This project will take you through the process of implementing a simplified version of the game *Wheel of Fortune*. Here are the rules of our game:

* **There are num\_human human players and num\_computer computer players.**
  + Every player has *some* amount of money ($0 at the start of the game)
  + Every player has a set of prizes (none at the start of the game)
* **The goal is to guess a phrase within a category. For example:**
  + Category: **Artist & Song**
  + Phrase: **Whitney Houston’s I Will Always Love You**
* **Players see the category and an obscured version of the phrase where every alphanumeric character in the phrase starts out as hidden (using underscores: \_):**
  + Category: **Artist & Song**
  + Phrase: \_\_\_\_\_\_\_ \_\_\_\_\_\_\_'\_ \_ \_\_\_\_ \_\_\_\_\_\_ \_\_\_\_ \_\_\_
* Note that case (capitalization) does not matter
* **During their turn, every player spins the wheel to determine a prize amount and:**
  + **If the wheel lands on a cash square, players may do one of three actions:**
    - **Guess any letter that hasn’t been guessed by typing a letter (a-z)**
      * Vowels (a, e, i, o, u) cost $250 to guess and can’t be guessed if the player doesn’t have enough money. All other letters are “free” to guess
      * The player can guess any letter that hasn’t been guessed and gets that cash amount for *every time* that letter appears in the phrase
      * If there is a prize, the user also gets that prize (in addition to any prizes they already had)
      * If the letter does appear in the phrase, the user keeps their turn. Otherwise, it’s the next player’s turn
      * **Example: The user lands on $500 and guesses ‘W’**
        + There are three W’s in the phrase, so the player wins $1500
    - **Guess the complete phrase by typing a phrase (anything over one character that isn’t *‘pass’*)**
      * If they are correct, they win the game
      * If they are incorrect, it is the next player’s turn
    - **Pass** their turn by entering 'pass'
  + If the wheel lands on **“lose a turn”**, the player loses their turn and the game moves on to the next player
  + If the wheel lands on **“bankrupt”**, the player loses their turn *and* loses their money but they keep all of the prizes they have won so far.
* The game continues until the entire phrase is revealed (or one player guesses the complete phrase)

—

First, let’s learn about a few functions and methods that we’ll use along the way to do this project. There are no questions to answer in the next four active code windows. They are just here to introduce you to some functions and methods that you may not be aware of. The active code window that starts with “Part A” is where you are first asked to complete code.

—

The time.sleep(s) function (from the time module) delays execution of the next line of code for s seconds. You’ll find that we can build a little suspense during gameplay with some well-placed delays. The game can also be easier for users to understand if not everything happens instantly.

import time

for x in range(2, 6):

print('Sleep {} seconds..'.format(x))

time.sleep(x) # "Sleep" for x seconds

print('Done!')

The random module includes several useful methods for generating and using random numbers, including:

* random.randint(min, max) generates a random number between min and max (inclusive)
* random.choice(L) selects a random item from the list L

import random

rand\_number = random.randint(1, 10)

print('Random number between 1 and 10: {}'.format(rand\_number))

letters = [letter for letter in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ']

rand\_letter = random.choice(letters)

print('Random letter: {}'.format(rand\_letter))

There are also several string methods that we haven’t gone over in detail but will use for this project:

* .upper() converts a string to uppercase (the opposite is .lower())
* .count(s) counts how many times the string s occurs inside of a larger string

myString = 'Hello, World! 123'

print(myString.upper()) # HELLO, WORLD! 123

print(myString.lower()) # hello, world! 123

print(myString.count('l')) # 3

s = 'python is pythonic'

print(s.count('python')) # 2

We’re going to define a few useful methods for you:

* getNumberBetween(prompt, min, max)) repeatedly asks the user for a number between min and max with the prompt prompt
* spinWheel() simulates spinning the wheel and returns a dictionary with a random prize
* getRandomCategoryAndPhrase() returns a tuple with a random category and phrase for players to guess
* obscurePhrase(phrase, guessed) returns a tuple with a random category and phrase for players to guess

Take some time to read their implementations below.

import json

import random

import time

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

# 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)))

category, phrase = getRandomCategoryAndPhrase()

guessed = []

for x in range(random.randint(10, 20)):

randomLetter = random.choice(LETTERS)

if randomLetter not in guessed:

guessed.append(randomLetter)

print("getRandomCategoryAndPhrase()\n -> ('{}', '{}')".format(category, phrase))

print("\n{}\n".format("-"\*5))

print("obscurePhrase('{}', [{}])\n -> {}".format(phrase, ', '.join(["'{}'".format(c) for c in guessed]), obscurePhrase(phrase, guessed)))

print("\n{}\n".format("-"\*5))

obscured\_phrase = obscurePhrase(phrase, guessed)

print("showBoard('{}', '{}', [{}])\n -> {}".format(phrase, obscured\_phrase, ','.join(["'{}'".format(c) for c in guessed]), showBoard(phrase, obscured\_phrase, guessed)))

print("\n{}\n".format("-"\*5))

num\_times\_to\_spin = random.randint(2, 5)

print('Spinning the wheel {} times (normally this would just be done once per turn)'.format(num\_times\_to\_spin))

for x in range(num\_times\_to\_spin):

print("\n{}\n".format("-"\*2))

print("spinWheel()")

print(spinWheel())

print("\n{}\n".format("-"\*5))

print("In 2 seconds, will run getNumberBetween('Testing getNumberBetween(). Enter a number between 1 and 10', 1, 10)")

time.sleep(2)

print(getNumberBetween('Testing getNumberBetween(). Enter a number between 1 and 10', 1, 10))

**Part A: WOFPlayer**

We’re going to start by defining a class to represent a Wheel of Fortune player, called WOFPlayer. Every instance of WOFPlayer has three instance variables:

* .name: The name of the player (should be passed into the constructor)
* .prizeMoney: The amount of prize money for this player (an integer, initialized to 0)
* .prizes: The prizes this player has won so far (a list, initialized to [])

Of these instance variables, only name should be passed into the constructor.

It should also have the following methods (note: we will exclude self in our descriptions):

* .addMoney(amt): Add amt to self.prizeMoney
* .goBankrupt(): Set self.prizeMoney to 0
* .addPrize(prize): Append prize to self.prizes
* **.\_\_str\_\_(): Returns the player’s name and prize money in the following format:**
  + Steve ($1800) (for a player with instance variables .name == 'Steve' and prizeMoney == 1800)

**Part B: WOFHumanPlayer**

Next, we’re going to define a class named WOFHumanPlayer, which should inherit from WOFPlayer (part A). This class is going to represent a human player. In addition to having all of the instance variables and methods that WOFPlayer has, WOFHumanPlayer should have an additional method:

* .getMove(category, obscuredPhrase, guessed): Should ask the user to enter a move (using input()) and **return whatever string they entered**.

.getMove()’s prompt should be:

{name} has ${prizeMoney}

Category: {category}

Phrase: {obscured\_phrase}

Guessed: {guessed}

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

For example:

Steve has $200

Category: Places

Phrase: \_L\_\_\_ER N\_\_\_\_N\_L P\_RK

Guessed: B, E, K, L, N, P, R, X, Z

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

The user can then enter:

* 'exit' to exit the game
* 'pass' to skip their turn
* a single character to guess that letter
* a complete phrase (a multi-character phrase other than 'exit' or 'pass') to guess that phrase

Note that .getMove() **does not** need to enforce anything about the user’s input; that will be done via the game logic that we define in the next ActiveCode window.

**Part C: WOFComputerPlayer**

Finally, we’re going to define a class named WOFComputerPlayer, which should inherit from WOFPlayer (part A). This class is going to represent a computer player.

Every computer player will have a difficulty instance variable. Players with a higher difficulty generally play “better”. There are many ways to implement this. We’ll do the following:

* If there aren’t any possible letters to choose (for example: if the last character is a vowel but this player doesn’t have enough to guess a vowel), we’ll 'pass'
* **Otherwise, semi-randomly decide whether to make a “good” move or a “bad” move on a given turn (a higher difficulty should make it more likely for the player to make a “good” move)**
  + To make a “bad” move, we’ll randomly decide on a possible letter.
  + To make a “good” move, we’ll choose a letter according to their overall frequency in the English language.

In addition to having all of the instance variables and methods that WOFPlayer has, WOFComputerPlayer should have:

**Class variable**

* .SORTED\_FREQUENCIES: Should be set to 'ZQXJKVBPYGFWMUCLDRHSNIOATE', which is a list of English characters sorted from least frequent ('Z') to most frequent ('E'). We’ll use this when trying to make a “good” move.

**Additional Instance variable**

* .difficulty: The level of difficulty for this computer (should be passed as the second argument into the constructor after .name)

**Methods**

* .smartCoinFlip(): This method will help us decide semi-randomly whether to make a “good” or “bad” move. A higher difficulty should make us more likely to make a “good” move. Implement this by choosing a random number between 1 and 10 using random.randint(1, 10) (see above) and returning True if that random number is greater than self.difficulty. If the random number is less than or equal to self.difficulty, return False.
* **.getPossibleLetters(guessed): This method should return a list of letters that can be guessed.**
  + These should be characters that are in LETTERS ('ABCDEFGHIJKLMNOPQRSTUVWXYZ') but **not** in the guessed parameter.
  + Additionally, if this player doesn’t have enough prize money to guess a vowel (variable VOWEL\_COST set to 250), then vowels (variable VOWELS set to 'AEIOU') should **not** be included
* **.getMove(category, obscuredPhrase, guessed): Should return a valid move.**
  + Use the .getPossibleLetters(guessed) method described above.
  + If there aren’t any letters that can be guessed (this can happen if the only letters left to guess are vowels and the player doesn’t have enough for vowels), return 'pass'
  + **Use the .smartCoinFlip() method to decide whether to make a “good” or a “bad” move**
    - If making a “good” move (.smartCoinFlip() returns True), then return the most frequent (highest index in .SORTED\_FREQUENCIES) possible character
    - If making a “bad” move (.smartCoinFlip() returns False), then return a random character from the set of possible characters (use random.choice())

VOWEL\_COST = 250

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

VOWELS = 'AEIOU'

# Write the WOFPlayer class definition (part A) here

class WOFPlayer():

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

self.name = initName

self.prizeMoney = 0

self.prizes = []

def goBankrupt(self):

self.prizeMoney = 0

def addPrize(self, prize):

self.prizes.append(prize)

def addMoney(self,amt):

self.prizeMoney = self.prizeMoney + amt

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

state = self.name + " ($" + str(self.prizeMoney) + ")"

return state

# Write the WOFHumanPlayer class definition (part B) here

class WOFHumanPlayer(WOFPlayer):

def getMove(category, obscuredPhrase, guessed):

str = input(self.name + "has $"+ str(self.prizeMoney)+ "/n" + ", Category:" + category + "/n" + ", Phrases:" + "/n" + obscuredPhrase + "/n" + ", Guessed:" + guessed + "/n" + "Guess a letter, phrase, or type 'exit' or 'pass':")

print(str)

# Write the WOFComputerPlayer class definition (part C) here

class WOFComputerPlayer(WOFPlayer):

SORTED\_FREQUENCIES = 'ZQXJKVBPYGFWMUCLDRHSNIOATE'

prizemoney = 0

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

self.name = name

self.difficulty = difficulty

self.prizeMoney = 0

self.prizes = []

def smartCoinFlip(self):

if random.randint(1, 10) > self.difficulty:

return True

else:

return False

def getPossibleLetters(self,guessed):

lst = []

if self.prizemoney >= 250:

for l in LETTERS:

if l!=guessed:

return False

lst.append(l)

else:

for l in LETTERS:

if l not in VOWELS:

if l!=guessed:

return False

lst.append(l)

return lst

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

return getPossibleLetters(guessed)

**Putting it together: Wheel of Python**

Below is the game logic for the rest of the “Wheel of Python” game. We have implemented most of the game logic. **Start by carefully reading this code and double checking that it all makes sense**. Then, paste your code from the previous code window in the correct places below.

**Note 1**: we added the following code to ensure that the Python interpreter gives our game time to run:

import sys

sys.setExecutionLimit(600000)

sys.setExecutionLimit(ms) says that we should be able to run our program for ms milliseconds before it gets stopped automatically.

**Note 2**: As you play, you will need to keep scrolling down to follow the game.

# PASTE YOUR WOFPlayer CLASS (from part A) HERE

# PASTE YOUR WOFHumanPlayer CLASS (from part B) HERE

# PASTE YOUR WOFComputerPlayer CLASS (from part C) HERE

VOWEL\_COST = 250

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

VOWELS = 'AEIOU'

# Write the WOFPlayer class definition (part A) here

class WOFPlayer():

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

self.name = initName

self.prizeMoney = 0

self.prizes = []

def goBankrupt(self):

self.prizeMoney = 0

def addPrize(self, prize):

self.prizes.append(prize)

def addMoney(self,amt):

self.prizeMoney = self.prizeMoney + amt

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

state = self.name + " ($" + str(self.prizeMoney) + ")"

return state

# Write the WOFHumanPlayer class definition (part B) here

class WOFHumanPlayer(WOFPlayer):

def getMove(category, obscuredPhrase, guessed):

str = input(self.name + "has $"+ str(self.prizeMoney)+ "/n" + ", Category:" + category + "/n" + ", Phrases:" + "/n" + obscuredPhrase + "/n" + ", Guessed:" + guessed + "/n" + "Guess a letter, phrase, or type 'exit' or 'pass':")

print(str)

# Write the WOFComputerPlayer class definition (part C) here

class WOFComputerPlayer(WOFPlayer):

SORTED\_FREQUENCIES = 'ZQXJKVBPYGFWMUCLDRHSNIOATE'

prizemoney = 0

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

self.name = name

self.difficulty = difficulty

self.prizeMoney = 0

self.prizes = []

def smartCoinFlip(self):

if random.randint(1, 10) > self.difficulty:

return True

else:

return False

def getPossibleLetters(self,guessed):

lst = []

if self.prizemoney >= 250:

for l in LETTERS:

if l!=guessed:

return False

lst.append(l)

else:

for l in LETTERS:

if l not in VOWELS:

if l!=guessed:

return False

lst.append(l)

return lst

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

return getPossibleLetters(guessed)

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))