COMP 1039

Problem Solving and Programming

Programming Assignment 2

School of Computer and Information Science
The University of South Australia
October, 2020

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 characters (heroes and villains).**

This assignment is an **individual task** that will require an **individual submission**. **You will be required to submit your work via learnonline before Tuesday 10 November (swot vac), 10:00 am (internal students).**

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 character (hero and villain) information

You are required to write a Python program that will manage character (heroes and villain) information. Character (hero and villain) information will be stored in a text file that will be read in when the program commences. Once the initial character (hero and villain) information has been read in from the file, the program should allow the user to interactively query and manipulate the character (hero and villain) information. 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 **`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.

4. 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.

5. 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.

6. 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.

Please note:
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`, `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.

Please ensure that you use Python 3.8.3 or a later version (i.e. the latest version) in order to complete your assignments. Your programs **MUST** run using Python 3.8.3 (or latest 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

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 slides… :)*
- to_string()
- find()
- remove_value()
- insert_value()
- get_slice()

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 CHARACTER INFORMATION

Write a **menu driven program** called `part2_yourEmailId.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 character information (**using two List objects – one that stores the character's name and one that stores the character's health rating**). Character information will be stored in a text file that will be read in when the program commences. Once the initial character data has been read in from the file, the program should allow the user to interactively query and manipulate the character information.

Input

When your program begins, it will read in character (hero and villain) information from a file called `characters.txt`. This is a text file that stores information pertaining to characters (heroes and villains). An example input file called `characters.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 character (hero or villain) is stored on a separate line. The very next line contains the hero or villain's health value. This information is seen in Figure 1 below:

After the program has stored the data (using two List objects, one that stores the character's name and one that stores the character's health value), it will enter interactive mode as described in the following section.

```
Wonder Woman
90
Batman
80
The Joker
80
Superman
100
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
80
Thanos
90
```

Figure 1: *Character information file format (`characters.txt`).*

Your program will maintain **two List objects** as follows:

```
character_list = []  # List to store character's name
health_list = []     # List to store character's health value
```

Once the above information is read in from the file, the two lists will be populated as follows:

character_list
Wonder Woman
Batman
The Joker
Superman
Catwoman
Aquaman
Iron Man
Hulk
Thanos

health_list
90
80
80
100
50
30
50
80
90

Note: That the character and their health value are stored in corresponding positions in each list, i.e. the character stored in character_list[0] has their health value stored in health_list[0], the character in character_list[1] has their health value stored in health_list[1] and so on...

Interactive Mode

Your program should enter an interactive mode after the character (hero and villain) 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. **list:**
   Displays for all characters, the character's name and their associated health value. Outputs the contents of the character's name and health lists 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. **search:**
   Prompts for and reads the character's name and searches for the character in the character list. If the character is found in the character list, the character's name and their health value, are displayed to the screen as seen below (in the section titled Screen Format). If the character is not found in the list of characters (heroes and villains), an error message stating the character has not been found is displayed to the screen.

3. **reset:**
   Prompts for and reads the character's name and searcher for the character in the character list. If the character is found in the list of characters, the character's corresponding health value (stored in the health list) is reset to 100. If the character is not found in the list of characters, an error message stating the character has not been found is displayed to the screen.

4. **add:**
   Prompts for and reads the character's name. If the character does not already exist (i.e. a match is not found on the character's name), the character is added to the list of characters and their corresponding health value of 100 is added to the health list. A message is displayed to the screen indicating that this has been successfully added.

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

5. **remove:**
   Prompts for the character's name. If the character is found, he/she is removed from the list of characters (along with their associated health value stored in the health list) and a message is displayed to the screen indicating that this has been done. If the character is not found in the character list, an error message is displayed.

6. **battle:**
   Prompts for the name of the two opponents to do battle. The program searches for each character in the list of characters and if the character is not found in the list of characters, an error message is displayed to the screen and the user is asked to enter another character.

   If the opponents are found in the list of characters, he/she is then able to do battle and the number of battle rounds the heroes/villains will undertake (a number between 1-5 inclusive) is prompted for and read. One battle may have many (1-5 inclusive) rounds. The heroes/villains battle until either an opponent dies (health status is reduced to zero) or the number of battle rounds have been completed. For each individual battle round, the hero/villain will sustain a certain amount of damage to their health rating (make sure you update the health list). The amount of damage sustained from the battle will be a randomly generated value between 0–50 inclusive. After each round, each opponents damage value (i.e. randomly generated number between 0–50 inclusive) and current health value (i.e. calculated by: health value – damage value) are displayed to the screen.

   After every battle (however many rounds), a winner is determined, i.e. the opponent with the higher health value wins the battle.

7. **quit:**
   Causes the program to quit, outputting the contents of the character and health lists to a file (see section '*Final Output*' below for format).

Note:

The program should display an appropriate message if a character 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
[list, search, reset, add, remove, battle, quit]:
```

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

Screen Format

The **`list`** command (`display_characters()` function) should display the character (hero and villain) information in the following format:

```
==================================
-        Character Summary         -
==================================
-  Name                    Health  -
----------------------------------
-  Wonder Woman              90  -
----------------------------------
-  Batman                    80  -
----------------------------------
-  The Joker                 80  -
----------------------------------
-  Superman                 100  -
----------------------------------
-  Catwoman                  50  -
----------------------------------
-  Aquaman                   30  -
----------------------------------
-  Iron Man                  50  -
----------------------------------
-  Hulk                      80  -
----------------------------------
==================================
```

The **`search`** command should display individual character (hero/villain) information to the screen in the following format:

```
Catwoman's current health: 50%
```

Final Output

When your program finishes (because you entered the quit command) your program should output the contents of the list of characters and the list of health values to a file called `new_characters.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 `part2_yourEmailId.py`.
- 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.
- You **must** 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, character_list, health_list)

      This function takes a file name and two lists (character_list and health_list) as parameters. This function reads the names of the characters stored in that file into a list called character_list and reads their associated health value into a list called health_list. The function does not return a value. 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 method `strip()` useful here. Please note: This function will be provided for you and explained in class. You may use your own function or use the function provided… the decision is yours. :)

    - write_to_file(filename, character_list, health_list)

      This function will output the contents of the character list (list of character names) and the health list (list of health values) 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, the list of characters and the list of health values as parameters. You **must** use a loop in your solution.

    - display_characters(character_list, health_list)

      This function will take the list of characters and the list of health values as parameters and will output the contents of the lists 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'.

- do_battle(character_list, health_list, opponent1_pos, opponent2_pos)

  This function takes the list of characters, the list of health values and the position of the two characters that are about to do battle (i.e. position is the location/index of the character stored in the list of characters and their associated health value stored in the health_list. This is useful so we can update the character's health value after every battle).

  This function prompts for and reads the number of battle rounds the heroes/villains will undertake (a number between 1-5 inclusive). The function allows the heroes/villains to battle until either an opponent dies (health status is reduced to zero) or the number of battle rounds have been completed. For each individual battle round, the hero/villain will sustain a certain amount of damage to their health rating. The amount of damage sustained from the battle will be a randomly generated value between (0 - 50 inclusive).

  General algorithm is as follows:

  while (number of battle rounds have not been completed and both opponents are still alive)

    randomly generate a damage value sustained from battle and update opponent 1's health value by health_list[opponent1_pos].

    randomly generate a damage value sustained from battle and update opponent 2's health value by health_list[opponent2_pos].

    display opponent 1 round results (as seen in the sample output)

    display opponent 2 round results (as seen in the sample output)

  determine the winner of the battles - the character with the most health left at the end of all the battle rounds is the winner.

  display the winner to the screen and also report if a character has died as a result of battle (refer to sample output for layout).

- 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]`.
- The `list_function.py` module (that you wrote in part I of this assignment). You should make use of some of the functions defined in the `list_function.py` module for this part of the assignment (as appropriate, i.e. find, insert_value and remove_value). 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 10 material.
- **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 3.8.3 or a later version (i.e. the latest version) in order to complete your assignments. Your programs **MUST** run using Python 3.8.3 (or latest 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 `part2_emailid.py`) and re-name part2_emailid.py to `part2_yourEmailId.py`.

Define an empty list to store the character information. For example:

```
character_list = []
```

Define an empty list to store the character's health information. For example:

```
health_list = []
```

Stage 2

Write the code for functions `read_file()` and `display_characters()` as specified above. Add code to call these two functions to ensure they are working correctly. You may write your own `read_file()` function or you may use the one provided for you. The read_file() function will be explained in class.

Stage 3

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

Stage 4

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
[list, search, reset, add, remove, battle, quit]: roger

Not a valid command - please try again.


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

In list command


Please enter choice
[list, search, reset, add, remove, battle, quit]: search

In search command


Please enter choice
[list, search, reset, add, remove, battle, quit]: reset

In reset command
```

```
Please enter choice
[list, search, reset, add, remove, battle, quit]: add

In add command


Please enter choice
[list, search, reset, add, remove, battle, quit]: remove

In remove command


Please enter choice
[list, search, reset, add, remove, battle, quit]: battle

In battle command


Please enter choice
[list, search, reset, add, remove, battle, 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 `list` 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) – found in either part I or part II.

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

- `list` command (`display_characters()` function).
- `search` command (calls the `find` function written in part I).
- `reset` command (calls the `find` function written in part I).
- `add` command (calls the `insert_value()` function written in part I).
- `remove` command (calls the `remove_value()` function written in part I).
- `battle` command (`do_battle()` 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 Tuesday 10 November (swot vac), 10 am (internal students)**.

  Assignments submitted to learnonline will be checked for plagiarism.

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`
- `part2_yourEmailId.py`

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

- `list_function.py, part2_bonjy007.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**. 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 – Part 2 - Weighting: 15% - Due: sp5, Swot-vac, 2020

NAME:	MAX MARK	MARK	COMMENT
PRODUCES CORRECT RESULTS (OUTPUT) - PART I	30 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

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']`
`['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

ADHERES TO SPECIFICATIONS (CODE) - PART I
 	
Function `length(my_list);`
`return length of list`
Function `to_string(my_list, sep=', ');`
`return str`
Function `find(my_list, value);`
`return index or -1`
Function `insert_value(my_list, value, position);`
`return copy of list`
Function `remove_value(my_list, position);`
`return copy of list`
Function `get_slice(my_list, start, stop);`
`return copy of list`
 	
Well constructed loops.
Appropriate if statements.
No 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).
 	
Use of `list_function.py` file.
Use of `assign2_partI_test_file.py` file.
			☐ -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
			
			☐ -1 No or incorrect loops
			☐ -1 No or incorrect if statements
			☐ -1 Use of global variables
			
			☐ -2 use of built-in functions not allowed
			☐ -2 use of slicing i.e. [::-1]
			☐ -2 use of methods not allowed
			
			☐ -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
 	
Please enter choice
[list, search, reset, add, remove, battle, quit]:
 	
list [4 marks]
`==========`
`- Character Summary -`
`==========`
`- Name Health -`
`----------`
`- Wonder Woman 90 -`
`- Batman 80 -`
`- The Joker 80 -`
`- Superman 100 -`
`- Catwoman 50 -`
`- Aquaman 30 -`
`- Iron Man 50 -`
`- Hulk 80 -`
`==========`
search [4 marks]
Catwoman's current health: 50%
reset [2 marks]
Please enter name: Batman
 	
Successfully updated Batman's health to 100
add [4 marks]
Please enter name: Deadpool
 	
Successfully added Deadpool
remove [4 marks]
Please enter name: Iron Man
 	
Successfully removed Iron Man
battle [8 marks]
-- Battle --
 	
Wonder Woman versus Aquaman - 1 rounds
 	
Round: 1
> Wonder Woman - Damage: 4 - Current health: 86
> Aquaman - Damage: 2 - Current health: 28
 	
-- End of battle --
 	
** Wonder Woman wins! **
Output file (new_characters.txt) [4 marks]
	30 marks		☐ -2 No or incorrect line spacing
			☐ -2 No or incorrect menu display
			☐ -1 For each missing or incorrect info (up to 2 marks)
			☐ -1 For each formatting error (up to 2 marks)
			☐ -1 For each output/prompt/msg not to specs (up to 4 marks)
			☐ -1 For each output/prompt/msg not to specs (up to 2 marks)
			☐ -1 For each output/prompt/msg not to specs (up to 4 marks)
			☐ -1 For each output/prompt/msg not to specs (up to 4 marks)
			☐ -1 For each output/prompt/msg not to specs (up to 8 marks)
			Check to see whether health is updating correctly as a result of battles. -2 if not updating correctly.
			☐ -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" or equivalent) Appropriate control structures (in general) Use of following functions: ▪ read_file(filename); ▪ display_characters(character_list, display_type) ▪ write_to_file(filename, character_list) ▪ do_battle(character_list, health_list, opp1_pos, opp2_pos) Use of `list_function.py` file. 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_characters ☐ -2 No or incorrect function write_to_file ☐ -2 No or incorrect function do_battle ☐ -2 No or incorrect use of functions within 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	**65 MARKS**		

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:

find Test
3
-1

insert_value Test
['one', 'two', 'three', 'four', 'five']
['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_partII.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
[list, search, reset, add, remove, battle, quit]: play

Not a valid command - please try again.


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman              90  -
------------------------------------
-  Batman                    80  -
------------------------------------
-  The Joker                 80  -
------------------------------------
-  Superman                 100  -
------------------------------------
-  Catwoman                  50  -
------------------------------------
-  Aquaman                   30  -
------------------------------------
-  Iron Man                  50  -
------------------------------------
-  Hulk                      80  -
------------------------------------
====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
80
The Joker
80
Superman
100
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
80
```

Sample output 2:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman              90  -
------------------------------------
-  Batman                    80  -
------------------------------------
-  The Joker                 80  -
```

```
-----------------------------------
-  Superman                  100  -
-----------------------------------
-  Catwoman                   50  -
-----------------------------------
-  Aquaman                    30  -
-----------------------------------
-  Iron Man                   50  -
-----------------------------------
-  Hulk                       80  -
-----------------------------------
===================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: search

Please enter name: Black Widow

Black Widow is not found in character list.


Please enter choice
[list, search, reset, add, remove, battle, quit]: search

Please enter name: Aquaman

Aquaman's current health: 30%


Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
80
The Joker
80
Superman
100
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
80
```

Sample output 3:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list

===================================
-        Character Summary        -
===================================
-  Name                    Health  -
-----------------------------------
-  Wonder Woman               90  -
-----------------------------------
-  Batman                     80  -
-----------------------------------
-  The Joker                  80  -
-----------------------------------
-  Superman                  100  -
-----------------------------------
-  Catwoman                   50  -
-----------------------------------
-  Aquaman                    30  -
-----------------------------------
-  Iron Man                   50  -
-----------------------------------
-  Hulk                       80  -
-----------------------------------
===================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: reset

Please enter name: Spiderman
```

```
Spiderman is not found in character list.


Please enter choice
[list, search, reset, add, remove, battle, quit]: reset

Please enter name: Iron Man

Successfully updated Iron Man's health to 100


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman             90  -
------------------------------------
-  Batman                   80  -
------------------------------------
-  The Joker                80  -
------------------------------------
-  Superman                100  -
------------------------------------
-  Catwoman                 50  -
------------------------------------
-  Aquaman                  30  -
------------------------------------
-  Iron Man                100  -
------------------------------------
-  Hulk                     80  -
------------------------------------
====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
80
The Joker
80
Superman
100
Catwoman
50
Aquaman
30
Iron Man
100
Hulk
80
```

Sample output 4:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman             90  -
------------------------------------
-  Batman                   80  -
------------------------------------
-  The Joker                80  -
------------------------------------
-  Superman                100  -
------------------------------------
-  Catwoman                 50  -
------------------------------------
-  Aquaman                  30  -
------------------------------------
-  Iron Man                 50  -
```

```
-----------------------------------
-  Hulk                         80  -
-----------------------------------
===================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: add

Please enter name: Catwoman

Catwoman already exists in character list.


Please enter choice
[list, search, reset, add, remove, battle, quit]: add

Please enter name: Black Widow

Successfully added Black Widow


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

===================================
-        Character Summary         -
===================================
-  Name                    Health  -
-----------------------------------
-  Wonder Woman                90  -
-----------------------------------
-  Batman                      80  -
-----------------------------------
-  The Joker                   80  -
-----------------------------------
-  Superman                   100  -
-----------------------------------
-  Catwoman                    50  -
-----------------------------------
-  Aquaman                     30  -
-----------------------------------
-  Iron Man                    50  -
-----------------------------------
-  Hulk                        80  -
-----------------------------------
-  Black Widow                100  -
-----------------------------------
===================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
80
The Joker
80
Superman
100
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
80
Black Widow
100
```

Sample output 5:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list

===================================
-        Character Summary         -
===================================
-  Name                    Health  -
-----------------------------------
```

```
-  Wonder Woman                90  -
-------------------------------------
-  Batman                      80  -
-------------------------------------
-  The Joker                   80  -
-------------------------------------
-  Superman                   100  -
-------------------------------------
-  Catwoman                    50  -
-------------------------------------
-  Aquaman                     30  -
-------------------------------------
-  Iron Man                    50  -
-------------------------------------
-  Hulk                        80  -
-------------------------------------
=====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: remove

Please enter name: Thor

Thor is not found in character list.


Please enter choice
[list, search, reset, add, remove, battle, quit]: remove

Please enter name: Superman

Successfully removed Superman


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

=====================================
-        Character Summary          -
=====================================
-  Name                     Health  -
-------------------------------------
-  Wonder Woman                90  -
-------------------------------------
-  Batman                      80  -
-------------------------------------
-  The Joker                   80  -
-------------------------------------
-  Catwoman                    50  -
-------------------------------------
-  Aquaman                     30  -
-------------------------------------
-  Iron Man                    50  -
-------------------------------------
-  Hulk                        80  -
-------------------------------------
=====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
80
The Joker
80
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
80
```

Sample output 6:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list
```

```
====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman             90  -
------------------------------------
-  Batman                   80  -
------------------------------------
-  The Joker                80  -
------------------------------------
-  Superman                100  -
------------------------------------
-  Catwoman                 50  -
------------------------------------
-  Aquaman                  30  -
------------------------------------
-  Iron Man                 50  -
------------------------------------
-  Hulk                     80  -
------------------------------------
====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: battle

Please enter opponent one's name: Thor
Thor is not found in character list - please enter another opponent!

Please enter opponent one's name: Hulk
Please enter opponent two's name: Spiderman
Spiderman is not found in character list - please enter another opponent!

Please enter opponent one's name: The Joker
Please enter number of battle rounds: 10
Must be between 1-5 inclusive.

Please enter number of battle rounds: 2


-- Battle --

Hulk versus The Joker - 2 rounds

Round: 1
  > Hulk - Damage: 32 - Current health: 48
  > The Joker - Damage: 14 - Current health: 66

Round: 2
  > Hulk - Damage: 21 - Current health: 27
  > The Joker - Damage: 5 - Current health: 61

-- End of battle --

** The Joker wins! **


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                    Health  -
------------------------------------
-  Wonder Woman             90  -
------------------------------------
-  Batman                   80  -
------------------------------------
-  The Joker                61  -
------------------------------------
-  Superman                100  -
------------------------------------
-  Catwoman                 50  -
------------------------------------
-  Aquaman                  30  -
------------------------------------
-  Iron Man                 50  -
------------------------------------
-  Hulk                     27  -
------------------------------------
====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
```

```
Batman
80
The Joker
61
Superman
100
Catwoman
50
Aquaman
30
Iron Man
50
Hulk
27
```

Sample output 7:

```
File     : wayby001_partII.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
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                     Health  -
------------------------------------
-  Wonder Woman               90  -
------------------------------------
-  Batman                     80  -
------------------------------------
-  The Joker                  80  -
------------------------------------
-  Superman                  100  -
------------------------------------
-  Catwoman                   50  -
------------------------------------
-  Aquaman                    30  -
------------------------------------
-  Iron Man                   50  -
------------------------------------
-  Hulk                       80  -
------------------------------------
====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: battle

Please enter opponent one's name: Batman
Please enter opponent two's name: Superman
Please enter number of battle rounds: 2


-- Battle --

Batman versus Superman - 2 rounds

Round: 1
  > Batman - Damage: 21 - Current health: 59
  > Superman - Damage: 18 - Current health: 82

Round: 2
  > Batman - Damage: 49 - Current health: 10
  > Superman - Damage: 43 - Current health: 39

-- End of battle --

** Superman wins! **


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

====================================
-        Character Summary         -
====================================
-  Name                     Health  -
------------------------------------
-  Wonder Woman               90  -
------------------------------------
-  Batman                     10  -
------------------------------------
-  The Joker                  80  -
------------------------------------
-  Superman                   39  -
------------------------------------
-  Catwoman                   50  -
------------------------------------
-  Aquaman                    30  -
```

```
-------------------------------------
-  Iron Man                   50  -
-------------------------------------
-  Hulk                       80  -
-------------------------------------
=====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: battle

Please enter opponent one's name: Aquaman
Please enter opponent two's name: Hulk
Please enter number of battle rounds: 5


-- Battle --

Aquaman versus Hulk - 5 rounds

Round: 1
  > Aquaman - Damage: 44 - Current health: 0
  > Hulk - Damage: 33 - Current health: 47

-- End of battle --

-- Aquaman has died!  :(

** Hulk wins! **


Please enter choice
[list, search, reset, add, remove, battle, quit]: list

=====================================
-        Character Summary          -
=====================================
-  Name                    Health  -
-------------------------------------
-  Wonder Woman               90  -
-------------------------------------
-  Batman                     10  -
-------------------------------------
-  The Joker                  80  -
-------------------------------------
-  Superman                   39  -
-------------------------------------
-  Catwoman                   50  -
-------------------------------------
-  Aquaman                     0  -
-------------------------------------
-  Iron Man                   50  -
-------------------------------------
-  Hulk                       47  -
-------------------------------------
=====================================

Please enter choice
[list, search, reset, add, remove, battle, quit]: quit


-- Program terminating --


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

Wonder Woman
90
Batman
10
The Joker
80
Superman
39
Catwoman
50
Aquaman
0
Iron Man
50
Hulk
47
```

USEFUL BUILT-IN PYTHON FUNCTIONS – 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
```