T1A3 Terminal App Assignment

Functionality

- The purpose of this app is entertainment and to test how well users know commonly used logos and icons for programming languages and frameworks.

Design

- In order to run this app you will need to install Python3, installation instructions can be found here: Python installation
- Install git, installation instructions can be found here : git installation
- Clone the repository from github by typing the following command into your terminal: `git clone https://github.com/E-Holt/t1a3.git`
- You then open your terminal on your computer and type in the following: `./src/executegg.sh help` for how to install and play this game. `./src/executegg.sh` to go to the main menu and play the game!
- If your terminal says you don't have permission, run this script then try again: `chmod +x src/executegg.sh`
- This program should work on any computer and operating system capable of installing python3.

How to play

- You can view the logos and icons used in this game in the README file in the folder containing the game.
- This game is played by following the prompts.
- In the menu, please type in the relevant number and press the enter key.
- If you've chosen to view the instructions you can then press enter again to return to the menu.
- If you've chosen to play the game, the game will give you a hint then prompt you for a colour name.
- Please type in the colour name in lower cased letters, then press enter.
- If you type in the incorrect colour or any other word that is not the colour, it will count as an incorrect guess and you'll be given a new hint for the same colour.
- Remember you only have five guesses!
- If you don't guess correctly within those five guesses, you'll be asked if you want to play again.
- Please type in “yes” or “no” and press enter.
- If you guessed correctly, you'll be given a score based on how many guesses it took and will be asked if you'd like to play again.
- If you guess correctly in multiple rounds, you'll be shown your total score for all the rounds you've played if you haven’t exited the game.
- When you're done playing, you can choose "no" to playing again, or exit the terminal at any point.
- Thanks for playing the game!

My repository can be found at:

`[Repository](https://github.com/E-Holt/t1a3)`

Styling Conventions

- I used the Python variable naming convention snake case for naming variables and functions (Python Variable Names, 2022).

Features

1. Menu feature

- The menu feature is a basic menu that allows the user to choose what they'd like to do. It has some error handling capabilities for inputs that aren't a menu option and uses imported data to connect with other files used in the program, as well as an imported module to allow the screen to be cleared. It also uses a while loop and conditional statements to allow the user to choose options

efficiently and with minimal risk of breaking if something goes wrong.

```
from os import system
import guess
import instruction

def welcome_options():
    print("Welcome to the super awesome web dev logo and icon colour guessing game!
    print("1. Show instructions")
    print("2. Right to the game")
    print("3. Exit")
    selection = input("Select what you'd like to do: ")
    return selection

user_input = ""
while user_input != "3":
    system('clear')
    user_input = welcome_options()
    system('clear')
    if user_input == "1":
        instruction.show_instructions()
    elif user_input == "2":
        guess.guess_game()
    elif user_input == "3":
        print("Thanks for playing!")
        exit()
    else:
        print("That isn't an option! Please choose an option from the menu only.")

    input("press Enter to continue...")
    system('clear')
```

2. Hints feature

- The hints feature is spread across a couple files and a couple functions to allow the program to choose a random colour and shuffle the related hints list in order to give hints that wont be in the same order on any given round. This involves using a list relevant to each colour, multiple if statements in order to correctly choose the list, an imported random module to randomise the colour choice and shuffle the hints list, as well as a while loop for giving the hints dependant on if the input was correct or not. The hints feature is mostly not interacted with by the user, but does have some error handling capabilities with the rest of the game_input function when the user inputs something that isn't usable.

```
def game_input(hints, colour_pick):
    guess_input_counter = 0
    i=0
    print("Try and guess the colour described in the following hint!")
    print (hints[i])
    guess_input = input("What is the colour? (red, organge, yellow, green, blue, purple, black, grey): ")
    while guess_input != colour_pick and i<4:
        guess_input_counter += 1
        i += 1
        print("Thats incorrect! Please try again!")
        print(hints[i])
        guess_input = input("What is the colour? (red, organge, yellow, green, blue, purple, black, grey): ")
    while guess_input != colour_pick and i == 4:
        print("Sorry, you're out of guesses!")
        play_again = input("Would you like to play again? (yes/no) ")
        if play_again == "yes":
            guess_game()
        if play_again == "no":
            print("Thanks for playing!")
            exit()
    total_points = 5-int(guess_input_counter)
    if guess_input == colour_pick:
        print("You guessed the colour! Great job!")
        print ("You got", total_points,"/5 points!")
        point_counter(total_points)
        play_again = input("Would you like to play again? (yes/no) ")
        if play_again == "yes":
            guess_game()
        if play_again == "no":
            print("Thanks for playing!")
            exit()
        if play_again != "yes" and play_again != "no":
            print("Sorry that's not an option!")
```

![Hints list example image](./docs/hints_list_example.png

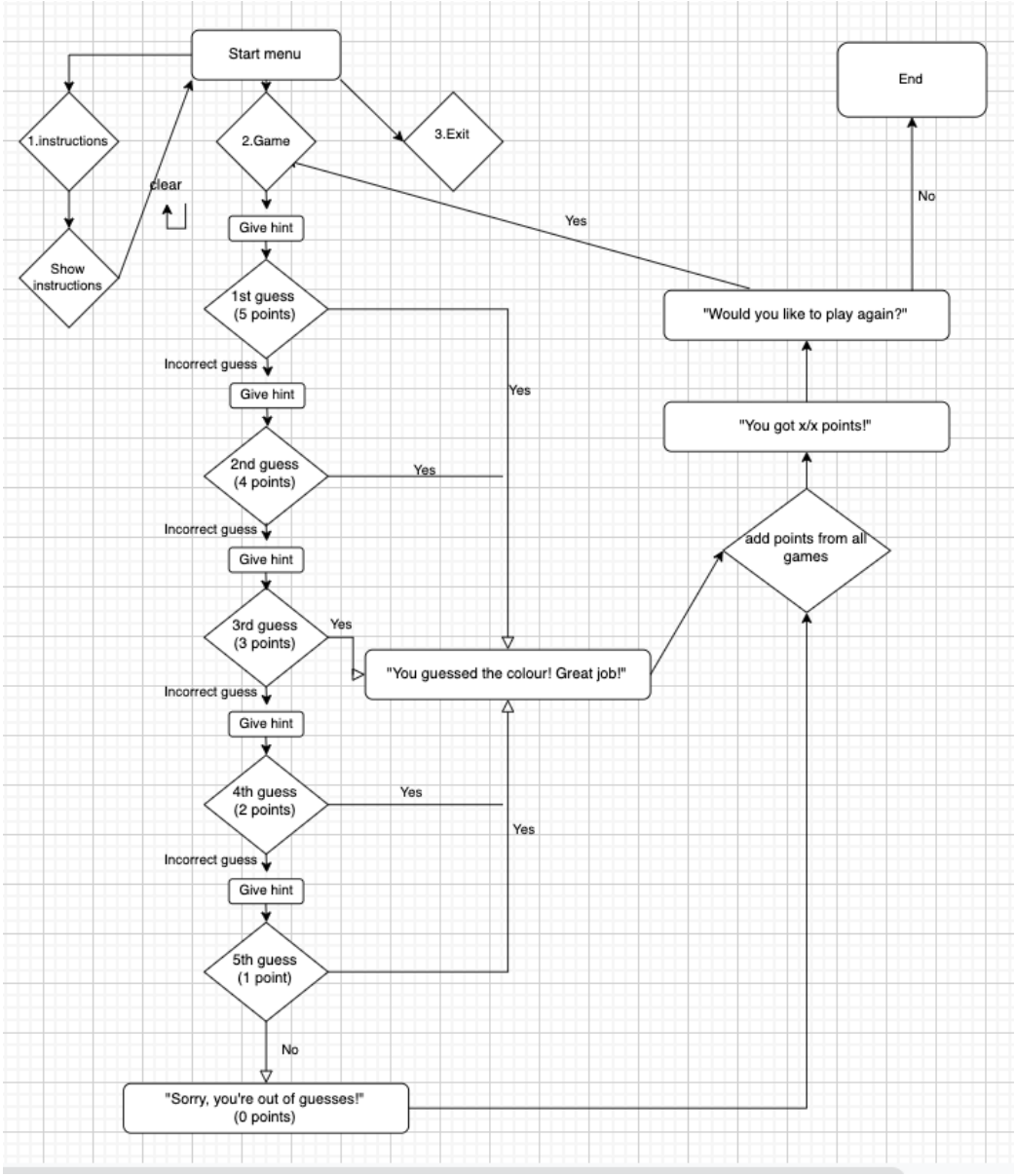
3. Scoring keeping feature

- The scoring feature if found in the game_input function and the point_counter function. It uses a variable that starts at zero and calculated the points in a round based on how many guesses were input before a correct guess, if a correct guess is given. Additionally, the point_counter function tracks those points and adds them together each round to give a total game score which is reset when the program is exited. The scoring feature is involved in a while loop along with the hints function mentioned above as well as conditional statements dependant on whether the input was a correct guess or not. The scoring feature itself doesn't have error handling function, but utilises the error handling in the game_input function when an incorrect guess is given.

```
def point_counter(total_points):
    global keep_adding_points
    keep_adding_points = keep_adding_points + total_points
    print("Your total points are", keep_adding_points,"!")
```

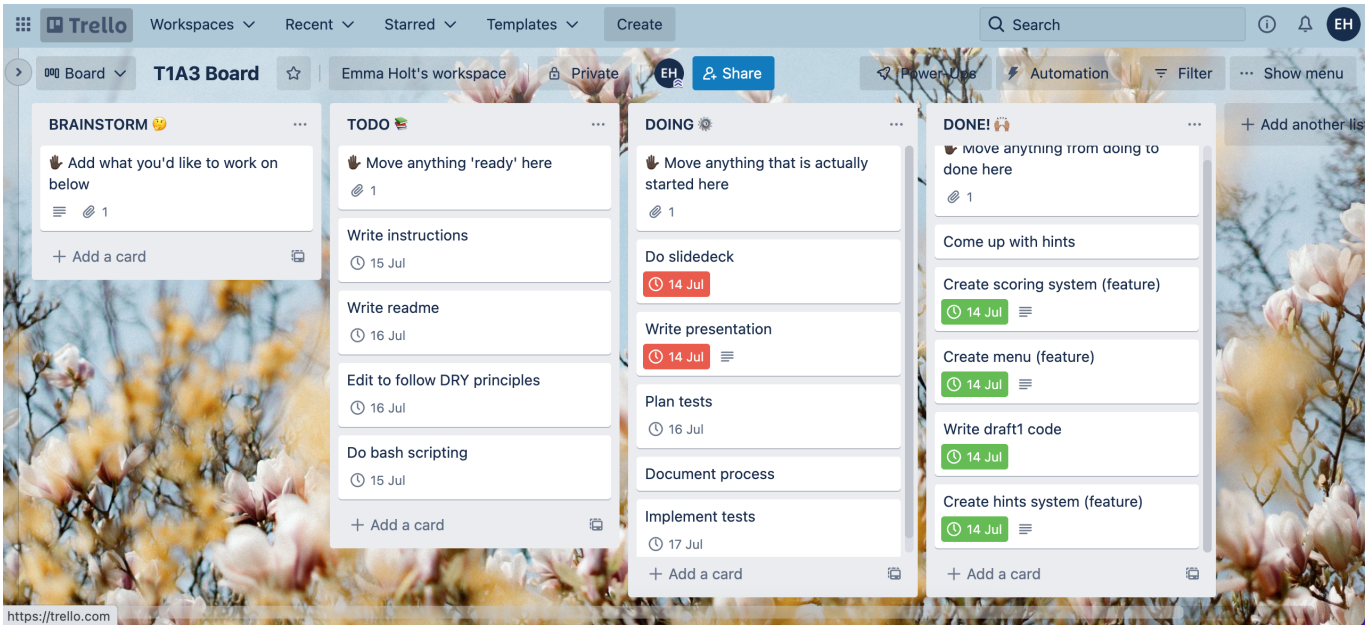
Development

- I used a flow chart to plan the general course of how the game would work and the different functions I needed to create.



Implementation Plan

- I used Trello to keep track and what I was working on and what needed to be done by a certain date. The Trello board can be found here [Trello board]



Menu feature

- The menu feature was one of the first I worked on as I used similar code to a tutorial we had done. I had to learn how to use the os library import to use the system module within the menu feature. I had all of my featured due on 14/07 to allow testing time before presenting on 15/07. This is the checklist I used for the menu feature.
    - List menu options
    - Link menu options with relevant functions
    - Figure out how to quit program
    - Create error response
    - Test menu function

Hints feature

- The hints feature is the second feature I focussed on as it is one of the more complicated features in the game. I had to take the time to understand the random module in Python and how to get functions to work together as well as import files into each other to get it working in a way that wasn't cluttered. This was also due on 14/07 to make sure I allowed time for testing and was ready for presenting on 15/07, however I made sure it was functional by 13/07 to allow for adequate testing with the other functions.This is the checklist I used for the hints feature:
    - List hints in an accessible manner
    - Create function for randomly choosing colour
    - Create function for randomly choosing hints
    - Get functions working together
    - Test performance of hints system

Score Keeping feature

- The score keeping feature was complicated to figure out for being smaller compared to the hints feature. I tried multiple different ways to get the feature to keep track of points between rounds, but ended up using a global variable and simplifying it as much as I could. This also involved figuring out how to get functions working together, which I had trouble with to begin with but thankfully figured out. This feature had the same time frame as the others, but I was able to complete it by the 13th with some minor fixes by the 15th. This is the checklist I used for the score keeping feature:
    - Create relevant scoring system
    - Create function for keeping score across games
    - Connect score functions with main game functions
    - Create error handling
    - Test score keeping function

Logos and Icons used

Angular logo (Angular, 2022) 

Apache Cordova logo (Apache Project logos, 2022) 

Bash logo (ProspectOne, 2016) 

Bootstrap logo (Otto, 2011) 

Bosque logo (Scarlet and LeBlanc, 2019) 

C# logo (Chavez, 2020) 

C++ logo (Kratz, 2017) 

C logo (C Programming Language Logo) 

CSS logo (daPhyre, 2016) 

Delphi logo (Embarcadero News, 2022) 

Django logo (Django, 2022) 

Elixir logo (Elixir, 2015) 

Hack logo (Hack, 2017) 

Hadoop logo (Apache Hadoop, 2016) 

HTML logo (W3C HTML5 Logo, 2011) 

Java logo (Java Licensing logo guidelines, 2016) 

Kotlin logo (Kotlin Programming Language, 2021) 

Linux mascot (Gabovitch, 2016) 

.NET logo (Massi, 2020) 

Microsoft logo (Microsoft Legal, 2012) 

Node logo (Node.js, 2016) 

PHP (PHP: Download Logos, 2001) 

Python logo (PSF Trademark Usage Policy, 2008) 

Pytorch logo (PyTorch Brand Guidelines, 2020) 

R logo (R: The R Project for Statistical Computing, 2016) 

Ruby logo (Matsumoto, 2022) 

Ruby on Rails logo (Trademarks policy, 2016) 

Rust logo (Rust Foundation, 2014) 

Spark Java logo(Åse, 2014) 

Spring logo (Spring Trademark Guidelines, 2018) 

Swift (Swift Resources, 2014) 

TensorFlow(TensorFlow Brand Guidelines, 2019) 

TypeScript (TypeScript Website, 2020) 

JavaScript (Williams, 2011) 

Reference List:

Angular.io. 2022. Angular. [online] Available at: https://angular.io/presskit [Accessed 16 July 2022].

Apache.org. 2022. Apache Project logos. [online] Available at: https://www.apache.org/logos/ [Accessed 16 July 2022].

Åse, D., 2014. Spark Framework: An expressive web framework for Kotlin and Java. [online] Spark Java. Available at: https://sparkjava.com/ [Accessed 16 July 2022].

Chavez, M., 2020. GitHub - MartinChavez/CSharp: C# : Test-Driven Learning. [online] GitHub. Available at: https://github.com/MartinChavez/CSharp [Accessed 16 July 2022].

daPhyre. 2016. File:CSS3 and HTML5 logos and wordmarks.svg - Wikimedia Commons. [online] Available at: https://commons.wikimedia.org/wiki/File:CSS3_and_HTML5_logos_and_wordmarks.svg#/media/File:CSS3_logo_and_wordmark.svg [Accessed 16 July 2022].

Developer.apple.com. 2014. Swift Resources. [online] Available at: https://developer.apple.com/swift/resources/ [Accessed 16 July 2022].

Django Project. 2022. Django. [online] Available at: https://www.djangoproject.com/ [Accessed 16 July 2022].

Elixir Language. 2015. Elixir. [online] Available at: https://elixir-lang.org/ [Accessed 16 July 2022].

Embarcadero. 2022. Embarcadero News. [online] Available at: https://www.embarcadero.com/news/logo [Accessed 16 July 2022].

Gabovitch, I., 2016. Tux (mascot) - Wikipedia. [online] En.wikipedia.org. Available at: https://en.wikipedia.org/wiki/Tux_(mascot)#/media/File:TuxFlat.svg [Accessed 16 July 2022].

GitHub. 2020. TypeScript Website. [online] Available at: https://github.com/microsoft/TypeScript-Website/blob/f407e1ae19e5e990d9901ac8064a32a8cc60edf0/packages/typescriptlang-org/static/branding/ts-logo-512.svg [Accessed 16 July 2022].

Hack Language. 2017. Hack. [online] Available at: https://hacklang.org/ [Accessed 16 July 2022].

Hadoop.apache.org. 2016. Apache Hadoop. [online] Available at: https://hadoop.apache.org/ [Accessed 16 July 2022].

Kotlin. 2021. Kotlin Programming Language. [online] Available at: https://kotlinlang.org/ [Accessed 16 July 2022].

Kratz, J., 2017. GitHub - isocpp/logos: C++ logos created for isocpp.org. [online] GitHub. Available at: https://github.com/isocpp/logos [Accessed 16 July 2022].

Massi, B., 2020. brand/dotnet-logo.svg at main · dotnet/brand. [online] GitHub. Available at: https://github.com/dotnet/brand/blob/main/logo/dotnet-logo.svg [Accessed 16 July 2022].

Matsumoto, Y., 2022. The Ruby Logo. [online] Ruby-lang.org. Available at: https://www.ruby-lang.org/en/about/logo/ [Accessed 16 July 2022].

Microsoft Legal. 2012. Trademark and Brand Guidelines | Microsoft Legal. [online] Available at: https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks [Accessed 16 July 2022].

Node.js. 2016. Logos and Graphics | Node.js. [online] Available at: https://nodejs.org/en/about/resources/ [Accessed 16 July 2022].

Oracle. 2016. Java Licensing logo guidelines. [online] Available at: https://www.oracle.com/a/ocom/docs/java-licensing-logo-guidelines-1908204.pdf [Accessed 16 July 2022].

Otto, M., 2011. Bootstrap Blog. [online] Blog.getbootstrap.com. Available at: https://blog.getbootstrap.com/ [Accessed 16 July 2022].

PHP. 2001. PHP: Download Logos. [online] Available at: https://www.php.net/download-logos.php [Accessed 16 July 2022].

ProspectOne, 2016. GitHub - odb/official-bash-logo: Everything you need to start using the official GNU Bash logo. [online] GitHub. Available at: https://github.com/odb/official-bash-logo [Accessed 16 July 2022].

Python. 2008. PSF Trademark Usage Policy. [online] Available at: https://www.python.org/psf/trademarks/ [Accessed 16 July 2022].

PyTorch. 2020. PyTorch Brand Guidelines. [online] Available at: https://pytorch.org/assets/brand-guidelines/PyTorch-Brand-Guidelines.pdf [Accessed 16 July 2022].

R-project.org. 2016. R: The R Project for Statistical Computing. [online] Available at: https://www.r-project.org/ [Accessed 16 July 2022].

Ruby on Rails. 2016. Trademarks policy. [online] Available at: https://rubyonrails.org/trademarks [Accessed 16 July 2022].

Rust Foundation. 2014. Rust Foundation. [online] Available at: https://foundation.rust-lang.org/policies/logo-policy-and-media-guide/ [Accessed 16 July 2022].

Scarlet, N. and LeBlanc, N., 2019. BosqueLanguage/Fabric-Bosque.svg at master · microsoft/BosqueLanguage. [online] GitHub. Available at: https://github.com/microsoft/BosqueLanguage/blob/master/resources/brand/combined/Fabric-Bosque.svg [Accessed 16 July 2022].

Seeklogo. C Programming Language Logo PNG Vector (SVG) Free Download. [online] Available at: https://seeklogo.com/vector-logo/306166/c-programming-language [Accessed 16 July 2022].

Spring. 2018. Spring Trademark Guidelines. [online] Available at: https://spring.io/trademarks [Accessed 16 July 2022].

Tensorflow.org. 2019. TensorFlow Brand Guidelines. [online] Available at: https://www.tensorflow.org/extras/tensorflow_brand_guidelines.pdf [Accessed 16 July 2022].

W3.org. 2011. W3C HTML5 Logo. [online] Available at: https://www.w3.org/html/logo/index.html [Accessed 16 July 2022].

W3schools.com. 2022. Python Variable Names. [online] Available at: https://www.w3schools.com/python/gloss_python_variable_names.asp [Accessed 15 July 2022].

Williams, C., 2011. logo js. [online] GitHub. Available at: https://github.com/voodootikigod/logo.js/blob/1544bdeed/js.svg [Accessed 16 July 2022].