FPP Report Nina Wyckoff, 2917069W, Transferring to Computing Science

1. “Own work” statement

The program submitted by me for the CS-1CT Free Programming Project is my own work, apart from the following parts, which I explain in the *How it works* section below:

1. N/A, all work is my own

2. What? Cookie Adventure

Cookie Adventure is an interactive choose-your-own-adventure style game. The user enters some personal details and chooses an adventure. They are prompted through the IDLE shell to make choices at certain junctures which affects the next part of the story. They earn points based off of their ending and can can have bonus points (or a penalty) from how they have interacted by the narrator persona, Geohn, throughout the game. At the end, turtle is used to display a graph of their results as well as a cookie (if their score was high enough to earn one)

3. What motivated you to choose this project

I am interested in the user experience as well as comedic creative writing. A choose-your-own-adventure style game seemed a natural solution! I really enjoyed adding complexity that enriched the user experience and made new ways to enjoy the game. I want to get better at making complex point systems given that I want to pursue computing science.

4. **Brief** user instructions

The main program (cookie.py) should be in the same folder as files storing functions (adventurePieces.py, turtlePieces.py, introPieces.py) as well as the adventure text files (spyAdventure.txt, antAdventure.txt). To run the program, open cookie.py and select “run” from the computer toolbar. The game will be text based through the IDLE shell.

5. How it works

There are three stages of the game: The introductory stage, the adventure stage, and the results page. The introductory stage gathers input from the user such as their name and pronouns and stores that into a variable and a dictionary respectively. The program inputs these values into preplanned sentences to personalize the user experience. The next stage is the adventure stage. The program prompts the user to choose between two stories, a spy story and an ant story. The program opens the corresponding text file. The program reads the text file line by line into a dictionary and further processes it (more explained below). The user makes choices by inputting ‘A’ or ‘B’ into the shell. Each choice corresponds with a different term in the dictionary. The user reaches the end of the adventure and their points are tallied up, including bonus points or penalty from their earlier text input in the intro stage. The results page is generated with the functions in the turtlePieces file. If the user reached the threshold of 25 points, a cookie is displayed, if not then the space is grayed out. A key and graph is also generated to give the user information on their performance as well as relevant score milestones for their information.

Data structures

When the user chooses which adventure they’d like, the program reads the text file into a dictionary of dictionaries, where the key terms of the AB strings are dictionaries that include the key term text

(related to the text of the story chunk) and the key term points (related to the points of the ending if applicable).
There is also a pronouns dictionary where the key terms are grammatical labels for different pronouns like subjPro and obj Pro associated with user input pronouns like she and her. I also have complex data structures with the points system. This is most visible in the scoreCalc function in the turtlePieces.py file. The points dictionary has a geohnHappy key that corresponds with accumulated mood score of the narrator character, Geohn, and the points dictionary also has an "adventure" key that corresponds to another dictionary with information on the points value from the adventure section as well as the corresponding color. geohnScore is a dictionary created that stores the converted points value associated with points["geohnHappy"] as well as adding a key for color that is either green or red depending on the integer value of points["geohnHappy"].
I use lists throughout the intro stage like the capitalLetters list and the blankEntry list to catch formatting errors that would interfere with the program. For instance, the userName string's 0 index must be found in the capitalLetters list.

Interesting / tricky / subtle aspects of the code meriting further explanation

The story text files are formatted in such a way that each new line features a colon separating a string of A or Bs and a long string that details a section of the story, as well as a third points section if the story is the end piece of the adventure.
A basic mechanic of the game is that when the user inputs A or B for a choice, that character is added to s variable called abChain. The program then looks for a string value that equals abChain in the constructed dictionary and then displays that to the user. The abChain builds over the course of the adventure which is how the relevant next branch is accessed.
Throughout the intro stage, points["geohnHappy"] is manipulated inadvertently by the user based on their input. For instance, in the nameAcquisition function in introPieces.py, if the user's name is the same as the narrator's, Geohn, integers are added to the value of points["geohnHappy"], whereas if the user inputs their name as "John", integers are subtracted from points["geohnHappy"] because the narrator has quite the complex. Additionally, the program will elect to change the value of userName to "J***" for comedic effect. There are penalties to points["geohnHappy"] if the user repeatedly incorrectly formats their input. I use the userPNs dictionary created in the pronounAcquisition function in the cookieFrolickMonologue to customize the blurb with the user's pronouns. In the adventure stage, the program always checks if the current term has the key "points" which indicates the story is over. In the turtlePieces file, I use the data in the points dictionary as parameters for graph creation. I have a lot of nested functions in turtlePieces.py, particularly in the graphGeneration function. It is possible to get a good score on the adventure but still not earn a cookie because geohnHappy was not high enough. The red sections of the graph represent the penalty from a low geohnScore and show either your negative score v alue or the section of your adventure score that is subtracted. The bar actually goes back over itself, whereas if the geohnScore is high, the bar ads to the existing length.

6. Particular challenges I overcame

I have my functions stored in separate files from the main program. This in combination with my implementation of nested functions meant I spent a lot of time figuring out when to use global variables, local variables, and parameters. This was especially complicated when I wanted to use the version of a variable that what modified in a previous function in another function. For instance, if the user's inputted name was John, the program then reassigns userName as J*** and I need to carry that variable forward to the next functions. I also needed help with incorporating the userName variable into the dictionary creation because I wanted the user's name to be embedded in the text of the story. I also spent a long time devising the point system in order to make the game fair to the player without being too easy.