T1A3 - Terminal App

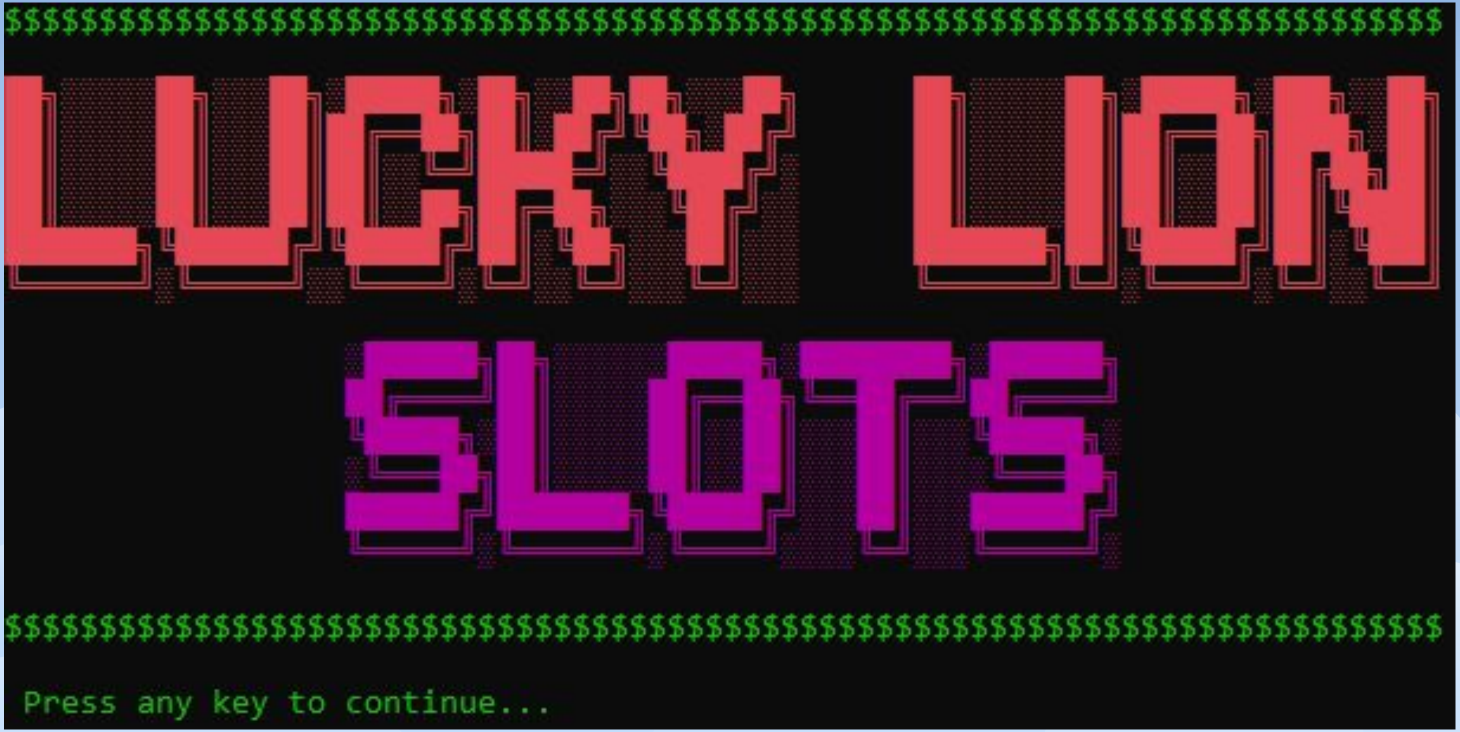
Timothy Nguyen

Requirements for slides

- A walk-through of the terminal application, its features and how it is used
- A walk-through of the logic of the terminal application and code
- A review of the development/build process including challenges, ethical issues, favourite parts
- An overview of the terminal application
  - Main features and overall structure of the app
- An overview of the code
  - Explanation of important parts of the code

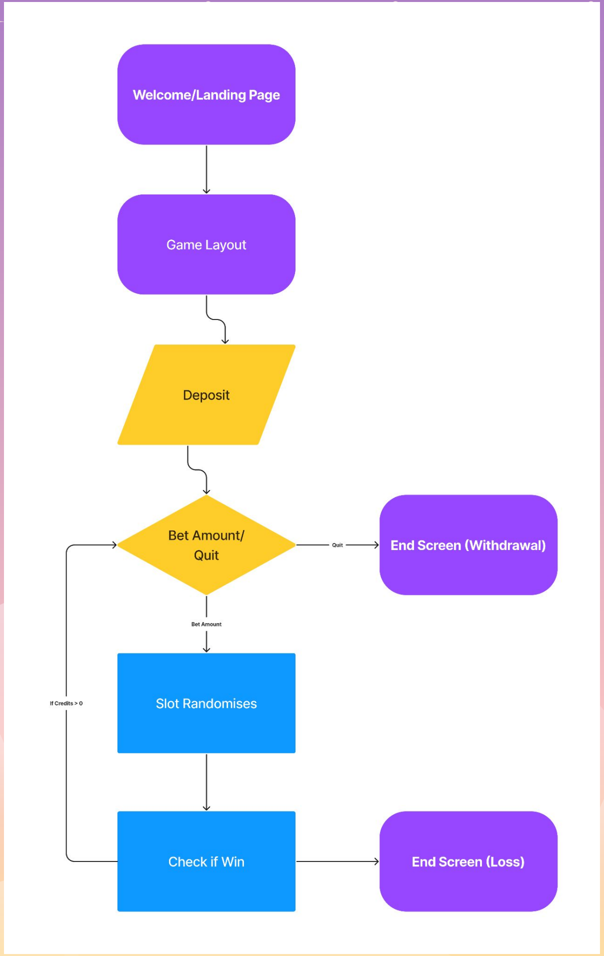
Slot Machine

Terminal Application



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

LUCKY LION

SLOTS

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Press any key to continue...
```

Flowchart



By using the flowchart, I can visually picture the data flow and can have an easier idea of where to begin

Style Guide

PEP 8 - Style Guide for Python

Overview

What is this application?

- This is a slot machine made in Terminal

Why did I make this application?

- In Australia, slot machines make up a major portion of gambling. I have personally played a few machines and I thought it would be fun to challenge myself to make something that I have knowledge of
- Slot machines are designed to be ‘fun’ as they require very little presses and they simulate a ‘game’ where a user can potentially win money (except gamblers tend to lose majority of the time)
- As it is such a large aspect in the gambling space in Australia, I feel like it would be a very well thought out process and it would be interesting to think in the minds of a developer associated with these machines
- I also wanted to make a gambling game that I could play without losing any money as well

Development

This is how I set up to begin the project and how I worked through it

Plan out in a document

Classes

- Game
  - Credits
  - Winnings
  - Bet amount

- Items
  - This creates variables for each slot item
  - Had variables to be used for the values of each item

Functions

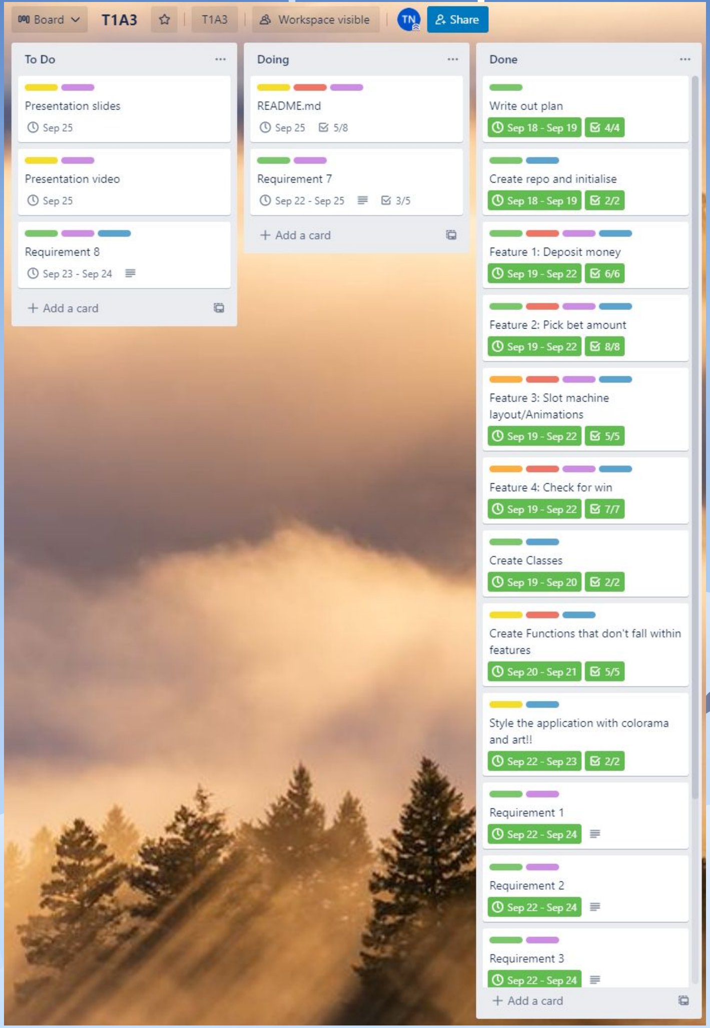
- play
- press_to_continue
- layout
- reel_randomiser
- spinning
- spin_animation
- check_win
- landing
- end_withdraw
- end_lost

A .pdf file was added to the README.md of this initial plan

Trello Board

Trello was then used to list out all the requirements

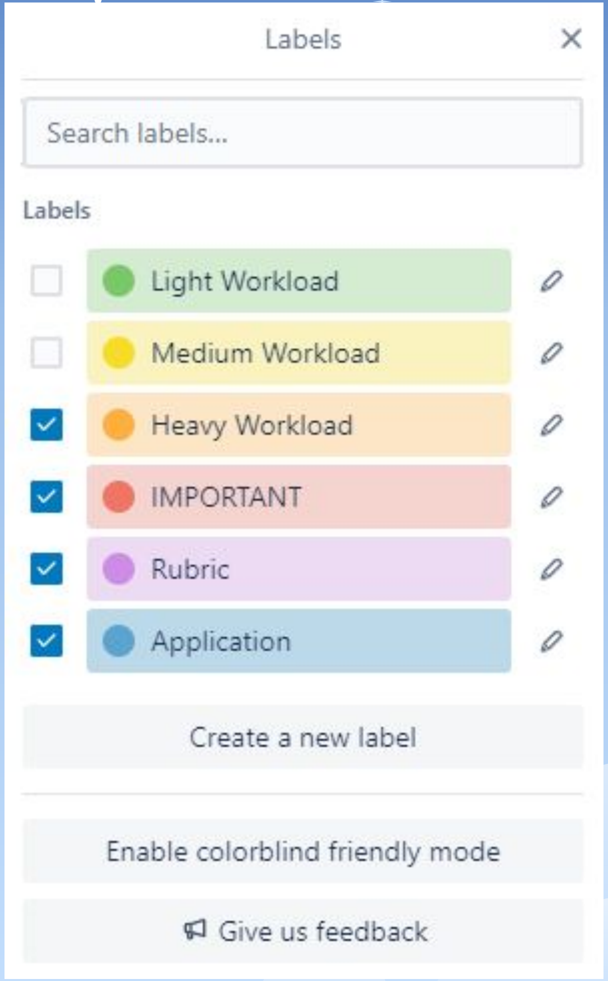
- Now that I wrote out that document I could easily list all the things that needed to be done
- Due dates were added to keep myself accountable and establish a timeline
- Easy to use and manage the board as well as track what I was on as I get distracted easily



To Do
- Presentation slides — Sep 25
- Presentation video — Sep 25
- Requirement 8 — Sep 23 - Sep 24

Doing
- README.md — Sep 25 — 5/8
- Requirement 7 — Sep 22 - Sep 25 — 3/5

Done
- Write out plan — Sep 18 - Sep 19 — 4/4
- Create repo and initialise — Sep 18 - Sep 19 — 2/2
- Feature 1: Deposit money — Sep 19 - Sep 22 — 6/6
- Feature 2: Pick bet amount — Sep 19 - Sep 22 — 8/8
- Feature 3: Slot machine layout/Animations — Sep 19 - Sep 22 — 5/5
- Feature 4: Check for win — Sep 19 - Sep 22 — 7/7
- Create Classes — Sep 19 - Sep 20 — 2/2
- Create Functions that don't fall within features — Sep 20 - Sep 21 — 5/5
- Style the application with colorama and art!! — Sep 22 - Sep 23 — 2/2
- Requirement 1 — Sep 22 - Sep 24
- Requirement 2 — Sep 22 - Sep 24
- Requirement 3 — Sep 22 - Sep 24


Labels

Search labels...

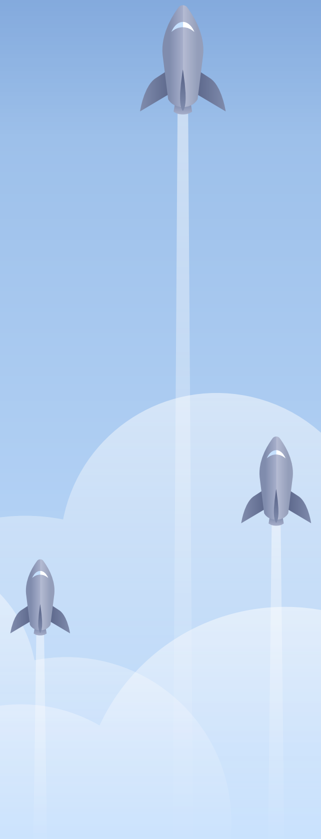
Labels

- [ ] Light Workload
- [ ] Medium Workload
- [x] Heavy Workload
- [x] IMPORTANT
- [x] Rubric
- [x] Application

These labels were used so I could understand which tasks were more important and I also sorted each task by their assumed workload

Features

1. deposit money
2. pick bet amount
3. feature for the slot machine
4. feature that checks final value and returns winnings

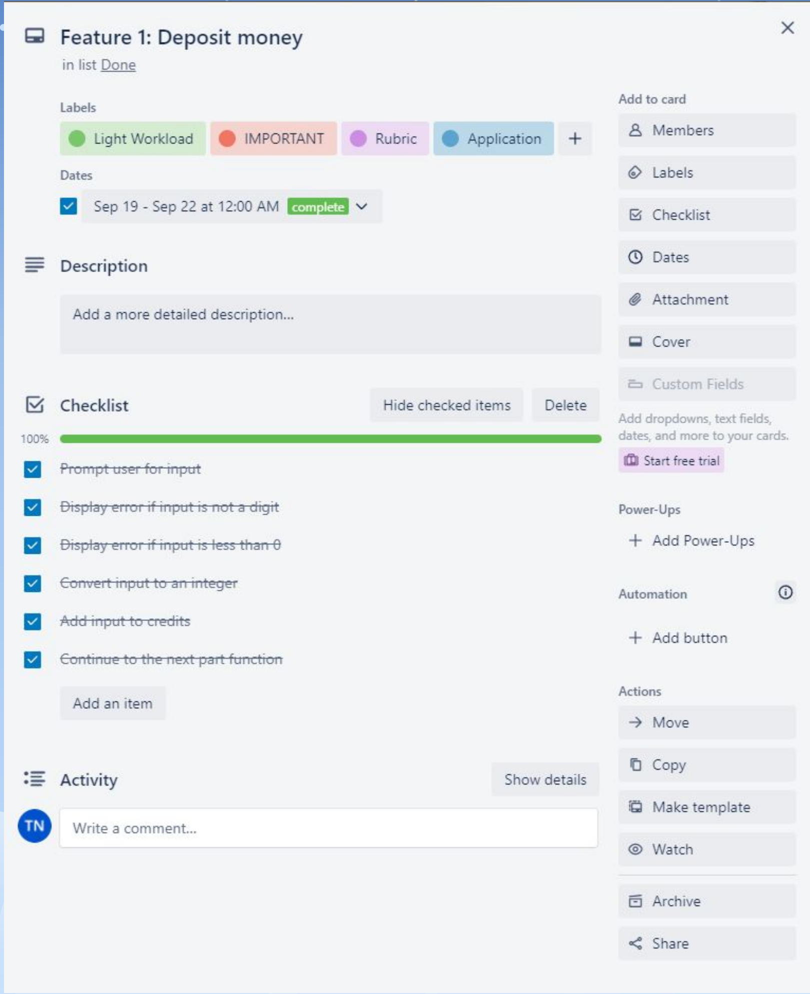


Quick demo of the game

Feature 1 - Deposit Money

Planning stage was done in Trello.

- It was labeled under the IMPORTANT category
- I added due dates to keep myself accountable
- Made a checklist of tasks I had to undertake to make the feature work



Feature 1: Deposit money

in list Done

Labels

Light Workload | IMPORTANT | Rubric | Application

Dates

Sep 19 - Sep 22 at 12:00 AM complete

Description

Add a more detailed description...

Checklist

100%

- [x] ~~Prompt user for input~~
- [x] ~~Display error if input is not a digit~~
- [x] ~~Display error if input is less than 0~~
- [x] ~~Convert input to an integer~~
- [x] ~~Add input to credits~~
- [x] ~~Continue to the next part function~~

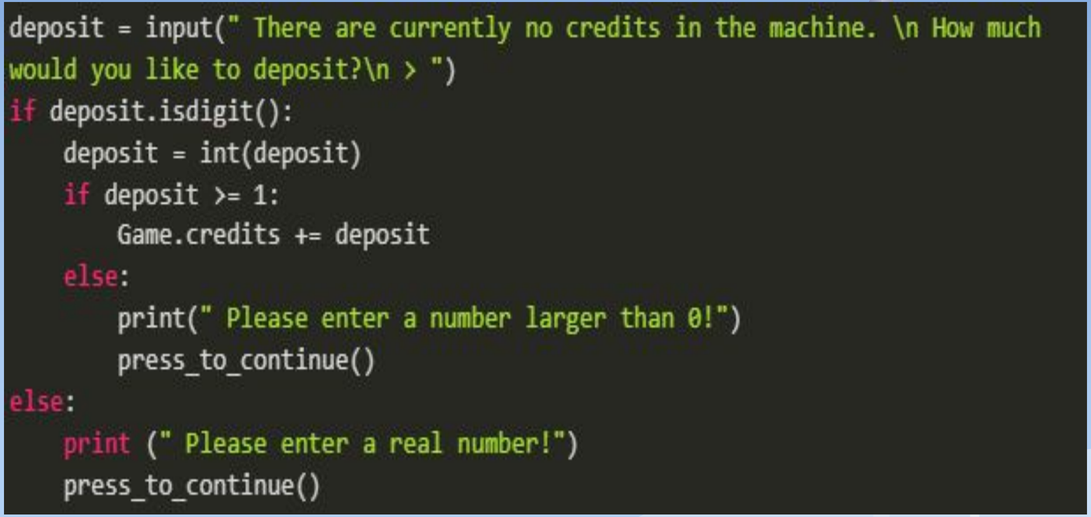
Activity

Code for deposit money

Deposit money feature begins after the welcome page.

- .isdigit() method is used to ensure the input is a digit

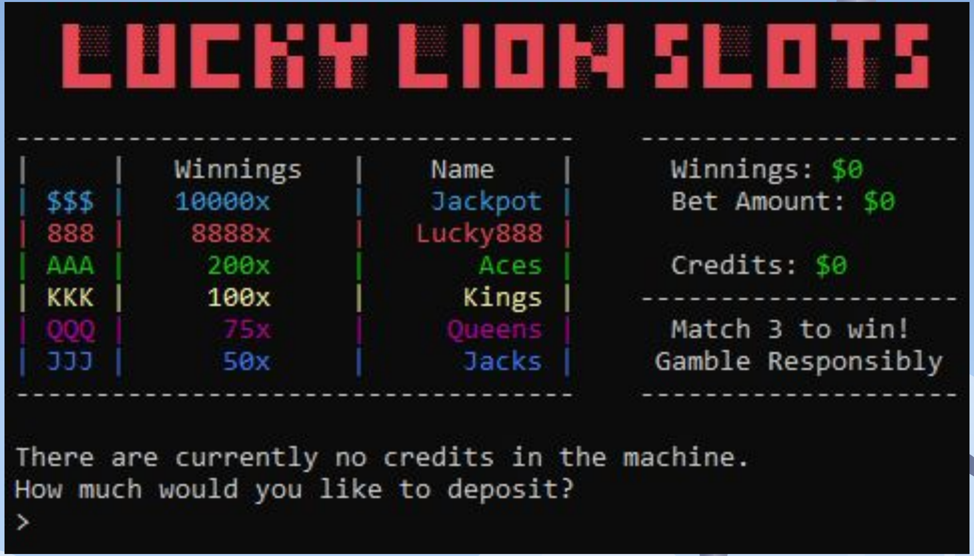
- The next conditions were added to determine what the next step of the code will be



```
deposit = input(" There are currently no credits in the machine. \n How much
would you like to deposit?\n > ")
if deposit.isdigit():
    deposit = int(deposit)
    if deposit >= 1:
        Game.credits += deposit
    else:
        print(" Please enter a number larger than 0!")
        press_to_continue()
else:
    print (" Please enter a real number!")
    press_to_continue()
```

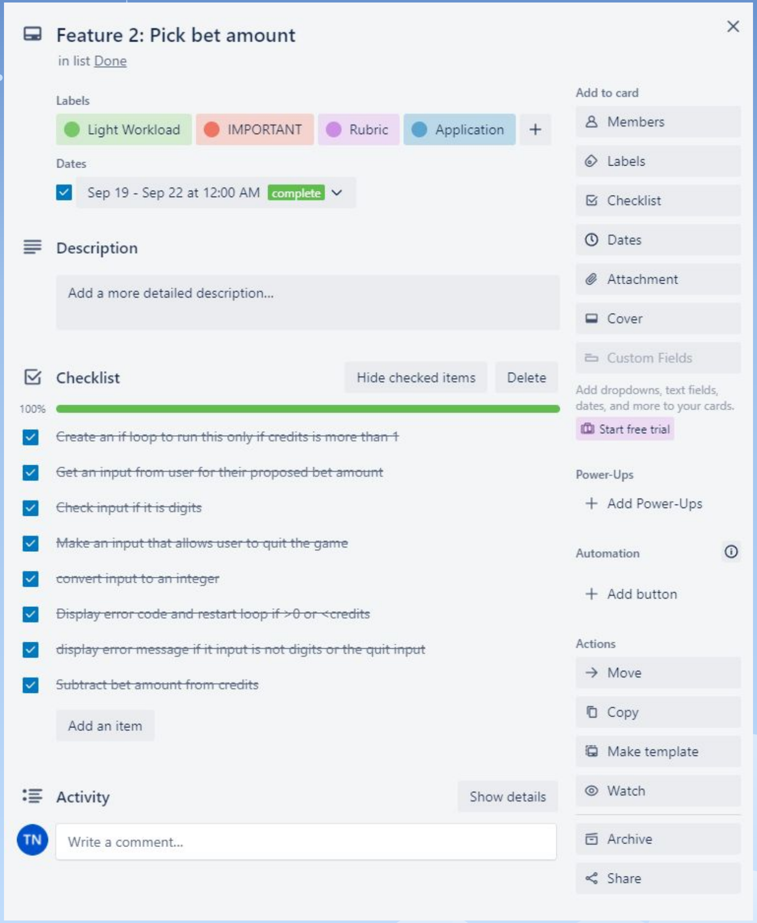
Deposit feature in game

- Here I will show a demo of what happens inside the game for this feature.



```
                    LUCKY LION SLOTS

----------------------------------          ------------------
|     |    Winnings    |     Name    |        Winnings: $0
| $$$ |  10000x        |    Jackpot  |        Bet Amount: $0
| 888 |   8888x        |   Lucky888  |
| AAA |    200x        |       Aces  |        Credits: $0
| KKK |    100x        |      Kings  |      ------------------
| QQQ |     75x        |     Queens  |        Match 3 to win!
| JJJ |     50x        |      Jacks  |      Gamble Responsibly
----------------------------------          ------------------

There are currently no credits in the machine.
How much would you like to deposit?
>
```

Feature 2 - Bet Input



Feature 2: Pick bet amount

in list Done

Labels

Light Workload | IMPORTANT | Rubric | Application

Dates

Sep 19 - Sep 22 at 12:00 AM complete

Description

Add a more detailed description...

Checklist

100%

- [x] ~~Create an if loop to run this only if credits is more than 1~~
- [x] ~~Get an input from user for their proposed bet amount~~
- [x] ~~Check input if it is digits~~
- [x] ~~Make an input that allows user to quit the game~~
- [x] ~~convert input to an integer~~
- [x] ~~Display error code and restart loop if >0 or <credits~~
- [x] ~~display error message if it input is not digits or the quit input~~
- [x] ~~Subtract bet amount from credits~~

Activity

Write a comment...

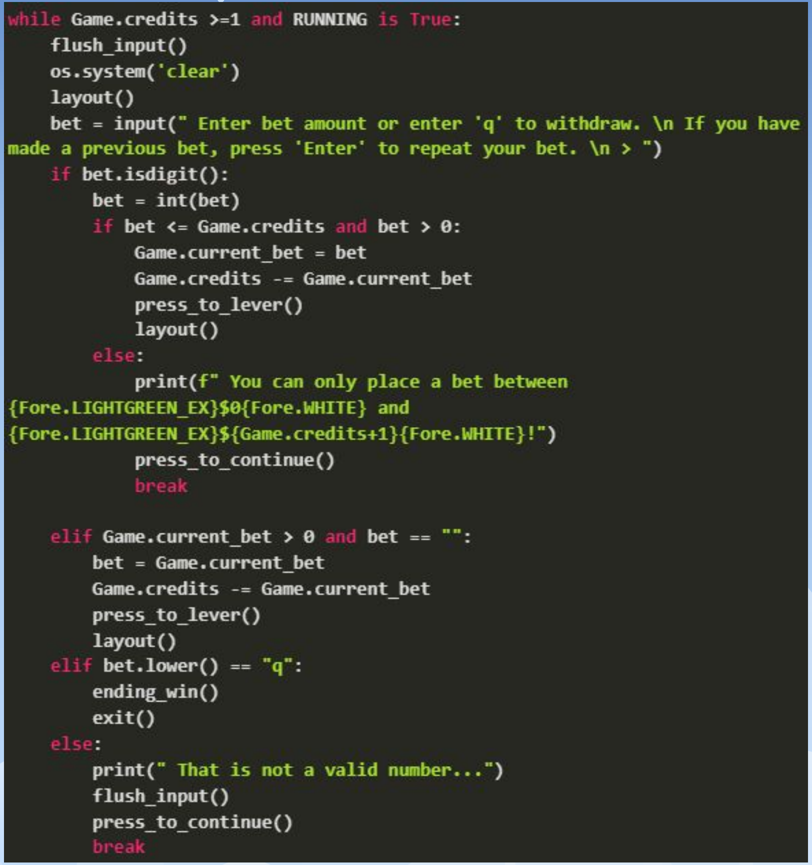
- This was also an IMPORTANT task
- Due dates also added for accountability
- Checklist was made to keep track of what was needed to be done

Code for Feature 2

Bet Input feature begins after the deposit feature.

- It will run under the conditions that Game.credits was more than or equal to 1, and the running loop returns true

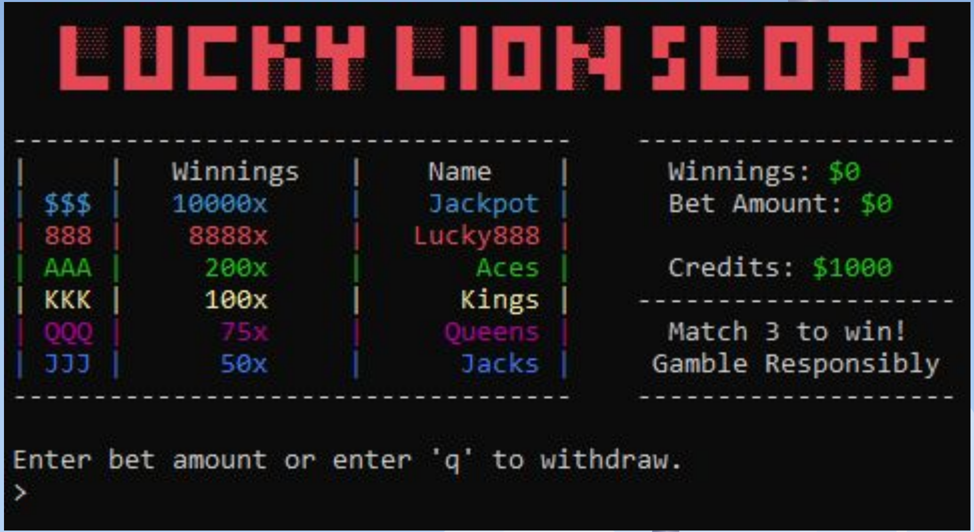
- 3 conditions from now:
  - If Input is digits
  - If current_bet > 0 and is an ‘enter’ input
  - If input is “q”



```python
while Game.credits >=1 and RUNNING is True:
    flush_input()
    os.system('clear')
    layout()
    bet = input(" Enter bet amount or enter 'q' to withdraw. \n If you have
made a previous bet, press 'Enter' to repeat your bet. \n > ")
    if bet.isdigit():
        bet = int(bet)
        if bet <= Game.credits and bet > 0:
            Game.current_bet = bet
            Game.credits -= Game.current_bet
            press_to_lever()
            layout()
        else:
            print(f" You can only place a bet between
{Fore.LIGHTGREEN_EX}$0{Fore.WHITE} and
{Fore.LIGHTGREEN_EX}${Game.credits+1}{Fore.WHITE}!")
            press_to_continue()
            break

    elif Game.current_bet > 0 and bet == "":
        bet = Game.current_bet
        Game.credits -= Game.current_bet
        press_to_lever()
        layout()
    elif bet.lower() == "q":
        ending_win()
        exit()
    else:
        print(" That is not a valid number...")
        flush_input()
        press_to_continue()
        break
```

Bet feature in game

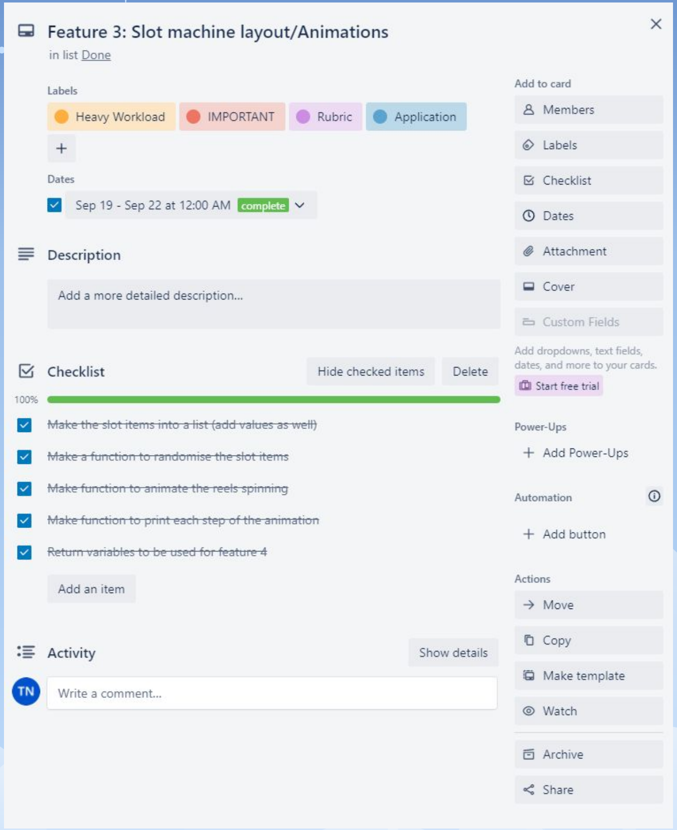
- Here is the bet feature within the application.



```
LUCKY LION SLOTS

-----------------------------------          --------------------
|     |    Winnings    |    Name    |          Winnings: $0
| $$$ |    10000x      |    Jackpot |          Bet Amount: $0
| 888 |     8888x      |   Lucky888 |
| AAA |      200x      |       Aces |          Credits: $1000
| KKK |      100x      |      Kings |        --------------------
| QQQ |       75x      |     Queens |          Match 3 to win!
| JJJ |       50x      |      Jacks |        Gamble Responsibly
-----------------------------------          --------------------

Enter bet amount or enter 'q' to withdraw.
>
```

Feature 3 - Reels

- As it was a feature, it is an IMPORTANT task
- This was my favourite part by far as it was the main part of the game itself and required a lot of problem solving to implement
- Due dates also added for accountability
- Checklist was made to keep track of what was needed to be done
- The most challenging feature of the application
  - Required 2 main parts consisting of 3 functions
    - 3.1 - Reel Randomiser
    - 3.2a - Reel Spin
    - 3.2b - Printing the Reel Spin



Feature 3: Slot machine layout/Animations

in list Done

Labels

Heavy Workload · IMPORTANT · Rubric · Application

Dates

Sep 19 - Sep 22 at 12:00 AM complete

Description

Add a more detailed description...

Checklist

100%

- [x] ~~Make the slot items into a list (add values as well)~~
- [x] ~~Make a function to randomise the slot items~~
- [x] ~~Make function to animate the reels spinning~~
- [x] ~~Make function to print each step of the animation~~
- [x] ~~Return variables to be used for feature 4~~

Activity

Write a comment...

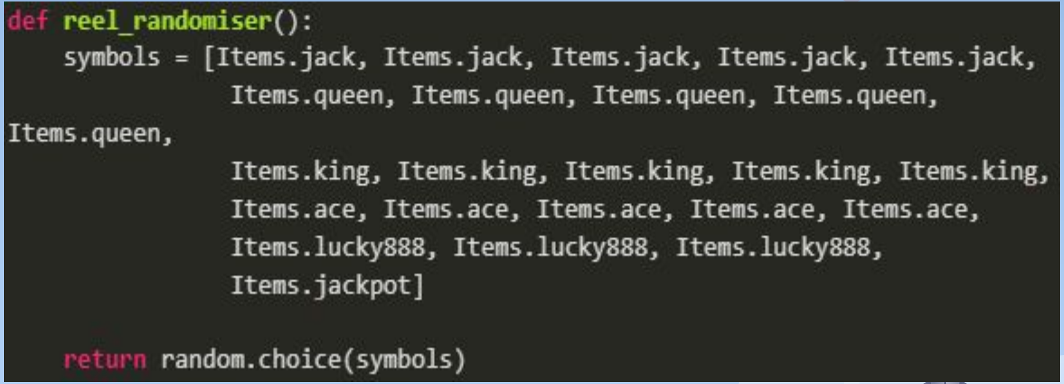
Code for Feature 3.1 - Randomiser

- This is the first part of the reel feature
    - Getting the list of symbols and randomising it

- First blocker appeared here

- Random.choice() method from the random module was used here to obtain a random selection from the given list

- The selection is then returned



```
def reel_randomiser():
    symbols = [Items.jack, Items.jack, Items.jack, Items.jack, Items.jack,
                   Items.queen, Items.queen, Items.queen, Items.queen,
Items.queen,
                   Items.king, Items.king, Items.king, Items.king, Items.king,
                   Items.ace, Items.ace, Items.ace, Items.ace, Items.ace,
                   Items.lucky888, Items.lucky888, Items.lucky888,
                   Items.jackpot]

    return random.choice(symbols)
```

Code for Feature 3.2a - Reel Spin

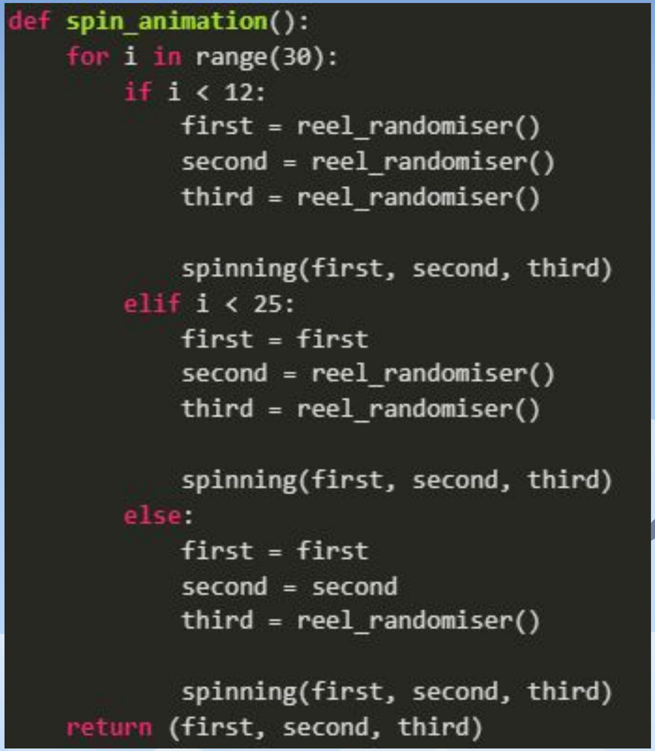
- This is the part 2a of the feature
  - Spinning each reel a certain amount to simulate a real slot machine

- Required a range

- All 3 reels spin

- When i = 13, first reel stops
- When i = 26, second reel stops
- When i = 30, third reel stops

- The final symbol on each reel is then returned in the format (first, second third)



```python
def spin_animation():
    for i in range(30):
        if i < 12:
            first = reel_randomiser()
            second = reel_randomiser()
            third = reel_randomiser()

            spinning(first, second, third)
        elif i < 25:
            first = first
            second = reel_randomiser()
            third = reel_randomiser()

            spinning(first, second, third)
        else:
            first = first
            second = second
            third = reel_randomiser()

            spinning(first, second, third)
    return (first, second, third)
```

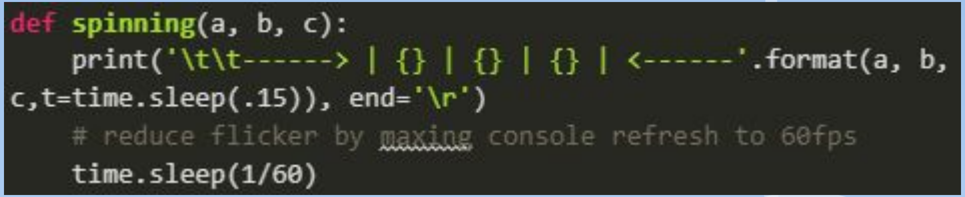
Code for Feature 3.2b - Print

- This is the part 2b of the feature
  - Printing the spinning reels on one singular line for every ‘i’ step

- Required the time module for t=time.sleep(.15) to delay the execution of the line for 0.15 seconds

- End = ‘\r’ is a carriage return which allows the line to be cleared instead of printing a new line every time

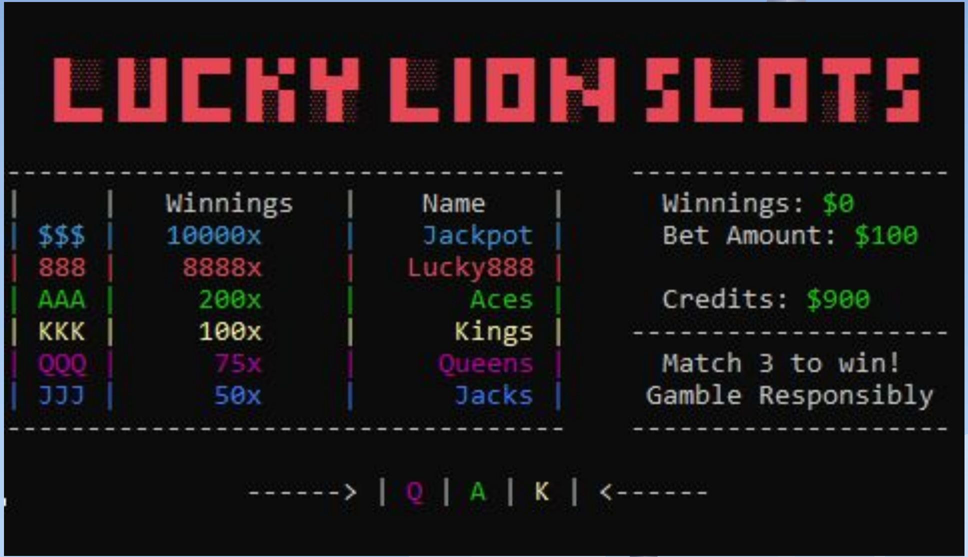
- Sometimes, the screen would flicker during the animation so time.sleep(1/60) helped max the console refresh to 60fps



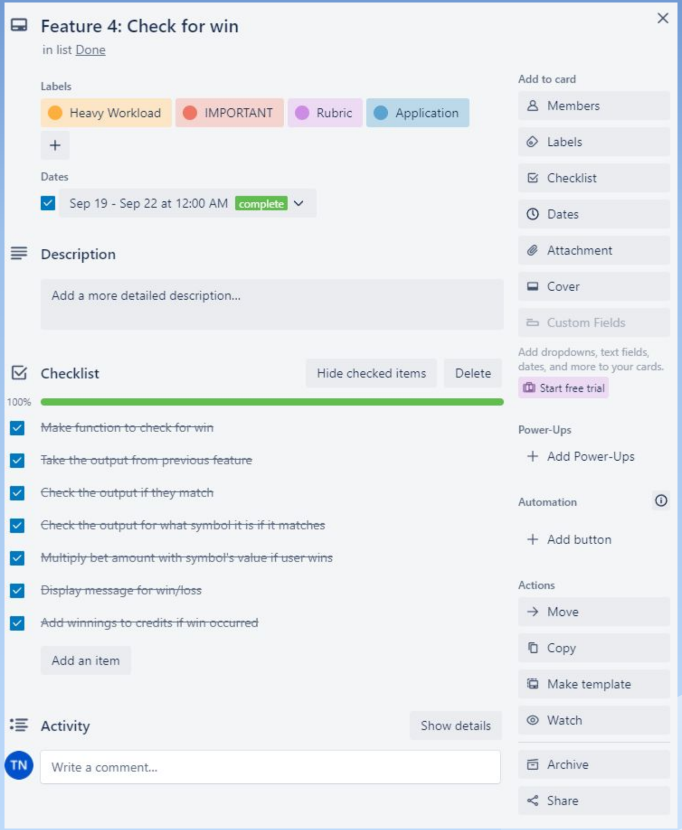
```
def spinning(a, b, c):
    print('\t\t------> | {} | {} | {} | <------'.format(a, b,
c,t=time.sleep(.15)), end='\r')
    # reduce flicker by maxing console refresh to 60fps
    time.sleep(1/60)
```

Reel Spin feature in game

- Here is a demo of the reel spinning after a bet input



```
LUCKY LION SLOTS

-----------------------------------        --------------------
|     |     Winnings     |     Name   |     Winnings: $0
| $$$ |     10000x       |    Jackpot |     Bet Amount: $100
| 888 |      8888x       |   Lucky888 |
| AAA |       200x       |       Aces |     Credits: $900
| KKK |       100x       |      Kings |    --------------------
| QQQ |        75x       |     Queens |     Match 3 to win!
| JJJ |        50x       |      Jacks |    Gamble Responsibly
-----------------------------------        --------------------

                ------> | Q | A | K | <------
```


Feature 4: Check for win
in list Done

Labels
Heavy Workload IMPORTANT Rubric Application

Dates
Sep 19 - Sep 22 at 12:00 AM complete

Description
Add a more detailed description...

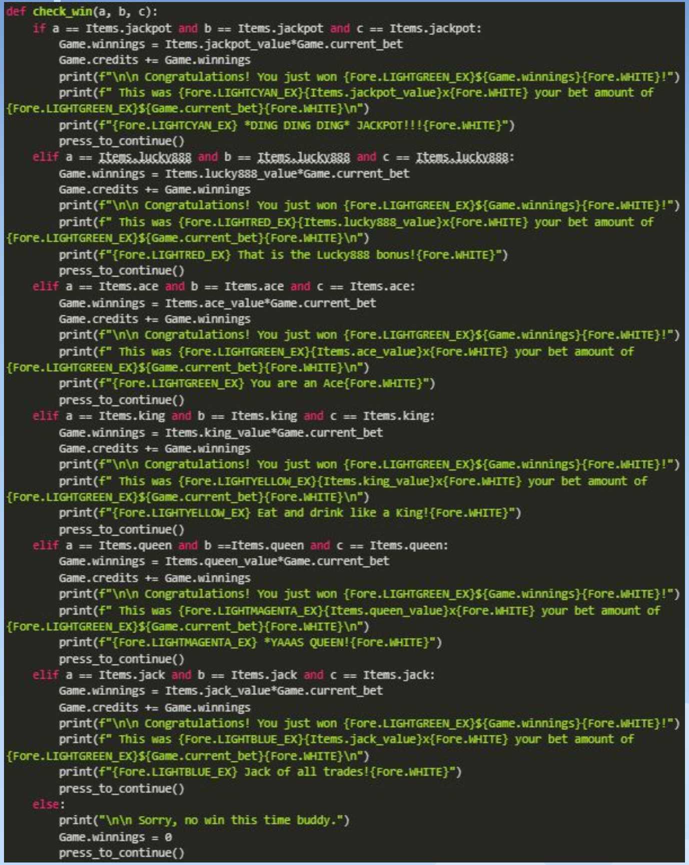
Checklist
100%
- [x] ~~Make function to check for win~~
- [x] ~~Take the output from previous feature~~
- [x] ~~Check the output if they match~~
- [x] ~~Check the output for what symbol it is if it matches~~
- [x] ~~Multiply bet amount with symbol's value if user wins~~
- [x] ~~Display message for win/loss~~
- [x] ~~Add winnings to credits if win occurred~~

Activity
Write a comment...

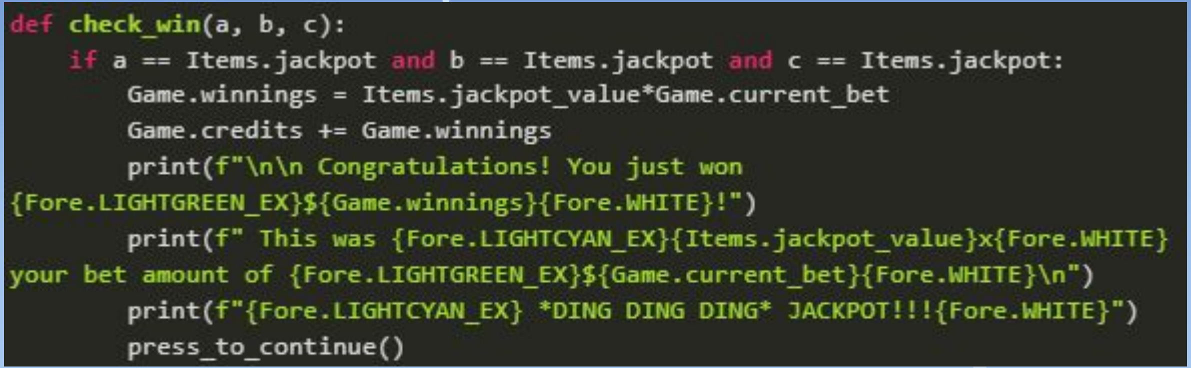
Feature 4 - Check for Win

- As it was a feature, it is an IMPORTANT task
- Due dates also added for accountability
- Checklist was made to keep track of what was needed to be done
- Required the use of returned variables from the previous feature to work

Code for Feature 4

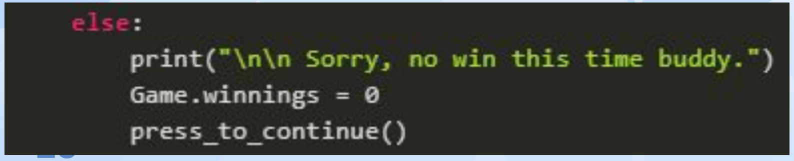


```python
def check_win(a, b, c):
    if a == Items.jackpot and b == Items.jackpot and c == Items.jackpot:
        Game.winnings = Items.jackpot_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTCYAN_EX}{Items.jackpot_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTCYAN_EX} *DING DING DING* JACKPOT!!!{Fore.WHITE}")
        press_to_continue()
    elif a == Items.lucky888 and b == Items.lucky888 and c == Items.lucky888:
        Game.winnings = Items.lucky888_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTRED_EX}{Items.lucky888_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTRED_EX} That is the Lucky888 bonus!{Fore.WHITE}")
        press_to_continue()
    elif a == Items.ace and b == Items.ace and c == Items.ace:
        Game.winnings = Items.ace_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTGREEN_EX}{Items.ace_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTGREEN_EX} You are an Ace{Fore.WHITE}")
        press_to_continue()
    elif a == Items.king and b == Items.king and c == Items.king:
        Game.winnings = Items.king_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTYELLOW_EX}{Items.king_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTYELLOW_EX} Eat and drink like a King!{Fore.WHITE}")
        press_to_continue()
    elif a == Items.queen and b ==Items.queen and c == Items.queen:
        Game.winnings = Items.queen_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTMAGENTA_EX}{Items.queen_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTMAGENTA_EX} *YAAAS QUEEN!{Fore.WHITE}")
        press_to_continue()
    elif a == Items.jack and b == Items.jack and c == Items.jack:
        Game.winnings = Items.jack_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won {Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTBLUE_EX}{Items.jack_value}x{Fore.WHITE} your bet amount of
{Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTBLUE_EX} Jack of all trades!{Fore.WHITE}")
        press_to_continue()
    else:
        print("\n\n Sorry, no win this time buddy.")
        Game.winnings = 0
        press_to_continue()
```



```python
def check_win(a, b, c):
    if a == Items.jackpot and b == Items.jackpot and c == Items.jackpot:
        Game.winnings = Items.jackpot_value*Game.current_bet
        Game.credits += Game.winnings
        print(f"\n\n Congratulations! You just won
{Fore.LIGHTGREEN_EX}${Game.winnings}{Fore.WHITE}!")
        print(f" This was {Fore.LIGHTCYAN_EX}{Items.jackpot_value}x{Fore.WHITE}
your bet amount of {Fore.LIGHTGREEN_EX}${Game.current_bet}{Fore.WHITE}\n")
        print(f"{Fore.LIGHTCYAN_EX} *DING DING DING* JACKPOT!!!{Fore.WHITE}")
        press_to_continue()
```

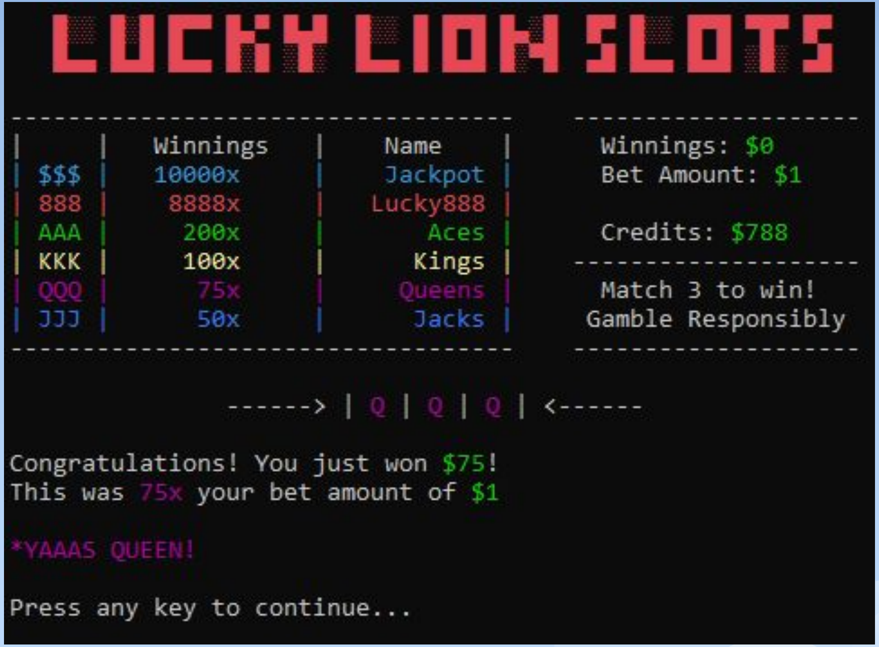
- A blocker appeared here (implementing classes helped)
- Checks if a, b, c matched and depending on what symbol it was, it displayed a different winning message as well as calculating the winnings and adding it to credits.



```python
    else:
        print("\n\n Sorry, no win this time buddy.")
        Game.winnings = 0
        press_to_continue()
```

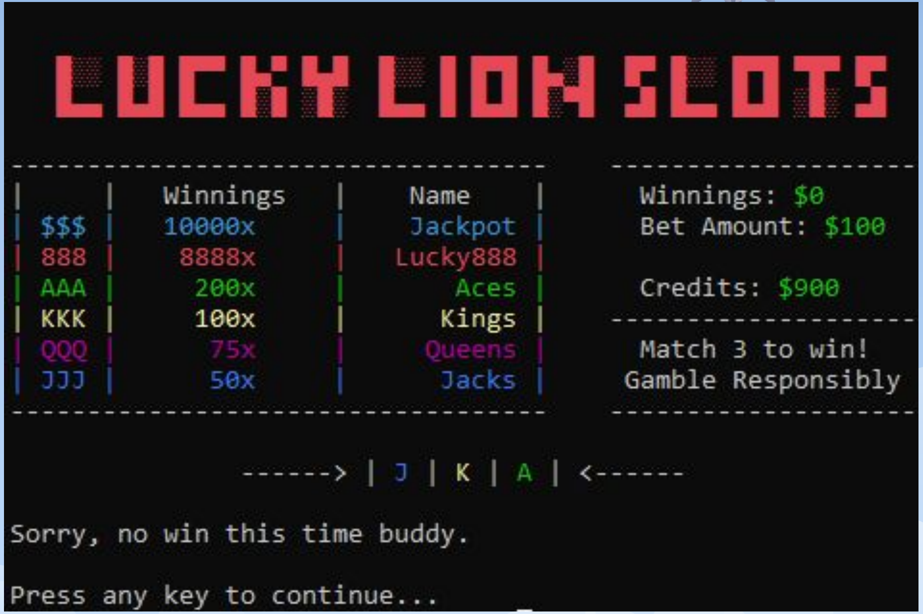
Check Win Feature in DEMO

WINNING OUTCOME



```
LUCKY LION SLOTS

-----------------------------------   --------------------
|     |   Winnings   |     Name    |    Winnings: $0
| $$$ |   10000x     |   Jackpot   |    Bet Amount: $1
| 888 |    8888x     |  Lucky888   |
| AAA |     200x     |      Aces   |    Credits: $788
| KKK |     100x     |     Kings   |  --------------------
| QQQ |      75x     |    Queens   |    Match 3 to win!
| JJJ |      50x     |     Jacks   |   Gamble Responsibly
-----------------------------------   --------------------

                 ------> | Q | Q | Q | <------

Congratulations! You just won $75!
This was 75x your bet amount of $1

*YAAAS QUEEN!

Press any key to continue...
```

LOSING OUTCOME



```
LUCKY LION SLOTS

-----------------------------------   --------------------
|     |   Winnings   |     Name    |    Winnings: $0
| $$$ |   10000x     |   Jackpot   |    Bet Amount: $100
| 888 |    8888x     |  Lucky888   |
| AAA |     200x     |      Aces   |    Credits: $900
| KKK |     100x     |     Kings   |  --------------------
| QQQ |      75x     |    Queens   |    Match 3 to win!
| JJJ |      50x     |     Jacks   |   Gamble Responsibly
-----------------------------------   --------------------

                 ------> | J | K | A | <------

Sorry, no win this time buddy.

Press any key to continue...
```

TESTS/ERROR HANDLING

Tests

TEST CASE ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail	Error Handling
TC01	Check Deposit with Integer	1. Get to deposit page 2. Enter integer	deposit = int()	Program should take the integer and continue running the code	As expected	PASS	nil
TC02	Check Deposit with 'word' input	1. Get to deposit page 2. Enter 'hello'	deposit = ‘hello’	Program should print error message and return back to loop	As expected	PASS	nil

Tests

TEST CASE ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail	Error Handling
TC03	Check Bet with Integer	1. Get to bet page 2. Enter integer	bet = int()	Program should take the integer and continue running the code	As expected	PASS	nil
TC04	Check Bet with 'word' input	1. Get to bet page 2. Enter 'hello'	bet = ‘hello’	Program should print error message and return back to loop	As expected	PASS	nil

Tests

TEST CASE ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail	Error Handling
TC05	Check Bet with 'q' input	1. Get to bet page 2. Enter 'q'	bet = lower.(‘q’)	Should withdraw credits and display end screen	As expected	PASS	nil
TC06	Press keys on keyboard while reel is spinning	1. Start reel spin. 2. Press keys while spinning	key presses whilst function is running	Nothing should happen	The letters show up on the terminal while reel is spinning. This affected the press_to_continue () function.	**FAIL**	Implemented flush_input() to make sure the inputted keypresses are flushed before press_to_continue() is called.

Tests

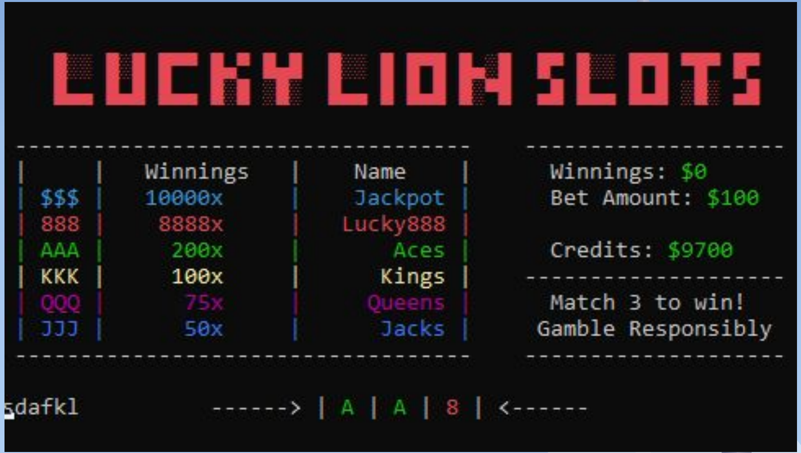
TEST CASE ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail	Error Handling
TC07	Press enter on keyboard while reel is spinning	1. Start reel spin. 2. Press 'enter' while function is running	Enter key input is recorded	Nothing should happen	The reel function gets printed multiple times while it is running	**FAIL**	Added warning to not press

Error TC06

- When the reel was spinning, the user can sometimes input keypresses.

- This showed up on the side of the terminal as shown in the image

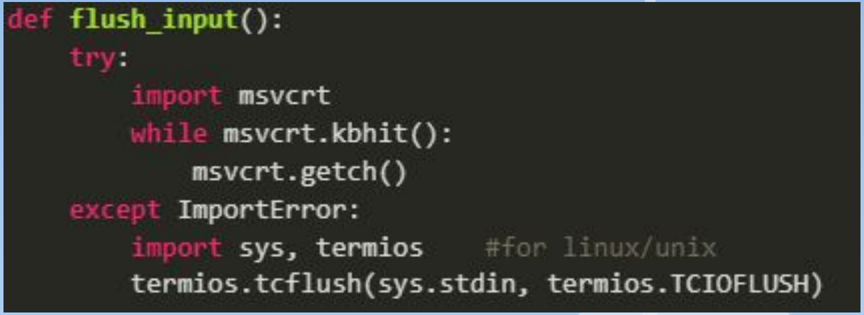
- This then affected the outcome page as this input was recognised for the following 'press_to_continue()' function and it made the next page get skipped



```
LUCKY LION SLOTS

-----------------------------------------       --------------------
|     |   Winnings    |     Name     |           Winnings: $0
| $$$ |   10000x      |   Jackpot    |           Bet Amount: $100
| 888 |    8888x      |   Lucky888   |
| AAA |     200x      |       Aces   |           Credits: $9700
| KKK |     100x      |      Kings   |          --------------------
| QQQ |      75x      |     Queens   |           Match 3 to win!
| JJJ |      50x      |      Jacks   |          Gamble Responsibly
-----------------------------------------       --------------------

sdafkl          ------> | A | A | 8 | <------
```

Error TC06 - Handling

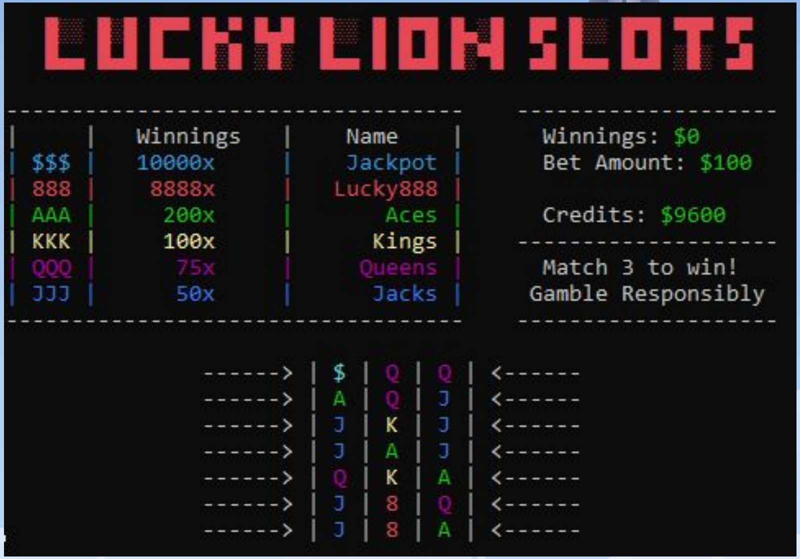
- This was handled by implementing a function to flush the input
- This function was called before the press_to_continue function was called
- The way this is written is that it can stay as its own little function and use its own module imports
- The function will check the import msvcrt for macOS uses and run the macOS function
    - If the msvcrt module cannot be imported the system is not a macOS
- It will then import sys, termios modules for linux/unix and run the function to clear inputs
- I chose to keep the imports here to make the flush_input function work as its own thing



```
def flush_input():
    try:
        import msvcrt
        while msvcrt.kbhit():
            msvcrt.getch()
    except ImportError:
        import sys, termios    #for linux/unix
        termios.tcflush(sys.stdin, termios.TCIOFLUSH)
```

Error TC07

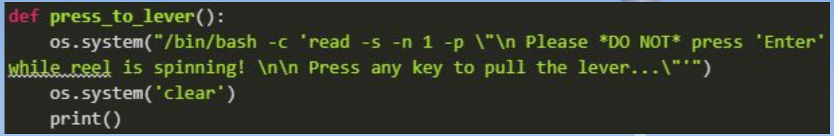
- When the reel is spinning, the user may accidentally press ‘Enter’ on the keyboard
- This causes the terminal to push to the next line
- This caused an error as it caused the print in the reel feature to remain on the screen



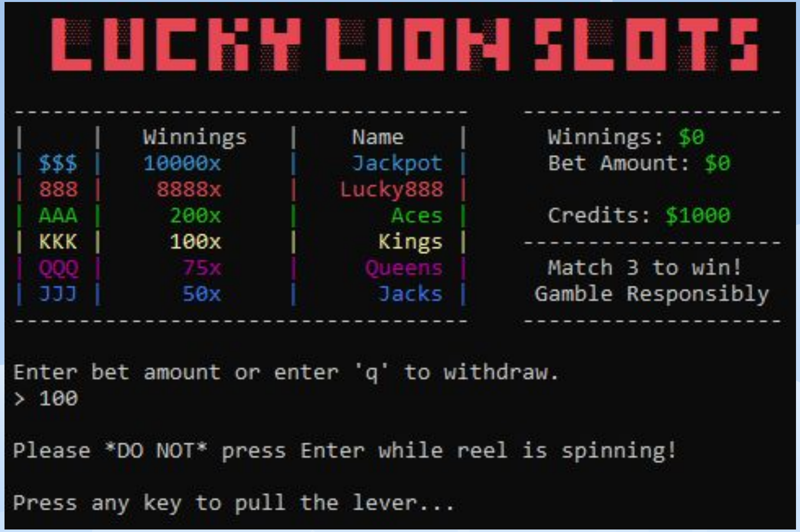
```
LUCKY LION SLOTS

-----------------------------------------     -----------------------
|     |   Winnings    |     Name     |        Winnings: $0
| $$$ |   10000x      |    Jackpot   |        Bet Amount: $100
| 888 |    8888x      |   Lucky888   |
| AAA |     200x      |       Aces   |        Credits: $9600
| KKK |     100x      |      Kings   |     -----------------------
| QQQ |      75x      |     Queens   |        Match 3 to win!
| JJJ |      50x      |      Jacks   |       Gamble Responsibly
-----------------------------------------     -----------------------

                  ------> | $ | Q | Q | <------
                  ------> | A | Q | J | <------
                  ------> | J | K | J | <------
                  ------> | J | A | J | <------
                  ------> | Q | K | A | <------
                  ------> | J | 8 | Q | <------
                  ------> | J | 8 | A | <------
```

Error TC07 - Handling

- To handle this, I decided to put in a warning to not press ‘Enter’ while the reel is spinning
- This is purely a visual error though and does not affect the output
- This is why I decided to place the warning as it does not break the code or the gameplay



```
def press_to_lever():
    os.system("/bin/bash -c 'read -s -n 1 -p \"\n Please *DO NOT* press 'Enter'
while reel is spinning! \n\n Press any key to pull the lever...\"'")
    os.system('clear')
    print()
```

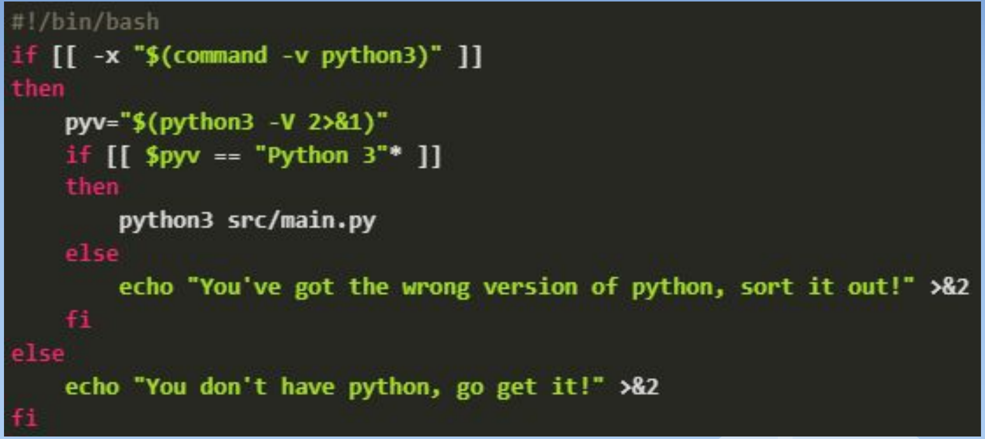


```
LUCKY LION SLOTS

-----------------------------------      --------------------
|     |     Winnings     |     Name    |       Winnings: $0
| $$$ |     10000x       |    Jackpot  |       Bet Amount: $0
| 888 |      8888x       |   Lucky888  |
| AAA |       200x       |       Aces  |       Credits: $1000
| KKK |       100x       |      Kings  |   --------------------
| QQQ |        75x       |     Queens  |     Match 3 to win!
| JJJ |        50x       |      Jacks  |   Gamble Responsibly
-----------------------------------      --------------------

Enter bet amount or enter 'q' to withdraw.
> 100

Please *DO NOT* press Enter while reel is spinning!

Press any key to pull the lever...
```

Bash Script Executable

Requirement:
Utilise developer tools to facilitate the execution of the application

- The purpose of this script is to allow users to more easily run the application
- Displayed is the Bash Script in the source folder
- This script, when run, will first check if python3 is installed
- If it is installed correctly, it will run the main.py python file
- However, if it is not installed, it will let the user know if it is the incorrect version or if they do not have it at all



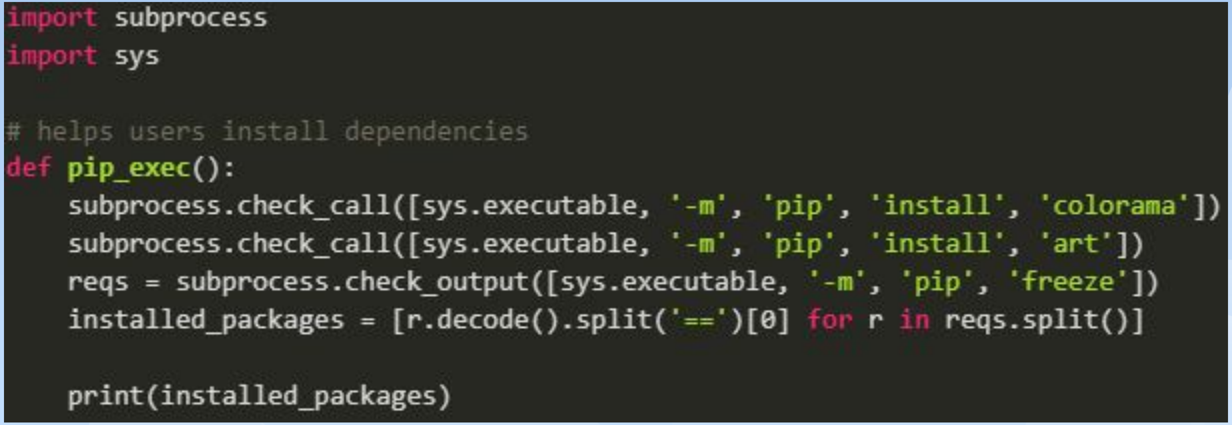
```bash
#!/bin/bash
if [[ -x "$(command -v python3)" ]]
then
    pyv="$(python3 -V 2>&1)"
    if [[ $pyv == "Python 3"* ]]
    then
        python3 src/main.py
    else
        echo "You've got the wrong version of python, sort it out!" >&2
    fi
else
    echo "You don't have python, go get it!" >&2
fi
```

Bash Script Executable

- This is the function for the bash script

- Just running the main.py file did not seem sufficient to help users run my application more easily

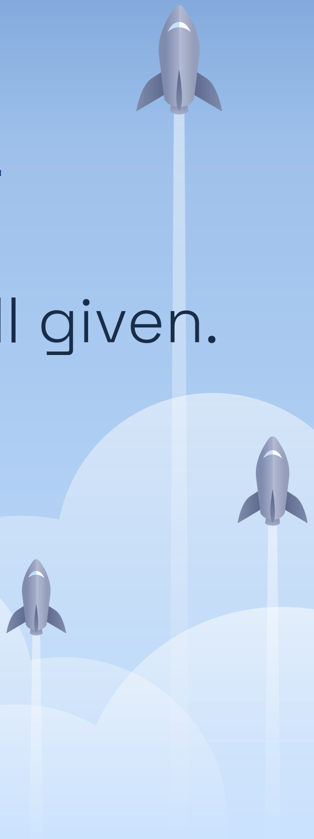
- This function checks if the third party modules required are installed and then installs them for the user



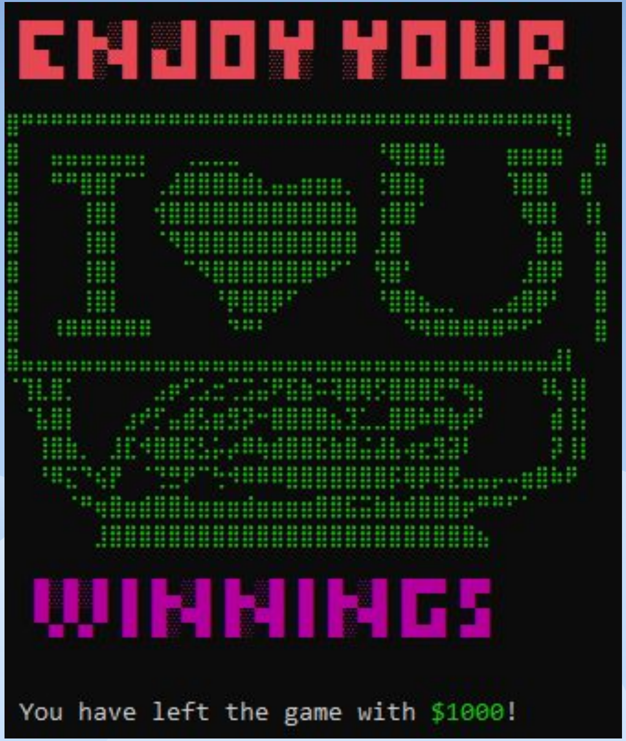
```python
import subprocess
import sys

# helps users install dependencies
def pip_exec():
    subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'colorama'])
    subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'art'])
    reqs = subprocess.check_output([sys.executable, '-m', 'pip', 'freeze'])
    installed_packages = [r.decode().split('==')[0] for r in reqs.split()]

    print(installed_packages)
```

Credits

- Presentation template by SlidesCarnival
- Thank you to all the educators at Coder Academy for all the fast help we were all given.



GAMBLE RESPONSIBLY



```
You have left the game with $1000!
```