# End of Unit 4: Yahtzee Project

#

# Purpose: To make a yahtzee game played by two players

#

# Author: Rodney Dong

#

# Date: May 6 2021

# The redo loop that is only useful at last when asking if the user wants to play again or not

redo = True

while redo == True:

import random

import pcinput

dice = { 1 : "âš€",

2 : "âš",

3 : "âš‚",

4 : "âšƒ",

5 : "âš„",

6 : "âš…"

}

# the dice cup which hold the dice (a list of dice) - starting with impossible values

# the parallel array boolean flags required to hold specific dice

cup = [0,0,0,0,0]

held = [False,False,False,False,False]

# Modular Functions

# rolling the dice

def rollDice():

for index in range(5):

if held[index] == False: # we can roll the die at that location

cup[index] = random.randint(1,6)

return cup, held

# This rolls dice index that are True and holds the False ones

def selectRollDice():

for index in range(5):

if held[index] == True: # we can roll the die at that location

cup[index] = random.randint(1,6)

# This is actually selecting the dices that the player wants to be rolled

def holdDie(index):

flag = False

if (index > 0 and index < 6):

index = index - 1 # update for correct list interaction

held[index] = True

flag = True

return flag # let the calling method know if successfully held

def chanceScore():

chance = 0

for index in range(5):

chance += cup[index]

return chance

# I used the most basic way to find out if a set of dice is a 3 of a Kind, 4 of a kind...etc

# These functions are pretty straight forwards except for the small straight one

def is3ofAKind(dice, TorF):

while True:

if dice[0] == dice[1] == dice[2] and dice[3] != dice[4] and dice[2]:

break

elif dice[1] == dice[2] == dice[3] and dice[0] != dice[4] and dice[1]:

break

elif dice[2] == dice[3] == dice[4] and dice[0] != dice[1] and dice[2]:

break

else:

message = False

TorF.append(message)

return TorF

message = True

TorF.append(message)

return TorF

def is4ofAKind(dice, TorF):

while True:

if dice[0] == dice[1] == dice[2] == dice[3] and dice[0] != dice[4]:

break

if dice[1] == dice[2] == dice[3] == dice[4] and dice[1] != dice[0]:

break

else:

message = False

TorF.append(message)

return TorF

message = True

TorF.append(message)

return TorF

def fullHouse (dice, TorF):

while True:

if dice[0] == dice[1] == dice[2] and dice[3] == dice[4] and dice[0] != dice[3]:

break

if dice[0] == dice[1] and dice[2] == dice[3] == dice[4] and dice[0] != dice[3]:

break

else:

message = False

TorF.append(message)

return TorF

message = True

TorF.append(message)

return TorF

def smallStraight(dice, TorF):

# In this function, I first made a copy of the set of dice

# Then the for loop will identify is there is dices that are similar

# later, it will remove one of the dices in a pair and finds if it is a small straight

# All the remove and indentify part is done on the transscript so it would not affect the actually set of dice

dice1 = []

for i in range (0,5):

dice1.append(dice[i])

for x in range(0,4):

if dice1[x] == dice1[x+1]:

dice1.remove(dice1[x])

break

while True:

if dice1[0]+1 == dice1[1] and dice1[1]+1 == dice1[2] and dice1[2]+1 == dice1[3]:

break

else:

message = False

TorF.append(message)

return dice, TorF

message = True

TorF.append(message)

return TorF

def largeStraight(dice, TorF):

while True:

if dice[0]+1 == dice[1] and dice[1]+1 == dice[2] and dice[2]+1 == dice[3] and dice[3]+1 == dice[4]:

break

else:

message = False

TorF.append(message)

return TorF

message = True

TorF.append(message)

return TorF

def yahtzee (dice, TorF):

while True:

if dice[1] == dice[2] == dice[3] == dice[4] == dice[0]:

break

else:

message = False

TorF.append(message)

return TorF

message = True

TorF.append(message)

return TorF

print("Welcome to the Game of Yahtzee")

# I also made two transcripts for each variable here

# One copy is for the turn or round that gets set back to zero every round

# Another one is for the whole game that only adds up

# Memory Input (This is probably the most challenging part of the project to know when a transcript should be used and when the actual variable shoud be used

player1Score = 0

threeKindTotal1 = 0

fourKindTotal1 = 0

fullHouseTotal1 = 0

smallStraightTotal1 = 0

largeStraightTotal1 = 0

yahtzeeTotal1 = 0

ones11 = 0

twos11 = 0

threes11 = 0

fours11 = 0

fives11 = 0

sixes11 = 0

player2Score = 0

threeKindTotal2 = 0

fourKindTotal2 = 0

fullHouseTotal2 = 0

smallStraightTotal2 = 0

largeStraightTotal2 = 0

yahtzeeTotal2 = 0

ones22 = 0

twos22 = 0

threes22 = 0

fours22 = 0

fives22 = 0

sixes22 = 0

# Main Game Loop

while True:

held = [False,False,False,False,False]

# First Player Game Loop

print("IT'S PLAYER 1'S TURN")

rollDice()

print(cup)

print('"r" to restart and "e" to end')

# Lets the user selects dices that they wants to be rolled

# Process Requirement (Anything that is not on the surface when the player runs the program are all process)

while True:

# Input Requirement (What is user inputs)

message = input("P1(firstRoll) Enter the dices you want to roll by sequence (1-5): ")

# "r" redos, so held needs to be all False also

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

# After the user "e"nds the program will roll the dice and breaks the loop

elif message.lower() == "e":

selectRollDice()

break

# Rolls specific dices only if the user inputs the correct numbers

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

# Output Requirement (Anything printed to the user for reference)

print(cup)

print('"r" to restart and "e" to end')

# Make held index all into False

held = [False,False,False,False,False]

while True:

message = input("P1(secondRoll) Enter the dices you want to roll by sequence (1-5): ")

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

elif message.lower() == "e":

selectRollDice()

break

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

print(cup)

print('"r" to restart and "e" to end')

held = [False,False,False,False,False]

while True:

message = input("P1(ThirdRoll) Enter the dices you want to roll by sequence (1-5): ")

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

elif message.lower() == "e":

selectRollDice()

break

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

print(cup)

# Sorts the cup so that it would be easier to indentify in the lower section

list.sort(cup)

print("Sorted: ",cup)

# These TorF is True or False

# Each TorF matches with one type of scoring in the lower section of the score sheet

TorF1 = []

TorF2 = []

TorF3 = []

TorF4 = []

TorF5 = []

TorF6 = []

onesCount = 0

twosCount = 0

threesCount = 0

foursCount = 0

fivesCount = 0

sixesCount = 0

threeKindTot1 = 0

fourKindTot1 = 0

fullHouseTot1 = 0

smallStraightTot1 = 0

largeStraightTot1 = 0

yahtzeeTot1 = 0

while True:

# Running all the functions and gives each TorF a value, which matches a corresponding function

# Returns if something is Ture or Not

is4ofAKind(cup, TorF2)

is3ofAKind(cup, TorF1)

fullHouse(cup, TorF3)

largeStraight(cup, TorF5)

smallStraight(cup, TorF4)

yahtzee(cup, TorF6)

if TorF2 == [True]:

# Again, one copy is for the turn score and gets sets back to zero every round

# And another copy adds the turn score into the total player score that only adds

fourKindTot1 = cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

fourKindTotal1 = fourKindTotal1 + cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

print("You got 4 of a kind")

break

elif TorF1 == [True]:

threeKindTot1 = cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

threeKindTotal1 = threeKindTotal1 + cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

print("You got 3 of a kind")

break

elif TorF3 == [True]:

fullHouseTot1 = 25

fullHouseTotal1 += 25

print("You got a full house")

break

elif TorF5 == [True]:

largeStraightTot1 = 40

largeStraightTotal1 += 40

print("You got a large straight")

break

elif TorF4 == [True]:

smallStraightTot1 = 30

smallStraightTotal1 += 30

print("You got a small straight")

break

elif TorF6 == [True]:

yahtzeeTot1 = 50

yahtzeeTotal1 += 50

print("Yahtzee!")

break

else:

print("You got nothing")

break

for x in cup:

# If encounter a similar value, it adds a count score

# It will later then be multiplied to each corresponding value to calculate the upper sections score

if x == 1:

onesCount += 1

elif x == 2:

twosCount += 1

elif x == 3:

threesCount += 1

elif x == 4:

foursCount += 1

elif x == 5:

fivesCount += 1

elif x == 6:

sixesCount += 1

else:

break

ones1 = onesCount \* 1

twos1 = twosCount \* 2

threes1 = threesCount \* 3

fours1 = foursCount \* 4

fives1 = fivesCount \* 5

sixes1 = sixesCount \* 6

ones11 += onesCount \* 1

twos11 += twosCount \* 2

threes11 += threesCount \* 3

fours11 += foursCount \* 4

fives11 += fivesCount \* 5

sixes11 += sixesCount \* 6

# Again, one adds to turn score, one adds to player total score

upperSheetTot1 = ones1 + twos1 + threes1 + fours1 + fives1 + sixes1

lowerSheetTot1 = threeKindTot1 + fourKindTot1 + yahtzeeTot1 + smallStraightTot1 + largeStraightTot1 + fullHouseTot1

upperSheetTotal1 = ones11 + twos11 + threes11 + fours11 + fives11 + sixes11

lowerSheetTotal1 = threeKindTotal1 + fourKindTotal1 + yahtzeeTotal1 + smallStraightTotal1 + largeStraightTotal1 + fullHouseTotal1

# Functnions for different score sheets

def upperSheet1():

print("P1 UPPER SHEET")

print("Ones = ",ones11)

print("Twos = ",twos11)

print("Threes = ", threes11)

print("Fours = ",fours11)

print("Fives = ",fives11)

print("Sixes = ",sixes11)

print("Player 1 upper Sheet Total = ",upperSheetTotal1)

def lowerSheet1():

print("P2 LOWER SHEET")

print("Three of a kind = ",threeKindTotal1)

print("Four of a kind = ",fourKindTotal1)

print("Small Straight = ",smallStraightTotal1)

print("Large Straight = ",largeStraightTotal1)

print("Full House = ",fullHouseTotal1)

print("YAHTZEE = ",yahtzeeTotal1)

print("Player 1 lower Sheet Total = ",lowerSheetTotal1)

def scoreSheet1():

print("P1 SCORE SHEET")

print("-----------")

upperSheet1()

lowerSheet1()

message = input("Enter (p) to print player 1 score sheet")

while True:

if message.lower() == "p":

# This whil loop is basically for users to know that the program is going to print the score sheet

scoreSheet1()

break

else:

print("Error: Invalid Input")

print("Upper Sheet Score = ",upperSheetTot1)

print("Lower Sheet Score = ",lowerSheetTot1)

print("Chance(sum of dices) = ",chanceScore())

answer = input("Do you want your score by the (u)pper section or (l)ower sectioin or (c)hance score?")

# In the game of yahtzee, players can only chose one part of the score sheet to score

while True:

if answer.lower() == "u":

print("Player 1 have selected the upper sheet as score")

print("Score: ", upperSheetTot1)

player1Score += upperSheetTot1

break

elif answer.lower() == "l":

print("Player 1 have selected the lower sheet as score")

print("Score: ",lowerSheetTot1)

player1Score += lowerSheetTot1

break

elif answer.lower() == "c":

print("Player 1 have selected the chance score as score")

print("Score: ", chanceScore())

player1Score += chanceScore()

break

else:

print("Error: Invalid Input")

# Second player Game Loop

print("IT'S PLAYER 2'S TURN")

# Sets everything back into zero and all held index into False

# Everything else is exactly same as above, except variable names differ

cup = [0,0,0,0,0]

held = [False, False, False, False, False]

rollDice()

print(cup)

print('"r" to restart and "e" to end')

while True:

message = input("P2(firstRoll) Enter the dices you want to roll by sequence (1-5): ")

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

elif message.lower() == "e":

selectRollDice()

break

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

print(cup)

print('"r" to restart and "e" to end')

held = [False,False,False,False,False]

while True:

message = input("P2(secondRoll) Enter the dices you want to roll by sequence (1-5): ")

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

elif message.lower() == "e":

selectRollDice()

break

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

print(cup)

print('"r" to restart and "e" to end')

held = [False,False,False,False,False]

while True:

message = input("P2(thirdRoll) Enter the dices you want to roll by sequence (1-5): ")

if message.lower() == "r":

held = [False,False,False,False,False]

print("You have restarted")

elif message.lower() == "e":

selectRollDice()

break

elif (message=="1" or message=="2" or message=="3" or message=="4" or message=="5"):

message = int(message)

while True:

if (message > 0 and message < 6):

holdDie(message)

print("Next...")

break

else:

print("Error: Needs to be numbers 1-5")

break

else:

print("Error: Invalid Input")

print(cup)

list.sort(cup)

print("Sorted: ",cup)

TorF1 = []

TorF2 = []

TorF3 = []

TorF4 = []

TorF5 = []

TorF6 = []

onesCount = 0

twosCount = 0

threesCount = 0

foursCount = 0

fivesCount = 0

sixesCount = 0

threeKindTot2 = 0

fourKindTot2 = 0

fullHouseTot2 = 0

smallStraightTot2 = 0

largeStraightTot2 = 0

yahtzeeTot2 = 0

while True:

is4ofAKind(cup, TorF2)

is3ofAKind(cup, TorF1)

fullHouse(cup, TorF3)

largeStraight(cup, TorF5)

smallStraight(cup, TorF4)

yahtzee(cup, TorF6)

if TorF2 == [True]:

fourKindTot2 = cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

fourKindTotal2 += cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

print("You got 4 of a kind")

break

elif TorF1 == [True]:

threeKindTot2 = cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

threeKindTotal2 += cup[0] + cup[1] + cup[2] + cup[3] + cup[4]

print("You got 3 of a kind")

break

elif TorF3 == [True]:

fullHouseTot2 = 25

fullHouseTotal2 += 25

print("You got a full house")

break

elif TorF5 == [True]:

largeStraightTot2 = 40

largeStraightTotal2 += 40

print("You got a large straight")

break

elif TorF4 == [True]:

smallStraightTot2 = 30

smallStraightTotal2 += 30

print("You got a small straight")

break

elif TorF6 == [True]:

yahtzeeTot2 = 50

yahtzeeTotal2 += 50

print("Yahtzee!")

break

else:

print("You got nothing")

break

for x in cup:

if x == 1:

onesCount += 1

elif x == 2:

twosCount += 1

elif x == 3:

threesCount += 1

elif x == 4:

foursCount += 1

elif x == 5:

fivesCount += 1

elif x == 6:

sixesCount += 1

else:

break

ones2 = onesCount \* 1

twos2 = twosCount \* 2

threes2 = threesCount \* 3

fours2 = foursCount \* 4

fives2 = fivesCount \* 5

sixes2 = sixesCount \* 6

ones22 += onesCount \* 1

twos22 += twosCount \* 2

threes22 += threesCount \* 3

fours22 += foursCount \* 4

fives22 += fivesCount \* 5

sixes22 += sixesCount \* 6

upperSheetTot2 = ones2 + twos2 + threes2 + fours2 + fives2 + sixes2

lowerSheetTot2 = threeKindTot2 + fourKindTot2 + yahtzeeTot2 + smallStraightTot2 + largeStraightTot2 + fullHouseTot2

upperSheetTotal2 = ones22 + twos22 + threes22 + fours22 + fives22 + sixes22

lowerSheetTotal2 = threeKindTotal2 + fourKindTotal2 + yahtzeeTotal2 + smallStraightTotal2 + largeStraightTotal2 + fullHouseTotal2

def upperSheet2():

print("P2 UPPER SHEET")

print("Ones = ",ones22)

print("Twos = ",twos22)

print("Threes = ", threes22)

print("Fours = ",fours22)

print("Fives = ",fives22)

print("Sixes = ",sixes22)

print("Upper Sheet Total = ",upperSheetTotal2)

def lowerSheet2():

print("P2 LOWER SHEET")

print("Three of a kind = ",threeKindTotal2)

print("Four of a kind = ",fourKindTotal2)

print("Small Straight = ",smallStraightTotal2)

print("Large Straight = ",largeStraightTotal2)

print("Full House = ",fullHouseTotal2)

print("YAHTZEE = ",yahtzeeTotal2)

def scoreSheet2():

print("P2 SCORE SHEET")

print("-----------")

upperSheet2()

lowerSheet2()

print("Chance(sum of dices) = ",chanceScore())

message = input("Enter (p) to print player 2 score sheet")

while True:

if message.lower() == "p":

scoreSheet2()

break

else:

print("Error: Invalid Input")

print("Upper Sheet Total: ", upperSheetTot2)

print("Lower Sheet Total: ", lowerSheetTot2)

print("Chance(sum of dices) = ",chanceScore())

answer = input("Do you want your score by the (u)pper section or (l)ower section or (c)hance score?")

while True:

if answer.lower() == "u":

print("Player 2 have selected the upper sheet as score")

print("Score: ", upperSheetTotal2)

player2Score += upperSheetTotal2

break

elif answer.lower() == "l":

print("Player 2 have selected the lower sheet as score")

print("Score: ",lowerSheetTotal2)

player2Score += lowerSheetTotal2

break

elif answer.lower() == "c":

print("Player 2 have selected the chanc score as score")

print("Score: ", chanceScore())

player2Score += chanceScore()

break

else:

print("Error: Invalid Input")

print ("Player 1 scores: ",player1Score)

print ("Player 2 scores: ",player2Score)

# At last, the program will check if a player completes the score sheet

player1Complete = []

player2Complete = []

# Below if blocks gives a value to the complete variable above if scores sheet is complete

if (ones11 > 0) and twos11 > 0 and threes11 > 0 and fours11 > 0 and fives11 > 0 and sixes11 > 0 and threeKindTotal1 > 0 and fourKindTotal1 > 0 and smallStraightTotal1 > 0 and largeStraightTotal1 > 0 and fullHouseTotal1 > 0 and yahtzeeTotal1 > 0:

player1Complete = True

if ones22 > 0 and twos22 > 0 and threes22 > 0 and fours22 > 0 and fives22 > 0 and sixes22 > 0 and threeKindTotal2 > 0 and fourKindTotal2 > 0 and smallStraightTotal2 > 0 and largeStraightTotal2 > 0 and fullHouseTotal2 > 0 and yahtzeeTotal2 > 0:

player2Complete = True

if player1Complete == True and player2Complete != True:

print("Player 1 is the winner!")

break

if player1Complete != True and player2Complete == True:

print("Player 2 is the winner!")

break

# If bloth player completed the score sheets, system will check whose total score is higher

if player1Complete == True and player2Complete == True:

if player1Score > player2Score:

print("Player 1 is the winner!")

break

elif player2Score > player1Score:

print("Player 2 is the winner!")

break

else:

print("This round is a tie")

break

if player1Complete != [True] and player2Complete != [True]:

print("Both player score sheets not complete, play again")

print("GAME OVER")

# Asks if the player wants to play again, else, program ends

while True:

response = input("Do you want to play again? (y)es or (n)o?")

if response.lower() == "y":

# Break the loop for redo at the bottom

break

elif response.lower() == "n":

print("..END OF PROGRAm..")

exit()

else:

print("Error: Invalid Input")

print("-----------------------------------")

redo = True

**Journal:**

I started off this project by splitting it into three different parts. The first part is the input part where the players “roll” the dice. The second part is where the program shows the players what they score like fullHouse, smallStraight, Yahtzee…etc or nothing. The third part is the score sheet printing and determining winner part. Through this project, I learned to split my work into pieces and put them together like a puzzle. I also learned another easier way for beginner programmers like me to utilize. I basically use two different files when coding the project, one for testing and another one is the actual program. If a function is functionable after testing, I will copy and paste it into the program in another py file. Problems pop out when I put the puzzle pieces together. Sometimes the program just stops in the middle, because I forgot to put a break in a for loop. I thought this project was very challenging, but after sorting everything out and coding it piece by piece; it was understandable. It just takes a lot of time to debug and find the source of the problem. I also encounter problems like not setting the variables back to zero after every round. Believe it or not, it takes a long time to find out this kind of problem sometimes. This becomes very frustrating when I created transcripts for variables. I need to know which variables need to be set back to zero and which variables does not need to set back to zero. Again, I would make a small program in another file and test it a bunch of times. Overall， through this project I have further learned about using functions to return True or False for something and learned the power of another copy of a similar variable. I think that this is pretty challenging, because I often make a lot of mistakes and it is very when the program does not work after hours of coding.