

course_4_project

Due: 2019-02-04 15:17:00

Description: Final Project for Course 4 - Wheel of Python

Score: 0 of 3 = 0.0%

Questions

Not yet
graded

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.

Save & Run

11/10/2020, 11:40:41 AM - 2 of 2

```
1 import time
2
3 for x in range(2, 6):
4     print('Sleep {} seconds..'.format(x))
5     time.sleep(x) # "Sleep" for x seconds
6 print('Done!')
7
```

Sleep 2 seconds..

ActiveCode (wof_ac_sleep)

Not yet
graded

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`

Save & Run

11/10/2020, 11:40:46 AM - 2 of 2

```
1 import random
2
3 rand_number = random.randint(1, 10)
4 print('Random number between 1 and 10: {}'.format(rand_number))
5
6 letters = [letter for letter in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ']
7 rand_letter = random.choice(letters)
8 print('Random letter: {}'.format(rand_letter))
9
```

```
Random number between 1 and 10: 8
Random letter: Z
Sleep 3 seconds..
```

ActiveCode (wof_ac_rand)

**Not yet
graded**

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

Save & Run

11/10/2020, 11:49:19 AM - 2 of 2

```
1 myString = 'Hello, World! 123'
2
3 print(myString.upper()) # HELLO, WORLD! 123
4 print(myString.lower()) # hello, world! 123
5 print(myString.count('l')) # 3
6
7 s = 'python is pythonic'
8 print(s.count('python')) # 2
9
```

```
HELLO, WORLD! 123
hello, world! 123
3
2
```

ActiveCode (wof_ac_str)

Not yet
graded

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.

Save & Run

11/10/2020, 11:49:43 AM - 2 of 2

```
1 import json
2 import random
3 import time
4
5 LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
6
7 # Repeatedly asks the user for a number between min & max (inclusive)
8 def getNumberBetween(prompt, min, max):
9     userinp = input(prompt) # ask the first time
10
11     while True:
12         try:
13             n = int(userinp) # try casting to an integer
14             if n < min:
15
```

```

getRandomCategoryAndPhrase()
-> ('Places', 'THE FOUR CORNERS OF THE EARTH')

-----

obscurePhrase('THE FOUR CORNERS OF THE EARTH', ['K', 'L', 'Z', 'N', 'H', 'V', 'A', 'C', 'Y'])
-> _H_ ____ C__N__ _ _H_ _A__H

-----

showBoard('THE FOUR CORNERS OF THE EARTH', '_H_ ____ C__N__ _ _H_ _A__H', ['K','L','Z','N','H','V']
->
Category: THE FOUR CORNERS OF THE EARTH
Phrase:   _H_ ____ C__N__ _ _H_ _A__H
Guessed:  A, C, H, K, L, N, V, Y, Z

-----

Spinning the wheel 5 times (normally this would just be done once per turn)

--

spinWheel()
{'type': 'cash', 'text': '$600', 'value': 600, 'prize': False}

--

spinWheel()
{'type': 'cash', 'text': '$900', 'value': 900, 'prize': False}

--

spinWheel()
{'type': 'cash', 'text': '$700', 'value': 700, 'prize': False}

--

spinWheel()
{'type': 'cash', 'text': '$800', 'value': 800, 'prize': False}

--

spinWheel()
{'type': 'cash', 'text': '$650', 'value': 650, 'prize': 'A brand new car!'}

-----

In 2 seconds, will run getNumberBetween('Testing getNumberBetween(). Enter a number between 1 and 10
Sleep 4 seconds..
5
Sleep 5 seconds..
Done!

```

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()`)

Save & Run

11/10/2020, 11:00:46 PM - 8 of 8

```

45     possible_letters = set(self.SORTED_FREQUENCIES) - set(guessed)
46     if self.prizeMoney < VOWEL_COST:
47         possible_letters = possible_letters - set(VOWELS)
48     return list(possible_letters)
49
50     def getMove(self, category, obscuredPhrase, guessed):
51         available_letters = self.getPossibleLetters(guessed)
52         if len(available_letters) == 0:
53             return 'pass'
54         if self.smartCoinFlip():
55             return sorted(available_letters, key=WOFComputerPlayer.SORTED_FREQUENCIES.index)
56         return random.choice(available_letters)
57
58

```

ActiveCode (wof_ac_wof_player)

Result	Actual Value	Expected Value	Notes
Pass	'Steve'	'Steve'	Testing that the name instance variable was set correctly
Pass	'Paul'	'Paul'	Testing that the name instance variable was set correctly
Pass	0	0	Testing that prizeMoney was initialized to 0
Pass	0	0	Testing that prizeMoney was initialized to 0
Pass	[]	[]	Testing that prizes was initialized to []
Pass	[]	[]	Testing that prizes was initialized to []
Pass	500	500	Testing that .addMoney modifies .prizeMoney
Pass	0	0	Testing that .addMoney does not modify .prizeMoney for other instances
Pass	None	None	Testing that .addMoney returns None
Pass	700	700	Testing that .addMoney adds .prizeMoney (rather than just setting it)
Pass	0	0	Testing that .addMoney does not modify .prizeMoney for other instances
Pass	700	700	Testing that .addMoney does not modify .prizeMoney for the wrong instance
Pass	800	800	Testing that .addMoney works
Pass	0	0	Testing that .goBankrupt works
Pass	None	None	Testing that .goBankrupt returns None

Pass	800	800	Testing that .goBankrupt only affects the correct instance	
Pass	['prize1']	['prize1']	Testing that .addPrize correctly adds prizes to self.prizes	
Pass	[]	[]	Testing that .addPrize only affects the correct instance	
Pass	['pri...ze2']	['pri...ze2']	Testing that .addPrize correctly adds prizes to self.prizes	Expand Differences
Pass	[]	[]	Testing that .addPrize only affects the correct instance	
Pass	None	None	Testing that .addPrize returns None	
Pass	['pri...ze2']	['pri...ze2']	Testing that .addPrize correctly adds prizes to self.prizes	Expand Differences
Pass	['prize3']	['prize3']	Testing that .addPrize correctly adds prizes to self.prizes	
Pass	200	200	Testing .goBankrupt()	
Pass	0	0	Testing .goBankrupt()	
Pass	['pri...ze2']	['pri...ze2']	Testing that .goBankrupt didn't affect prizes	Expand Differences
Pass	['prize3']	['prize3']	Testing that .goBankrupt didn't affect prizes	
Pass	'Steve (\$200)'	'Steve (\$200)'	Testing __str__	
Pass	True	True	Testing that WOFHumanPlayer inherits from WOFPlayer	
Pass	'p'	'p'	Testing that the name was correctly set	
Pass	0	0	Testing that the prizeMoney was correctly set	
Pass	[]	[]	Testing that the prizes was correctly set	
Pass	True	True	Testing that WOFComputerPlayer inherits from WOFPlayer	
Pass	'c'	'c'	Testing that the name was correctly set	
Pass	5	5	Testing that the difficulty was correctly set	
Pass	0	0	Testing that the prizeMoney was correctly set	
Pass	[]	[]	Testing that the prizes was correctly set	
Pass	'ZQXJK...IOATE'	'ZQXJK...IOATE'	Testing that WOFComputerPlayer.SORTED_FREQUENCIES was correctly set	Expand Differences
Pass	'ZQXJK...IOATE'	'ZQXJK...IOATE'	Testing that WOFComputerPlayer.SORTED_FREQUENCIES was set as a class variable	Expand Differences
Pass	9999	9999	Testing that the prizeMoney was correctly set	
Pass	0	0	Testing that the prizeMoney was correctly set	
Pass	False	False	Guessed a vowel when there wasn't enough prizeMoney	
Pass	'pass'	'pass'	When there aren't any possible moves, getMove() should return 'pass'	

You passed: 100.0% of the tests

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.

Save & Run

11/10/2020, 11:02:45 PM - 2 of 2

```
16
17     def __str__(self):
18         return '{} ({{}})'.format(self.name, self.prizeMoney)
19
20 # PASTE YOUR WOFHumanPlayer CLASS (from part B) HERE
21 class WOFHumanPlayer(WOFPlayer):
22     def getMove(self, category, obscuredPhrase, guessed):
23         print('{} has {}'.format(self.name, self.prizeMoney))
24         print('Category: {}'.format(category))
25         print('Phrase: {}'.format(obscuredPhrase))
26         print('Guessed: {}'.format(guessed))
27         return input("Guess a letter, phrase, or type 'exit' or 'pass':")
28
29
30
```

=====

WHEEL OF PYTHON

=====

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed:

Sandipan spins...

Bankrupt!

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed:

Computer 1 spins...

One Million!

Computer 1 guesses "B"

There is no B

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed: B

Sandipan spins...

\$700!

Sandipan has \$0

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed: ['B']

Need \$250 to guess a vowel. Try again.

Sandipan has \$0

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed: ['B']

Need \$250 to guess a vowel. Try again.

Sandipan has \$0

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed: ['B']

Need \$250 to guess a vowel. Try again.

Sandipan has \$0

Category: Phrase

Phrase: _ ' _ _ _ _ _ _ _ _ _

Guessed: ['B']

Sandipan guesses "N"

There are 2 N's

Category: Phrase

Phrase: _ ' _ N _ _ _ _ N _ _ _ _ _
Guessed: B, N

Sandipan spins...

\$900!

Sandipan has \$1400

Category: Phrase

Phrase: _ ' _ N _ _ _ _ N _ _ _ _ _

Guessed: ['B', 'N']

Sandipan guesses "P"

There is one P

Category: Phrase

Phrase: _ ' _ N _ _ _ _ N _ _ _ _ P _ _ _ _

Guessed: B, N, P

Sandipan spins...

\$950!

Sandipan has \$2300

Category: Phrase

Phrase: _ ' _ N _ _ _ _ N _ _ _ _ P _ _ _ _

Guessed: ['B', 'N', 'P']

Sandipan guesses "X"

There is no X

Category: Phrase

Phrase: _ ' _ N _ _ _ _ N _ _ _ _ P _ _ _ _

Guessed: B, N, P, X

Computer 1 spins...

\$600!

Computer 1 guesses "T"

There are 2 T's

Category: Phrase

Phrase: _ T ' _ N _ _ _ _ N _ _ _ _ P _ T _ _ _

Guessed: B, N, P, T, X

Computer 1 spins...

\$800!

Computer 1 guesses "E"

There are 2 E's

Category: Phrase

Phrase: _ T ' _ N _ _ E _ _ N _ _ E _ P _ T _ _ _

Guessed: B, E, N, P, T, X

Computer 1 spins...

\$700!

Computer 1 guesses "W"

There is no W

Category: Phrase

Phrase: _T'_ N_ _E _N_ _E_ _P__T ____

Guessed: B, E, N, P, T, W, X

Sandipan spins...

\$600!

Sandipan has \$2300

Category: Phrase

Phrase: _T'_ N_ _E _N_ _E_ _P__T ____

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W']

T has already been guessed. Try again.

Sandipan has \$2300

Category: Phrase

Phrase: _T'_ N_ _E _N_ _E_ _P__T ____

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W']

Sandipan guesses "R"

There are 2 R's

Category: Phrase

Phrase: _T'_ N_ _E _R_N_ _ER _P__T ____

Guessed: B, E, N, P, R, T, W, X

Sandipan spins...

\$600!

Sandipan has \$3500

Category: Phrase

Phrase: _T'_ N_ _E _R_N_ _ER _P__T ____

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R']

Sandipan guesses "S"

There are 3 S's

Category: Phrase

Phrase: _T'S N_ _SE _R_N_ _ER SP__T ____

Guessed: B, E, N, P, R, S, T, W, X

Sandipan spins...

\$700!

Sandipan has \$5300

Category: Phrase

Phrase: _T'S N_ _SE _R_N_ _ER SP__T ____

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S']

Sandipan guesses "A"

There is no A

Category: Phrase

Phrase: _T'S N_ _SE _R_N_ _ER SP__T ____

Guessed: A, B, E, N, P, R, S, T, W, X

Computer 1 spins...

\$800!

Computer 1 guesses "O"

There are 2 O's

Category: Phrase

Phrase: _T'S NO _SE _R__N_ O_ER SP__T ____

Guessed: A, B, E, N, O, P, R, S, T, W, X

Computer 1 spins...

\$600!

Computer 1 guesses "U"

There is one U

Category: Phrase

Phrase: _T'S NO USE _R__N_ O_ER SP__T ____

Guessed: A, B, E, N, O, P, R, S, T, U, W, X

Computer 1 spins...

900!

Computer 1 guesses "I"

There are 4 I's

Category: Phrase

Phrase: IT'S NO USE _R_IN_ O_ER SPI_T _I__

Guessed: A, B, E, I, N, O, P, R, S, T, U, W, X

Computer 1 spins...

\$600!

Computer 1 guesses "H"

There is no H

Category: Phrase

Phrase: IT'S NO USE _R_IN_ O_ER SPI_T _I__

Guessed: A, B, E, H, I, N, O, P, R, S, T, U, W, X

Sandipan spins...

One Million!

Sandipan has \$5050

Category: Phrase

Phrase: IT'S NO USE _R_IN_ O_ER SPI_T _I__

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H']

Sandipan guesses "L"

There are 2 L's

Category: Phrase

Phrase: IT'S NO USE _R_IN_ O_ER SPILT _IL__

Guessed: A, B, E, H, I, L, N, O, P, R, S, T, U, W, X

Sandipan spins...

\$600!

Sandipan has \$2005050

Category: Phrase

Phrase: IT'S NO USE _R_IN_ O_ER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L']

Sandipan guesses "C"

There is one C

Category: Phrase

Phrase: IT'S NO USE CR_IN_ O_ER SPILT _IL_

Guessed: A, B, C, E, H, I, L, N, O, P, R, S, T, U, W, X

Sandipan spins...

\$900!

Sandipan has \$2005650

Category: Phrase

Phrase: IT'S NO USE CR_IN_ O_ER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C']

Sandipan guesses "Y"

There is one Y

Category: Phrase

Phrase: IT'S NO USE CRYIN_ O_ER SPILT _IL_

Guessed: A, B, C, E, H, I, L, N, O, P, R, S, T, U, W, X, Y

Sandipan spins...

\$700!

Sandipan has \$2006550

Category: Phrase

Phrase: IT'S NO USE CRYIN_ O_ER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C', 'Y']

Sandipan guesses "G"

There is one G

Category: Phrase

Phrase: IT'S NO USE CRYING O_ER SPILT _IL_

Guessed: A, B, C, E, G, H, I, L, N, O, P, R, S, T, U, W, X, Y

Sandipan spins...

\$600!

Sandipan has \$2007250

Category: Phrase

Phrase: IT'S NO USE CRYING O_ER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C', 'Y', 'G']

Sandipan guesses "V"

There is one V

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: A, B, C, E, G, H, I, L, N, O, P, R, S, T, U, V, W, X, Y

Sandipan spins...

\$700!

Sandipan has \$2007850

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C', 'Y', 'G'],

Sandipan guesses "F"

There is no F

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: A, B, C, E, F, G, H, I, L, N, O, P, R, S, T, U, V, W, X, Y

Computer 1 spins...

\$600!

Computer 1 guesses "J"

There is no J

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: A, B, C, E, F, G, H, I, J, L, N, O, P, R, S, T, U, V, W, X, Y

Sandipan spins...

Bankrupt!

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: A, B, C, E, F, G, H, I, J, L, N, O, P, R, S, T, U, V, W, X, Y

Computer 1 spins...

One Million!

Computer 1 guesses "D"

There is no D

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: A, B, C, D, E, F, G, H, I, J, L, N, O, P, R, S, T, U, V, W, X, Y

Sandipan spins...

\$700!

Sandipan has \$0

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _IL_

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C', 'Y', 'G'],

Sandipan guesses "K"

There is one K

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _ILK

Guessed: A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, V, W, X, Y

Sandipan spins...

\$600!

Sandipan has \$700

Category: Phrase

Phrase: IT'S NO USE CRYING OVER SPILT _ILK

Guessed: ['B', 'N', 'P', 'X', 'T', 'E', 'W', 'R', 'S', 'A', 'O', 'U', 'I', 'H', 'L', 'C', 'Y', 'G'],

Sandipan guesses "M"

There is one M

Sandipan wins! The phrase was IT'S NO USE CRYING OVER SPILT MILK

Sandipan won \$1300

ActiveCode (wof_ac_final)

Score Me