Yahtzee Game Journal

November 17 2023

Planning:

Timeline:

[2:00pm]

| **Date** | **Goal** |
| --- | --- |
| November 17 2023 | Review code and do planning  Requirements: flowcharts, Timeline, pseudocode |
| November 18 2023 | Program Journal  Implement code  (bug test) - if time |
| November 19 2023 | Complete if not finished  Crash test |

Program requirements:

* Must be uncrashable
* Sheet - displaying a player's full score sheet
* Score (1-13) - selecting a spot to score the current dice
* RollDice() - rolling all free dice
* DisplayDice() - showing the player their current dice.
* HoldDie (1-5) - a method of holding / keeping dice not to be rolled
* ReleaseDie(1-5) - a method of releasing dice so they can be rolled.
* Help() - prints out a list of all valid commands for your CLI
* Simulate rolling five dice (some or all)
* Simulate a turn by counting the number of rolls
* Switch between player (1) and player (2)
* Score each player’s game on a score sheet
* Tabulate a score and declare a winner

Firstly, I changed the dice symbols to something I prefer and which I find easier to read and then I will create the pseudocode and flowcharts for the functions that will be reused in my game such as, calculating totals based on category, as well as displaying the score card. For the four of a kind, I used the same principle I used for three of a kind but I upped the index and I kept receiving an index error. To fix this I had to make the range be from the length of the dice-4 and it seemed to work well after that.

Pseudocode Functions:

[3:23pm]

**Calculate Total of all categories**

function calculate\_totalscore(player\_scores):

total\_score = 0

for each category in player\_scores:

score = player\_scores[category]

if score is not None:

total\_score = total\_score + score

return total\_score

**Display Score Card**

function display\_score\_card(player):

print("Player " + player + "'s Score Card:")

for each category in score\_categories:

score = player\_scores[player][category]

print(category + ": " + score)

**Calculate Ones**

Function calculate\_ones(cup):

return sum([1 for die in cup if die ==1])

**Calculate Twos**

Function calculate\_twos(cup):

return sum([2 for die in cup if die ==2])

**Calculate Three of a Kind**

function calculate\_threeofakind(dice):

sort(dice)

for i from 0 to 2:

if dice[i] is equal to dice[i+2]:

return sum(dice)

return 0 // Return 0 if no three of a kind is found

**Calculate Four of a Kind**

function calculate\_fourofakind(dice):

sort(dice)

for i from 0 to length(dice) - 4:

if dice[i] is equal to dice[i+3]:

return sum(dice)

return 0 // Return 0 if no four of a kind is found

**Calculate Full House**

function calculate\_fullhouse(cup):

counts = create a list of counts for each number from 1 to 6 in cup

// For example count = number of 1s in cup...

if 2 is in counts and 3 is in counts:

return 25

return 0

**Calculate Small Straight**

function calculate\_smallstraight(cup):

sorted\_cup = sort the cup list in ascending order

if sorted\_cup is equal to [1, 2, 3, 4] or sorted\_cup is equal to [2, 3, 4, 5] or sorted\_cup is equal to [3, 4, 5, 6]:

return 30

return 0 // 0 points earned if no small straight

**Calculate Large Straight**

function calculate\_largestraight(cup):

sorted\_cup = sort the cup list in ascending order

if sorted\_cup is equal to [1, 2, 3, 4, 5] or sorted\_cup is equal to [2, 3, 4, 5, 6]:

return 40

return 0

**Calculate Yahtzee**

function calculate\_yahtzee(cup):

if number of occurrences of cup[0] in cup is equal to 5:

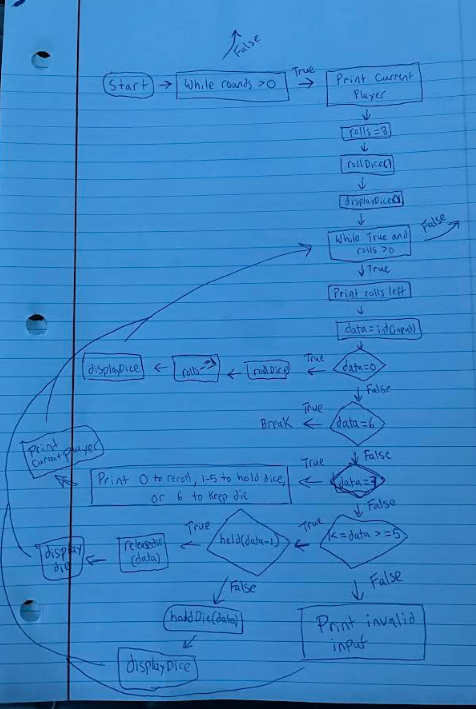
return 50

return 0 // if not yahtzee return 0 points

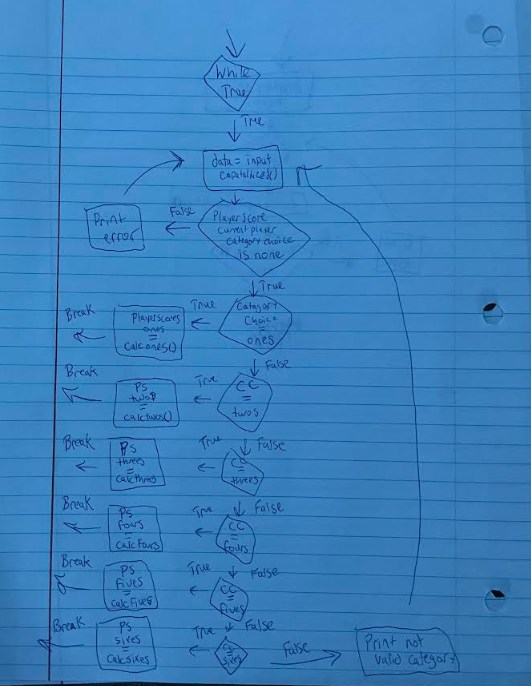
Flowcharts:

[8:00pm]

Next, I will create the flowcharts for the main game loop and select the category for points.



[9:45pm]



Next I will add the yahtzee screen and the command line instructions and then I will add some global variables to help my game such as rounds, rolls and current player turn.

\_\_ \_\_ \_ \_

\ \ / /\_ \_| |\_| |\_ \_\_\_\_\_\_ \_\_\_

\ V / \_` | ' \ \_|\_ / -\_) -\_)

|\_|\\_\_,\_|\_||\_\\_\_/\_\_\\_\_\_\\_\_\_|

Command Line Instructions

-choose 1-5 to hold that rolled die

-choose 0 to reroll all non held die

-choose 6 to keep all rolled die

current\_player = '1'

rounds = 13

rolls = 0

November 18 2023

Implementing code:

[12:00pm]

Firstly, I must create my score categories for each player as this will be a big impact on the game and to test to see if my display scorecard function works correctly.

# Define score categories

score\_categories = [

"Ones", "Twos", "Threes", "Fours", "Fives", "Sixes",

"Three of a Kind", "Four of a Kind", "Full House",

"Small Straight", "Large Straight", "Yahtzee", "Chance"

]

# Initialize player score sheets

player\_scores = {

'1': {category: None for category in score\_categories},

'2': {category: None for category in score\_categories},

}

I was having trouble making it all work properly and account for user errors such as entering a wrong value and outputting an error message to try again. So I looked back at older projects I did and thought it would be best to try a try and except loop to have it work correctly and that seems to have fixed the issues I was having.

while rounds > 0:

print(f"Current Player: Player {current\_player}")

# roll the dice

rolls = 3

rollDice()

displayDice()

# shows chance score potential

print(f"Player {current\_player} has a chance score of {chanceScore()}"

while True and rolls > 0:

try:

print(f"rolls left: {rolls}")

data = int(input("enter your choice (enter 7 for help): "))

if data == 0: # reroll non held die

rollDice()

rolls -= 1

displayDice()

elif data ==6 : # keep die

break

elif data == 7:

print("\nCommands…\n0 to reroll all non held die\n1-5 to hold the die in that slot\n6 to keep all die")

print(f"Current Player: Player {current\_player}"), displayDice()

elif 1 <= data <= 5:

if held[data - 1]:

releaseDie(data)

displayDice()

else:

holdDie(data)

displayDice()

else:

print("Invalid input. Please choose a die between 1 and 5.")

except ValueError:

print("Invalid input. Please enter a number.")

I will then show the chance score for the whole cup and display the scorecard for the player

.

# shows chance score potential

print(f"Player {current\_player} has a chance score of {chanceScore()}")

# display score sheet

display\_score\_card(current\_player)

Next, I will implement the choosing the category for the player turn that is easy to read, I must capitalize the input so that it matches the right category. It must also account for user error such as choosing a category that is already picked.

while True:

# choose category

category\_choice = input("Choose a category: ").capitalize()

if player\_scores[current\_player].get(category\_choice) is None:

if category\_choice.lower() == "ones":

player\_scores[current\_player]["Ones"] = calculate\_ones(cup)

break

elif category\_choice.lower() == "twos":

player\_scores[current\_player]["Twos"] = calculate\_twos(cup)

break

elif category\_choice.lower() == "threes":

player\_scores[current\_player]["Threes"] = calculate\_threes(cup)

break

elif category\_choice.lower() == "fours":

player\_scores[current\_player]["Fours"] = calculate\_fours(cup)

break

elif category\_choice.lower() == "fives":

player\_scores[current\_player]["Fives"] = calculate\_fives(cup)

break

elif category\_choice.lower() == "sixes":

player\_scores[current\_player]["Sixes"] = calculate\_sixes(cup)

break

elif category\_choice.lower() == "three of a kind":

player\_scores[current\_player]["Three of a Kind"] = calculate\_threeofakind(cup)

break

elif category\_choice.lower() == "four of a kind":

player\_scores[current\_player]["Four of a Kind"] = calculate\_fourofakind(cup)

break

elif category\_choice.lower() == "full house":

player\_scores[current\_player]["Full House"] = calculate\_fullhouse(cup)

break

elif category\_choice.lower() == "small straight":

player\_scores[current\_player]["Small Straight"] = calculate\_smallstraight(cup)

break

elif category\_choice.lower() == "large straight":

player\_scores[current\_player]["Large Straight"] = calculate\_largestraight(cup)

break

elif category\_choice.lower() == "yahtzee":

player\_scores[current\_player]["Yahtzee"] = calculate\_yahtzee(cup)

break

elif category\_choice.lower() == "chance":

player\_scores[current\_player]["Chance"] = chanceScore()

break

else:

print("Not a valid category, please try again.")

else:

print("Invalid category choice or category already scored. Please choose a valid and unscored category.")

Then I will display the updated scorecard and switch the player and take one off of the rounds and then loop back to the start so that when rounds = 0 the game will end.

# Display the updated score card

display\_score\_card(current\_player)

# switch to the next player

current\_player = '2' if current\_player == '1' else '1'

rounds -= 1

While testing everything I noticed that it would only let me get through half the game and then it would end. This is because I had the rounds set to 13 which is only enough for one player so I had to double it to account for both players.

Next, when the game ends I will print the final scorecards that the players have gotten. Then I will use my calculate total function to declare a winner!

# print final scores

print("Final Scores:")

display\_score\_card('1')

display\_score\_card('2')

# calculate totals and determin winner

total\_playerscore1 = calculate\_totalscore(player\_scores['1'])

total\_playerscore2 = calculate\_totalscore(player\_scores['2'])

if total\_playerscore1 > total\_playerscore2:

print("Player 1 wins!")

elif total\_playerscore1 < total\_playerscore2:

print("Player 2 wins!")

else:

print("Tie!")

Bug testing:

[8:00pm]

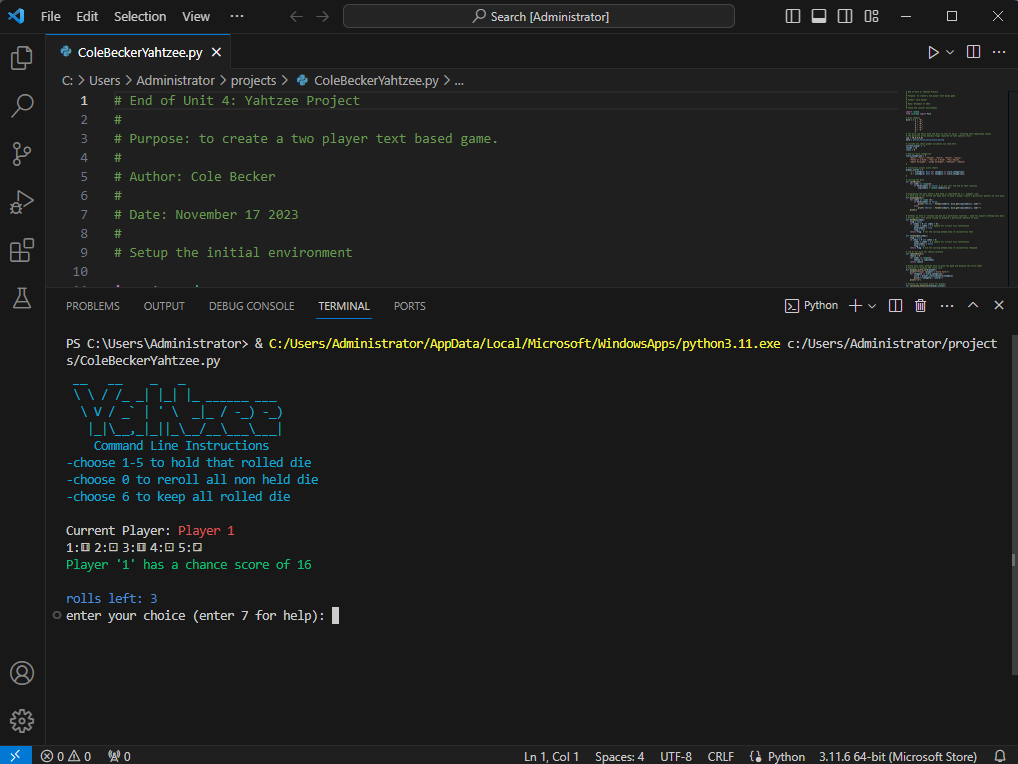
Everything seems to be working as intended now and the game seems to account for user errors and I couldn’t get the code to crash.

November 19 2023

Color and Looks

[2:00pm]

I will now add colorama to give the code some colors so it is appealing to look at, as well as space out the code with \n to my liking until I am satisfied with how it looks.



Conclusion

[4:00pm]

I had a lot of fun working on this project and I am happy I got the chance to do this game because it helped me with my pseudocode writing skills and implementing bigger functions into my code. Also, It was more of a challenge as I came into contact with more challenging obstacles that I needed to fix so that my game would work correctly. I also played yahtzee with my family as a kid so it was a lot of fun to create a virtual version of it and implement all the different aspects of the board game.