CS 115 Homework 1: Choose Your Own Adventure

In this homework, you will develop a text-based interactive story: a game where a user makes choices in a story. What happens in the story depends on their choices. Here is the beginning of an example with the same interface as what you will make.

```
You wake up in a silent cathedral of steel. After taking a rest you
a) walk through a small opening opening to the left
b) climb through a ceiling hatch
c) look around the walls
Type in your action and press <Enter>: b

 It looks like you're in a building's ventilation system. It's cold.
 There's a large fan behind you. In front of you, you see a faint light.
 In response, you
a) walk forward towards the small light
b) run towards the fan
c) freeze
Type in your action and press <Enter>:
```

Figure 1: Example gameplay using the interface we will use for the assignment

Many parts of this assignment are deliberately open-ended to account for the diverse backgrounds in this class. You are welcome to use your creativity however you like *as long as your code meets the autograder requirements.*

In particular, **all decisions must be selected by the user by typing in a, b, or c as shown in the example above.** Your code does not need to handle the case where the user types in incorrect output.

We hope that by the end of this assignment, all students have

1. Drawn a visualization of their story
2. Implemented code that makes their story
3. Developed a program that they would want to share with others

Task 1: Design

Brainstorm

Think of a story you'd like to share. Start with something simple and then you can iterate on it. In `report .txt`, briefly share the following (1-3 sentences per question):

1. Who is the player in the story? What is their goal?
2. List 3 possible endings the player might reach.
3. What types of decisions would lead to those endings?

For inspiration, you can try some text-based adventure games or watch a Let's Play of Zork I (1977)

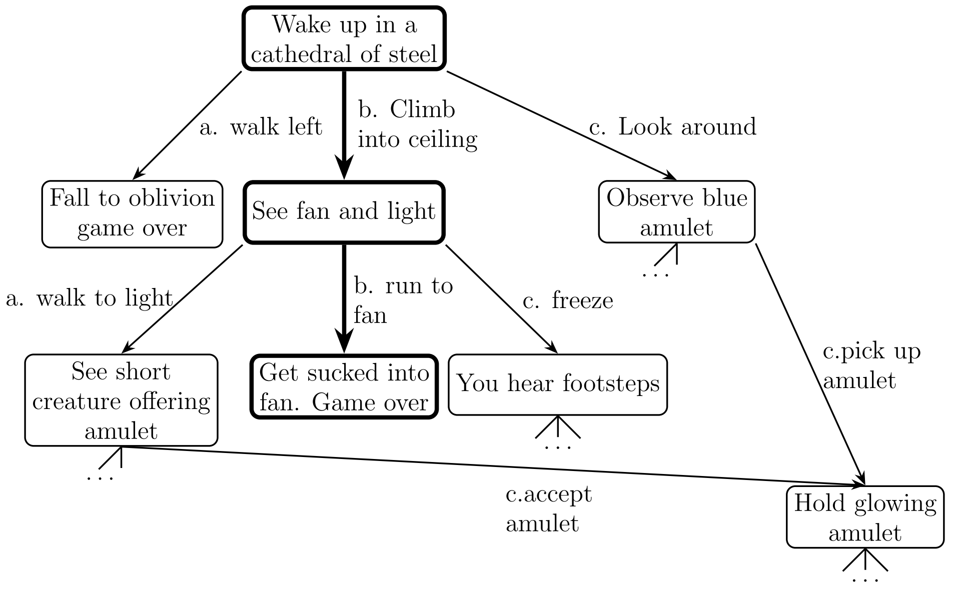

Figure 3: Sample control flow with example path formatted in bold. Not all choices shown.

Abstraction of a text-based adventure

Create a *simple diagram* describing *how a user interacts with the text-based adventure game* ***in this assignment***. This diagram should contain fewer than 3 boxes and 5 arrows (ex. 2 boxes and 3 arrows is okay - 4 boxes and 10 arrows would need to be simplified.

Include the **inputs and outputs** of the game without describing how the game works. Below (Fig. 2)is an example diagram of the interface for a vending machine. The purpose is to show the *flow of data*.

Submit your diagram as `interface_abstraction.jpg`. You may draw it and take a picture or draw it electronically, as long as we can read it.

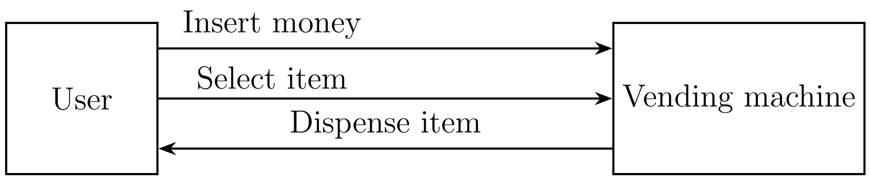

Figure 2: Sample interface diagram for a vending machine

Control flow diagram

A control flow diagram show the possible ways a program progresses through decision points. In this activity, we map how situations are connected in the story to a logical flow that can be translated into executable code. The purpose is to show the *flow of decisions and situations*.

1. Pick a starting scenario for the game
2. Draw arrows between the starting scenario and **three** additional scenarios labelled with the decisions that lead to those additional scenarios.
3. Continue your story by adding scenarios where **three different decisions** can be taken to lead to new scenarios. The new scenario can be an "ending"

4. This control flow diagram should have at least one path where in a single run of the game, a player observes *at least* ***three different scenarios*** as a result of inputting a decision *at least twice*. For example, the path shown in bold in Figure 3 would minimally satisfy this requirement.
5. Save the image as `control_flow_diagram.jpg` and upload with the rest of your files.

Task 2: Implementation

Helper functions

In this section, you will implement a few helper functions to help structure your code and warm up to programming

Getting user input

First, write a function called `get_user_input` that 1. prompts the user with the following string: `"Type in your action and press <Enter>: "` 2. returns the single character that the user inputs. For example, the following should print *c* if the user types in *c* and presses `Enter`.

```
>>> alexs_choice = get_user_input()
Type in your action and press <Enter>: c
>>> print(alexs_choice)
c
```

Printing choices

Each function will include a portion where 3 choices are listed next to *a*), *b*), and *c*) corresponding to the 1st, 2nd, and 3rd inputs. This helper function will print each choice on a line next to the corresponding letter. For example:

```
>>> print_choices("go left", "go right", "pick up book")
a) go left
b) go right
c) pick up book
```

Implementing the control flow

Fill in `my_story` with the story you designed by writing + calling functions (`get_user_input, print_choices` , and any functions you define), printing story content, and encoding the control flow with Python logic (`if, elif, else` etc ...).

Tips:

- If you're not sure how to start, try implementing just the beginning: printing the starting scenario. Then you can add the first choice. Always start small.
- Programming is iterative. **Run the code frequently**, such as by calling `my_story` at the end of your file).
- Post any questions on Piazza or come to office hours if you have *any* questions or otherwise need help with the assignment.

In your writeup, submit

1. a printout of **one path** in your story that you find interesting that requires at least 2 user decisions
2. an annotation on your control flow diagram with this path connecting what happens in the code to your design.

Technical requirements list

Your code *must* follow the following specifications:

1. Each scenario must either provide **three** choices or end the game.
2. The user inputs are restricted to *exactly a*, *b*, or *c* (no more and no less). No other inputs are permitted.
3. `get_user_input` and `print_choices` must be implemented exactly as described
4. The story needs to be able to run by calling `my_story()`
5. At least one path must take the player through **at least three** different scenarios, including the starting scenario (i.e. both `get_user_input` and `print_choices` is called at least twice).
6. The story's text must depend on user input
7. All input and output in the submission must be text.
8. **All code in the submission (other than imports if applicable) is inside functions**. If you run `my_story()` in `hw1.py` to test it, make sure to delete that line before submitting.
9. **Each function must include a docstring that describes what that function does**
10. For advanced students: You may use any functionality from the Python 3.13 standard library as long as the program satisfies this list of requirements.

Task 3: User testing

Run your program on your computer and have a volunteer (who is not you) play the game while you observe. You are welcome to change your code in your submission without changing your answers to these questions. Share the following in the corresponding box in `report.txt`:

1. How did the program behave compared to what you expected? Were there any times where the story did not match what you designed?
2. What aspect of the game are you proudest of?
3. What (if anything) did you want to change after seeing another person play the game?

Submissions

Submit all files to the Gradescope assignment for Homework 1. You can access Gradescope through Canvas. You can check basic assignment requirements by looking at the autograder results.

- `report.txt` including your
  - written answers to Task 1 and Task 3
  - Stevens Pledge
  - name
  - labor log (at least 2 sentences – more info on template `report.txt`)
- `hw1.py`
- `interface_abstraction.jpg`
- `control_flow_diagram.jpg`

An autograder with *some* tests checking very basic requirements will be provided two days after the homework is released. Once released, the autograder will run automatically whenever you upload a new version of your homework to Gradescope. However, you are ultimately responsible for running your own code as well as ensuring your submission complete and correct.

If you are having trouble submitting to Gradescope, please let us (the CAs and professor) know as soon as possible so that we can assist you with a timely submission.