VOWEL\_COST = 250

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

VOWELS = 'AEIOU'

# Write the WOFPlayer class definition (part A) here

class WOFPlayer:

def \_\_init\_\_(self, name):

self.name = name

self.prizeMoney = 0

self.prizes=[]

def addMoney(self, amt):

self.prizeMoney=self.prizeMoney+amt

def goBankrupt(self):

self.prizeMoney=0

def addPrize(self, prize):

self.prizes=self.prizes+[prize]

def \_\_str\_\_(self):

return "{} (${})".format(self.name, self.prizeMoney)

# Write the WOFHumanPlayer class definition (part B) here

class WOFHumanPlayer(WOFPlayer):

def getMove(self,category, obscuredPhrase, guessed):

user\_input = input('''{} has ${}

Category: {}

Phrase: {}

Guessed: {}

Guess a letter, phrase, or type 'exit' or 'pass':

'''.format(self.name,self.prizeMoney,category,obscuredPhrase,guessed))

print (user\_input)

# Write the WOFComputerPlayer class definition (part C) here

class WOFComputerPlayer(WOFPlayer):

SORTED\_FREQUENCIES='ZQXJKVBPYGFWMUCLDRHSNIOATE'

def \_\_init\_\_(self, name, difficulty):

WOFPlayer.\_\_init\_\_(self, name) # call the parent class's constructor

# basically, call the SUPER -- the parent version

# of the constructor, with all the parameters that it needs.

self.difficulty = difficulty

def smartCoinFlip(self):

x=random.randint(1,10)

if x > self.difficulty:

return True

else:

return False

def getPossibleLetters(guessed):

a=[]

for letter in self.guessed:

a= a + [letter.upper()] ## converting guessed to upper case

possletters=[]

if self.prizeMoney < VOWEL\_COST:

for letter in LETTERS:

if (letter not in a) and (letter not in VOWELS):

possletters=possletters+[letter]

else:

for letter in LETTERS:

if (letter not in a):

possletters=possletters+[letter]

return possletters

def getMove(category, obscuredPhrase, guessed):

x=self.getPossibleLetters(guessed)

max\_index=0

if x == []:

return 'pass'

elif self.smartCoinFlip() == True:

for letter in x:

temp\_index=SORTED\_FREQUENCIES.index(letter)

if max\_index < temp\_index:

max\_index=temp\_index

return SORTED\_FREQUENCIES[max\_index]

else:

if self.smartCoinFlip() == False:

return random.choice(x)

import sys

sys.setExecutionLimit(600000) # let this take up to 10 minutes

import json

import random

import time

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

VOWELS = 'AEIOU'

VOWEL\_COST = 250

# Repeatedly asks the user for a number between min & max (inclusive)

def getNumberBetween(prompt, min, max):

userinp = input(prompt) # ask the first time

while True:

try:

n = int(userinp) # try casting to an integer

if n < min:

errmessage = 'Must be at least {}'.format(min)

elif n > max:

errmessage = 'Must be at most {}'.format(max)

else:

return n

except ValueError: # The user didn't enter a number

errmessage = '{} is not a number.'.format(userinp)

# If we haven't gotten a number yet, add the error message

# and ask again

userinp = input('{}\n{}'.format(errmessage, prompt))

# Spins the wheel of fortune wheel to give a random prize

# Examples:

# { "type": "cash", "text": "$950", "value": 950, "prize": "A trip to Ann Arbor!" },

# { "type": "bankrupt", "text": "Bankrupt", "prize": false },

# { "type": "loseturn", "text": "Lose a turn", "prize": false }

def spinWheel():

with open("wheel.json", 'r') as f:

wheel = json.loads(f.read())

return random.choice(wheel)

# Returns a category & phrase (as a tuple) to guess

# Example:

# ("Artist & Song", "Whitney Houston's I Will Always Love You")

def getRandomCategoryAndPhrase():

with open("phrases.json", 'r') as f:

phrases = json.loads(f.read())

category = random.choice(list(phrases.keys()))

phrase = random.choice(phrases[category])

return (category, phrase.upper())

# Given a phrase and a list of guessed letters, returns an obscured version

# Example:

# guessed: ['L', 'B', 'E', 'R', 'N', 'P', 'K', 'X', 'Z']

# phrase: "GLACIER NATIONAL PARK"

# returns> "\_L\_\_\_ER N\_\_\_\_N\_L P\_RK"

def obscurePhrase(phrase, guessed):

rv = ''

for s in phrase:

if (s in LETTERS) and (s not in guessed):

rv = rv+'\_'

else:

rv = rv+s

return rv

# Returns a string representing the current state of the game

def showBoard(category, obscuredPhrase, guessed):

return """

Category: {}

Phrase: {}

Guessed: {}""".format(category, obscuredPhrase, ', '.join(sorted(guessed)))

# GAME LOGIC CODE

print('='\*15)

print('WHEEL OF PYTHON')

print('='\*15)

print('')

num\_human = getNumberBetween('How many human players?', 0, 10)

# Create the human player instances

human\_players = [WOFHumanPlayer(input('Enter the name for human player #{}'.format(i+1))) for i in range(num\_human)]

num\_computer = getNumberBetween('How many computer players?', 0, 10)

# If there are computer players, ask how difficult they should be

if num\_computer >= 1:

difficulty = getNumberBetween('What difficulty for the computers? (1-10)', 1, 10)

# Create the computer player instances

computer\_players = [WOFComputerPlayer('Computer {}'.format(i+1), difficulty) for i in range(num\_computer)]

players = human\_players + computer\_players

# No players, no game :(

if len(players) == 0:

print('We need players to play!')

raise Exception('Not enough players')

# category and phrase are strings.

category, phrase = getRandomCategoryAndPhrase()

# guessed is a list of the letters that have been guessed

guessed = []

# playerIndex keeps track of the index (0 to len(players)-1) of the player whose turn it is

playerIndex = 0

# will be set to the player instance when/if someone wins

winner = False

def requestPlayerMove(player, category, guessed):

while True: # we're going to keep asking the player for a move until they give a valid one

time.sleep(0.1) # added so that any feedback is printed out before the next prompt

move = player.getMove(category, obscurePhrase(phrase, guessed), guessed)

move = move.upper() # convert whatever the player entered to UPPERCASE

if move == 'EXIT' or move == 'PASS':

return move

elif len(move) == 1: # they guessed a character

if move not in LETTERS: # the user entered an invalid letter (such as @, #, or $)

print('Guesses should be letters. Try again.')

continue

elif move in guessed: # this letter has already been guessed

print('{} has already been guessed. Try again.'.format(move))

continue

elif move in VOWELS and player.prizeMoney < VOWEL\_COST: # if it's a vowel, we need to be sure the player has enough

print('Need ${} to guess a vowel. Try again.'.format(VOWEL\_COST))

continue

else:

return move

else: # they guessed the phrase

return move

while True:

player = players[playerIndex]

wheelPrize = spinWheel()

print('')

print('-'\*15)

print(showBoard(category, obscurePhrase(phrase, guessed), guessed))

print('')

print('{} spins...'.format(player.name))

time.sleep(2) # pause for dramatic effect!

print('{}!'.format(wheelPrize['text']))

time.sleep(1) # pause again for more dramatic effect!

if wheelPrize['type'] == 'bankrupt':

player.goBankrupt()

elif wheelPrize['type'] == 'loseturn':

pass # do nothing; just move on to the next player

elif wheelPrize['type'] == 'cash':

move = requestPlayerMove(player, category, guessed)

if move == 'EXIT': # leave the game

print('Until next time!')

break

elif move == 'PASS': # will just move on to next player

print('{} passes'.format(player.name))

elif len(move) == 1: # they guessed a letter

guessed.append(move)

print('{} guesses "{}"'.format(player.name, move))

if move in VOWELS:

player.prizeMoney -= VOWEL\_COST

count = phrase.count(move) # returns an integer with how many times this letter appears

if count > 0:

if count == 1:

print("There is one {}".format(move))

else:

print("There are {} {}'s".format(count, move))

# Give them the money and the prizes

player.addMoney(count \* wheelPrize['value'])

if wheelPrize['prize']:

player.addPrize(wheelPrize['prize'])

# all of the letters have been guessed

if obscurePhrase(phrase, guessed) == phrase:

winner = player

break

continue # this player gets to go again

elif count == 0:

print("There is no {}".format(move))

else: # they guessed the whole phrase

if move == phrase: # they guessed the full phrase correctly

winner = player

# Give them the money and the prizes

player.addMoney(wheelPrize['value'])

if wheelPrize['prize']:

player.addPrize(wheelPrize['prize'])

break

else:

print('{} was not the phrase'.format(move))

# Move on to the next player (or go back to player[0] if we reached the end)

playerIndex = (playerIndex + 1) % len(players)

if winner:

# In your head, you should hear this as being announced by a game show host

print('{} wins! The phrase was {}'.format(winner.name, phrase))

print('{} won ${}'.format(winner.name, winner.prizeMoney))

if len(winner.prizes) > 0:

print('{} also won:'.format(winner.name))

for prize in winner.prizes:

print(' - {}'.format(prize))

else:

print('Nobody won. The phrase was {}'.format(phrase))