Documentation Template

| ***You MUST provide evidence showing how the problem has been decomposed, how the components have been developed and trialled, and of how they have been assembled and tested to create a final, working outcome.*** |
| --- |

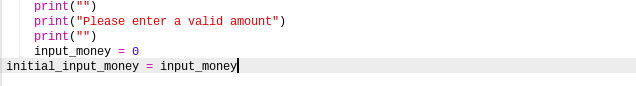
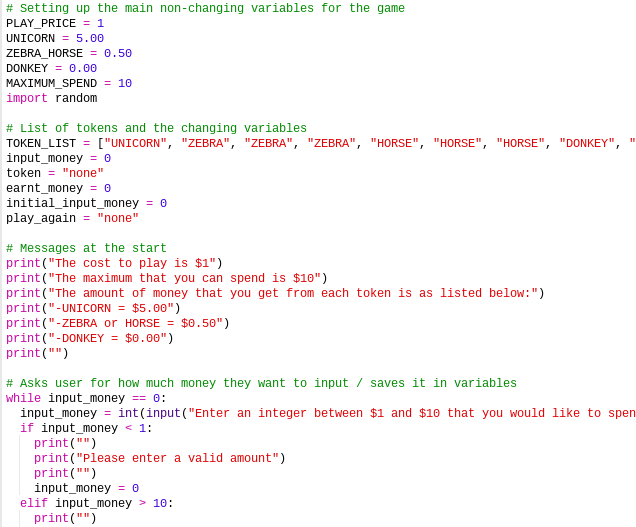
### GIthub Link: <https://github.com/AnnieC123/EXDTE_AnnieC.git>

### Outline / Decomposition

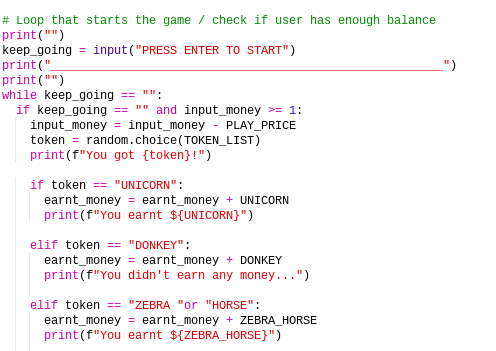
*Please write down your task decomposition here (a numbered list is a good idea)*

1. *Ask how much the user wants to play with (must be more / equal to $1 and less than / equal to $10)*
2. *Randomly generate a token (zebra, horse, unicorn or donkey). Display the token*
3. *Check that the house has an advantage. Adjust list to ensure that users don't get too many unicorns.*
4. *Set up winnings system*
5. *If token is zebra or horse, subtract $0.5 from the total, set ‘amount\_won to 50c*
6. *If token is unicorn, add $5 to the total, set ‘amount\_won to $5*
7. *Otherwise subtract $1 from the total, set ‘amount\_won to $0*
8. *State how much the user won.*
9. *If the amount\_won is more than 0, feedback is “Congratulations, you won {amount}”*
10. *Otherwise feedback is “Sorry, you did not win anything this round”.*
11. *State the new total*
12. *Setup end mechanics*
13. *If the total is more than $1.00, allow user to press ‘q’ to quit or <enter> to continue*
14. *If user presses <enter> generate a new token*
15. *If the total is less than $1.00 output farewell message and end game*

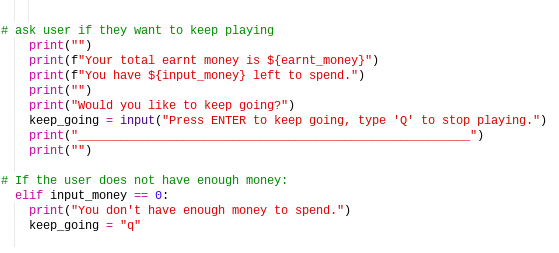
### Version Log

*Your version log should go here. Annotated screenshots are a good idea at this point*

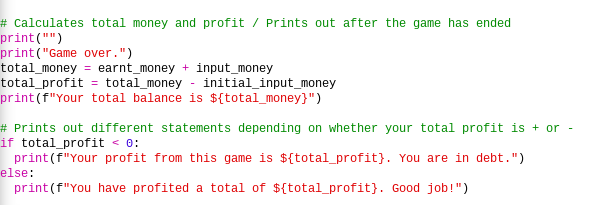
*1. Creating variables and getting user input.*

**

*2. Looping and generating the tokens*

**

*3. Printing out statements after each roll and asking if they want to play again / if they have enough money left.*

**

*4. Printing out the final balance after the game ends.*

### Component Testing

*Show that you have tested each component here. You should have a test plan and then screenshot proof for each component. You should also include notes that justify the major decisions you made.*

*Test plan: test expected inputs from user and also unexpected inputs to see what will happen.*

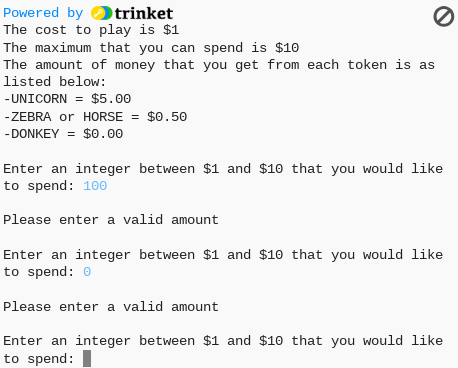
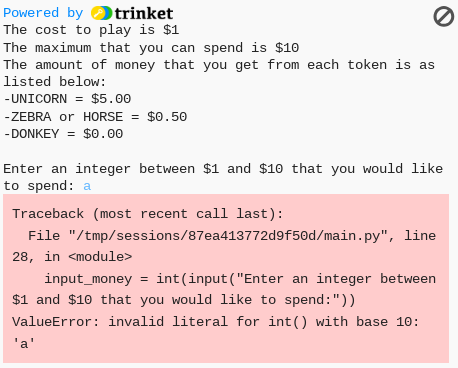
*Major decisions:*

*Creating a list for the tokens: so that we can change the odds and rates to get each token*

*Creating a constant variable for the amount each token gives you: So that it is easy to add each amount without getting confused with random numbers (e.g. + 5, +0.5, +0).*

### Assembled Outcome Testing

*Please show testing for your assembled outcome below. This should include a test plan followed by screenshot proof*

**

*The testing done above works because when you enter a value that is not an integer (e.g. “a”, “5.6” or “ “), it will return an ValueError message because the expected input is an int() and not a string or float.*

*The second screenshot shows that you have to enter a proper integer between 1-10 for the system to continue running, or else it will just keep asking you to enter a valid integer.*

### Usability Testing

*Write a list of things improvements which need to be made based on your usability testing. Then write down what you changed.*

*Improvements:*

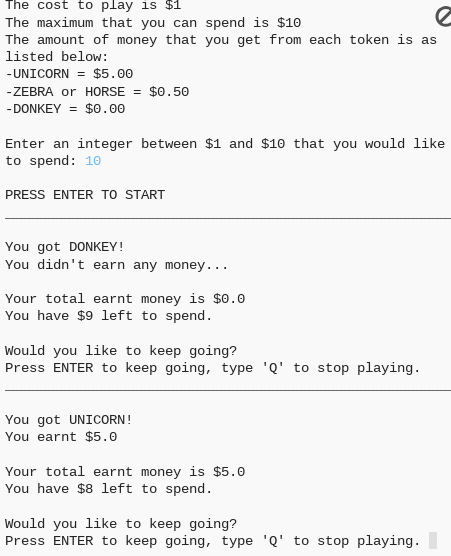
*Saying at the end whether the user has lost or won money in total.*

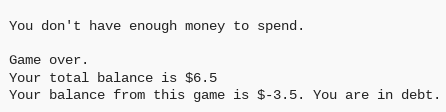
*Creating an option for the user to press q to stop the game after each token.*

*Showing your balance and money spent after each token.*

### Post Usability Test…

*Show that your post usability testing program works correctly*

**

*What it prints after you spend all your money or stop the game:*

### Social and End User Considerations…

**How did you ensure that your task was suitable for your chosen audience?**

*I made it user friendly and easy to understand how it works.*

**How have you honoured copyright?**

*No*

**How did you make your quiz easy to use?**

*By making it so that you only need to press enter to continue and press q to stop. It also shows how much you earnt after each round and how much you have left to spend.*