one of the die to 11 (this is to avoid the player busting on the first roll).

  ***Note:*** *This should NOT be achieved by changing the arguments to the random.randint() function*

- Now let's include code to play out the rest of the player's hand.
    - o Prompt for and read whether the player would like to hit ('h') or stand ('s').
    - o Implement a loop that continues to roll a die until the user enters stand ('s').
    - o Display the value of die 1, die 2 and the total roll value as seen below.

    Note: when the player chooses to 'hit' the new dice value is added to the previous total, to make the new player total

  *Sample output:*
```
| Dealer hand: 7
| Player hand: 6 + 2 = 8
|
| Please choose to hit or stand (h/s): h
| Player hand: 8 + 2 = 10
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 8 = 18
| Please choose to hit or stand (h/s): h
| Player hand: 18 + 5 = 23
| Please choose to hit or stand (h/s): h
| Player hand: 23 + 8 = 31
| Please choose to hit or stand (h/s): h
| Player hand: 31 + 7 = 38
| Please choose to hit or stand (h/s): s
```

- It does not make sense to continue rolling when the player busts (exceeds 21). Modify the loop developed in the previous stage so that it also stops looping if the player busts (**more** than 21).

  *Sample output:*
```
| Dealer hand: 6
| Player hand: 7 + 3 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 6 = 16
| Please choose to hit or stand (h/s): h
| Player hand: 16 + 8 = 24
| PLAYER BUSTS!
```

- Add code to ensure a die roll of one (1) counts as eleven (11) - unless this will causes the player/dealer to bust. In this case it is worth one (1).

- Add code which ensures that the player cannot stand ('s') on a value less than 15. Notice that the player automatically rolls again when this happens (see sample output below).

  *Sample output:*
```
| Dealer hand: 3
| Player hand: 10 + 3 = 13
|
| Please choose to hit or stand (h/s): s
|
| Cannot stand on value less than 15!
|
| Player hand: 13 + 2 = 15
| Please choose to hit or stand (h/s): s
```

- Return the player's total hand value from the function (you will need to use a return statement to do this).

**Stage 4:** Implement the function that plays out the dealer's hand (called `deal_dealers_hand()`).

This function accepts a die value as a parameter (the initial roll), and returns the dealer's hand total. This function is called from function `play_blackjack()`.

- The dealer (computer) rolls the second 10-sided dice and displays his/her initial hand to the screen. Use the `random.randint()` function to create a simulation of rolling the dealer’s (computer’s) hand. Display the value of die 1 (passed in as a parameter), die 2 and the total roll value as seen below.

  *Sample output:*
  ```
  | Dealer hand: 8 + 7 = 15
  ```

- Include code that checks the dealer’s hand for a die roll of one (1). If a die roll of one (1) is found, include code that changes the value to eleven (11). If the values of both dice rolled are one, then only one of the die should be changed to eleven (this is to avoid the dealer busting on the first roll).

- Now let’s add a loop that simulates the computer’s turn. That is, plays out the dealer’s (computer’s) hand. The dealer must continue to roll until they have a point value of 17 or greater.

  *Sample output 1:*
  ```
  | Dealer hand: 3
  | Player hand: 4 + 10 = 14
  |
  | Please choose to hit or stand (h/s): h
  | Player hand: 14 + 3 = 17
  | Please choose to hit or stand (h/s): s
  |
  | Dealer hand: 3 + 2 = 5
  | Dealer hand: 5 + 7 = 12
  | Dealer hand: 12 + 5 = 17
  ```

  *Sample output 2:*
  ```
  | Dealer hand: 4
  | Player hand: 2 + 10 = 12
  |
  | Please choose to hit or stand (h/s): h
  | Player hand: 12 + 4 = 16
  | Please choose to hit or stand (h/s): h
  | Player hand: 16 + 4 = 20
  | Please choose to hit or stand (h/s): s
  |
  | Dealer hand: 4 + 3 = 7
  | Dealer hand: 7 + 7 = 14
  | Dealer hand: 14 + 9 = 23
  | DEALER BUSTS!
  ```

- Add code that ensures a die roll of one (1) counts as eleven (11) unless this would cause the player/dealer to bust, in which case it is worth one (1).

- Return the dealer's total hand value from the function (you will need to use a return statement to do this).

**Stage 5:** Implement the function `determine_winner()`to determine the winner and display the result to the screen.

- The function `determine_winner()` accepts the player and dealer's total hand values and determines the winner adhering to the Blackjack rules.
- If the dealer wins, return 0 from the function, if the player wins, the function should return the value 3, if it's a draw (push), the function should return the value 1.
- This function is called from function `play_blackjack()`.

*Sample output 1:*
```
| Dealer hand: 11
| Player hand: 5 + 5 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 8 = 18
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 11 + 7 = 18
|
| Dealer = 18 Player = 18 *** Push - no winners. ***
```

*Sample output 2:*
```
| Dealer hand: 8
| Player hand: 6 + 4 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 3 = 13
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 6 = 19
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 8 + 6 = 14
| Dealer hand: 14 + 6 = 20
|
| Dealer = 20 Player = 19 *** Dealer Wins! ***
```

*Sample output 3:*
```
| Dealer hand: 10
| Player hand: 3 + 8 = 11
|
| Please choose to hit or stand (h/s): h
| Player hand: 11 + 8 = 19
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 4 = 14
| Dealer hand: 14 + 4 = 18
|
| Dealer = 18 Player = 19 *** Player Wins! ***
```

**Stage 6:** In `play_blackjack()`,add a loop that asks the player whether they wish to play again ['y'|'n'].

- This function allows a player to play Blackjack against the computer until he/she does not wish to continue playing (enters 'n' when prompted to 'Play again [y|n]?').
- After every game, the player's total game score is updated: 3 points are awarded if the player wins, 0 points are awarded if the player loses and 1 point is awarded if the player draws with the dealer.
- Update the total_score variable to keep a cumulative total of the score returned from function `determine_winner()`.
- Function `play_blackjack()` returns the total score resulting from all games played.

**Stage 7:** Implement function `get_choice()`

- This function should prompts for and reads whether the player would like to ‘h’ or ‘s’ (hit or stand).
- The function `get_choice()` does not accept any parameters and returns the user’s choice (of ‘h’ or ‘s’).
- It should validate the input.

**Stage 8:** Add code to validate all user input. Hint: use a while loop to validate input.

*Sample output:*
```
| Dealer hand: 3
| Player hand: 9 + 5 = 14
|
| Please choose to hit or stand (h/s): r
| Please choose to hit or stand (h/s): h
| Player hand: 14 + 2 = 16
|
| Please choose to hit or stand (h/s): p
| Please choose to hit or stand (h/s): q
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 9 = 12
| Dealer hand: 12 + 4 = 16
| Dealer hand: 16 + 4 = 20
|
| Dealer = 20 Player = 16 *** Dealer Wins! ***


Your score: 0 - Play again [y|n]? p
Please enter y or n. Play again [y|n]? z
Please enter y or n. Play again [y|n]? t
Please enter y or n. Play again [y|n]? n
```

**Stage 9**

**Finally**, check the sample output (see section titled ‘Sample Output – Part I’ towards the end of this document) and if necessary, modify your code so that:

- The output produced by your program **EXACTLY** adheres to the sample output provided.
- Your program behaves as described in these specs **and** the sample output provided.

## PART II SPECIFICATION – HIGH SCORES INFORMATION

Write a **menu driven program** called `yourId_highscores.py` that will allow the user to enter commands and process these commands until the quit command is entered. The program will keep a record of the players who achieve high scores while playing Blackjack. The program will allow players to play Blackjack against the computer and will maintain information on high scoring players (using Lists). Player information will be stored in a text file that will be read in when the program commences. Once the initial high scores data has been read in from the file, the program should allow the user to interactively query and manipulate the high scores information as well as play Blackjack against the computer (using the program you developed in Part I).

**Note: Although this assignment has two related parts, if you are not able to complete Part I of the assignment (or don't intend to), you can still implement Part II. Likewise, if you are not able to complete Part II of the assignment (or don't intend to), you can still implement Part I. Refer to stage two (2) in this section (Part II) for information on how to handle an incomplete Part I.**

### Input

When your program begins, it will read in player information from a file called `highscores.txt`. This is a text file that stores high score information relating to players.

You may assume that all data is in the correct format. The name (firstname surname) of the player and his/her high score is stored on one line and are separated by the space character as seen in Figure 1 below.

After the program has stored the data (using two Lists, one List to store the player names and the other List to store the high scores), it will enter interactive mode as described in the following section.

```
Tony Stark 10
Fox Mulder 8
Phil Dunphy 6
Buster Bluth 1
```

Figure 1: *Player information file format (`highscores.txt`).*

### Interactive Mode

Your program should enter an interactive mode after the player high score information has been read from the file. The program will allow the user to enter commands and process these commands until the quit command is entered. The following commands should be allowed:

1. **Scores:**
   Outputs the contents of the high scores lists (as shown in the section titled Screen Format)

2. **Search:**
   Prompts for and reads the player's name and searches for the player in the player names (high scores) list.

   - If the player is found, the player's name, and high score, are displayed to the screen:

     ```
     Phil Dunphy has a high score of 6
     ```

   - If the player is not found, an error message stating the player has not been found is displayed.

     ```
     Bruce Banner does not have a high scores entry.
     ```

3. **Play:**
   Allows a player to play Blackjack against the computer until he/she does not wish to continue playing (enters ‘n’ when prompted to ‘Play again [y|n]?’).

   - After every game, the player’s total (cumulative) score is updated. 3 points are awarded if the player wins, 0 points are awarded if the player loses and 1 point is awarded if the player draws with the dealer.
   - Once the player enters ‘n’ (i.e. does not wish to continue playing), it is determined whether the player has qualified to be stored in the high scores list.
   - If the player is to be added to the high scores list:
     - The player’s name is prompted for and read.
     - The player’s information is then added to the player names and high scores lists.
     - Player information must be added to the high scores lists maintaining the sorted order of the lists (descending order of total score).
     - Where two players have the same total score, the new player should appear first. (*Hint: to add a record, shift all elements one position down the list*).
     - A message should be displayed to the screen indicating they have been added to the high scores lists (player names and high scores).
   - If the player does not qualify for the high scores lists, a message should be displayed to the screen.

4. **Quit:**
   Causes the program to quit, outputting the contents of the high scores lists (player names and player scores) to a file called `new_scores.txt.`
   The format of this file should exactly match that of the input file.

**Note:**

Appropriate messages should also be displayed to indicate whether a command has been successfully/uncessfully completed.

Please refer to the sample output (at the end of this handout) to ensure that your program is behaving correctly and that you have the correct output messages.

Each time your program prompts for a command, it should display the list of available commands. See the sample output (at the end of this handout) to ensure that you have the output format correct.

For example:

```
Please enter command [scores, search, play, quit]:
```

Menu input should be validated with an appropriate message being displayed if incorrect input is entered.

**Screen Format**

When the high scores are displayed (scores command) or written to a file () they should be printed in the following fomat:

The maximum number of high scores is 5. If less than 5 scores are recorded, it should display ????????

```
*****************************************************
******************   Blackjack    *****************
******************  HIGH SCORES   *****************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Buster Bluth                     1           *
*---------------------------------------------------*
*      ????????                         0           *
*---------------------------------------------------*
*****************************************************
```

## PRACTICAL REQUIREMENTS (PART II)

It is recommended that you develop this part of the assignment in the suggested stages. Each stage is worth a portion of the marks.

**It is expected that your solution will include the use of:**

- Your solution in a file called `yourId_highscores.py`.
- Appropriate and well constructed `while` and/or `for` loops without `break` statements to exit from loops
- Appropriate `if, if-else, if-elif-else` statements (as necessary).
- Appropriate constants, i.e. MAX_SCORES = 5
- The following functions:
    1. read_file(filename, player_names, player_scores):
       reads the contents of a file into player_name and player_scores lists
        - o filename, player_name and player_score lists are passed as a parameters and returns nothing
        - o The player_names and player_scores lists should not exceed 5 elements. You must use a loop in your solution. **You may use String and/or List methods in this function only.**
    2. write_to_file(filename, player_names, player_scores):
       output the contents of the player_names and player_scores lists to a file in the same format as the input file.
        - o filename (of the output file), player_names and player_scores lists are passed as parameters and returns nothing
        - o The file will need to be opened before and closed once all writing has been done. You must use a loop in your solution.
    3. display_high_scores(player_names, player_scores): output the contents of the lists to the screen
        - o player_names and player_scores list passed as parameters and returns nothing
        - o Output the contents of the lists to the screen in the format specified in the assignment specifications under the section - 'Screen Format'.
        - o You must use a loop in your solution.
    4. find_player(player_names, name): find the position of the player in the high scores
        - o player's name, player_names list are passed as parameters
        - o Returns the position (index) of the player found in the player_names list
        - o If more than one player exists with the same name, return the position of the player who has the highest score (i.e. first match found).
        - o If the player is not found, the function returns -1.
        - o You **must** use a loop in your solution. You **must not** use list methods in your solution.
    5. is_high_score(player_scores, new_score): determines if player is to be added to the high scores
        - o Takes a player's score, and player_scores list as parameters
        - o Returns the insertion position is returned from this function (if the player is to be added)
        - o If the player does not qualify in the high scores lists, the function returns -1.
        - o You must use a loop in your solution.

6. add_player(player_names, player_scores, new_name, new_score, insert_position, num_players): adds a player to the high scores lists at the insert position
    - o player_names and player_scores lists, the player's name and score, the number of high scores (stored in the lists) and the position to insert (the player into the lists) are parameters
    - o The function returns the number of high scores stored in the lists.
    - o The high scores lists (player_names and player_scores) must be maintained in descending order of total score.
    - o Where two players have the same total score, the new player being added should appear first. (Hint: when inserting a player, simply shift all elements one position down the lists.
    - o You must make sure you are not exceeding list bounds and that your lists do not contain more than five elements).
    - o You **must** use a loop in your solution. You **must not** use list methods in your solution.

- Output that **strictly** adheres to the assignment specifications. If you are not sure about these details, you should check with the 'Sample Output – Part II' provided at the end of this document.
- Good programming practice:
    1. Consistent commenting, layout and indentation. You are to provide comments to describe: your details, program description, all variable definitions, and every significant section of code.
    2. Meaningful variable names.

- Your solutions **MAY** make use of the following:
    1. Built-in functions `int()`, `input()`, `print()`, `range()`, `open()`, `len()`, `format()` and `str()`.
    2. Concatenation (+) operator to create/build new strings.
    3. Access the individual elements in a string with an index (one element only). i.e. `string_name[index]`.
    4. Access the individual elements in a list with an index (one element only). i.e. `list_name[index]`.
    5. You **must** use a loop in each function.
- Your solutions **MUST NOT** use:
    1. Built-in functions (**can** use `int()`, `input()`, `print()`, `range()`, `open()`, `len()`, `format()` and `str()` functions).
    2. Slice expressions to select a range of elements from a string or list. i.e. `name[start:end]`.
    3. String or list methods (i.e. *`list_name`*`.append(item)`, etc).
    4. Global variables as described in the second lecture about functions.

NOTE: **Do not** use `break`, or `continue` statements (or any other technique to break out of loops) in your solution – doing so will result in a significant mark deduction.

**PLEASE NOTE: You are reminded that you should ensure that all input and output conform to the specifications listed here; if you are not sure about these details you should check with the sample output provided at the end of this document or post a message to the discussion forum in order to seek clarification.**

Please ensure that you use Python 3.4.3 (or later) in order to complete your assignments.

## STAGES (PART II)

It is recommended that you develop this part of the assignment in the suggested stages. Many problems in later stages are due to errors in early stages. **Make sure you have finished and thoroughly tested each stage before continuing.**

The following stages of development are recommended:

**Stage 1**
To begin, download the provided file (available on the course website called highscores.py) and name it `yourId_highscores.py`. Define two lists (with five elements each) in order to store high score information, that is, player names and their corresponding scores. For example:

```
player_names = ["","","","",""]
player_scores = [0,0,0,0,0]
```

**Stage 2**
Call function `play_blackjack()` (created in Part I of this assignment). If you have not yet implemented (or don't intend on implementing) Part I of the assignment, you should at least implement Stage 1 (of Part I) so that you can call the function as described below.

**Note: file `yourID_blackjack.py` should be placed in the same directory as *`yourId`*`_highscores.py` file.**

In the file called `yourID_blackjack.py` that you created in Part I, comment out the code below (and any other related code outside the functions) and save the file.

```
          :
          :
          :

    # Call function play_blackjack() - plays blackJack against the computer
    ### score = play_blackjack()

    # Display score to the screen
    ### print('\n\nYour score is:', score)
```

Okay, back to the file called *`yourID`*`_highscores.py`. Import the blackjack module and call function `play_blackjack()` as seen below. You may wish to read the material on modules posted on the course website (under the assessment tab).

```
    import yourID_blackjack


    score = yourID_blackjack.play_blackjack()
```

Make sure the program runs correctly. Once you have that working, back up your program. *Note: When developing software, you should always have fixed points in your development where you know your software is bug free and runs correctly.*

**Stage 3**
Write the code for function `read_file()` and `display_high_scores()` according to the function requirements (specified in practical requirements). Add code to call these two functions to ensure they are working correctly.

**Stage 4**

Now that you know the information is being correctly stored in your high scores lists (player_names and player_scores lists), write the code for function `write_to_file()`. Add code to call this function to ensure it is working correctly.

**Stage 5: Implement the interactive menu, i.e. to prompt for and read menu commands.**

Set up a loop to obtain and process commands. Test to ensure that this is working correctly before moving onto the next stage. You do not need to call any functions at this point, you may simply display an appropriate message to the screen, for example:

*Sample output:*

```
Please enter command [scores, search, play, quit]: roger

Not a valid command - please try again.

Please enter command [scores, search, play, quit]: scores

In scores command


Please enter command [scores, search, play, quit]: search

In search command


Please enter command [scores, search, play, quit]: play

In play command


Please enter command [scores, search, play, quit]: quit
```

Menu input should be validated with an appropriate message being displayed if incorrect input is entered by the user.

**Stage 6: Implement one command at a time.**

For each command you will need to complete and call the required functions. Test to ensure the command and associated functions are working correctly before starting the next command.

**<u>See the program specification for detailed requirements for each command and function.</u>**

The following order is suggested:

- `quit` – Write scores to file (call `write_to_file()` function) and end the program
- `scores` – Display the scores in screen format (call display_high_scores() function)
- `search` – Ask user to enter name and display the score for player or error message if not found (call `find_player()` function).
- `play`– plays blackjack (calls `playgame()` function) until user stops and calculates score

  Checks if score is a high score (call `is_high_score()`function, and (if makes it) adds player to high scores (call `add_player()` function)

**Stage 7**

Ensure that you have validated all user input with an appropriate message being displayed for incorrect input entered by the user. Add code to validate all user input. Hint: use a while loop to validate input.

**Stage 8**

Finally, check the sample output (see section titled ‘Sample Output – Part II’ towards the end of this document) and if necessary, modify your code so that:

- The output produced by your program **EXACTLY** adheres to the sample output provided.
- Your program behaves as described in these specs and the sample output provided.

## SUBMISSION DETAILS

**You are required to demonstrate your assignment to your practical supervisor during your week 12 class for marking. The supervisor will mark your work using the marking criteria included in this document. You MUST attend the practical session that you have been attending all study period in order to have your assignment marked.** You are **also** required to submit an electronic copy of your program via Moodle. Assignments submitted to Moodle, but not demonstrated during your allocated practical session, will NOT be marked. Likewise, assignments that have been demonstrated during the practical session, but have not been submitted via Moodle, will NOT be marked. Assignments are submitted to Moodle in order to check for plagiarism.

Ensure that your files are named correctly (as per instructions outlined in this document).

Ensure that the following files are included in your submission:

- `yourID_blackjack.py`
- `yourId_highscores.py`
- `high_scores.txt`

For example:

- `ABCD1601_blackjack.py`
- `ABCD1601_highscores.py`
- `high_scores.txt`

All files that you submit must include the following comments.

```
#
# File: fileName.py
# Author: your name
# SAIBT Id: your saibt id
# Description: Assignment 2 – place assignment description here…
# This is my own work as defined by SAIBT's
# Academic Misconduct policy.
#
```

Assignments that do not contain these details may not be marked.

**It is expected that students will make copies of all assignments and be able to provide these if required.**

## EXTENSIONS AND LATE SUBMISSIONS

There will be **no** extensions/late submissions for this course without one of the following exceptions:

1. A medical certificate is provided that has the timing and duration of the illness and an opinion on how much the student's ability to perform has been compromised by the illness. **Please note** if this information is not provided the medical certificate WILL NOT BE ACCEPTED. Late assessment items will not be accepted unless a medical certificate is presented to the Course Coordinator. The certificate must be produced as soon as possible and must cover the dates during which the assessment was to be attempted. In the case where you have a valid medical certificate, the due date will be extended by the number of days stated on the certificate up to five working days.

2. A SAIBT counsellor contacts the Course Coordinator on your behalf requesting an extension. Normally you would use this if you have events outside your control adversely affecting your course work.

3. Unexpected work commitments. In this case, you will need to attach a letter from your work supervisor with your application stating the impact on your ability to complete your assessment.

4. Military obligations with proof.

Applications for extensions must be lodged with via Moodle before the due date of the assignment.

Note: Equipment failure, loss of data, 'Heavy work commitments' or late starting of the course are not sufficientgrounds for an extension.

## ACADEMIC MISCONDUCT

Students are reminded that they should be aware of the academic misconduct guidelines available from the SAIBT website.

Deliberate academic misconduct such as plagiarism is subject to penalties.

## MARKING CRITERIA

*Please note that the following is a guide only and may be subject to change.*

*Possible deductions:*

- *Programming style:* Things to watch for are poor or no commenting, poor variable names, etc.
- *Submitted incorrectly:* -10 marks if assignment is submitted incorrectly (i.e. not adhering to the specs).

Name: ___________________

| Problem Solving and Programming (COMP1039) Assignment 2 – Weighting 15% | | | |
|---|---|---|---|
| **Criteria** | **MAX** | **MARK** | **COMMENT** |
| **PRODUCES CORRECT REUSLTS (OUTPUT) – PART 1** | | | |
| **One game**<br><br>`---------------------- START GAME ----------------------`<br>`\| Dealer hand: 8`<br>`\| Player hand: 9 + 4 = 13`<br>`\|`<br>`\| Please choose to hit or stand (h/s): h`<br>`\| Player hand: 13 + 1 = 14`<br>`\|`<br>`\| Please choose to hit or stand (h/s): h`<br>`\| Player hand: 14 + 5 = 19`<br>`\|`<br>`\| Please choose to hit or stand (h/s): s`<br>`\|`<br>`\| Dealer hand: 8 + 3 = 11`<br>`\| Dealer hand: 11 + 7 = 18`<br>`\|`<br>`\| Dealer = 18 Player = 19 *** Player Wins! ***`<br>`----------------------- END GAME -----------------------`<br>`Your score: 3 - Play again [y\|n]? y`<br><br>`---------------------- START GAME ----------------------`<br>`\| Dealer hand: 7`<br>`\| Player hand: 10 + 2 = 12`<br>`\|`<br>`\| Please choose to hit or stand (h/s): s`<br>`\|`<br>`\| Cannot stand on value less than 15!`<br>`\|`<br>`\| Player hand: 12 + 9 = 21`<br>`\|`<br>`\| Please choose to hit or stand (h/s): h`<br>`\| Player hand: 21 + 8 = 29`<br>`\| PLAYER BUSTS!`<br>`\|`<br>`\| Dealer hand: 7 + 4 = 11`<br>`\| Dealer hand: 11 + 9 = 20`<br>`\|`<br>`\| Dealer = 20 Player = 29 *** Player Wins! ***`<br>`----------------------- END GAME -----------------------`<br>`Your score: 3 - Play again [y\|n]? n`<br><br>`Your scores is: 3` | 16 | | **One game:**<br>-1 No or incorrect line spacing<br>-1 No start/end game decoration<br><br>-1 No or incorrect display of initial dealer hand<br><br>-1 No or incorrect display of player hand<br>-1 No or incorrect values for player hand<br><br>-1 No or incorrect prompt('h' or 's')<br><br>-1 No or incorrect display of dealer hand<br>-1 No or incorrect values for dealer hand<br><br>-1 No or incorrect win/push message<br><br>-1 No or incorrect display of score<br>Your score: <value><br>-1 No or incorrect score for score<br>Your score: <value><br><br>-1 No or incorrect prompt ('y' or 'n')<br><br>**Play again:**<br>-1 No or incorrect score for score<br>Your score: <value><br>(cumulative score)<br><br>**End game:**<br>-1 game doesn't end correctly |
| Player bust test: >21<br>`\| PLAYER BUSTS!`<br><br>Dealer bust test: >21<br>`\| DEALER BUSTS!`<br><br>Stand on less than 15 test (player):<br>`\| Cannot stand on value less than 15!`<br><br>Stand on 17 or more (dealer) :<br>`\| Dealer hand: 12 + 6 = 18` | 4 | | **Case tests:**<br>-1 No or incorrect player busts message<br><br>-1 No or incorrect dealer busts message<br><br>-1 No or incorrect value < 15 message<br><br>-1 Dealer doesn't stand on >= 17 |
| **Adheres to Specifications (Code) – PART 1** | | | |
| Use of random.randint(1,10) to simulate die roll<br><br>Use of following functions:<br>`get_choice()`<br>`deal_players_hand()`<br>`deal_dealers_hand(die_value)`<br>`determine_winner(player_die,dealer_die)`<br>`play_blackjack()`<br>While loop for repeated play<br><br>Appropriate if statements<br><br>Validation of user input<br>`Please choose to hit or stand (h/s): h`<br>`Please enter y or n. Play again [y\|n]?` | | | -2 No or incorrect use of randint<br>-2 No or incorrect function (-2 per function)<br><br>-2 No or incorrect while loop<br><br>-2 No or incorrect if statements<br><br>-2 No validation of user input<br><br>-2 Use of break/return/exit incorrectly |
| **PART 1 TOTAL** | **20** | | |

| | | | |
|---|---|---|---|
| **PRODUCES CORRECT REUSLTS (OUTPUT) – PART 2** | | | |
| **Menu [1 mark]**<br>`Please enter command [scores, search, play, quit]: scores` | 1 | | -2 No or incorrect line spacing<br>-1 No or incorrect menu display |
| **Scores [5 marks]**<br>`****************************************************`<br>`******************    Blackjack    ******************`<br>`******************   HIGH SCORES   ******************`<br>`****************************************************`<br>`*      Player Name                      Score      *`<br>`****************************************************`<br>`*--------------------------------------------------*`<br>`*      Tony Stark                       10         *`<br>`*--------------------------------------------------*`<br>`*      Fox Mulder                       8          *`<br>`*--------------------------------------------------*`<br>`*      Phil Dunphy                      6          *`<br>`*--------------------------------------------------*`<br>`*      Buster Bluth                     1          *`<br>`*--------------------------------------------------*`<br>`*      ????????                         0          *`<br>`*--------------------------------------------------*`<br>`****************************************************` | 5 | | -1 Each missing or incorrect info (up to 5 marks) |
| **Search [5 marks]**<br>`Please enter player’s name: Bruce Banner`<br>`Bruce Banner does not have a high scores entry.`<br><br>`Please enter player’s name: Fox Mulder`<br>`Fox Mulder has a high scores of 8.` | 5 | | -1 Each output/prompt/msg not to requirements (up to 5 marks) |
| **`Play [6 marks]`**<br>`Please enter command [scores, search, play, quit]: play`<br><br>`---------------------- START GAME ----------------------`<br>`\| Dealer hand: 7`<br>`\| Player hand: 4 + 9 = 13`<br>`\|`<br>`\| Please choose to hit or stand (h/s): h`<br>`\| Player hand: 13 + 6 = 19`<br>`\|`<br>`\| Please choose to hit or stand (h/s): s`<br>`\|`<br>`\| Dealer hand: 7 + 11 = 18`<br>`\|`<br>`\| Dealer = 18  Player = 19  *** Player Wins! ***`<br>`----------------------- END GAME -----------------------`<br><br>`Your score: 3 - Play again [y\|n]? n`<br><br>`Congratulations! You have made it into the BlackJack Hall of Fame!`<br><br>`Please enter your name: Bruce Banner`<br><br>`Successfully added Bruce Banner to BlackJack Hall of Fame.` | 6 | | -1 import Part 1 incorrect<br><br>-3 does not add to list<br>-3 does not add in order<br>-1 each output/prompt/msg not to requirements (up to 5 marks) |
| **Output file [3 marks]**<br>`Please enter command [scores, search, play, quit]: quit`<br><br>`Goodbye - thanks for playing!`<br>`Output file format:           Tony Stark 10`<br>`                              Fox Mulder 8`<br>`                              Phil Dunphy 6`<br>`                              Bruce Banner 3`<br>`                              Buster Bluth 1` | 3 | | -3 output file does not exist<br>-2 incorrect results in file<br>-2 output not to specs<br>-1 if not called ‘new scores.txt’ |
| **Adheres to Specifications (Code) – PART 2** | | | |
| While loop for menu<br>Appropriate control (if) structures<br><br>Use of following functions:<br>`read_file(filename, player_names, player_scores)`<br>`write_to_file(filename, player_names, player_scores)`<br>`display_high_scores(player_names, player_scores)`<br>`find_player(player_names, name)`<br>`is_high_score(player_names, new_scores)`<br>`add_player(player_names, player_scores, new_name, new_score, insert_position, num_players)`<br><br>Validation of user input<br>`Not a valid command - please try again.` | | | -2 No or incorrect loop<br>-2 No or incorrect if statements<br><br>-2 No or incorrect function (-2 per function)<br>-2 Not following rules (-2 per broken rule) (built in functions/methods, slicing, global variables)<br><br>-2 No validation of user input<br>-2 Use of break/return/exit incorrectly |
| **PART 2 TOTAL** | **20** | | |
| **Style – Both parts** | | | -2 Insufficient comments<br>-2 Inconsistent code layout<br>-2 Non-descriptive variable names |
| **ASSIGNMENT 2 TOTAL** | **40**<br><br>**/20** | | |

## SAMPLE OUTPUT – PART I

Sample output 1:

```
---------------------- START GAME ----------------------
| Dealer hand: 11
| Player hand: 8 + 5 = 13
|
| Please choose to hit or stand (h/s): s
|
| Cannot stand on value less than 15!
|
| Player hand: 13 + 10 = 23
| PLAYER BUSTS!
|
| Dealer hand: 11 + 5 = 16
| Dealer hand: 16 + 9 = 25
| DEALER BUSTS!
|
| Dealer = 25  Player = 23
| *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? n


Your score is: 0
```

Sample output 2:

```
---------------------- START GAME ----------------------
| Dealer hand: 2
| Player hand: 9 + 6 = 15
|
| Please choose to hit or stand (h/s): z
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 6 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 2 + 9 = 11
| Dealer hand: 11 + 7 = 18
|
| Dealer = 18  Player = 21  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? n


Your score is: 3
```

Sample output 3:

```
---------------------- START GAME ----------------------
| Dealer hand: 5
| Player hand: 5 + 4 = 9
|
| Please choose to hit or stand (h/s): h
| Player hand: 9 + 2 = 11
|
| Please choose to hit or stand (h/s): h
| Player hand: 11 + 9 = 20
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 5 + 6 = 11
| Dealer hand: 11 + 5 = 16
| Dealer hand: 16 + 4 = 20
|
| Dealer = 20  Player = 20  *** Push - no winners. ***
----------------------- END GAME -----------------------

Your score: 1 - Play again [y|n]? n


Your score is: 1
```

Sample output 4:

```
---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 9 + 7 = 16
|
| Please choose to hit or stand (h/s): s
```

```
|
| Dealer hand: 3 + 9 = 12
| Dealer hand: 12 + 2 = 14
| Dealer hand: 14 + 4 = 18
|
| Dealer = 18  Player = 16  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? n


Your score is: 0
```

Sample output 5:

```
---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 3 + 10 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 1 = 14
|
| Please choose to hit or stand (h/s): h
| Player hand: 14 + 9 = 23
| PLAYER BUSTS!
|
| Dealer hand: 7 + 8 = 15
| Dealer hand: 15 + 9 = 24
| DEALER BUSTS!
|
| Dealer = 24  Player = 23  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 11
| Player hand: 3 + 6 = 9
|
| Please choose to hit or stand (h/s): h
| Player hand: 9 + 3 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 2 = 14
|
| Please choose to hit or stand (h/s): h
| Player hand: 14 + 7 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 11 + 4 = 15
| Dealer hand: 15 + 1 = 16
| Dealer hand: 16 + 4 = 20
|
| Dealer = 20  Player = 21  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 4 + 9 = 13
|
| Please choose to hit or stand (h/s): s
|
| Cannot stand on value less than 15!
|
| Player hand: 13 + 6 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 10 = 20
|
| Dealer = 20  Player = 19  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
```

```
| Dealer hand: 5
| Player hand: 10 + 10 = 20
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 5 + 11 = 16
| Dealer hand: 16 + 7 = 23
| DEALER BUSTS!
|
| Dealer = 23  Player = 20  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 4 + 11 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 8 = 23
| PLAYER BUSTS!
|
| Dealer hand: 7 + 8 = 15
| Dealer hand: 15 + 10 = 25
| DEALER BUSTS!
|
| Dealer = 25  Player = 23  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 2
| Player hand: 7 + 3 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 5 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 8 = 23
| PLAYER BUSTS!
|
| Dealer hand: 2 + 2 = 4
| Dealer hand: 4 + 5 = 9
| Dealer hand: 9 + 10 = 19
|
| Dealer = 19  Player = 23  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 3 + 6 = 9
|
| Please choose to hit or stand (h/s): h
| Player hand: 9 + 9 = 18
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 3 = 6
| Dealer hand: 6 + 9 = 15
| Dealer hand: 15 + 7 = 22
| DEALER BUSTS!
|
| Dealer = 22  Player = 18  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 3 + 3 = 6
|
| Please choose to hit or stand (h/s): h
| Player hand: 6 + 2 = 8
|
| Please choose to hit or stand (h/s): h
| Player hand: 8 + 2 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 8 = 18
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 7 + 11 = 18
|
| Dealer = 18  Player = 18  *** Push - no winners. ***
----------------------- END GAME -----------------------

Your score: 10 - Play again [y|n]? n


Your score is: 10
```

Sample output 6:

```
---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 5 + 4 = 9
|
| Please choose to hit or stand (h/s): h
| Player hand: 9 + 8 = 17
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 8 = 11
| Dealer hand: 11 + 10 = 21
|
| Dealer = 21  Player = 17  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 9 + 7 = 16
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 7 + 4 = 11
| Dealer hand: 11 + 7 = 18
|
| Dealer = 18  Player = 16  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 2
| Player hand: 8 + 6 = 14
|
| Please choose to hit or stand (h/s): h
| Player hand: 14 + 10 = 24
| PLAYER BUSTS!
|
| Dealer hand: 2 + 8 = 10
| Dealer hand: 10 + 8 = 18
|
| Dealer = 18  Player = 24  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 8 + 11 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 8 = 11
| Dealer hand: 11 + 7 = 18
|
| Dealer = 18  Player = 19  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? n
Your score is: 3
```

Sample output 7:

```
---------------------- START GAME ----------------------
| Dealer hand: 9
| Player hand: 11 + 10 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 9 + 3 = 12
| Dealer hand: 12 + 4 = 16
| Dealer hand: 16 + 2 = 18
|
| Dealer = 18  Player = 21  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 5
| Player hand: 11 + 2 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 2 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 3 = 18
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 5 + 10 = 15
| Dealer hand: 15 + 10 = 25
| DEALER BUSTS!
|
| Dealer = 25  Player = 18  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? n


Your score is: 6
```

Sample output 8:

```
---------------------- START GAME ----------------------
| Dealer hand: 8
| Player hand: 5 + 4 = 9
|
| Please choose to hit or stand (h/s): s
|
| Cannot stand on value less than 15!
|
| Player hand: 9 + 4 = 13
|
| Please choose to hit or stand (h/s): s
|
| Cannot stand on value less than 15!
|
| Player hand: 13 + 6 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 8 + 11 = 19
|
| Dealer = 19  Player = 19  *** Push - no winners. ***
----------------------- END GAME -----------------------

Your score: 1 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 8 + 4 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 7 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 11 = 14
| Dealer hand: 14 + 1 = 15
| Dealer hand: 15 + 2 = 17
|
| Dealer = 17  Player = 19  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 4 - Play again [y|n]? n


Your score is: 4
```

## SAMPLE OUTPUT – PART II

Sample output 1:
```
Please enter command [scores, search, play, quit]: find

Not a valid command - please try again.

Please enter command [scores, search, play, quit]: scores


*****************************************************
******************    Blackjack    ******************
******************   HIGH SCORES   ******************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Buster Bluth                     1           *
*---------------------------------------------------*
*      ????????                         0           *
*---------------------------------------------------*
*****************************************************

Please enter command [scores, search, play, quit]: search

Please enter player's name: Fox Mulder

Fox Mulder has a high score of 8

Please enter command [scores, search, play, quit]: search

Please enter player's name: Bruce Banner

Bruce Banner does not have a high scores entry.


Please enter command [scores, search, play, quit]: quit

Goodbye - thanks for playing!

NOTE: Your program should output the following information to a file called new_scores.txt.

Tony Stark 10
Fox Mulder 8
Phil Dunphy 6
Buster Bluth 1
```

Sample output 2:
```
Please enter command [scores, search, play, quit]: scores


*****************************************************
******************    Blackjack    ******************
******************   HIGH SCORES   ******************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Buster Bluth                     1           *
*---------------------------------------------------*
*      ????????                         0           *
*---------------------------------------------------*
*****************************************************
```

```
Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 6 + 4 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 9 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 5 = 8
| Dealer hand: 8 + 4 = 12
| Dealer hand: 12 + 6 = 18
|
| Dealer = 18  Player = 19  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? n

Congratulations! You have made it into the BlackJack Hall of Fame!

Please enter your name: Dana Scully

Successfully added Dana Scully to BlackJack Hall of Fame.


Please enter command [scores, search, play, quit]: scores


*****************************************************
******************    Blackjack    ******************
******************   HIGH SCORES   ******************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Dana Scully                      3           *
*---------------------------------------------------*
*      Buster Bluth                     1           *
*---------------------------------------------------*
*****************************************************

Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 5 + 3 = 8
|
| Please choose to hit or stand (h/s): h
| Player hand: 8 + 4 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 3 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 4 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 2 = 12
| Dealer hand: 12 + 5 = 17
|
| Dealer = 17  Player = 19  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 6
| Player hand: 10 + 6 = 16
```

```
|
| Please choose to hit or stand (h/s): h
| Player hand: 16 + 7 = 23
| PLAYER BUSTS!
|
| Dealer hand: 6 + 6 = 12
| Dealer hand: 12 + 1 = 13
| Dealer hand: 13 + 8 = 21
|
| Dealer = 21  Player = 23  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 3
| Player hand: 2 + 8 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 11 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 3 + 8 = 11
| Dealer hand: 11 + 9 = 20
|
| Dealer = 20  Player = 21  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? n

Congratulations! You have made it into the BlackJack Hall of Fame!

Please enter your name: Bruce Wayne

Successfully added Bruce Wayne to BlackJack Hall of Fame.


Please enter command [scores, search, play, quit]: scores


*****************************************************
******************    Blackjack    ******************
******************   HIGH SCORES   ******************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Bruce Wayne                      6           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Dana Scully                      3           *
*---------------------------------------------------*
*****************************************************

Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 7 + 6 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 2 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 4 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 11 = 21
|
| Dealer = 21  Player = 19  *** Dealer Wins! ***
----------------------- END GAME -----------------------
```

```
Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 4
| Player hand: 8 + 3 = 11
|
| Please choose to hit or stand (h/s): h
| Player hand: 11 + 2 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 9 = 22
| PLAYER BUSTS!
|
| Dealer hand: 4 + 9 = 13
| Dealer hand: 13 + 6 = 19
|
| Dealer = 19  Player = 22  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 9
| Player hand: 8 + 4 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 10 = 22
| PLAYER BUSTS!
|
| Dealer hand: 9 + 8 = 17
|
| Dealer = 17  Player = 22  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 4
| Player hand: 6 + 2 = 8
|
| Please choose to hit or stand (h/s): h
| Player hand: 8 + 5 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 8 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 4 + 4 = 8
| Dealer hand: 8 + 7 = 15
| Dealer hand: 15 + 4 = 19
|
| Dealer = 19  Player = 21  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 6
| Player hand: 6 + 6 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 7 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 6 + 2 = 8
| Dealer hand: 8 + 4 = 12
| Dealer hand: 12 + 7 = 19
|
| Dealer = 19  Player = 19  *** Push - no winners. ***
----------------------- END GAME -----------------------

Your score: 4 - Play again [y|n]? n

Congratulations! You have made it into the BlackJack Hall of Fame!

Please enter your name: Homer Simpson
```

```
Successfully added Homer Simpson to BlackJack Hall of Fame.


Please enter command [scores, search, play, quit]: scores


****************************************************
*****************    Blackjack    *****************
*****************   HIGH SCORES   *****************
****************************************************
*                                                  *
*      Player Name                     Score       *
****************************************************
*--------------------------------------------------*
*      Tony Stark                      10          *
*--------------------------------------------------*
*      Fox Mulder                      8           *
*--------------------------------------------------*
*      Bruce Wayne                     6           *
*--------------------------------------------------*
*      Phil Dunphy                     6           *
*--------------------------------------------------*
*      Homer Simpson                   4           *
*--------------------------------------------------*
****************************************************

Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 11
| Player hand: 6 + 7 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 5 = 18
|
| Please choose to hit or stand (h/s): h
| Player hand: 18 + 8 = 26
| PLAYER BUSTS!
|
| Dealer hand: 11 + 6 = 17
|
| Dealer = 17  Player = 26  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? n

Sorry - you did not make it into the BlackJack Hall of Fame!


Please enter command [scores, search, play, quit]: quit

Goodbye - thanks for playing!

NOTE: Your program should output the following information to a file called new_scores.txt.

Tony Stark 10
Fox Mulder 8
Bruce Wayne 6
Phil Dunphy 6
Homer Simpson 4
```

Sample output 3:

```
Please enter command [scores, search, play, quit]: scores


****************************************************
*****************    Blackjack    *****************
*****************   HIGH SCORES   *****************
****************************************************
*                                                  *
*      Player Name                     Score       *
****************************************************
*--------------------------------------------------*
*      Tony Stark                      10          *
*--------------------------------------------------*
*      Fox Mulder                      8           *
*--------------------------------------------------*
*      Phil Dunphy                     6           *
*--------------------------------------------------*
*      Buster Bluth                    1           *
```

```
*---------------------------------------------------*
*      ????????                        0            *
*---------------------------------------------------*
*****************************************************

Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 6
| Player hand: 9 + 6 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 8 = 23
| PLAYER BUSTS!
|
| Dealer hand: 6 + 5 = 11
| Dealer hand: 11 + 7 = 18
|
| Dealer = 18  Player = 23  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 2
| Player hand: 11 + 4 = 15
|
| Please choose to hit or stand (h/s): h
| Player hand: 15 + 3 = 18
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 2 + 2 = 4
| Dealer hand: 4 + 3 = 7
| Dealer hand: 7 + 7 = 14
| Dealer hand: 14 + 10 = 24
| DEALER BUSTS!
|
| Dealer = 24  Player = 18  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 3 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 8 + 2 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 10 = 20
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 7 = 17
|
| Dealer = 17  Player = 20  *** Player Wins! ***
----------------------- END GAME -----------------------

Your score: 6 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 5 + 7 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 2 = 14
|
| Please choose to hit or stand (h/s): h
| Player hand: 14 + 2 = 16
|
| Please choose to hit or stand (h/s): h
| Player hand: 16 + 3 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 7 + 10 = 17
|
| Dealer = 17  Player = 19  *** Player Wins! ***
----------------------- END GAME -----------------------
```

```
Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 4
| Player hand: 7 + 4 = 11
|
| Please choose to hit or stand (h/s): h
| Player hand: 11 + 8 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 4 + 9 = 13
| Dealer hand: 13 + 8 = 21
|
| Dealer = 21  Player = 19  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 9 + 7 = 16
|
| Please choose to hit or stand (h/s): h
| Player hand: 16 + 10 = 26
| PLAYER BUSTS!
|
| Dealer hand: 7 + 7 = 14
| Dealer hand: 14 + 1 = 15
| Dealer hand: 15 + 10 = 25
| DEALER BUSTS!
|
| Dealer = 25  Player = 26  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 2 + 6 = 8
|
| Please choose to hit or stand (h/s): h
| Player hand: 8 + 11 = 19
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 3 = 13
| Dealer hand: 13 + 8 = 21
|
| Dealer = 21  Player = 19  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 2
| Player hand: 11 + 2 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 9 = 22
| PLAYER BUSTS!
|
| Dealer hand: 2 + 3 = 5
| Dealer hand: 5 + 2 = 7
| Dealer hand: 7 + 2 = 9
| Dealer hand: 9 + 5 = 14
| Dealer hand: 14 + 5 = 19
|
| Dealer = 19  Player = 22  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 2 + 7 = 9
```

```
|
| Please choose to hit or stand (h/s): h
| Player hand: 9 + 7 = 16
|
| Please choose to hit or stand (h/s): h
| Player hand: 16 + 8 = 24
| PLAYER BUSTS!
|
| Dealer hand: 10 + 7 = 17
|
| Dealer = 17  Player = 24  *** Dealer Wins! ***
---------------------- END GAME ----------------------

Your score: 9 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 10
| Player hand: 4 + 8 = 12
|
| Please choose to hit or stand (h/s): h
| Player hand: 12 + 8 = 20
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 10 + 10 = 20
|
| Dealer = 20  Player = 20  *** Push - no winners. ***
---------------------- END GAME ----------------------

Your score: 10 - Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer hand: 11
| Player hand: 3 + 4 = 7
|
| Please choose to hit or stand (h/s): h
| Player hand: 7 + 3 = 10
|
| Please choose to hit or stand (h/s): h
| Player hand: 10 + 11 = 21
|
| Please choose to hit or stand (h/s): s
|
| Dealer hand: 11 + 5 = 16
| Dealer hand: 16 + 3 = 19
|
| Dealer = 19  Player = 21  *** Player Wins! ***
---------------------- END GAME ----------------------

Your score: 13 - Play again [y|n]? n

Congratulations! You have made it into the BlackJack Hall of Fame!

Please enter your name: Bruce Banner

Successfully added Bruce Banner to BlackJack Hall of Fame.


Please enter command [scores, search, play, quit]: scores


*****************************************************
******************    Blackjack    ******************
******************   HIGH SCORES   ******************
*****************************************************
*                                                   *
*      Player Name                      Score       *
*****************************************************
*---------------------------------------------------*
*      Bruce Banner                     13          *
*---------------------------------------------------*
*      Tony Stark                       10          *
*---------------------------------------------------*
*      Fox Mulder                       8           *
*---------------------------------------------------*
*      Phil Dunphy                      6           *
*---------------------------------------------------*
*      Buster Bluth                     1           *
*---------------------------------------------------*
*****************************************************
```

```
Please enter command [scores, search, play, quit]: play


---------------------- START GAME ----------------------
| Dealer hand: 7
| Player hand: 3 + 10 = 13
|
| Please choose to hit or stand (h/s): h
| Player hand: 13 + 6 = 19
|
| Please choose to hit or stand (h/s): h
| Player hand: 19 + 5 = 24
| PLAYER BUSTS!
|
| Dealer hand: 7 + 11 = 18
|
| Dealer = 18  Player = 24  *** Dealer Wins! ***
----------------------- END GAME -----------------------

Your score: 0 - Play again [y|n]? n

Sorry - you did not make it into the BlackJack Hall of Fame!


Please enter command [scores, search, play, quit]: quit

Goodbye - thanks for playing!

NOTE: Your program should output the following information to a file called new_scores.txt.

Bruce Banner 13
Tony Stark 10
Fox Mulder 8
Phil Dunphy 6
Buster Bluth 1
```

**Formatting Integers:**
You can use the `format()` function to format the way integers and strings are displayed to the screen.
In the following example:

```
print(format(total_user_score, '10d'))
```

the integer value assigned to variable `total_user_score` is printed in a field that is 10 spaces wide. By default, it is right-aligned within the field width.

There are other alignment options as follows:

| Option | Meaning |
|---|---|
| `'<'` | Forces the field to be left-aligned within the available space (this is the default for most objects). |
| `'>'` | Forces the field to be right-aligned within the available space (this is the default for numbers). |
| `'^'` | Forces the field to be centered within the available space. |

More examples of use (including output):

```
>>> total_user_score = 100
>>> format(total_user_score, '10d')
'       100'
>>> format(total_user_score, '^10d')
'   100    '
>>> format(total_user_score, '<10d')
'100       '
>>> format(total_user_score, '>10d')
'       100'
```

Use nested with the print function:

```
>>> total_user_score = 100
>>> print(format(total_user_score, '10d'))
       100
>>> print(format(total_user_score, '^10d'))
   100
>>> print(format(total_user_score, '<10d'))
100
>>> print(format(total_user_score, '>10d'))
       100
```

**Formatting Text (aligning the text and specifying a width)**
Examples of use, nested with the print function (including output):

```
>>> format("You", '10s')
'You       '
>>> format("You", '^10s')
'   You    '
>>> format("You", '<10s')
'You       '
>>> format("You", '>10s')
'       You'

>>> print(format("You", '10s'))
You
>>> print(format("You", '^10s'))
   You
>>> print(format("You", '<10s'))
You
>>> print(format("You", '>10s'))
       You
```