Homework 8 Infinite Story

All parts of HW8 are due Wed Dec 4th at 11:55 pm.

This project is a modification of Chris Piech's neat AI project.

The idea with this project is we have an adventure story stored in a dict. The story has different scenes, and the user can go from one scene to another. The trick is, AI is used to come up with new scenes as needed, so the story actually just goes on forever. So in part, this project gives realistic practice with dictionaries and function, and of course it's neat to get to see AI integrated to provide such an audacious feature.

Downloads infinite-story.zip to get started.

Not Open AI Key

Go to not open ai to get an API key string.

Paste your notopenai key into the line

```
CLIENT = NotOpenAI(api_key="paste-your-key-here")
```

Aside: Why is the API called NotOpenAI? Its just a joke. OpenAI is the company that created GPT-3.5. To keep things free for you we are routing all of your requests through our CS106A paid account. We call it NotOpenAI because we are not OpenAI 🙂 The API is identical to the OpenAI API, and if you want to switch over to your own paid OpenAI key, you can!

Pillow + requests

You also need to install Pillow if not already present, and also the "requests" module:

```
$ python3 -m pip install Pillow requsts
```

Story Structure

First take a look at the "story" structure. The structure is complicated, but is a

realistic example. The "story" is the outer dict, containing elements to build an adventure story. Within the story is a "scenes" key, which points to a dict of all the scenes. Each scene is a dict, known by its key in the scenes dict. In the example below, from the file tiny.json, the scenes are `'start'` and `'scene_aaa'`. Each scene contains a "choices" list, and each choice is a little dict holding a scene_key of a scene the user might go to. In the tiny.json example, from the "start" scene, it's possible to go to either `'scene_aaa'` or `'scene_bbb'`.

```
{
  "plot": "plot words blah blah blah",
  "scenes": {
    "start": {
      "text": "You are at the start.",
      "scene_summary": "Words about start.",
      "choices": [
        {
          "text": "blah blah",
          "scene_key": "scene_aaa"
        },
        {
          "text": "blah blah",
          "scene_key": "scene_bbb"
        }
      ]
    },
    "scene_aaa": {
      "text": "blah blah",
      "scene_summary": "blah",
      "choices": [
        {
          "text": "Go back to the start",
          "scene_key": "start"
        },
        {
          "text": "blah",
          "scene_key": "scene_ccc"
        }
      ]
    }
  }
}
```

Milestone 1 - Unknown Scene

It will turn out to be interesting when a scene_key is referenced, but there is no such key in the story. We'll call this an "unknown" scene_key.

Complete code for the is_unknown_scene() function which checks if a key is unknown or not. Doctests are provided. The story dict put together for the Doctest is incredibly small, with just a couple scenes, and leaving out everything else not

needed for this test.

```
def is_unknown_scene(story, scene_key):
    """
    Return True if this scene_key is not present in the story scenes,
    i.e. it is an unknown scene.
    >>> story = {'plot': 'xyz', 'scenes': {'start': {}, 'go_outside':
{}}}
    >>> is_unknown_scene(story, 'go_outside')
    False
    >>> is_unknown_scene(story, 'selfie_with_celeb')
    True
    """
    pass
```

Mileston 2 - List Unknown Scenes

Complete code for the unknown_scenes() function. Given a story, look at all the scenes, and within them, all the choices. Return a list of all the unknown scene_keys from among the choices. Doctests are provided. One test uses the `'tiny.json'` example shown above, which should return the list `['scene_bbb', 'scene_ccc']`. The other test uses the `'original_small.json'` story, and you can look at that file to see its scene details.

```
def unknown_scenes(story):
    """
    Look at all the scenes, and within them,
    all the choices. Return a list of all the scene_key
    strings which are unknown.
    >>> story = json.load(open(f'data/tiny.json'))
    >>> unknown_scenes(story)
    ['scene_bbb', 'scene_ccc']
    >>> story = json.load(open(f'data/original_small.json'))
    >>> unknown_scenes(story)
    ['next_to_gully', 'descend_into_valley', 'watching_sunset',
'continue_exploring_hilltop', 'return_to_small_brick_building']
    """
    pass
```

Milestone 3 - Create New Scene

This function calls the AI to create a new scene when the player goes to a scene that does not exist yet.

The boilerplate code to call the AI is provided, but your code needs to construct the prompt string.

The prompt string should have the following contents, with text you need to include

shown in brackets. Get data you need from inside the story dict, e.g. the `'start'` scene.

```
Return the next scene of a story for key [scene key]. An example scene
should be formatted in json like this: [json form of start scene dict].
The main plot line of the story is [plot text from story].
```

The provided code sends this prompt to the AI which should compose a new scene dict, which the function returns.

Milestone 4 - Next Scene

This function handles the key logic for every step by the user through the story. Once this function is done, the program should be able to run.

Given a story and a scene_key, return the scene dict for that scene_key. There is a tricky case: the scene_key may be unknown. In that case, the function should create a new scene for this scene_key and insert the new scene dict into the story. Then in all cases, return the scene dict. So essentially, it always returns the scene dict for the given scene_key, it may just be that the dict was created just now. If you create a new scene, remember to insert it into the story dict under its scene_key.

```
def next_scene(story, scene_key):
    """
    Given a story and scene_key. If the scene_key is unknown,
    construct a new scene dict via the AI and insert the new scene
    dict into the story. In all cases, return the scene dict
    from the story for this scene_key.
    """
    pass
```

Milestone 5 - Let's Go!

The other functions are provided. They call your next_scene() and the other functions to make the story work.

Run the program like this:

```
$ python3 infinite_story.py data/original_small.json
```

The stories for running are `original_small.json` `original_big.json`, and `engineer_story.json`. The syntax `'data/original_small.json'` is the way to refer to the file `'original_small.json'` **inside** the folder `'data'`. Use the tab-key to autocomplete parts of the filenames on the command line.

When it prompts the to user to select their next scene, a `'*'` next to the option marks that that will be an AI generated room (using your is_unknown_scene() function of course!).

There are DallE graphics for the scenes in the story files, but then once the AI starts making up new scenes, the graphics window will just be a blue rectangle. The graphics files are simply stored in the `'img'` folder. Take look in there if you are curious. It would be easy for you to add your own images if you want to create a story.

Explore around, see how the AI does. You can mess around with the our stories or write your own.

As a last step, we have some ethics questions

Ethics Reflection

Run the story engineer_story.json a few times. This story only has a single scene, so it should start generating as soon as the user makes a choice.

Take note of the names that it generates for your co-workers. You can explore statistics behind any names generated using this website: https://forebears.io/forenames.

After, please answer these questions in the infinite_ethics.txt file. We do not need very long answers, but something showing that you have put a little thought into it.

Q1: You should notice that these names generated by OpenAI for this story are quite popular in a lot of countries. What type of countries/regions keep showing up for these names? Do you feel like they are distributed evenly across the world population? As a hint, compare the names generated in your story to the list of authors of this handout! Justify your answer. Write at least two sentences.

Q2: How comfortable do you feel letting the AI model decide the names used in the story? Are you willing to trust the model with other storytelling decisions? There is no wrong position, but please justify your answer. Write at least two sentences.

Q3: Given how ChatGPT handled names in this context, what sorts of issues could you imagine coming up if instead of asking ChatGPT to generate a story, you were using ChatGPT to evaluate candidates for a job?

Well and Truly All Done!

Please turn in your infinite_story.py and infinite_ethics.txt files on on Paperless as usual.

History and Background

This neat project was first built by Chris Piech and others, and then Nick Parlante modified it, adding in Doctests and giving the students more of the boilerplate to make a smaller assignment. Here is the acknowledgement from Chris's version:

Assignment designed by Chris Piech, inspired by Eric Roberts. Handout written with Anjali Sreenivas, Yasmine Alonso, Katie Liu. Ethics by Javokhir Arifov and Dan Webber. Test scripts by Iddah Mlauzi and Tina Zheng. Advised by Mehran Sahami and Ngoc Nguyen, and more!