COMP 1039

Problem Solving and Programming

Programming Assignment 1

UniSA STEM
The University of South Australia
March 2024

Contents

INTRODUCTION

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

The assignment is intended to provide you with the opportunity to put into practice what you have learnt in the course by applying your knowledge and skills to the implementation of a game called **Dragon Battleground.**

This assignment is an **individual task** that will require an **individual submission (i.e. the work that you submit MUST be your own)**. If you are an **internal student,** you will be required to submit your work via learnonline before **Monday 22 April, 9.00 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

You are required to write a Python program that allows a player to play a text-based **game of Dragon Battleground**. The program will allow a player to play as many games of Dragon Battleground as they wish. Once the player chooses to stop playing, the program will report the game summary to the screen.

Please ensure that you read the section titled 'Dragon Battleground 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).

DRAGON BATTLEGROUND SPECIFICATION – PART 1

You are required to write a Python program called *yourEmailId*_battle_p1.py that allows a player to play a text-based **game of Dragon Battleground**. The program will allow a player to play as many games of Dragon Battleground as they wish.

Dragon Battleground

In Dragon Battleground, a player goes into battle with a dragon. One battle may have many (1-5 inclusive) rounds. The player is able to go into battle with a dragon as many times as they wish. Once the player chooses to stop playing (battling dragons), the program will report the game summary to the screen.

If the player chooses to battle a dragon, the number of battle rounds the player will undertake (a number between 1-5 inclusive) is prompted for and read. One battle may have many (1-5 inclusive) rounds. The player battles a dragon until either an opponent dies (their health value is reduced to zero) or the number of battle rounds have been completed. For each individual battle round, the player and dragon will sustain a certain amount of damage to their health rating. The amount of damage sustained from the battle round will be determined by the rolling of five six-sided dice. Each battle round will involve the player rolling five six-sided dice to deal damage to the dragon. The dragon will then attack the player by rolling five six-sided dice to deal damage to the player. During each round, the opponents damage dealt (i.e. calculated by rolling five six-sided dice) is displayed to the screen. Each opponents' damage taken (i.e. calculated by rolling five six-sided dice) and current health value (i.e. calculated by: health value – damage taken) 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) and the opponents' battle statistics are updated (i.e. number of battles, no won, no lost, etc…) accordingly.

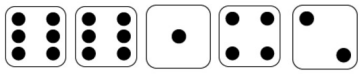
This will repeat until the player chooses to stop battling dragons. Once the player chooses to stop playing, the program will report the game summary to the screen.

We will be adhering to the following rules and game play for the assignment.

Dragon Battleground Game Play and Rules:

- To begin, the player is asked to enter the number of battle rounds (1-5 inclusive).
- The player and the dragon start each new battle with a health value of 100.
- For each battle round, the player rolls five six-sided dice to deal damage to the dragon and the dragon rolls five six-sided dice to deal damage to the player.
- The player and the dragon rolls are displayed to the screen along with the damage dealt in that round. The sum of the five die values will be the damage inflicted on the player and the dragon. To make things interesting, both the player and the dragon have the chance to score double the damage (a hit), triple the damage (a critical hit) or miss their attacks.
    - If two of the same die value are rolled (a pair), the damage dealt (calculated by summing the five die values) is doubled.
    - If three of the same die value are rolled (three of a kind), the damage inflicted is tripled resulting in a critical hit.
    - However, if three ones, three threes or three fives are rolled, the attack will miss resulting in 0 damage being dealt.

- Example rolls (damage dealt):

Roll	Description	Example
Three of a kind (critical hit)	Three dice have the same face value and the other two have different values. Triple the damage dealt. Damage dealt: 54	
Three of a kind (swing and miss)	Three dice have the same face value and the other two have different values. If three of a kind is 1, 3 or 5, then no damage inflicted. Damage dealt: 0	
Pair (hit)	Two dice have the same value and the others have different values. Double the damage dealt. Damage dealt: 38	
Standard (regular damage)	All five dice have different values or combination is not as listed above (i.e. critical hit, swing and miss, or hit). Damage dealt: 17	

- The player and dragon's damage taken and current health are displayed after every round.
- The player battles a dragon until either an opponent dies (i.e. their health value is reduced to zero) or the number of battle rounds have been completed.
- At the end of the battle (after all rounds have been completed or an opponent has died), the opponent with the highest health value wins the battle! If both the dragon and the player have equal health status, the battle is a draw. If one opponent dies in battle, the other opponent wins.
- Once the player no longer wishes to battle dragons, the game summary is displayed to the screen (i.e. games played, won, lost, drawn and dragons killed in battle).

You do not have to write the code that displays the die face values to the screen, a module containing a function that does that for you has been provided. The `dice.py` file is a module that contains a function called `display_dice()` that displays the face values of the dice to the screen for you. You are required to use this as part of this assignment, however, **please do not modify the `dice.py` file.**

PRACTICAL REQUIREMENTS – PART 1

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

- Your solution in a file called *yourEmailId*_battle_p1.py.
- Appropriate and well-constructed `while` and/or `for` loops (as necessary).
- Appropriate `if, if-else, if-elif-else` statements (as necessary).
- The use of `random.randint(1,6)` function in order to simulate the roll of a six sided die.
- The supplied `dice.py` module (containing the `display_dice` function). This is provided for you – **please DO NOT modify this file**.
- A list to represent/store the player's roll (damage dealt), i.e. player_roll = [0, 0, 0, 0, 0]
- A list to represent/store the dragon's roll (damage dealt), i.e. dragon_roll = [0, 0, 0, 0, 0]
- A list to store how many times each die face value was rolled.
- A loop in order to count how many times each die face value was rolled.
- Output that **strictly** adheres 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.
- 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 (no single letter identifier names).

Your solutions **MAY** use:

- You may make use of the `print(), input(), int(), len()` and `range()` built-in functions.
- You may make use of the `list.append()` method.
- Access the individual elements in a list with an index (one element only). i.e. `list_name[index]`.

Your solutions **MUST NOT** use:

- `break` or `continue` statements in your solution. **Do not** use the `quit()` or `exit()` functions or the `break` or `return` statements (or any other techniques) as a way to break out of loops. Doing so will result in a significant mark deduction.
- List or String **methods** in order to count how many times each die face value was rolled or to sum the list.
- Built-in functions in order to count how many times each die face value was rolled or to sum the list.
- The enumerate() function.
- List comprehensions.
- Dictionary to store data items.
- Your own (user-defined) functions.

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

Please ensure that you use Python 3.12.2 or a later version (i.e. the latest version) in order to complete your assignments. Your programs **MUST** run using Python 3.12.2 (or latest version).

STAGES – PART 1

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 the `dice.py` file for this assignment. This has been provided for you. Please download this file from the course website (Assessment tab) and ensure that it is in the same directory as the *`yourEmailId`*`_battle_p1.py` file.

Test to ensure that this is working correctly by entering the following in your *`yourEmailId`*`_battle_p1.py` file:

```
import dice

player_roll = [2, 3, 4, 5, 6]

print("Player rolled:")
dice.display_dice(player_roll)
```

Run the *`yourEmailId`*`_battle_p1.py` file. If this is working correctly, you should now see the following output in the Python shell when you run your program:

```
Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [2]       [3]       [4]       [5]       [6]
                *         *         * *       * *       * *
                           *                   *         * *
                  *        *         * *       * *       * *
```

Note, this is for developmental purposes only, and you will need to modify and correctly position the above code.

Stage 2

Create a list to store the player's roll, i.e. the five dice values. For example:

```
player_roll = [0, 0, 0, 0, 0]   …or…
player_roll = []
```

Add code to simulate the rolling of five dice, i.e. roll player's damage dealt. That is, you should generate five random numbers, each in the range of 1 through 6 (inclusive) and assign each number to a list element. Use the `random.randint(1,6)` function to simulate the roll of a die. Hint: Use a loop in order to assign each random number (die roll) to a list element.

Display the player's roll (i.e. the randomly generated dice roll) to the screen, e.g.:

```
:
# Place code to randomly generate the roll of five dice here…
:
:
# Display player's roll to the screen.
print("Player rolled:")
dice.display_dice(player_roll)
```

Sample output (this will look different given we are generating random values here):

```
Player rolled:
               Die 1      Die 2      Die 3      Die 4      Die 5
                [3]        [6]        [1]        [4]        [2]
                *          * *                   * *        *
                 *         * *          *
                  *        * *                   * *          *
```

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

Stage 3

Add code to count how many times each die face value was rolled. **Hint**: You **MUST** use a list in order to store this information.

To define a list in order to count how many times each die face value was rolled:

```
die_count = [0, 0, 0, 0, 0, 0, 0]
```

die_count

0	0	0	0	0	0	0
0	1	2	3	4	5	6

Given that die face values are 1 – 6 inclusive, we create a list with seven elements but ignore the zero element of the list. This will make it easier to increment the appropriate list element. That is, die_count[1] should contain the number of 1s that were rolled, die_count[2] the number of 2s rolled, etc.

For example: In the example provided in stage 1, the player's roll is assigned the following dice values: 5, 2, 1, 1, 6. In light of this, the die_count should be as follows:

die_count

0	2	1	0	0	1	1
0	1	2	3	4	5	6

To access and update the appropriate list element (using the value of the die as an index):

```
die_value = player_roll[0]
die_count[die_value] = die_count[die_value] + 1
```

For example: If die_value is assigned the value 3.

die_count

0	0	0	1	0	0	0
0	1	2	3	4	5	6

Hint: You **MUST** use a loop in order to count how many times each die face value was rolled. In the example above, this would mean that the index in the statement **player_roll[0] would need to be modified if placed within a loop**.

Stage 4

Calculate the damage dealt by the player's roll by summing the five die values. You MUST use a loop in order to sum the five die values. Determine the damage dealt by the player's roll as described in the section 'Dragon Battleground Game Play and Rules'. Hint: Use the `die_count` list that holds how many times each die face value was rolled in order to determine the damage dealt.

Stage 5

Given the damage calculated in stage 4, display the corresponding text and damage dealt to the screen (as seen in the sample output at the end of this handout).

Stage 6

Add code to simulate the dragon's roll (i.e. dragon's damage dealt). That is, repeat steps 1 – 5 in order to do this for the dragon.

Stage 7

Add code to display the player's damage taken and current health and the dragon's damage taken and current health to the screen (as seen in the sample output at the end of this handout).

Stage 8

Add code to determine whether the player wins, loses or draws with the dragon. Display the appropriate message to the screen, i.e.:

- `** Player wins! **`
- `** Dragon wins! **`
- `** Draw! **`

Don't forget to check to see whether the player has died in battle.

Stage 9

Now… it's time to allow the player to play more than one game. Let's add a loop that loops until the user enters 'n' (to stop playing the game). Think about where this code should go – what needs to be repeated, etc.

Stage 10

Allow the player to enter the number of battle rounds (per game). Let's add a loop that loops until either the player or the dragon dies (i.e. their health value is reduced to zero) or the number of battle rounds have been completed. Think about where this code should go – what needs to be repeated, etc.

Stage 11

Add code to validate **all** user input:

- Would you like to play Dragon Battleground [y|n]?

  Sample output:

```
Would you like to play Dragon Battleground [y|n]? p
Please enter either 'y' or 'n'.

Would you like to play Dragon Battleground [y|n]? y
```

- Please enter number of battle rounds:

  Sample output:

```
Please enter number of battle rounds: 9
Must be between 1-5 inclusive.

Please enter number of battle rounds: -1
Must be between 1-5 inclusive.

Please enter number of battle rounds: 2
```

- Play again [y|n]?

  Sample output:

```
Play again [y|n]? z
Please enter either 'y' or 'n'.

Play again [y|n]? n
```

Stage 12

Add code to keep track of how many games were played, won, lost, drawn and dragons killed in battle. Display this to the screen (as seen in the sample output at the end of this handout).

Stage 13 – THIS IS IMPORTANT!

Finally, check the **sample output (see section titled 'Sample Output') and if necessary, modify your code so that:**

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

DRAGON BATTLEGROUND SPECIFICATION – PART 2

You are required to write a Python program called *yourEmailId*_battle_p2.py that allows a player to play **a game of Dragon Battleground**. The program will allow a player to play as many games of Dragon Battleground as they wish. Please refer to section titled 'Dragon Battleground Specification – Part 1' for full specification.

You are to use your code solution from part 1 for part 2 of this assignment. **Please copy your code solution from part 1 to the file called *yourEmailId*_battle_p2.py and edit that file for part 2. Please note**: **DO NOT** edit the code contained in the file *yourEmailId*_battle_p1.py. **The code contained in file *yourEmailId*_battle_p1.py should remain unchanged as you are required to submit two files; the solution for part 1 and the solution for part 2.**

You will be required to submit two files as follows:

- yourEmailId_battle_p1.py (which contains your code solution for part 1 with **no** user-defined functions).
- yourEmailId_battle_p2.py (which contains your code solution for part 1 with the user-defined functions described in part 2).

PRACTICAL REQUIREMENTS – PART 2

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

- Your solution in a file called *yourEmailId*_battle_p2.py.
- Your code solution from part 1. Please refer to section titled 'Practical Requirements - Part 1' for full practical requirements. Copy your code solution from part 1 to a file called *yourEmailId*_battle_p2.py and edit that file for part 2.
- The following four functions (refer to stage 1 for the description of functions):
    - display_details()
    - roll_die()
    - roll_damage(max_dice)
    - calculate_damage(roll)

STAGES – PART 2

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

Copy your code solution from part 1 to a file called *yourEmailId*_battle_p2.py.

Modify your code solution from part 1 (in file *yourEmailId*_battle_p2.py) to include and make use of the following **four** functions:

- `display_details()`
- `roll_die()`
- `roll_damage(max_dice)`
- `calculate_damage(roll)`

DO NOT modify your code solution in the file called *yourEmailId*_battle_p1.py. Make the following changes to the *yourEmailId*_battle_p2.py file only.

1. Write a function called `display_details()` that will display your details to the screen. The function takes no parameters and does not return a value. The function simply displays your details to the screen. Your function should produce the following output (with your details).

   Output:
   ```
   File     : wayby001_battle.py
   Author   : Batman
   Stud ID  : 0123456X
   Email ID : wayby001
   This is my own work as defined by the
   University's Academic Misconduct Policy.
   ```

2. Write a function called `roll_die()` that simulates the rolling of one die. The function will generate a random number in the range 1 – 6 (the face value on the die are 1 – 6 inclusive). The function takes no parameters and returns the random number generated.

   For example:
   ```
   die = roll_die()
   ```

3. Write a function called `roll_damage()` that takes the size of the list to create as a parameter. The function generates five (size) random numbers, each in the range of 1 through 6 (inclusive) and assigns each number to a list element, i.e. the player/dragon's roll. The function must call the roll_die() function to simulate the roll of a die. Hint: Use a loop in order to assign each random number (die roll) to a list element. The function returns a list containing the randomly generated numbers (i.e. the player/dragon's roll).

4. Write a function called `calculate_damage()` that takes a list of integers (i.e. player/dragon's roll) as a parameter. The function determines the damage dealt by summing the die values stored in the list and based on the combination of die values stored in the list. The function displays the text (Swing and miss – no damage inflicted!, Critical hit – triple the damage!, Hit – double the damage! as appropriate and depending on the die values stored in the list). The function returns the damage dealt by the roll depending on the combination of die values as described in the section 'Dragon Battleground Game Play and Rules'.

Reminder:
Defining a function does not execute the function – you will need to call the function(s) from the appropriate place(s) from within your program (some functions, more than once).

SUBMISSION DETAILS

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

- You are required to submit an electronic copy of your program via learnonline **before Monday 22 April, 9.00 am.**

All students must follow the submission instructions below:

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

Do NOT zip (or rar or the like) your files.

Ensure that the following files are included in your submission:

- `yourEmailId_battle_p1.py (which contains your code solution for part 1 with no user-defined functions)`
- `yourEmailId_battle_p2.py (which contains your code solution for part 1 with user-defined functions)`

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

- `bonjy007_battle_p1.py`
- `bonjy007_battle_p2.py`

All files that you submit must include the following comments:

```
#
# File: fileName.py
# Author: your name
# Email Id: your email id
# Description: Assignment 1 - 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 due date**. Work that has not been correctly submitted to learnonline may not be marked.

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

EXTENSIONS AND LATE SUBMISSIONS

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

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

Students are reminded that they should be aware of what constitutes academic misconduct. Information about academic integrity and what constitutes academic misconduct can be found in the Academic Integrity Policy and Procedure (https://i.unisa.edu.au/policies-and-procedures/university-policies/academic/ab-69).

Deliberate academic misconduct (such as plagiarism, contract cheating, the use of ChatGPT (or similar tool)) is subject to penalties as outlined in the **Academic Integrity Policy and Procedure**.

You are **NOT** permitted to use ChatGPT (or other similar Artificial Intelligence tools) for your assessment items in this course. **Please note**: as stated in the University's Academic Integrity Policy (Policy AB-69), academic misconduct includes (but is not limited to) the use of artificial intelligence (AI) software or paraphrasing tools as a form of contract cheating. Please refer to the Academic Integrity Policy (Policy AB-69) for further information.

MARKING CRITERIA

University of South Australia

Assessment feedback

COMP 1039 Problem Solving and Programming – Assignment 1, SP2, 2024

NAME:	AVAILABLE MARKS	MARK	COMMENT
PRODUCES CORRECT RESULTS (OUTPUT) – PART 1	**52 MARKS**		
	☐ Line spacing (3 marks)		
File : wayby001_battle.py Author : Batman Email ID : wayby001 This is my own work as defined by the University's Academic Misconduct Policy.	☐ Details display (2 marks)		
Would you like to play Dragon Battleground [y|n]? y	☐ Prompt [y|n] (2 marks)		
Please enter number of battle rounds: 5	☐ Prompt rounds (2 marks)		
-- Battle -- Player versus Dragon: 5 rounds --	☐ Text display (1 mark) ☐ Correct rounds (1 mark)		
Round: 1	☐ Text display (1 mark) ☐ Correct rounds (1 mark)		
Player rolled: Die 1 Die 2 Die 3 Die 4 Die 5 [3] [2] [1] [2] [4] * * * * * * * * * * * * -- Hit - double the damage! -- Player has dealt 24 damage -- Hit - double the damage! -- Critical hit - triple the damage! -- Swing and miss - no damage inflicted!	☐ Player display (1 mark) ☐ Dice display (1 mark) ☐ Dealt text (1 mark) ☐ Correct damage (2 mks) ☐ Hit display (1 mark) ☐ Critical display (1 mark) ☐ Miss display (1 mark)		
Dragon rolled: Die 1 Die 2 Die 3 Die 4 Die 5 [2] [5] [2] [2] [6] * * * * * * * * * * * * -- Critical hit - triple the damage! -- Dragon has dealt 51 damage -- Hit - double the damage! -- Critical hit - triple the damage! -- Swing and miss - no damage inflicted!	☐ Dragon display (1 mark) ☐ Dice display (1 mark) ☐ Dealt text (1 mark) ☐ Correct damage (2 mks) ☐ Hit display (1 mark) ☐ Critical display (1 mark) ☐ Miss display (1 mark)		
> Player - Damage taken: 51 - Current health: 49 > Dragon - Damage taken: 24 - Current health: 76	☐ Correct player text (1) ☐ Correct damage (1) ☐ Correct health (1) ☐ Correct dragon text (1) ☐ Correct damage (1) ☐ Correct health (1)		
-- End of battle --	☐ End display (1 mark)		

`-- Player has died! :(`
 	
`-- Dragon has died! :(`
	☐ Player died msg (1 mark)
	
	☐ Dragon died msg (1 mark)		

`** Player wins! **`
`** Dragon wins! **`
`** Draw! **`
	☐ Correct layout (1 mark)
	☐ Player wins msg (1 mark)
	☐ Dragon win msg (1 mark)
	☐ Draw msg (1 mark)		

`Play again [y|n]? n`	☐ Prompt [y|n] (2 marks)		

`Game Summary`
`============`
`You played 3 games`
`  |--> Games won: 1`
`  |--> Games lost: 2`
`  |--> Games drawn: 0`
`  |--> Dragons killed: 1`
	☐ Game Sum layout (1 mk)
	☐ Game Sum text (1 mk)
	
	☐ Played value (1 mark)
	☐ Won value (1 mark)
	☐ Lost value (1 mark)
	☐ Drawn value (1 mark)
	☐ Killed value (1 mark)		

`Thanks for playing!`	☐ Thanks display (1 mark)		
ADHERES TO SPECIFICATIONS (CODE) – PART 1		**COMMENT**	
While loop for play again, i.e. while play == 'y' (or equivalent)		☐ -2 No or incorrect while	
While loop for rounds (or equivalent) i.e. while count <= battle_rounds and player_health > 0 and dragon_health > 0:		☐ -2 No or incorrect while	
Use of `random.randint(1,6)` or similar for die roll Use of provided `display_dice(roll)` function.		☐ -2 No or incorrect use of randint() ☐ -2 No or incorrect use of display_dice() function	
List of integers to represent player and dragon rolls		☐ -1 Not to spec or -2 not implemented	
List to store how many times each die face value rolled		☐ -1 Not to spec or -2 not implemented	
Loop to count how many times each die face value rolled		☐ -1 Not to spec or -2 not implemented	
Should **not use** the following: • Built-in functions to count die face value rolled or sum list • List or String **methods** to count die face value rolled • enumerate() function • List comprehensions • Dictionary to store data items		☐ -5 use of built-in functions ☐ -5 use of methods to count ☐ -10 use of enumerate() ☐ -10 use of list comprehensions ☐ -10 use of dictionary	

Validation of user input messages (validate [y|n] choice only. Not required to check for numeric input or strings when expecting characters, etc):
 	
`Would you like to play Dragon Battleground [y|n]? p`
`Please enter either 'y' or 'n'.`
 	
`Please enter number of battle rounds: 9`
`Must be between 1-5 inclusive.`
 	
`Play again [y|n]? p`
`Please enter either 'y' or 'n'.`
		☐ -2 No validation of play
		
		☐ -2 No validation of rounds
		
		☐ -2 No validation of play again	

No user-defined functions		☐ -2 If using own functions	
Good loops (i.e. no break, continue, return or similar statements to exit loops)		☐ -2 For using break/return/etc statements to exit loops	

PART 2	AVAILABLE MARKS	MARK	COMMENT
ADHERES TO SPECIFICATIONS (CODE) – PART 2	**12 MARKS**		
Function `display_details()` *no return* Function `roll_die()` *returns random number* Function `roll_damage(max_dice)` *returns list of 5 random numbers (i.e. player/dragon roll)* Function `calculate_damage(hand)` *returns damage*	☐ 2 Correct details function ☐ 2 Correct roll function ☐ 3 Correct damage funct. ☐ 5 Correct calc function *-1 For **each** function that does not adhere to the specs*		
PRODUCES CORRECT RESULTS (OUTPUT) – PART 2		**COMMENT**	
Functionality and output as per part 1		☐ -2 If does not adhere to part 1 functionality ☐ -2 If does not adhere to part 1 output	

STYLE (BOTH PARTS)	6 MARKS	MARK	COMMENT
Comments: ○ your details at top of file ○ program description ○ all variable definitions ○ all function definitions ○ significant sections of code			☐ -2 Insufficient comments ○ your details at top of file, ○ program description, ○ all variable definitions, ○ functions, and ○ significant sections of code
Consistent code layout and indentation.			☐ -2 Inconsistent indentation and layout
Meaningful variable names (no single letter variable names).			☐ -2 Non-descriptive variable names
TOTAL	**70 MARKS**		

The Graduate qualities being assessed by this assignment are indicated by an X:

X	GQ1: operate effectively with and upon a body of knowledge		GQ5: are committed to ethical action and social responsibility
	GQ2: are prepared for lifelong learning	X	GQ6: communicate effectively
	GQ3: are effective problem solvers		GQ7: demonstrate an international perspective
	GQ4:can work both autonomously and collaboratively		

This form meets the 2006 requirements of UniSA's Code of Good Practice: Student Assessment

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).

SAMPLE OUTPUT – PART 1 AND PART 2

Sample output 1:

```
File     : wayby001_battle.py
Author   : Batman
Email ID : wayby001
This is my own work as defined by the
University's Academic Misconduct Policy.

Would you like to play Dragon Battleground [y|n]? n


No worries... you live to battle another day... :)
```

Sample output 2:

```
File     : wayby001_battle.py
Author   : Batman
Email ID : wayby001
This is my own work as defined by the
University's Academic Misconduct Policy.

Would you like to play Dragon Battleground [y|n]? p
Please enter either 'y' or 'n'.

Would you like to play Dragon Battleground [y|n]? y

Please enter number of battle rounds: -2
Must be between 1-5 inclusive.

Please enter number of battle rounds: 7
Must be between 1-5 inclusive.

Please enter number of battle rounds: 1


-- Battle -- Player versus Dragon: 1 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [3]       [2]       [3]       [6]       [2]
                *         *         *         * *       *
                 *                   *        * *
                  *         *         *       * *         *

-- Hit - double the damage!
-- Player has dealt 32 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [2]       [2]       [1]       [6]       [2]
                *         *                   * *       *
                                      *        * *
                  *         *                 * *         *

-- Critical hit - triple the damage!
-- Dragon has dealt 39 damage

> Player - Damage taken: 39 - Current health: 61
> Dragon - Damage taken: 32 - Current health: 68


-- End of battle --

** Dragon wins! **


Play again [y|n]? n


Game Summary
============
You played 1 games
  |--> Games won:    0
  |--> Games lost:   1
```

```
  |--> Games drawn:  0
  |--> Dragons killed:  0

Thanks for playing!
```

Sample output 3:

```
File     : wayby001_battle.py
Author   : Batman
Email ID : wayby001
This is my own work as defined by the
University's Academic Misconduct Policy.

Would you like to play Dragon Battleground [y|n]? y

Please enter number of battle rounds: 2


-- Battle -- Player versus Dragon: 2 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [4]       [4]       [6]       [2]       [1]
                * *       * *       * *       *
                                    * *                   *
                * *       * *       * *         *

-- Hit - double the damage!
-- Player has dealt 34 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [2]       [6]       [4]       [5]       [6]
                *         * *       * *       * *       * *
                          * *                   *         * *
                  *       * *       * *       * *       * *

-- Hit - double the damage!
-- Dragon has dealt 46 damage

> Player - Damage taken: 46 - Current health: 54
> Dragon - Damage taken: 34 - Current health: 66


Round: 2

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [3]       [5]       [6]       [2]       [2]
                *         * *       * *       *         *
                 *         *        * *
                  *       * *       * *         *         *

-- Hit - double the damage!
-- Player has dealt 36 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [1]       [1]       [6]       [6]
                * *                           * *       * *
                * *         *         *       * *       * *
                * *                           * *       * *

-- Critical hit - triple the damage!
-- Dragon has dealt 60 damage

> Player - Damage taken: 60 - Current health: 0
> Dragon - Damage taken: 36 - Current health: 30


-- End of battle --

-- Player has died!  :(

** Dragon wins! **


Play again [y|n]? n
```

```
Game Summary
============
You played 1 games
  |--> Games won:    0
  |--> Games lost:   1
  |--> Games drawn:  0
  |--> Dragons killed:  0

Thanks for playing!
```

Sample output 4:

```
File     : wayby001_battle.py
Author   : Batman
Email ID : wayby001
This is my own work as defined by the
University's Academic Misconduct Policy.

Would you like to play Dragon Battleground [y|n]? y

Please enter number of battle rounds: 5


-- Battle -- Player versus Dragon: 5 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [1]       [5]       [1]       [5]       [1]
                          * *                 * *
                 *         *         *         *         *
                          * *                 * *

-- Swing and miss - no damage inflicted!
-- Player has dealt 0 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [5]       [1]       [1]       [4]       [4]
                * *                           * *       * *
                 *         *         *
                * *                           * *       * *

-- Hit - double the damage!
-- Dragon has dealt 30 damage

> Player - Damage taken: 30 - Current health: 70
> Dragon - Damage taken: 0 - Current health: 100


Round: 2

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [5]       [5]       [5]       [1]       [2]
                * *       * *       * *                 *
                 *         *         *         *
                * *       * *       * *                   *

-- Swing and miss - no damage inflicted!
-- Player has dealt 0 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [4]       [6]       [1]       [3]       [3]
                * *       * *                 *         *
                          * *         *         *         *
                * *       * *                   *         *

-- Hit - double the damage!
-- Dragon has dealt 34 damage

> Player - Damage taken: 34 - Current health: 36
> Dragon - Damage taken: 0 - Current health: 100


Round: 3

Player rolled:
```

```
               Die 1     Die 2     Die 3     Die 4     Die 5
                [1]       [2]       [3]       [6]       [4]
                          *         *         * *       * *
                 *                  *         * *
                             *       *       * *       * *

-- Player has dealt 16 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [3]       [5]       [1]       [2]
                * *       *         * *                 *
                * *         *         *         *
                * *           *       * *                 *

-- Dragon has dealt 17 damage

> Player - Damage taken: 17 - Current health: 19
> Dragon - Damage taken: 16 - Current health: 84


Round: 4

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [1]       [1]       [2]       [3]       [6]
                                    *         *         * *
                 *         *                  *         * *
                                       *         *       * *

-- Hit - double the damage!
-- Player has dealt 26 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [3]       [6]       [1]       [1]       [2]
                *         * *                           *
                 *        * *       *         *
                   *      * *                           *

-- Hit - double the damage!
-- Dragon has dealt 26 damage

> Player - Damage taken: 26 - Current health: 0
> Dragon - Damage taken: 26 - Current health: 58


-- End of battle --

-- Player has died!  :(

** Dragon wins! **


Play again [y|n]? y

Please enter number of battle rounds: 3


-- Battle -- Player versus Dragon: 3 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [1]       [4]       [2]       [6]
                * *                 * *       *         * *
                * *         *                           * *
                * *                 * *         *       * *

-- Hit - double the damage!
-- Player has dealt 38 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [5]       [1]       [2]       [4]       [2]
                * *                 *         * *       *
                 *         *
                * *                   *       * *         *

-- Hit - double the damage!
```

```
-- Dragon has dealt 28 damage

> Player - Damage taken: 28 - Current health: 72
> Dragon - Damage taken: 38 - Current health: 62


Round: 2

Player rolled:
                Die 1     Die 2     Die 3     Die 4     Die 5
                 [1]       [4]       [2]       [6]       [2]
                           * *       *         * *       *
                  *                            * *
                           * *         *       * *         *

-- Hit - double the damage!
-- Player has dealt 30 damage

Dragon rolled:
                Die 1     Die 2     Die 3     Die 4     Die 5
                 [1]       [3]       [4]       [4]       [3]
                           *         * *       * *       *
                  *         *                             *
                             *       * *       * *         *

-- Hit - double the damage!
-- Dragon has dealt 30 damage

> Player - Damage taken: 30 - Current health: 42
> Dragon - Damage taken: 30 - Current health: 32


Round: 3

Player rolled:
                Die 1     Die 2     Die 3     Die 4     Die 5
                 [1]       [6]       [4]       [6]       [2]
                           * *       * *       * *       *
                  *        * *                 * *
                           * *       * *       * *         *

-- Hit - double the damage!
-- Player has dealt 38 damage

Dragon rolled:
                Die 1     Die 2     Die 3     Die 4     Die 5
                 [4]       [2]       [1]       [4]       [5]
                 * *       *                   * *       * *
                                       *                   *
                 * *         *                 * *       * *

-- Hit - double the damage!
-- Dragon has dealt 32 damage

> Player - Damage taken: 32 - Current health: 10
> Dragon - Damage taken: 38 - Current health: 0


-- End of battle --

-- Dragon has died!  :(

** Player wins! **


Play again [y|n]? y

Please enter number of battle rounds: 2


-- Battle -- Player versus Dragon: 2 rounds --


Round: 1

Player rolled:
                Die 1     Die 2     Die 3     Die 4     Die 5
                 [4]       [2]       [4]       [3]       [5]
                 * *       *         * *       *         * *
                                                 *         *
                 * *         *       * *         *       * *
```

```
-- Hit - double the damage!
-- Player has dealt 36 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [3]       [3]       [4]       [4]       [4]
                *         *         * *       * *       * *
                 *         *
                  *         *       * *       * *       * *

-- Critical hit - triple the damage!
-- Dragon has dealt 54 damage

> Player - Damage taken: 54 - Current health: 46
> Dragon - Damage taken: 36 - Current health: 64


Round: 2

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [4]       [6]       [3]       [5]       [1]
                * *       * *       *         * *
                          * *        *         *         *
                * *       * *         *       * *

-- Player has dealt 19 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [6]       [3]       [1]       [4]
                * *       * *       *                   * *
                * *       * *        *         *
                * *       * *         *                 * *

-- Hit - double the damage!
-- Dragon has dealt 40 damage

> Player - Damage taken: 40 - Current health: 6
> Dragon - Damage taken: 19 - Current health: 45


-- End of battle --

** Dragon wins! **


Play again [y|n]? n


Game Summary
============
You played 3 games
  |--> Games won:    1
  |--> Games lost:   2
  |--> Games drawn:  0
  |--> Dragons killed:  1

Thanks for playing!
```

Sample output 5:

```
File     : wayby001_battle.py
Author   : Batman
Email ID : wayby001
This is my own work as defined by the
University's Academic Misconduct Policy.

Would you like to play Dragon Battleground [y|n]? y

Please enter number of battle rounds: 3


-- Battle -- Player versus Dragon: 3 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [1]       [2]       [6]       [2]       [2]
                          *         * *       *         *
```

```
                *                   * *
                          *         * *           *           *

-- Critical hit - triple the damage!
-- Player has dealt 39 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [5]       [4]       [3]       [4]
                * *       * *       * *       *         * *
                * *         *                   *
                * *       * *       * *         *       * *

-- Hit - double the damage!
-- Dragon has dealt 44 damage

> Player - Damage taken: 44 - Current health: 56
> Dragon - Damage taken: 39 - Current health: 61


Round: 2

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [2]       [2]       [4]       [4]
                * *       *         *         * *       * *
                * *
                * *         *         *       * *       * *

-- Hit - double the damage!
-- Player has dealt 36 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [6]       [2]       [1]       [5]
                * *       * *       *                   * *
                * *       * *                   *         *
                * *       * *         *                 * *

-- Hit - double the damage!
-- Dragon has dealt 40 damage

> Player - Damage taken: 40 - Current health: 16
> Dragon - Damage taken: 36 - Current health: 25


Round: 3

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [4]       [4]       [6]       [6]       [5]
                * *       * *       * *       * *       * *
                                    * *       * *         *
                * *       * *       * *       * *       * *

-- Hit - double the damage!
-- Player has dealt 50 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [1]       [6]       [1]       [6]       [5]
                          * *                 * *       * *
                 *        * *         *       * *         *
                          * *                 * *       * *

-- Hit - double the damage!
-- Dragon has dealt 38 damage

> Player - Damage taken: 38 - Current health: 0
> Dragon - Damage taken: 50 - Current health: 0


-- End of battle --

-- Player has died!  :(

-- Dragon has died!  :(

** Draw! **


Play again [y|n]? y
```

```
Please enter number of battle rounds: 2


-- Battle -- Player versus Dragon: 2 rounds --


Round: 1

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [5]       [6]       [2]       [2]       [1]
                * *       * *       *         *
                 *        * *                           *
                * *       * *         *         *

-- Hit - double the damage!
-- Player has dealt 32 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [6]       [2]       [4]       [5]       [2]
                * *       *         * *       * *       *
                * *                           *
                * *         *       * *       * *         *

-- Hit - double the damage!
-- Dragon has dealt 38 damage

> Player - Damage taken: 38 - Current health: 62
> Dragon - Damage taken: 32 - Current health: 68


Round: 2

Player rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [2]       [6]       [5]       [5]       [5]
                *         * *       * *       * *       * *
                          * *         *         *         *
                  *       * *       * *       * *       * *

-- Swing and miss - no damage inflicted!
-- Player has dealt 0 damage

Dragon rolled:
               Die 1     Die 2     Die 3     Die 4     Die 5
                [5]       [1]       [2]       [5]       [3]
                * *                 *         * *       *
                 *         *                  *         *
                * *                   *       * *         *

-- Hit - double the damage!
-- Dragon has dealt 32 damage

> Player - Damage taken: 32 - Current health: 30
> Dragon - Damage taken: 0 - Current health: 68


-- End of battle --

** Dragon wins! **


Play again [y|n]? n


Game Summary
============
You played 2 games
  |--> Games won:    0
  |--> Games lost:   1
  |--> Games drawn:  1
  |--> Dragons killed:  1

Thanks for playing!
```