W200 Python Project 1

Name: Kuangwei Huang   Date: March 15, 2018

WAH!-Gyu Farm

This is the final write-up for my project which is a simple text-based game.

Thoughts and reflections

I enjoyed writing this program, and the game was fun to envision and code. My initial design document listed out classes, attributes and methods which have changed slightly as I coded and tested different sections of the code. The final script basically creates a menu object as an instance and then runs a launch method. The menu then creates the subsequent gameplay object which drives most of the game. When the game ends, the program loops back to menu for further selection and options.

In order to simply and deliver the project, I ended up deleting the feature of the game which kills the cows if they have been starved or are depressed to the point of suicidal. That allowed me more time to do play testing and balancing to adjust the metrics to allow for the game to be hopefully sufficiently challenging, and having re-play value.

One challenge which I was not able to resolve is setting up my Git Bash for Windows to print out the UTF-8 characters denoting the music note "♫" (\u266b). I will run the program "wahgyu_farm.py" in Command Prompt for Windows for my demonstration as it seems to work fine in that environment. I have not tested this on a Mac OS. In case it is not able to run in Mac or Unix Terminal due to encode errors, please run the separate file in my project submission folder "wahgyu_farm_non_utf.py" which replaces the music note symbols with a "la~la de~da~".

No.	Filename	Remarks
1	w200_project1_Design_doc-Kuangwei.pdf	Initial design document submitted
2	w200_project1_Final_writeup-Kuangwei.pdf	Final write up (this document)
3	wahgyu_farm.py	Main version of the program
4	wahgyu_farm_non_utf.py	Alternate version (without UTF-8 ♫ char)

The final "shipped" Game premise" and "Rules of the game" are provided below. I have also provide the updated class, attributes and methods table.

Game premise:

You are a new hire at a farm whereby you are in charge to care for a number of special genetically engineered, low-carbon footprint, ultra-sustainable cows. These cows consume very little food, have minimal biological waste and produce extremely desirable high-grade marbled beef. These cows are called WAH!-Gyu, and are true a marvel for sustainable meat production!

The only drawback is that WAH!-Gyu's have a very special low calorie diet of only a bucket of craft trappist beer a day, and have fickle moods. These special cows would only produce their superior marbling when they have enough calories and are kept sufficiently happy. While WAH-Gyu's benefit from a good diet of beer, they have a 50% chance of getting emotionally depressed if they have had 2 beers, and need constant massaging and new indie folk tunes to keep them happy.

Rules for the game:

1. Depending on difficulty selected the number of cows would range from 2 to 4 cows.
2. Each game starts with cows at a calorie rating and a mood rating depending on number of cows.
3. All cows start with a marbling rating of 0.
4. Each cow loses one calorie (-1) and one mood point (-1) at the start of each successive turn.
5. As long as a cow's calorie and mood are both above 1, the cow gains a marbling point (+1); if either calorie or mood rating reaches zero, the cow loses a marbling point (-1).
6. If a cow's calorie rating is more than 6, the cow has a 50% chance to be depressed (mood -2)
7. There are 10 turns in a game and the player can do only 1 action per turn:
    a. Feed a cow a bucket of beer (+6 calorie), -1 beer stock
    b. Massage a cow (+5 mood)
    c. Play music (randomly +1, +2 or +3 mood to all cows, they have different music tastes), -1 tunes stock
    d. Go to the abbey and get some trappist beer (+5 beer buckets)
    e. Go to iTunes and buy a bunch of indie songs (+5 songs)
8. At the end of 10 turns, your dear WAH!-Gyu cows will be sent for slaughter and you will get scored by the owner of the farm, based on the sum of the marbling of all the cows under your charge.

Classes:

No.	Class	Attributes	Methods
1	Gameplay	▪ Difficulty setting ○ Easy, medium, or rare ▪ Total number of cows ▪ Cow names ▪ Turn counter ▪ Maximum number of turns in a game ▪ Player action and cow choice ▪ Cows marbling rating at end game ▪ Player rank	▪ __init__ ▪ startup ○ Prints starting cows, ipod and beer fridge ○ Set ups a loop to execute method *new_turn* until max turns are reached ○ Subseqently execute method *end_game* ▪ new_turn ○ Executes another class *Player* and its method *query_action* to get the choice of the player on which action to take and which cow to act on. ○ Executes *game_turn* to effectuate the action ▪ game_turn ○ Runs through the results for each possible choice of player action: ○ If feed cow, reduce stocks in beer fridge and execute method *drink_beer* on the affected cow. ○ If massage cow, execute method *get_massage* on the affected cow. ○ If play music, reduce tunes stock in ipod and execute method *listen_music* on all cows. ○ If buy more beer, execute *add_stocks* on beer fridge. ○ If buy more tunes, execute *add_stocks* on ipod. ○ Execute *end_turn* method when complete. ▪ end_turn ○ Print status of the beer fridge and ipod. ○ Set the emotional string to an empty string if the last action was not to feed a cow a beer. ○ Execute *reset_music* on all cows if the last action was not to play music. ○ Print all cows. ▪ end_game ○ Calculates the final cows marbling, the total score for all cows and the ranking ○ Publish congratulations and rank ○ Executes a static method from menu calls *enter_to_cont*.
2	Menu	▪ Difficulty setting ○ Easy, medium, or rare	▪ __init__ ▪ @staticmethod: enter_to_cont ▪ launch ○ Enter a loop with 4 choices, looping

No.	Class	Attributes	Methods
			unless option to quit is selected. ○ Option 1: Set difficulty ○ Option 2: Start a new game, creates a new *gameplay* instance and executes *startup* method ○ Option 3: Displays game premise and rules ○ Option 4: Quit.
2	Player	▪ Player's action in a turn	▪ __init__ ▪ query_action ○ Enter a loop to query the player for a choice of 5 actions in a turn and for options to feed a cow or give a cow a massage, prompt for which cow to perform the action on: ○ 1. Feed a cow some beer ○ 2. Give a cow a massage ○ 3. Play some music ○ 4. Buy more beer ○ 5. Buy more tunes ○ Error checking if action is feeding beer when beer stocks are zero, prompt for alternate choice. ○ Error checking if action is playing music when there are no new songs in iPod, prompt for alternate choice. ○ Return the selected action and which cow to perform this action on (where applicable).
3	WAH!-Gyu cow	▪ Cow name ▪ Index number ▪ Calories (starts at a value equivalent to number of cows) ▪ Mood ▪ Min Calories ▪ Max Calories ▪ Calories a bucket of beer provides ▪ Min mood ▪ Max mood ▪ Effect of depression on mood ▪ Effect of massage on mood ▪ Effect of music on mood ▪ Emotional depression	▪ __init__ ▪ drink_beer ○ + beer calories ○ 50% chance to execute *depressed* method ▪ depressed ○ effect mood with depression ○ Print a statement ○ Set emotional string ▪ get_massage ○ effect mood with massage ▪ listen_music ○ effect mod with music ○ set strings to print musical note ▪ reset_music ○ reset strings to not print musical note ▪ update_status ○ increases marbling or decreases marbling based on the mood and

No.	Class	Attributes	Methods
		string for printing ▪ Marbling ▪ Miscellanous print strings for graphics	calories of the cow. o decrease the mood and marbling by 1 for the next turn. ▪ __str__ o Returns a string ascii art to print out the cow and its status.
4	Farm supply	▪ Name ▪ Stocks ▪ Units description ▪ Unit description ▪ use_stock: How much stock to use at a time ▪ add_stock: How much stock to add when a restock occurs	▪ __init__ ▪ any_left o Check if there are remaining stocks prior to the selection of the player to either feed a cow a bucket of beer or play music. ▪ add_stock o Increase stocks by the re_stock amount ▪ reduce_stock o Decrease stocks by the use_stock amount