

COSC 101 Homework 6: Fall 2024

! Tasks 1 and 2 must be completed with the **partner assigned by your instructor** and are due **Tuesday, October 22, 11:59pm EDT**

! Task 3 must be completed **individually** and is due **Friday, October 25, 11:59pm EDT**

Introduction

In this homework, you will practice your **design and programming skills to create a simplified version of the Wheel of Fortune Game**. This assignment is designed to give you practice with the following topics:

- Top Down Design
- Functions
- Boolean Values
- While Loops

! Important Tips

- Labs and homeworks build on each other. You should **draw on the skills you used in prior homeworks and labs**.
- **You are not allowed to use constructs/methods/statements, that we have not yet covered in this course**—which includes string methods, list assignment, list methods.
- Remember, computer science is a science. **Always write code with a prediction in mind**. While you can sparingly code to try and see output, you should focus on thinking through what you want your code to do, what you expect as a result, and compare what your code actually does to your result.
- Before you type, **trace the execution of the starter code** already provided.
- Use extra steps if needed for your thought process, printing results to test your code. Just remember to **remove these extra prints** once you know your code is correct.

Your assignment

Your assignment is to complete following steps:

1. Download the `hw6.zip` file and open it. You will see one python file (`hw6_wof.py`) and two PDF files (`wof_games_basic.pdf` and `wof_games_full.pdf`) in the

- unzipped folder. Do **not** change the name of the Python file.
2. Review the assignment overview and grading criteria below.
 3. Complete tasks 1 and 2 with the **partner assigned by your instructor** and submit these tasks by **Tuesday, October 22, 10am EDT**
 4. Complete task 3 in `hw6_wof.py` **individually** and submit this task by **Friday, October 25, 11:59pm EDT**.

Similar to homework 4, the **tasks in this assignment build on one another**. You need to complete Task 1 before you can do Task 2, and you need to complete Task 2 before you can do Task 3.

Notice that the starter `.py` file has a header with some information for you to fill in. Please do so. Your feedback helps the instructors better understand your experiences doing the homeworks and where we can provide better assistance.

Game overview

Wheel of Fortune is an American game show where contestants compete to guess a mystery phrase and win money.

You will both design and implement a program for a simplified single-player version of Wheel of Fortune. The overarching goal of the game is to solve the puzzle (that is, guess the phrase) and win as many points as possible along the way.

At the beginning of every game:

- The player starts with 0 points and 5 tokens.
- A phrase is shown to the player with all letters hidden. Players can see spaces (if applicable).

Below the phases of the game are sketched.

Game start

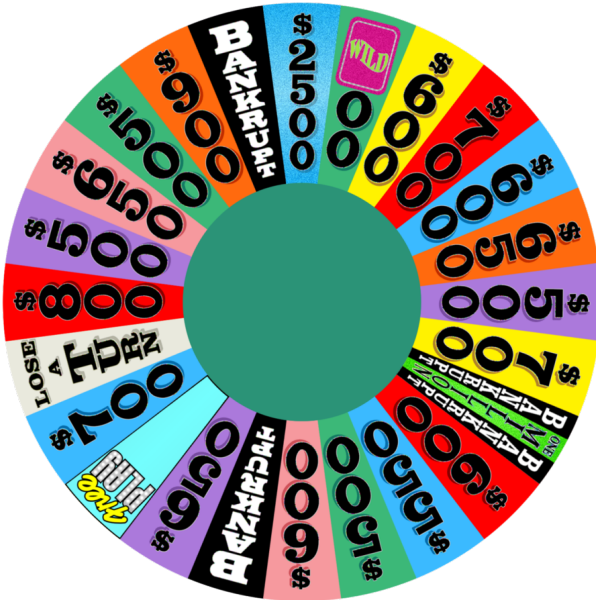
At the start of the game, a welcome message, the state of the game, and the phrase (with letters hidden) is displayed.

Player's turn

For each turn, the player can choose one of three options:

- Spin the wheel
- Guess the puzzle
- Quit

Spin the Wheel



If the player chooses to Spin the wheel, the wheel is spun. As depicted above, the wheel has different states it can land on. We will simplify this wheel to just positive points (500, 1000, ...) or BANKRUPT.

- If the wheel shows 'BANKRUPT', the player loses all of the points they've accumulated so far and they lose one of their tokens.
- If the outcome is a positive point value, the user can enter a letter in the puzzle.
 - If the letter is not in the puzzle, they lose one of their remaining tokens.
 - If the letter is in the puzzle they win points equal to the spin value times the number of times that letter appears in the puzzle.

Guess the puzzle

If the player chooses to Guess the puzzle, the player will be prompted to provide a solution.

- If the guess is correct, the player wins the game.
- If the guess is incorrect, they lose one of their remaining tokens.

Quit

If the player chooses to Quit, the game ends with the player losing.

End of Game

At the end of the game, the player earns points equal to the points accumulated during the game times the number of tokens they have remaining. If they lose, the correct phrase is displayed along with a losing message.

Sample Games

Sample runs of the basic game can be found in `wof_games_basic.pdf` and of the fuller game in `wof_games_full.pdf`.

Grading

When we are assessing your code, higher levels of achievement are demonstrated, in part, by

- Starter code left unmodified
- All lines of output are present
- Lines of output have proper formatting, including spacing and blank lines
- Use spaces and blank lines in code for readability
- Variables are appropriately named and used
- Functions are appropriately named and definitions include type annotations and docstrings
- Functions are used to capture patterns and minimize repeated code
- Functions are used to break down problems and employ abstraction
- Loops are appropriately used to abstract repeated patterns
- Code is clean (for example, remove code you commented out)
- Code is elegant (for example, limit excessive code, ask yourself, can I do this a simpler way)

Task 1: Problem Solving (with a partner)

For this task **you will not code**. With more complicated programs, we need to breakdown our thought process. First we will orient ourselves to game and its flow by focusing on the basic game. **Start by reviewing the sample basic game runs in `wof_games_basic.pdf`.**

For the following questions, you will write your answers by annotating the pdf copy of the basic game runs called `wof_games_basic.pdf`. A printed copy can be found outside of your instructor's office.

1. Label the code output in each of the 3 basic games with the phases of the game the output corresponds to. That is, is it the start of the game, end of the game, or player turn.
2. For each player turn label the round number and summarize the important action (what was done and what was updated or outputted).
3. What are the conditions that end the game? How does the player win? How does the player lose?
4. In a player turn, how are the points calculated (provide an example calculation)?
5. At the end of the game, how are the total points calculated (provide an example calculation)?

Task 2: Program Design (with a partner)

For this task **you will not code**. For this task, we will think about top down program design. You will produce a **Design Doc** that provides the outline of a solution for the basic game. In building this design doc, consider the tips below. Additionally, **ignore the total score for now**.

You can write your answers on paper and scan them or submit a photo of a whiteboard with your answer. Regardless of which method you choose, you will submit a single PDF file with your answer to Task 2.

1. Re-review the sample basic game runs you annotated for Task 1. Review the provided functions and their docstrings in `hw6_wof.py`. Using these existing functions and by designing functions of your own group the game output with the relevant functions. **Do not write functions**. Rather, provide a function definition with parameters and the output you expect (assume that the function can do the computation you want).
2. Diagram your program showing how your functions (and the provided ones) relate to one another (when does your function get called, what function is called after, etc.).

Task 3: Implementation (individually)

! Task 3 should be completed individually.

Now that we have oriented ourselves to the game and proposed a possible high-level description of a solution, it's time to implement our program. **Note: Your actual solution may differ from your proposal. That is to be expected. Do not modify your answer to Task 2 based on your final solution to Task 3. Rather, you can copy and change your Task 2 as needed in helping you think through Task 3.** We will proceed in two steps. First implementing the basic game flow and then filling in a fuller solution.

Sub-task A: Basic Game Flow

For sub-task A, you should use the provided functions and code you write to recreate the basic game runs in `wof_games_basic.pdf` (that you annotated for **Task 1** and that you proposed a design for in **Task 2**). Begin by completing the `display_board` and `end_game` functions.

Sub-task B: The Program

Now that you have the basic scaffold of your game, modify your code (**Reminder: Do not change the provided functions**) to handle the full game. Review the sample runs in `wof_games_full.pdf`. Note the changes. What new information did you learn about the player choice part? What happens when the wheel lands on BANKRUPT?

Submission Instructions

Submit a single PDF with your answers for Task 1, a single PDF with your answers to Task 2, and your Python file `hw6_wof.py` to the platform indicated in your class section. Recall, that tasks 1 and 2 are due **Tuesday Oct. 22 at 10AM** and are completed with a partner. Task 3 is due **Friday Oct. 25 at 11:59PM** and **must be completed individually**.