**Saint Augustine’s College, Sydney**

**Harrison Dearing**

**Software Engineering Year 11: Connections Assessment - BOSSNNECTIONS**

Table of Contents

[Planning 3](#_Toc150971422)

[Task Definition 3](#_Toc150971423)

[Structure Chart 4](#_Toc150971424)

[Algorithm Design 5](#_Toc150971425)

[Flowchart 6](#_Toc150971426)

[Data Dictionary 7](#_Toc150971427)

[Implementation 8](#_Toc150971428)

[GitHub Repository 8](#_Toc150971429)

[Testing 9](#_Toc150971430)

[Test Table 9](#_Toc150971431)

[Release and Patch Notes 10](#_Toc150971432)

[Release 1.0.0 10](#_Toc150971433)

[Release 1.1.0 11](#_Toc150971434)

[Release 1.1.1 11](#_Toc150971435)

[Project Reflection 13](#_Toc150971436)

# Planning

## Task Definition

I have been assigned the task of developing ‘ConnectionsPY’, a command-line powered Python application for the New York Times Connections game, ensuring that the game is intuitive for players of all ages. I aim to create a bugless application with a user-friendly and engaging command-line interface by making it robust as well as using an assortment of features for improved display.

The functional requirements I will need to implement include:

Randomly select 4 categories with 4 words corresponding to each category from a predefined list at the start of each game.

Generate a 4 x 4 grid that displays the selected 16 words and places them in a randomized order.

Capture player guesses through a command line system of incorporating coordinates on the grid (e.g, 1,4 representing X,Y coordinates.

Validate player guesses and reveal correct guesses by floating guesses to the top in the pre-determined 4 colourised categories.

Track incorrect guesses, updating lives and ending the game when player reaches guess limits or all 4 categories are correct.

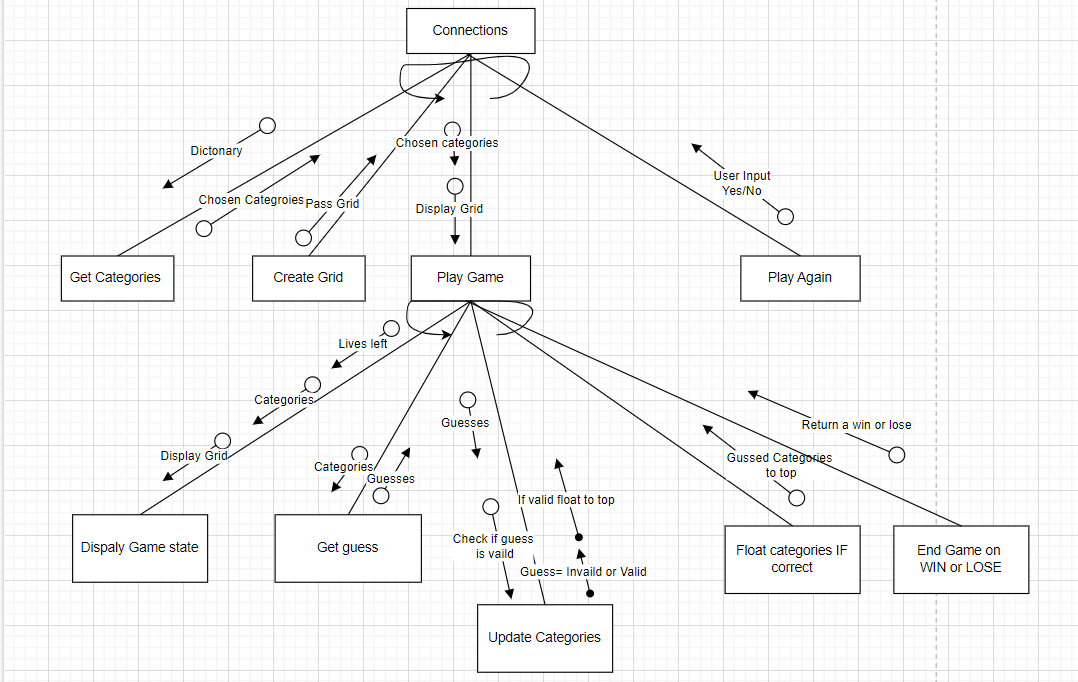
If incorrect, when game has ended with no guesses left reveal the answer by floating words into colourised categories.

Allow a new game to start once current game concludes using 4 new randomized categories.

## Structure Chart

As I will be taking a functional approach to the development of ‘ConnectionsPy’, it is important to create a structure chart that will decompose the game logic into a mainline and individual functions within, and help visualise the data/parameters that will be passed around.

The following flowchart maps out the functions within my program, a simple run down is here:



*This diagram was generated using* [*.drawio*](https://www.drawio.com/)*. It can be viewed as a template* [*here*](https://drive.google.com/file/d/1uzQsjF8thjtgjTTYEHFJa-khEq4BfrPz/view?usp=sharing)*.*

**Connections** will be the top-level mainline that starts the game.

**Get Categories** will be a function that creates Category for my game.

**Create Grid** will be a function that creates a 4x4 grid for my game.

**Play Game** is the main game loop where the gameplay occurs, including getting guesses and updating the game state and grid. It continues until the player runs out of attempts or guesses the word.

**Display Game State** will show the status of the word being guessed and the remaining attempts of the players.

**Get Guess**: This function simply gets 4 guesses from the player.

**Update Categories Checks** If the guess is invalid or valid and changes the category

**End Game on WIN or Lose** will determine whether the guessed categories have all been guessed correctly indicating a win, or they have been guessed incorrectly.

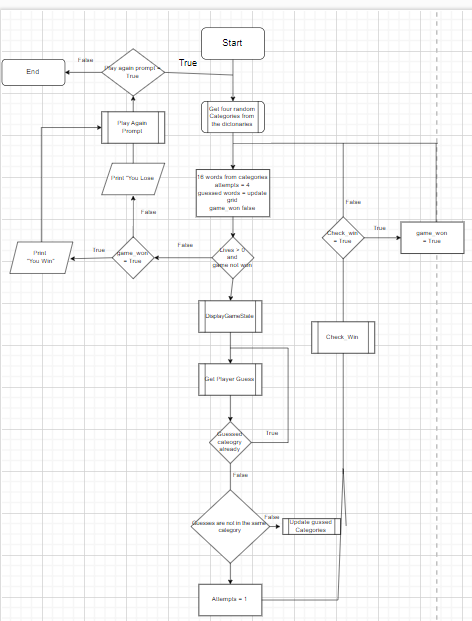
**Play Again Prompt**: After the game concludes, this asks the player if they would like to play again or stop playing.

## Algorithm Design

The mainline logic of the 'Connections' game proceeds as follows:

1. **Start**:
   * Start the game by initializing the four randomly chosen categories, the four attempts, and other necessary game states.
2. **Gameplay**:
   * Provides the user with instructions on how to play the game
   * Displays the original Grid with 12 randomized words
   * Asks for 4 words from a player
   * Begin the main game loop which continues until the player guesses all the correct words in the same category or runs out of attempts.
     + **Game Loop**:
       - Display the current game state using the Display Game State function.
       - Capture the player's guess with the Get Player Guess function.
       - If the guess is new, use the Update Guesses function to add it to the grid if correct.
       - Decrement attempts if the guess is incorrect.
       - Determine if the player has won with the Check Win function.
3. **Win/Loss Screen and Replay**:
   * Once out of the loop, display a win or loss message to the player
   * Prompt the player to play again using the Play Again Prompt function.
     + If the player chooses to replay, reset the game variables and restart the game.
     + End the game if the player decides not to continue.

## Flowchart

This algorithm's logic can be effectively illustrated through a flowchart to visually augment comprehension. While the detailed operations of the subfunctions are simplified, this overview should adequately convey the workings of the Bossnnections game.

*This diagram was generated using* [*.drawio*](https://www.drawio.com/)*.*

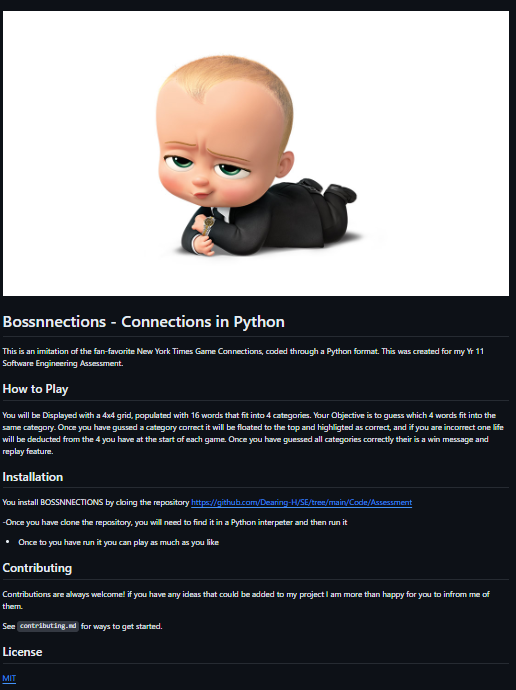
## Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable | Data Type | Format for display | Description | Example | Validation |
| Categories | Array | Text | List of all possible game categories | [ 'Axiom', 'Enigma', 'Paradox', 'Epiphany ] | Must be from Dictionary |
| Replay | String | Text | The string variable that stores the user’s choice to replay or not | Replay = ‘Yes’ | Ensure that the reply is either yes or no |
| Correct\_Words | List[String] | Text | This list holds the correct words guessed by a player | [‘Blue’, ‘Spalding’, | Ensures that the list contains strings that representing correct words |
| Lives | Integer | Numeric | Number of attempts for incorrect guesses | 3 Lives left | At 0 the game ends with a lose screen |
| game\_won | Boolean | True/False | Flag to determine if the game has been won | You won | True or False only |
| guesses | List[strings] | A word | This list holds the user’s guesses as inputs in the game | 'Red', 'Green', 'Blue', 'Yellow' | Ensures the list contains strings representing the players guess |
| Grid | List of rows | Text | The grid that displays at start of the game that displays the words | --- --- ---  | | | |  --- --- -- | This grid printed into the correct format |

# Implementation

## GitHub Repository

https://github.com/Dearing-H/SE/tree/main#readme

****

*This GitHub README.md was created using* [*https://readme.so/*](https://readme.so/)

# Testing

## Test Table does the game end when lives go to zero

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Category | Test Case Description | Input to Provide | Expected Output | Actual Output | Pass/Fail |
| Test 1 | Normal | Check if game ends if lives go to 0 | N/A | Game over! You ran out of lives. | “Game over! You ran out of lives.” | Pass |
| Test 2 | Normal | Check if game ends if all categories are guessed | N/A | You WIN!!!!!, Congratulations | “You WIN!!!!!, Congratulations”. | Pass |
| Test 3 | Normal | Playing again after winning a game | “Yes” for replay | Restart the game with new categories | Restarted the game with new categories. | Pass |
| Test 4 | Exceptional | Check how game responds to no wors being imputed rather than 4 | “ ”,” ”,” ”, | Please enter four words one by one | Program accepts and treats it as an incorrect guess. | Fail |
| Test 5 | Exceptional | Players enter numbers rather than words | 4, 2, 78, 21 | Incorrect Input, please guess again | Program accepts and treats it as an incorrect guess. | Fail |
| Test 6 | Exceptional | Guessing words that do not belong in any category | “Dog”, “Plumer”, “Power” | Sorry, that's incorrect. | Sorry, that's incorrect. | Pass |
| Test 7 | Extreme | When the game chooses four categories from a larger list of dictionaries all dictionaries are including first and last | N/A | First and list are included randomly | First and list are included randomly | Pass |
| Test 8 | Extreme | Playing multiple games consecutively without, reloading the file | “Yes” for the replay | Restart the game with new categories each time | Restart the game with new categories each time | Pass |
| Test 9 | Extreme | Players can win the game on 1 life remaining | N/A | Restarts the game with new categories each time | Restarts the game with new categories each time | Pass |

# Release and Patch Notes

## Release 1.0.0

<https://github.com/Dearing-H/SE/releases/tag/v1.0.0>

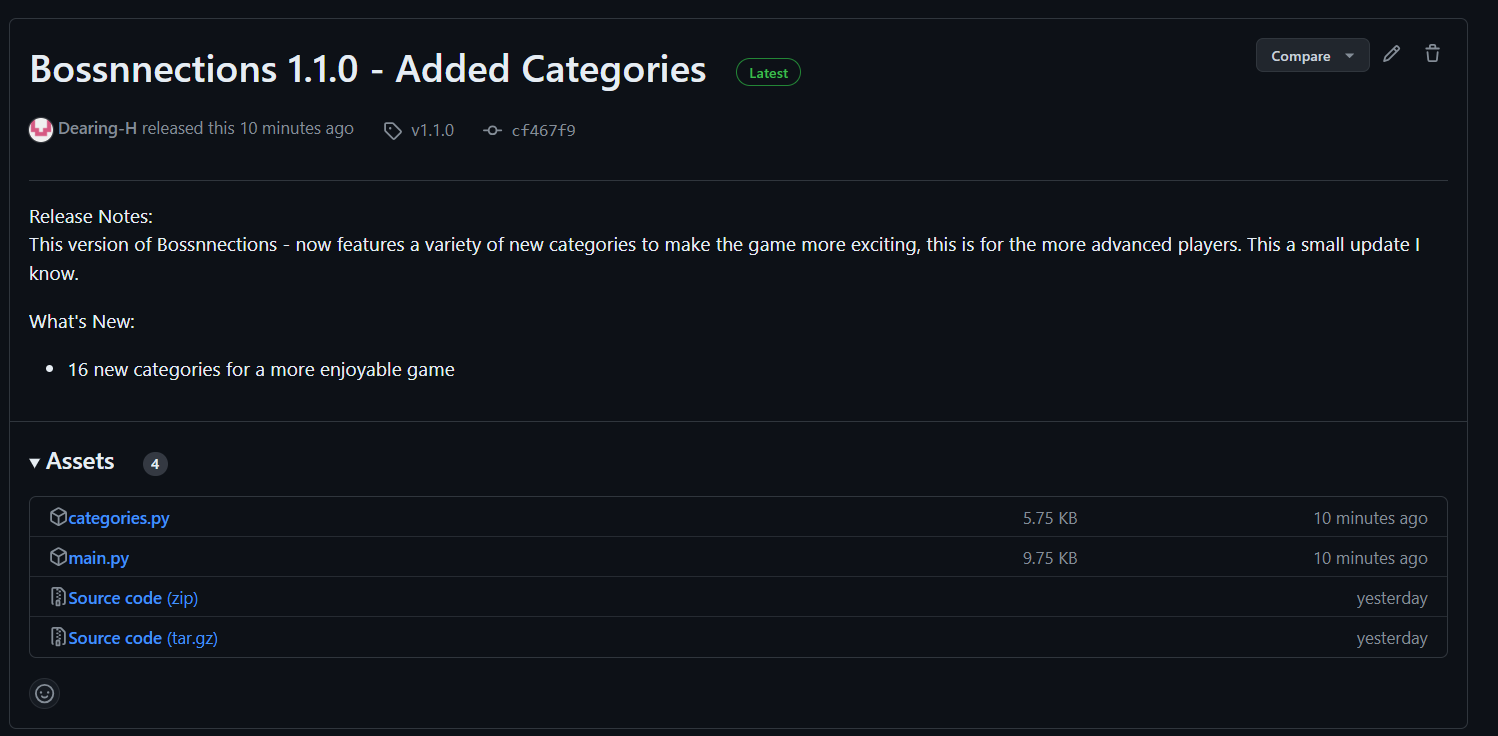
A screenshot of a computer

Description automatically generated

## Release 1.1.0

<https://github.com/Dearing-H/SE/releases/tag/v1.1.0>

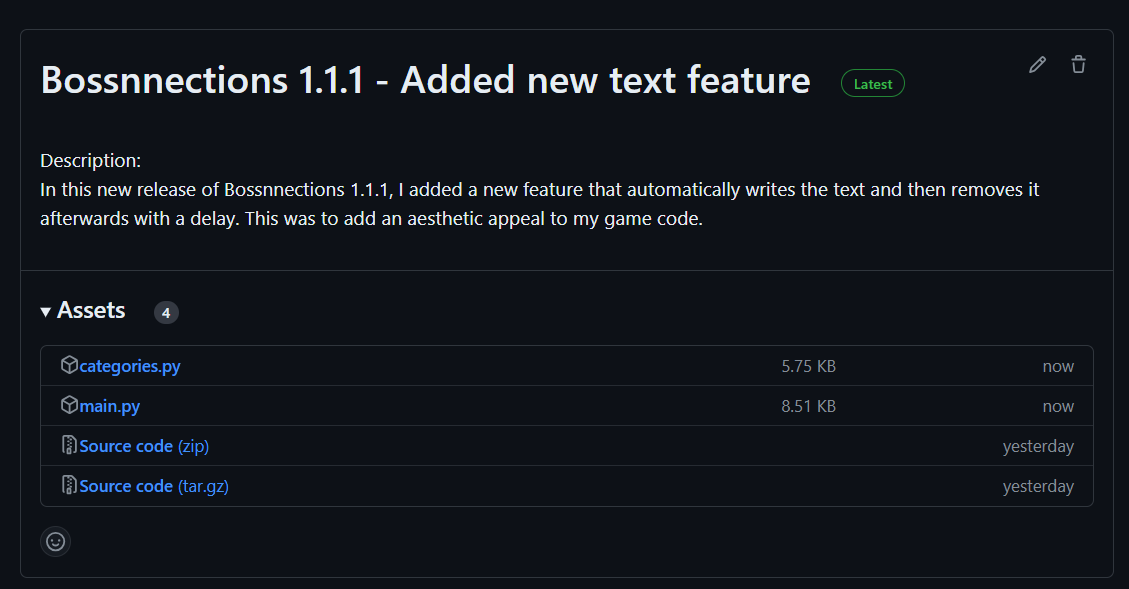
Patch 1.1.0 is a feature update, where I introduced more categories into the game from my dictionary. This now adds more depth to the game as it is less likely for the same categories to be used multiple times if a player constantly replays the game.



## Release 1.1.1

<https://github.com/Dearing-H/SE/releases/tag/v1.1.1>

I wanted to add a more aesthetically pleasing appearance to my game, I accomplished this by using a feature that writes text automatically and then deletes it after before starting my game.



# Project Reflection

The planning phase of the algorithms, albeit initially met with scepticism due to my lack of overall coding knowledge as I only recently started the course, however they taught me the value of a structured approach for the most optimal results. Although it extended the time required to accomplish tasks, it ensured the achievement of the set objectives with greater precision for me due to my lack of knowledge.

Initially, I encountered difficulties in wrapping my head around the multiple aspects of coding techniques that I had to implement into my game. This was due to python being a whole new language for me, I had to learn all the variables and data types while learning to code and I had trouble understanding, the overall concepts of the code in the beginning. However, after I experimented with the code, through forms of trial and error, asking reddit, chat GPT or Mr Fong for assistance I begun to learn the aspects of coding and have an overall creator understanding. I learnt a sense of independence for coding, I could go and research on my own and use techniques that I had learned in class or things that I had found on the internet to upgrade my knowledge of the topic, I believe this allowed me to be less stressed and really enjoy the coding as it was a fun experience to learn something completely foreign to me.

After I have completed this assessment task I can fully understand and see my progression from my first of Yr 11 and walking into my software class to being utterly confused with no understanding of what was going on, to now being able to understand the concepts behind coding and what the individual factors can accomplish. I learnt problem solving and that I need to be open minded and not as concerned about getting all the little gritty aspects I needed to focus on my development in the course. The GitHub repository management proved to be a rewarding aspect of the project, particularly with the utilization of readme.io, which facilitated efficient documentation formatting. This was due to me being able to represent milestones in a form of documentation that I had never experienced before. My proficiency in Python has advanced considerably through this first project. I have mastered the structure of a basic game loop and the method of breaking down complex problems into manageable segments. I now understand how to solve complex problems in python and look forward to doing more coding.

However I regret my poor time management in the start of the term, this was were I didn’t understand the code and was at a loss. The delay in my understanding of python caused me to produce a game that I believed lacked a few factors that I wished to include being a shuffle feature, colour codding the categories and a more streamlined user experience. I hope to develop my skills in my next assessment task, and learn from my mistakes by not doing enough work and procrastinating because I didn’t understand and was to scared to ask at the beginning, as I was in a class full of individuals who had experience apart from the odd 1 or 2.