COMP 1039

Problem Solving and Programming

Programming Assignment 2

School of Computer and Information Science
The University of South Australia
May, 2018

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 **Python module** (that contains functions that operate on lists) and **a program that will maintain information on players playing blackjack (dice) against the computer.**

This assignment is an **individual task** that will require an **individual submission**. **You will be required to submit your work via learnonline before Monday 4 June (week 13), 12 noon (internal students). You will also be required to present your work to your practical supervisor during your practical session held in week 13 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. External student will be required to submit their work via Learnonline before Friday 8 June (week 13), 11:30pm.

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 parts** to this assignment:

Part I: Writing a Python Module (list manipulation functions)

You are required to implement a Python module that contains functions that manipulate lists. Please ensure that you read sections titled 'Part I specification' below for further details.

Part II: Manage player information

You are required to write a Python program that will manage player information. The program will allow players to play blackjack (dice version) against the computer and will maintain information on players (using a List of Player objects). The program will keep a record of the players who play blackjack (dice version). Player information will be stored in a text file that will be read in when the program commences. Once the initial player information has been read in from the file, the program should allow the user to interactively query and manipulate the player information as well as play blackjack against the computer. Please ensure that you read sections titled 'Part II specification' below for further details.

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.

- The application of international standards (GQ7) by making sure your solution conforms to the standards presented in the Python Style Guide slides (available on the course website).

PART I SPECIFICATION – WRITING A PYTHON MODULE (LIST MANIPULATION FUNCTIONS)

You are required to write a list_function.py module (containing only the functions listed below). This file is provided for you (on the course website) however, you will need to modify this file by writing code that implements the functions listed below. **Please read the slides on modules available on the course website if you would like more information on modules.**

You are required to implement a Python module containing the following functions:

1. Write a function called **length(my_list)** that takes a list as a parameter and returns the length of the list. You must use a loop in your solution. You **must not** use built-in functions, list methods or string methods in your solution.

2. Write a function called **to_string(my_list, sep=', ')** that takes a list and a separator value as parameters and returns the **string** representation of the list (separated by the separator value) in the following form:

   item1, item2, item3, item4

   The separator value **must** be a default argument. i.e. sep=', '

   You must use a loop in your solution. You **must not** use built-in functions (other than the range() and str() functions), slice expressions, list methods or string methods in your solution. You may use the concatenation (+) operator to build the string. You **must** return a string from this function.

3. Write a function called **count(my_list, value)** that takes a list and a value as parameters. The function searches for the value in the list and returns how many times the value appears in the list. You may assume that the elements of the list can be compared using the comparison operators ==, !=, etc. You must use a loop in your solution. You **must not** use built-in functions (other than the range() function), list methods or string methods in your solution.

4. Write a function called **find(my_list, value)** that takes a list, and a value as parameters. The function searches for the value in the list and returns the index at which the first occurrence of value is found in the list. The function returns -1 if the value is not found in the list.

5. Write a function called **insert_value(my_list, value, insert_position)** that takes a list, a value and an insert_position as parameters. The function returns a **copy** of the list with the value inserted into the list (my_list) at the index specified by insert_position. Check for the insert_position value exceeding the list (my_list) bounds. If the insert_position is greater than the length of the list, insert the value at the end of the list. If the insert_position is less than or equal to zero, insert the value at the start of the list. You **must** use a loop(s) in your solution. You may make use of the *list_name*.append(item) method in order to build the new list. You **must not** use built-in functions (other than the range() function), slice expressions, list methods (other than the append() method) or string methods in your solution.

6. Write a function called **remove_value(my_list, remove_position)** that takes a list and a remove_position as parameters. The function returns a **copy** of the list with the item at the index specified by remove_position, removed from the list. Check for the remove_position value exceeding the list (my_list) bounds. If the remove_position is greater than the length of the list, remove the item at the end of the list. If the remove_position is less than or equal to zero, remove the item stored at the start of the list. You must use a loop in your solution. You may make use of the *list_name*.append(item) method in order to build the new list. You **must not** use built-in functions (other than the range() function), slice expressions, list methods (other than the append() method) or string methods in your solution.

7. Write a function called **`get_slice(my_list, start, stop)`** that takes a list, a start value and a stop value as parameters. The function returns a **copy** of the list between start and stop-1 (inclusive). Check for the start and stop values exceeding the list bounds. If the stop value exceeds the list bounds, then make the stop value the length of the list. If the start value exceeds the list bounds, then make the start value zero (0). Check for the start value being less than the stop value. If the start value is greater than the stop value, return an empty list. You must use a loop in your solution. You may make use of the *`list_name`*`.append(item)` method in order to build the new list. You **must not** use built-in functions (other than the `range()` function), slice expressions, list methods or string methods in your solution.

You must test your functions to ensure that they are working correctly. **So you do not have to write your own test file, one has been provided for you.** The `assign2_partI_test_file.py` file is a test file that contains code that calls the functions contained in the `list_function.py` module. **Please do not modify the test file.**

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:

- The supplied `list_function.py` module (containing the functions listed below). This is provided for you – **you will need to modify this file**.
- Functions (`length`, `to_string`, `count`, `find`, `insert_value`, `remove_value` and `get_slice`) implemented adhering to the assignment specifications.
- The supplied `assign2_partI_test_file.py` file. This is provided for you – **please DO NOT modify this file**.
- Well constructed `while` loops. (Marks will be lost if you use `break` statements or the like in order to exit from loops).
- Well constructed `for` loops. (Marks will be lost if you use `break` statements or the like in order to exit from loops).
- Appropriate `if/elif/else` statements.
- Output that **strictly** adheres to the assignment specifications.
- Good programming practice:
    - Consistent commenting and code layout. You are to provide comments to describe: your details, program description, all variable definitions, all functions, and every significant section of code.
    - Meaningful variable names.

- Your solutions **MAY** make use of the following:
    - Built-in functions `range()` and `str()`.
    - List method `append()` to create/build new lists. i.e. `*list_name*.append(item)`.
    - Concatenation (+) operator to create/build new strings.
    - Comparison operators (==, !=, <, >, etc).
    - Access the individual elements in a list with an index (one element only). i.e. `list_name[index]`.
    - Use of any of the functions you have written as part of the assignment. i.e. length() function.

- Your solutions **MUST NOT** use:
    - Built-in functions (other than `range()` and `str()` functions).
    - Slice expressions to select a range of elements from a list. i.e. `list_name[start:end]`.
    - List methods (other than the `append()` method. i.e. `*list_name*.append(item)`).
    - String methods.
    - **Do not** use `break`, or `continue` statements in your solution – doing so will result in a significant mark deduction. **Do not** use the `return` statement as a way to break out of loops. **Do not** use `quit()` or `exit()` functions as a way to break out of loops.

It is recommended that you use Python 3.6.4 (or most current version) in order to complete your assignments. Your programs **MUST** run using Python 3.6.4 (or most current version).

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

You will need both the `list_function.py` and `assign2_partI_test_file.py` files for this assignment. **These have been provided for you.** Please **download both of these files from the course website** and ensure that they are in the same directory as each other.

Test to ensure that this is working correctly by opening and running the `assign2_partI_test_file.py` file. If this is working correctly, you should now see the following output in the Python shell when you run your program:

```
Start Testing!

length Test
In function length()
List length: None
In function length()
List length: None

to_string Test
In function to_string()
List is: None
In function to_string()
List is: None
In function to_string()
List is: None

count Test
In function count()
None
In function count()
None
In function count()
None

find Test
In function find()
None
In function find()
None

insert_value Test
In function insert_value()
None
In function insert_value()
None
In function insert_value()
None
In function insert_value()
None

remove_value Test
In function remove_value()
None
In function remove_value()
None
In function remove_value()
None
```

```
get_slice Test
['r', 'i', 'n', 'g', 'i', 'n', 'g']
In function get_slice()
Slice is: None
In function get_slice()
Slice is: None
In function get_slice()
Slice is: None
In function get_slice()
Slice is: None

End Testing!
```

Stage 2

Implement one function at a time. The following implementation order is a recommendation only:

- length() – *you may find this function in the lecture slides… :)*
- to_string()
- count()
- find()
- get_slice()
- remove_value()
- insert_value()

Place the code that implements each function in the appropriate place in the `list_function.py` file.

For example, if you were implementing the `length()` function, you would place the code that calculates and returns the length of the list under the comment 'Place your code here' (within the length function definition) seen below.

```
# Function length() – place your own comments here…  : )
def length(my_list):

    # This line will eventually be removed - used for development purposes only.
    print("In function length()")

    # Place your code here
```

Test your function by running the `assign2_partI_test_file.py` test file to ensure each function is working correctly before starting on the next function.

Compare your output with the sample output provided (at the end of this document) to ensure that your function is working as it should.

Stage 3

Finally, check the sample output (see section titled 'Sample Output – Part I' towards the end of this document) and if necessary, modify your functions 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 – MANAGE PLAYER INFORMATION

Write a **menu driven program** called *yourEmailId*_player_info.py that will allow the user to enter commands and process these commands until the quit command is entered. The program will store and maintain player information (using a List of Player objects). Player information will be stored in a text file that will be read in when the program commences. Once the initial player data has been read in from the file, the program should allow the user to interactively query and manipulate the player information.

Input

When your program begins, it will read in player information from a file called players.txt. This is a text file that stores information pertaining to players. An example input file called players.txt can be found on the course website (under the Assessment tab). You may assume that all data is in the correct format. The name of the player is stored on a separate line. The very next line contains the number of games played, games won, games lost, games drawn, chip balance and the total score. This information is stored on one line and is separated by the space character as seen in Figure 1 below:

After the program has stored the data (using a List of Player objects), it will enter interactive mode as described in the following section.

```
Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
```

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

Interactive Mode

Your program should enter an interactive mode after the player 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. **players:**
   Displays for all players, the player's name, games played, won, lost, drawn, number of chips and their total score. Outputs the contents of the player list as seen below in the section titled Screen Format. Please read the section at the end of this handout titled – 'Useful Built-In Python Functions – required for part II'.

2. **buy:**
   Prompts for and reads the player's name and searches for the player in the list of players. If the player is found in the list of player objects, the number of chips the player would like to buy is prompted for and read (the player can only buy between 1-100 chips inclusive at a time). The player's chip balance is updated accordingly and a message indicating that this has been done is displayed to the screen. If the player is not found in the player list, an error message is displayed.

3. **search:**
   Prompts for and reads the player's name and searches for the player in the player list. If the player is found in the player list, the player's name, games played, won, lost, drawn, number of chips and their

total score, are displayed to the screen as seen below (in the section titled Screen Format). If the player is not found in the list of players, an error message stating the player has not been found is displayed to the screen. Please have a read of the section at the end of this handout titled – 'Useful Built-In Python Functions – required for part II'.

4. **clear:**
   Prompts for and reads the player's name and searches for the player in the list of players. If the player is found, the player's game statistics (games played, number won, number lost, number drawn and player's total score) and chip balance are set to zero and a message is displayed to the screen indicating that this has been done. If the player is not found in the player list, an error message is displayed to the screen.

5. **add:**
   Prompts for and reads the player's name. If the player does not already exist (i.e. a match is not found on the player's name), the player is added to the list of players. If the player is added, chip balance is initialised to 100 and all other data members (games played, games won, games lost, games drawn, and the player's total score), are initialised to zero and a message is displayed to the screen indicating that this has been successfully added.

   The player information must be added after the last player entry stored in the list (i.e. at the end of the list). If the player is already stored in the player list, an error message is displayed. No duplicate entries are allowed.

6. **remove:**
   Prompts for the player's name. If the player is found, he/she is removed from the list of players and a message is displayed to the screen indicating that this has been done. If the player is not found in the player list, an error message is displayed.

7. **play:**
   Prompts for the player's name. The program searches for the player in the list of players and if the player is not found in the list of players, an error message is displayed to the screen.

   If the player is found in the list of players, he/she is then able 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 game statistics are updated (i.e. games played, total score, etc…). Three (3) points are awarded if the player wins, zero (0) points are awarded if the player loses and one (1) point is awarded if the player draws with the dealer. The player's chip balance is also updated. If the player wins, the player's chip balance will be increased by the amount of chips bet. If the player loses, the player's chip balance will be decreased by the amount of chips bet. A push (draw) result does not change the player's chip balance in any way.

8. **quit:**
   Causes the program to quit, outputting the contents of the player list (list of player objects) to a file (see section '*Final Output*' below for format).

Note:

The program should display an appropriate message if a player is not found matching a search criteria. Appropriate messages should also be displayed to indicate whether a command has been successfully 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 choice
      [players, buy, search, clear, add, remove, play, quit]:
```

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

Screen Format

The **`players`** command (`display_players()` function) should display the player information in the following format:

```
===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================
```

The **`search`** command should display individual player information to the screen in the following format:

```
Johnny Rose stats:

P  W  L  D  Score
6  2  0  4  10

Chips:  20
```

Final Output

When your program finishes (because you entered the quit command) your program should output the contents of the list of players to a file called `new_players.txt`.

The format of this file should **exactly** match that of the input file.

PRACTICAL REQUIREMENTS (PART II)

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 called *yourEmailId*_player_info.py.
- The supplied player.py module (that defines the Player class). This is provided for you – **do NOT modify this file**.
- The supplied blackjack.py module (that plays one game of Blackjack (dice) against the computer). This is provided for you – **do NOT modify this file**.
- Player objects and methods (as appropriate) from the Player class definition. You are not required to implement the Player class. This is provided for you (player.py module) – **do NOT modify this file**.
- play_one_game(no_chips) function from the blackjack.py module. You are not required to implement the play_one_game(no_chips) function or the blackjack.py module. This is provided for you (blackjack.py module) – **do NOT modify this file**.
- Appropriate and well constructed while and/or for loops. (Marks will be lost if you use break statements or the like in order to exit from loops).
- You **must** implement **each** function listed below (**or** you may call the appropriate function(s) defined in the list_function module (that you wrote in part I of this assignment)).
- Appropriate if, if-else, if-elif-else statements (as necessary).
- The following functions:
    - read_file(filename, player_list)

      This function takes a file name and reads the contents of that file into the player_list (list) passed as a parameter into the function. The function returns the list of player objects. You must use a loop in your solution. You **may** use String and/or List methods in this function only. You may find the String methods split() and strip() useful here.
    - write_to_file(filename, player_list)

      This function will output the contents of the player list (list of player objects) to a file in the same format as the input file. The file will need to be opened for writing in this function (and of course closed once all writing has been done). The function accepts the filename of the file to write to and the list of player objects. You must use a loop in your solution.
    - display_players(player_list)

      This function will take the list of player objects as a parameter and will output the contents of the list to the screen. This function displays the information to the screen in the format specified in the assignment specifications under the section - 'Screen Format'. You must use a loop in your solution. Please have a read of the section at the end of this handout titled – 'Useful Built-In Python Functions – required for part II'.
    - find_player(player_list, name)

      This function will take the player's name as input along with the list of player objects (player_list) and will return the position (index) of the player found in the player_list. If the player is not found, the function returns -1. You **must** use a loop in your solution. You **must not** use list methods in your solution.

- buy_player_chips(player_list, player_pos)

  This function takes the player list and the player's position (index) in the list and updates the player's chip balance. The function prompts for and reads the number of chips the player would like to buy (between 1-100) and updates the player's chip balance by that amount. A message indicating that this has been done is displayed to the screen. You should validate the number of chips to ensure the player only enters between 1-100. You should also validate the position (index) of the player (second parameter) making sure it does not exceed list bounds. If the position exceeds list bounds, an error message is displayed.

- clear_player_stats (player_list, name)

  This function takes the list of players and the player's name, and determines whether the player exists in the list of players. If the player is found, the player's game statistics (games played, number won, number lost, number drawn, player's total score) and chip balance are set to zero and the function returns 1 (indicating that the player's statistics and chip balance have been set to zero). If the player is not found in the player list, the function returns 0 (indicating the player's statistics and chip balance have not been set to zero). Hint: this function MUST make use of the `find_player`() function

- add_player(player_list)

  This function takes the list of player objects as input. The function prompts for and reads the player's (to be added) name. If the player already exists in the list of players, display an error message to the screen. If the player does not exist in the players list, the player is added. Create a new player object with the information read in (i.e. name) and add the player object to the end of the list of players. If the player is added, chip balance is initialised to 100 and all other data members (games played, games won, games lost, games drawn, and the player's total score), are initialised to zero and a message is displayed to the screen indicating this has been successfully added. This function returns the list of players. You may use the `list_name.append(item)` method to add the new player object to the list of players. You **must** call function `find_player()` from this function.

- remove_player(player_list)

  This function takes the list of player objects as input. The function prompts for and reads the player's (to be removed) name. If the player is not found in the list of players, display an error message to the screen. If the player does exist in the players list, this function removes the player object and displays a message to the screen indicating that the player has been removed from the players list. This function returns the list of players. You may use the `list_name.append(item)` method in this function. You **must** call function `find_player()` from this function.

- play_blackjack_games(player_list, player_pos)

  The `play_blackjack_games()` 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 chip balance and game stats are 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. The function takes the player list and the player's position (index) in the player list as parameters. This function calls function `play_one_game(no_chips)` (i.e. the function provided for you in the `blackjack` module; see note below). You **must** use a loop in your solution.

  Please note: You DO NOT have to write your own game of Blackjack. One has been provided for you. The `blackjack.py` file contains the function called `play_one_game(no_chips)` that allows you to play one game of Blackjack (dice version) against the computer. **Please do not modify this module.**

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

Your solutions **MAY** make use of the following:

- Built-in functions `int()`, `input()`, `print()`, `range()`, `open()`, `close()`, `len()`, `format()` and `str()`.
- Concatenation (+) operator to create/build new strings.
- The `list_name.append(item)` method to update/create lists.
- Access the individual elements in a string with an index (one element only). i.e. `string_name[index]`.
- Access the individual elements in a list with an index (one element only). i.e. `list_name[index]`.
- `Player` objects and methods (as appropriate).
- The `list_function.py` module (that you wrote in part I of this assignment). You may like to make use of some of the functions defined in the `list_function.py` module for this part of the assignment (as appropriate). Not all will be suitable or appropriate.

Your solutions **MUST NOT** use:

- Built-in functions (other than the `int()`, `input()`, `print()`, `range()`, `open()`, `close()` `len()`, `format()` and `str()` functions).
- Slice expressions to select a range of elements from a string or list. i.e. `name[start:end]`.
- String or list methods (i.e. other than those mentioned in the 'MAY make use' of section above).
- Global variables as described in week 7 lecture.
- **Do not** use `break`, or `continue` statements in your solution – doing so will result in a significant mark deduction. **Do not** use the `return` statement as a way to break out of loops. **Do not** use `quit()` or `exit()` functions as a way to break out of loops.

PLEASE NOTE: You are reminded that you should ensure that all input and output conform to the assignment specifications. 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 3.6.4 (or most current version) in order to complete your assignments. Your programs **MUST** run using Python 3.6.4 (or most current version).

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 files (available on the course website called `player_info.py, blackjack.py` and `player.py`) and re-name player_info.py *`yourEmailId`*`_player_info.py.` Define an empty list to store the player information. For example:

```
player_list = []
```

Stage 2
Create a player object from class `Player` (provided for you in the `player.py` module). Note: **you may simply download it from the course website under the Assessment tab. The `player.py` module should be placed in the same directory as *`yourEmailId`*`_player_info.py` file.** Please do **NOT** modify the `player.py` module.

Import the `player` module and create a player object as seen below. You may wish to read the material on modules posted on the course website (under the assessment tab). You should also read the section titled 'Description of `player.py` module' included in this document.

```
import player

player_list = []

new_player = player.Player("Buster Bluth")
print(new_player)

player_list.append(new_player)
```

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.* Once you are sure that your program is working correctly, you can comment out or remove the above statements – we just wanted to make sure it was working correctly before continuing! You can bring it back later (appropriately positioned).

Stage 3
Call function `play_one_game()` (provided for you in the `blackjack.py` module). Note: **you may simply download it from the course website under the Assessment tab. The file `blackjack.py` should be placed in the same directory as *`yourEmailId`*`_player_info.py` file.** Please do **NOT** modify the `blackjack.py` module.

Import the `blackjack` module and call function `play_one_game()` as seen below. You may wish to read the material on modules posted on the course website (under the assessment tab).

```
import blackjack
import player

player_list = []

new_player = player.Player("Buster Bluth")
print(new_player)
```

```
player_list.append(new_player)

# Plays one game of blackJack and assigns the result of the game and the chip balance
# to variables game_result and no_chips respectively.
game_result, no_chips = blackjack.play_one_game(player_list[0].get_chips())

# Display to the screen the result of play_one_game() –game result (value of 3, 1 or 0)
# and number of chips remaining.
print(game_result, no_chips)

# Update player stats - games played and chip balance
player_list[0].increment_games_played()
player_list[0].set_chips(no_chips)
print(player_list[0])
```

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.*

Once you are sure that your program is working correctly, you can either remove or comment out the above statements – we just wanted to make sure it was working correctly! You can bring it back later (appropriately positioned).

Stage 4

Write the code for function `read_file()` and `display_players()` as specified above. Add code to call these two functions to ensure they are working correctly.

Stage 5

Now that you know the information is being correctly stored in your players list, write the code for function `write_to_file()`. Add code to call this function to ensure it is working correctly.

Stage 6

Implement the interactive mode, 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 choice
[players, buy, search, clear, add, remove, play, quit]: roger

Not a valid command - please try again.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

In players command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: buy

In buy command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: search

In search command
```

```
Please enter choice
[players, buy, search, clear, add, remove, play, quit]: clear

In clear command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: add

In add command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

In remove command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

In play command


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit
```

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

Stage 7

Implement one command at a time. Test to ensure the command is working correctly before starting the next command. Start with the `quit` and `players` commands as they do not need you to add anything further to the file other than ensuring that the function calls are in the correct place.

You should be able to see that for *most* commands there is a corresponding function(s).

For the remaining commands, the following implementation order is suggested (note: this is a guide only):

- `players` command `(display_players()` function`)`.
- `search` command `(find_player()` function`)`.
- `clear` command `(clear_player_stats()` function`)`.
- `buy` command `(buy_player_chips()` function`)`.
- `add` command `(add_player()` function`)`.
- `remove` command `(remove_player()` function`)`.
- `play` command `(play_blackjack_games()` function`)`.

Stage 8

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 9

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 do the following in order to submit your work and have it marked:

- Internal students:
  - You are required to submit an electronic copy of your program via learnonline **before Monday 4 June (week 13), 12 noon (internal students)**.
  - **Internal students are also required to demonstrate your assignment to your practical supervisor during your week 13 practical 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.**

  Assignments submitted to learnonline, 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 learnonline, will NOT be marked. Assignments are submitted to learnonline in order to check for plagiarism.

- External students:

  If you are an **external student**, you are only required to submit an electronic copy of your program via learnonline before Friday 8 June (week 13), 11:30pm (external students only). External students are **not** required to demonstrate in person.

All students (internal and external) must follow the submission instructions below:

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

Ensure that the following two files are included in your submission:

- `list_function.py`
- `yourEmailId_player_info.py`

For example (if your name is James Bond, your submission files would be as follows):

- `list_function.py, bonjy007_player_info.py`

All files that you submit must include the following comments.

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

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

You must submit your program **before the online due date** and demonstrate your work to your marker. You will also be required to demonstrate that you have correctly submitted your work to learnonline. Work that has not been correctly submitted to learnonline will 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 Learning and Teaching Unit councillor 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 via learnonline before the due date of the assignment.

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

ACADEMIC MISCONDUCT

ACADEMIC MISCONDUCT

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

Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the *Assessment policies and procedures manual* at:
http://www.unisa.edu.au/policies/manual/

MARKING CRITERIA

Please note that the following is a guide only and may be subject to change (see next page for breakdown).

Other 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).

University of South Australia

Assessment feedback

Problem Solving and Programming (COMP 1039)
Assignment 2 - Weighting: 15% - Due: sp2, Week 13, 2018

NAME:	MAX MARK	MARK	COMMENT
PRODUCES CORRECT RESULTS (OUTPUT) - PART I	35 marks		

length Test [5 marks]
`List length: 7`
`List length: 0`
			□ -1 No or incorrect output
			□ -1 No or incorrect output

to string Test [5 marks]
`List is: r, i, n, g, i, n, g`
`List is: r-i-n-g-i-n-g`
`List is:`
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output

count Test [5 marks]
`2`
`0`
`0`
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output

find Test [5 marks]
`3`
`-1`
			□ -1 No or incorrect output
			□ -1 No or incorrect output

insert value Test [5 marks]
`['one', 'two', 'three', 'four', 'five', 'six']`
`['p', 'i', 't']`
`['s', 'p', 'i', 't']`
`['s', 'p', 'i', 't', 's']`
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output

remove value Test [5 marks]
`['r', 'i', 'g']`
`['i', 'n', 'g']`
`['r', 'i', 'n']`
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output

get slice Test [5 marks]
`['r', 'i', 'n', 'g', 'i', 'n', 'g']`
`Slice is: ['i', 'n', 'g']`
`Slice is: []`
`Slice is: ['r', 'i', 'n', 'g']`
`Slice is: ['n', 'g', 'i']`
`End Testing!`
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output
			□ -1 No or incorrect output

ADHERES TO SPECIFICATIONS (CODE) - PART I			
Function `length(my_list)` Function `to_string(my_list, sep=', ')` Function `count(my_list, value)` Function `find(my_list, value)` Function `insert_value(my_list, value, position)` Function `remove_value(my_list, value, position)` Function `get_slice(my_list, start, stop)`			□ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented □ -2 Not to specs or -5 not implemented
Well constructed loops. Appropriate if statements. No global variables.			□ -1 No or incorrect loops □ -1 No or incorrect if statements □ -1 Use of global variables
Should **not use** the following: • Built-in functions (other than range() and str() functions) • Slice expressions to select range of elements • String or list methods (other than list_name.append() method).			□ -2 use of built-in functions not allowed □ -2 use of slicing i.e. [::-1] □ -2 use of methods not allowed
Use of `list_function.py` file. Use of `assign2_partI_test_file.py` file.			□ -1 No or incorrect use of file □ -1 No or incorrect use of file
			□ -2 Using break/return to exit loops

PRODUCES CORRECT RESULTS (OUTPUT) - PART II
`[2 marks]`
`Please enter choice`
`[players, buy, search, clear, add, remove, play, quit]:`
	35 marks		□ -3 No or incorrect line spacing
			□ -2 No or incorrect menu display

players `[5 marks]`
`=========================================================`
`-                     Player Summary                    -`
`=========================================================`
`-                             P  W  L  D   Chips   Score  -`
`---------------------------------------------------------`
`-  Jake Peralta                5  5  0  0    100      15  -`
`---------------------------------------------------------`
`-  Johnny Rose                 6  2  0  4     20      10  -`
`---------------------------------------------------------`
`-  Amy Santiago                7  4  0  3    300      15  -`
`---------------------------------------------------------`
`-  Jessica Jones              12  0  6  6     50       6  -`
`---------------------------------------------------------`
`=========================================================`
			□ -1 For each missing or incorrect info (up to 5 marks)
			□ -1 For each formatting error (up to 2 marks)

buy `[4 marks]`
`Please enter name: Johnny Rose`
`Johnny Rose currently has 20 chips.`
`How many chips would you like to buy? 100`
`Successfully updated Johnny Rose's chip balance to 120`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)

search `[4 marks]`
`Please enter name: Jake Peralta`
`Jake Peralta stats:`
`P  W  L  D  Score`
`5  5  0  0  15`
`Chips:  100`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)

clear `[4 marks]`
`Please enter name: Amy Santiago`
`Successfully cleared Amy Santiago's statistics.`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)

add `[4 marks]`
`Please enter name: Gob Bluth`
`Successfully added Gob Bluth to player list.`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)

remove `[4 marks]`
`Please enter name: Johnny Rose`
`Successfully removed Johnny Rose from player list.`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)

play `[4 marks]`
`Please enter name: Jessica Jones`
`--------------------- START GAME ----------------------`
`:`
`:`
`---------------------- END GAME ----------------------`
`Play again [y|n]? n`
			□ -1 For each output/prompt/msg not to specs (up to 4 marks)
			Check to see whether stats are updating correctly as a result of played games

Output file `(new players.txt)` `[4 marks]`			□ -2 If output file does not exist □ -1 If incorrect results in file □ -1 If output not to specs in file
ADHERES TO SPECIFICATIONS (CODE) - PART II While loop for menu/prompt (choice != "quit") Appropriate control structures (in general) Use of following functions: ▪ read_file(filename, player_list): ▪ display_players(player_list): ▪ write_to_file(filename, player_list): ▪ find_player(player_list, name): ▪ add_player(player_list): ▪ remove_player(player_list): ▪ buy_player_chips(player_list, pos): ▪ clear_player_stats (player_list, name): ▪ play_blackjack_games (player_list, player_pos): Use of `player.py` file and `Player` class Should **not use** the following: Built-in functions, Slice expressions to select range of elements, String or list methods (other than list_name.append()), Global variables. *Validation of user input - messages:* `Not a valid command - please try again.`			□ -2 No or incorrect loop □ -2 No or incorrect control structures □ -2 No or incorrect function read_file □ -2 No or incorrect function display_players □ -2 No or incorrect function write_to_file □ -2 No or incorrect function find_player □ -2 No or incorrect function add_player □ -2 No or incorrect function remove_player □ -2 No or incorrect function buy_player_chips □ -2 No or incorrect function clear_player_stats □ -2 No or incorrect function play_blackjack_games □ -2 No or incorrect use of file □ -2 for **each** should not be using □ -2 No validation of user input □ -2 For using break/return/exit statements to exit loops
STYLE (BOTH PARTS): Comments (your details, prog. description, all variable definitions, functions & code), meaningful variable names.	5 marks		□ -4 Insufficient comments □ -4 Inconsistent code layout □ -4 Non-descriptive variable names
TOTAL	**75 MARKS**		

DESCRIPTION OF `player.py` MODULE

The `player.py` module provides a player class definition called `Player`. This class definition is to be used to create player objects and is designed to be used for assignment 2 (part II) work. `Player` objects will be used to store player information.

To make use of `class Player` defined within the `player` module, you will need to import the `player` module as seen below:

```
import player
```

To create a `Player` object, with name (and all other game stats; games played, no won, no lost, no drawn and total score initialised to zero and number of chips initialised to 100) you will need to do the following:

```
new_player = player.Player("Buster Bluth")
```

The above code constructs a `Player` object storing player information about Buster Bluth.

`Player(name)`	Constructs a `Player` object with name data member; all other data members are initialised to zero and chips is initialised to 100.

`Player` Object Methods:

set _name(name)	Sets the player's name.
get_name()	Returns the player's name.
set_games_played(played)	Sets the number of games player has played.
get_games_played()	Returns the number of games player has played.
increment_games_played(games)	Increments the number of games player has played by games (default 1).
set_no_won(won)	Sets the number of games player has won.
get_no_won()	Returns the number of games player has won.
increment_no_won(won)	Increments the number of games player has won by won (default 1).
set_no_lost(lost)	Sets the number of games player has lost.
get_no_lost()	Returns the number of games player has lost.
increment_no_lost(lost)	Increments the number of games player has lost by lost (default 1).
set_no_drawn(drawn)	Sets the number of games player has drawn.
get_no_drawn()	Returns the number of games player has drawn.
increment_no_drawn(drawn)	Increments the number of games player has drawn by drawn (default 1).
set_chips(chips)	Sets the number of chips player owns.
get_chips()	Returns the number of chips the player owns.

set_total_score(score) Sets the total score resulting from all games played.

get_total_score() Returns the player’s total score.

increment_total_score(score) Increments the total score of the player by score (default of one (1)).

__str__() Returns a string representation of the `Player` object.

A simple example:

You can construct a `Player` object and call it’s methods as seen in the following example:

```
import player


# Create Player object with name "Buster Bluth"
new_player1 = player.Player("Buster Bluth")

# Display the Player object to the screen as specified by the __str__ method.
print(new_player1)

# Update the number of games played by one
new_player1.increment_games_played()

# Display the Player object to the screen as specified by the __str__ method.
print(new_player1)

# Create another Player object
new_player2 = player.Player("Gob Bluth")

# Update chip balance
new_player2.set_chips(50)

# Display the Player object to the screen as specified by the __str__ method.
print(new_player2)

# Create yet another Player object
new_player3 = player.Player("Buster Bluth")

# Update total score of player
new_player3.increment_total_score(3)

# Display the Player object to the screen as specified by the __str__ method.
print(new_player3)


# Create list of players (Player objects)
player_list = []

# Append player objects to list
player_list.append(new_player1)
player_list.append(new_player2)
player_list.append(new_player3)

print("\n\nDisplaying player_list:")

# Iterate over player list and display to screen
for player in player_list:
    print(player.get_name(), player.get_games_played(), player.get_total_score(), player.get_chips())
```

Output:

```
Buster Bluth              0  0  0  0     100       0
Buster Bluth              1  0  0  0     100       0
Gob Bluth                 0  0  0  0      50       0
Buster Bluth              0  0  0  0     100       3


Displaying player_list:
Buster Bluth 1 0 100
Gob Bluth 0 0 50
Buster Bluth 0 3 100
```

SAMPLE OUTPUT – PART I

Sample output 1:

```
Start Testing!

length Test
List length: 7
List length: 0

to_string Test
List is: r, i, n, g, i, n, g
List is: r-i-n-g-i-n-g
List is:

count Test
2
0
0

find Test
3
-1

insert_value Test
['one', 'two', 'three', 'four', 'five', 'six']
['p', 'i', 't']
['s', 'p', 'i', 't']
['s', 'p', 'i', 't', 's']

remove_value Test
['r', 'i', 'g']
['i', 'n', 'g']
['r', 'i', 'n']

get_slice Test
['r', 'i', 'n', 'g', 'i', 'n', 'g']
Slice is: ['i', 'n', 'g']
Slice is: []
Slice is: ['r', 'i', 'n', 'g']
Slice is: ['n', 'g', 'i']

End Testing!
```

SAMPLE OUTPUT – PART II

Sample output 1:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: list

Not a valid command - please try again.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: buy

Please enter name: Tony Stark

Tony Stark is not found in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: buy

Please enter name: Johnny Rose

Johnny Rose currently has 20 chips.

How many chips would you like to buy? 1000
You may only buy between 1-100 chips at a time!

How many chips would you like to buy? -1
You may only buy between 1-100 chips at a time!

How many chips would you like to buy? 50

Successfully updated Johnny Rose's chip balance to 70


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: summary

Not a valid command - please try again.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      70      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --
```

```
NOTE: Your program should output the following information to a file - new_players.txt.

Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 70 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
```

Sample output 2:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --


NOTE: Your program should output the following information to a file - new_players.txt.

Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
>>> ================================ RESTART ================================
>>>
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: search

Please enter name: Gob Bluth

Gob Bluth is not found in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: search

Please enter name: Amy Santiago

Amy Santiago stats:

P  W  L  D  Score
7  4  0  3  15

Chips:  300


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

==========================================================
-                     Player Summary                     -
==========================================================
-                             P  W  L  D  Chips   Score  -
----------------------------------------------------------
-  Jake Peralta               5  5  0  0    100      15  -
----------------------------------------------------------
-  Johnny Rose                6  2  0  4     20      10  -
----------------------------------------------------------
-  Amy Santiago               7  4  0  3    300      15  -
----------------------------------------------------------
-  Jessica Jones             12  0  6  6     50       6  -
----------------------------------------------------------
==========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --
```

NOTE: Your program should output the following information to a file - new_players.txt.

```
Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
```

Sample output 3:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: clear

Please enter name: Captain Holt

Captain Holt is not found in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: clear

Please enter name: Jessica Jones

Successfully cleared Jessica Jones's statistics.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones              0  0  0  0       0       0  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --
```

NOTE: Your program should output the following information to a file - new_players.txt.

```
Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
0 0 0 0 0 0
```

Sample output 4:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: add

Please enter name: Jake Peralta

Jake Peralta already exists in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: add

Please enter name: Buster Bluth

Successfully added Buster Bluth to player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
-  Buster Bluth               0  0  0  0     100       0  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: add

Please enter name: Buster Bluth

Buster Bluth already exists in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
-  Buster Bluth               0  0  0  0     100       0  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --
```

```
NOTE: Your program should output the following information to a file - new_players.txt.

Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
Buster Bluth
0 0 0 0 100 0
```

Sample output 5:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

Please enter name: Denny Crane

Denny Crane is not found in players.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

Please enter name: Jake Peralta

Successfully removed Jake Peralta from player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================
Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

Please enter name: Johnny Rose

Successfully removed Johnny Rose from player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

Please enter name: Jessica Jones

Successfully removed Jessica Jones from player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: remove

Please enter name: Amy Santiago

Successfully removed Amy Santiago from player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------


No players to display!

===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: add

Please enter name: Denny Crane

Successfully added Denny Crane to player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Denny Crane                0  0  0  0     100       0  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

Please enter name: Batman

Batman is not found in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Denny Crane                0  0  0  0     100       0  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

Please enter name: Denny Crane


---------------------- START GAME ----------------------
| Dealer's hand is: 7
| Player's hand is: 6
|
| --> Number of chips: 100
| --> Place your bet:  10
|
| Player's hand is: 6 + 7 = 13
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 13 + 9 = 22
| *** Player busts!
|
| Dealer's hand is: 7 + 3 = 10
| Dealer's hand is: 10 + 4 = 14
| Dealer's hand is: 14 + 4 = 18
|
| *** Dealer: 18  Player: 22  -  Dealer Wins! ***
| --> 90 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 4
| Player's hand is: 5
|
| --> Number of chips: 90
| --> Place your bet:  20
|
| Player's hand is: 5 + 1 = 6
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 6 + 6 = 12
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 12 + 3 = 15
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 15 + 5 = 20
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 4 + 10 = 14
| Dealer's hand is: 14 + 3 = 17
|
| *** Dealer: 17  Player: 20  -  Player Wins! ***
| --> 110 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 8
| Player's hand is: 4
|
| --> Number of chips: 110
| --> Place your bet:  50
|
| Player's hand is: 4 + 4 = 8
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 8 + 3 = 11
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 11 + 9 = 20
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 8 + 9 = 17
|
| *** Dealer: 17  Player: 20  -  Player Wins! ***
| --> 160 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? n

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Denny Crane                3  2  1  0     160       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --


NOTE: Your program should output the following information to a file - new_players.txt.

Denny Crane
3 2 1 0 160 6
```

Sample output 6:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
```

```
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

Please enter name: Peter Parker

Peter Parker is not found in player list.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

Please enter name: Jake Peralta


---------------------- START GAME ----------------------
| Dealer's hand is: 10
| Player's hand is: 5
|
| --> Number of chips: 100
| --> Place your bet:  20
|
| Player's hand is: 5 + 1 = 6
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 6 + 2 = 8
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 8 + 11 = 19
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 10 + 7 = 17
|
| *** Dealer: 17  Player: 19  -  Player Wins! ***
| --> 120 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 6
| Player's hand is: 10
|
| --> Number of chips: 120
| --> Place your bet:  20
|
| Player's hand is: 10 + 9 = 19
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 6 + 3 = 9
| Dealer's hand is: 9 + 4 = 13
| Dealer's hand is: 13 + 8 = 21
|
| *** Dealer: 21  Player: 19  -  Dealer Wins! ***
| --> 100 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 7
| Player's hand is: 5
|
| --> Number of chips: 100
| --> Place your bet:  30
|
| Player's hand is: 5 + 1 = 6
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 6 + 6 = 12
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 12 + 1 = 13
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 13 + 6 = 19
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 7 + 8 = 15
| Dealer's hand is: 15 + 6 = 21
|
| *** Dealer: 21  Player: 19  -  Dealer Wins! ***
| --> 70 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 9
| Player's hand is: 11
|
| --> Number of chips: 70
| --> Place your bet:  30
|
| Player's hand is: 11 + 7 = 18
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 9 + 2 = 11
| Dealer's hand is: 11 + 7 = 18
|
| *** Dealer: 18  Player: 18  -  Push - no winners! ***
| --> 70 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? n

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               9  6  2  1      70      19  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --


NOTE: Your program should output the following information to a file - new_players.txt.

Jake Peralta
9 6 2 1 70 19
Johnny Rose
6 2 0 4 20 10
Amy Santiago
7 4 0 3 300 15
Jessica Jones
12 0 6 6 50 6
```

Sample output 7:

```
File     : wayby001_player_info.py
Author   : Batman
Stud ID  : 0123456X
Email ID : wayby001
This is my own work as defined by the University's Academic Misconduct Policy.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               7  4  0  3     300      15  -
```

```
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: clear

Please enter name: Amy Santiago

Successfully cleared Amy Santiago's statistics.


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               0  0  0  0       0       0  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: buy

Please enter name: Amy Santiago

Amy Santiago currently has 0 chips.

How many chips would you like to buy? 100

Successfully updated Amy Santiago's chip balance to 100


Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players

===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               0  0  0  0     100       0  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: play

Please enter name: Amy Santiago


---------------------- START GAME ----------------------
| Dealer's hand is: 2
| Player's hand is: 4
|
| --> Number of chips: 100
| --> Place your bet:  30
|
| Player's hand is: 4 + 7 = 11
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 11 + 6 = 17
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 2 + 7 = 9
| Dealer's hand is: 9 + 6 = 15
| Dealer's hand is: 15 + 5 = 20
|
| *** Dealer: 20  Player: 17  -  Dealer Wins! ***
| --> 70 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 5
| Player's hand is: 7
```

```
|
| --> Number of chips: 70
| --> Place your bet:  20
|
| Player's hand is: 7 + 10 = 17
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 5 + 5 = 10
| Dealer's hand is: 10 + 11 = 21
|
| *** Dealer: 21  Player: 17  -  Dealer Wins! ***
| --> 50 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 10
| Player's hand is: 5
|
| --> Number of chips: 50
| --> Place your bet:  40
|
| Player's hand is: 5 + 5 = 10
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 10 + 11 = 21
|
| Dealer's hand is: 10 + 4 = 14
| Dealer's hand is: 14 + 3 = 17
|
| *** Dealer: 17  Player: 21  -  Player Wins! ***
| --> 90 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 11
| Player's hand is: 9
|
| --> Number of chips: 90
| --> Place your bet:  10
|
| Player's hand is: 9 + 4 = 13
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 13 + 6 = 19
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 11 + 2 = 13
| Dealer's hand is: 13 + 1 = 14
| Dealer's hand is: 14 + 3 = 17
|
| *** Dealer: 17  Player: 19  -  Player Wins! ***
| --> 100 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? y


---------------------- START GAME ----------------------
| Dealer's hand is: 2
| Player's hand is: 7
|
| --> Number of chips: 100
| --> Place your bet:  20
|
| Player's hand is: 7 + 2 = 9
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 9 + 4 = 13
| Please enter h or s (h = Hit, s = Stand): h
| Player's hand is: 13 + 5 = 18
| Please enter h or s (h = Hit, s = Stand): s
|
| Dealer's hand is: 2 + 4 = 6
| Dealer's hand is: 6 + 6 = 12
| Dealer's hand is: 12 + 3 = 15
| Dealer's hand is: 15 + 1 = 16
| Dealer's hand is: 16 + 4 = 20
|
| *** Dealer: 20  Player: 18  -  Dealer Wins! ***
| --> 80 chips left!
----------------------- END GAME -----------------------


Play again [y|n]? n

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: players
```

```
===========================================================
-                     Player Summary                      -
===========================================================
-                             P  W  L  D   Chips   Score  -
-----------------------------------------------------------
-  Jake Peralta               5  5  0  0     100      15  -
-----------------------------------------------------------
-  Johnny Rose                6  2  0  4      20      10  -
-----------------------------------------------------------
-  Amy Santiago               5  2  3  0      80       6  -
-----------------------------------------------------------
-  Jessica Jones             12  0  6  6      50       6  -
-----------------------------------------------------------
===========================================================

Please enter choice
[players, buy, search, clear, add, remove, play, quit]: quit


-- Program terminating --
```

NOTE: Your program should output the following information to a file - new_players.txt.

```
Jake Peralta
5 5 0 0 100 15
Johnny Rose
6 2 0 4 20 10
Amy Santiago
5 2 3 0 80 6
Jessica Jones
12 0 6 6 50 6
```

USEFUL BUILT-IN PYTHON FUNCTIONS – REQUIRED FOR PART II

Formatting Integers (useful for part II):
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
```