Pokeguesser

For our final project, we were inspired by akinator and wanted to create a game simulated like it. While the initial idea was to have our game guess celebrities, we thought it would be less vague to use pokemon characters because there is a definite number of them. To install and run the project on a mac, be sure to have tkmacosx installed via pip. Once tkmacosx is installed, type python pokeguesser.py into the terminal. For the py file, you also need the ListofQs.txt, the droak.png, the droaktalk.png, music1.mp3, congrats.mp3, pokeball-1(dragged).png through pokeball-19 (dragged).png, and all the folders of pokemon images. Be sure that all of these files are in the same location for the code to run accurately.

The Pokemon class takes in a database of raw pokemon information, creates a dictionary and parses it into an easy-to-use python class. The way we do this is by downloading the database into a csv file that we open in python. Next, we use the numpy package to open it, create a dataframe and use a function to turn a dataframe into a dictionary. The dictionary is not perfect, therefore, we continue to edit the dictionary so that we get the pokemon name as a key and the rest of the columns as values. Furthermore, The python class has two functions, one for debugging purposes that lists all details of a given pokemon, and another function that compares one pokemon to another for the final stages of our game. The Pokemon class stores lots of information, including types, size, whether it is a multitype, generation, if it evolves, and more. In the future, I would try to set initial values for all attributes that a Pokemon object has, and then modify those values in the future so there's no risk of a Pokemon object not having a particular attribute. I'm not sure what I would do about the file paths, as some file paths don't work

The Question class is a class that takes in a dictionary key, a list of questions and on occasion, and a singular pokemon. A question object is constructed to carry a singular question in the form of a String, using the dictionary key and question list within a function of the Question class. The function uses a loop to find what question has the key word in it. That object is then used to call show_ques(). The dictionary key is found by another loop that creates a dictionary item and value of every single attribute. Which then gets sent to a function that finds the key that will split the amount of pokemon by about half.

When our application runs, it loads a tkinter interface that displays to the user four buttons, the question, and an image of Dr. Oak. The user is able to click yes or no for a series of questions, and click indeed or not once the game gives a pokemon suggestion. For this project, we made the decision to use tkinter as the interface because the tkinter package came with widgets that would make our code more simple. We used a total of four buttons: Yes, No, Indeed, Not. The yes or no buttons sent a command to the function pertaining to either yes or no; these functions were called ans_yes, and ans_no. The ans_yes, and ans_no functions update the pokeleft dictionary based on the user's response and remove said question from our list of

questions. Afterward, the update function is called. This function destroys all the widgets being displayed and calls the start function which proceeds with the next questions. Once the user is asked “is blank your pokemon”, they can hit indeed which triggers the ans_ind function that calls the user_reponse function with its perimeters, clears the screen, and uses tkinter label to print “I guessed your pokemon” and displays the image of said pokemon using the image features of Pillow(another package of tkinter). If an incorrect pokemon was given, the user can hit Not which triggers ans_not function that tells the user, we couldn’t guess their Pokemon.

While we thought using tkinter would provide us with more simplicity, because this was the first time we were working with tkinter, it was a bit difficult to know what pieces of the package to use. In terms of the user interface, for the future, we would like to utilize more buttons like probably, probably not, and I don’t know. I think we also wanted to only need one yes button. The way our code runs, we have a yes button for the questions, and an indeed button reserved for the final “is blank your pokemon” question. I think while working through this project, we had to learn to use collaborative code. Because we each were working on different aspects of the code, and we had to merge each of our separate pieces to create a functioning code. Learning how to read to understand others' code was pivotal to the success of this project.

After we had a more or less complete prototype, we tried to improve the aesthetics of the game. Some examples are changing the background colors, adding animations and playing music. Fun fact, tkinter sucks for animations display.We first started by adding a background color to a yellow color. For animations, we continue by adding loops that constantly update the pictures on the screen. The way they work is that in the for loop, we iterate through a list of image paths, and open the one that corresponds with the index of the loop, and display it into the screen. I had a lot of trouble figuring this out. At first, I tried displaying a gif but the gif was never shown on the screen. I spent a while until I came up with this idea of displaying images to give animated effects. Sometimes I had trouble with the directory too and the type of pictures I used but luckily, I figured this stuff on time .In some cases, the final session is not shown due to a malfunction on our drive and renaming of folders, that is one thing I would love to fix. The music part was not too hard to figure out, the pygame.mixer module was pretty simple to use. However, it took me some time to figure out how to stop the music when the user exited the game, because the music was still playing, luckily I fixed this by checking if the user clicked on the button to close the window. If I had more time, I would work on a few things. First, fixing the pokemon images directory, basically renaming files. I would work on adding more music in case the music ends before someone can finish the game. Usually the game does not take too much time to finish but I am unsure if it will throw an error if the song finishes and then we try to change the song. It is very unlikely users will run into that problem, but I would like to check in that case. Finally I would love to do further decorations and make this problem more original e.g. changing colors of buttons, having personalized backgrounds, etc. I would have loved to work on more backend stuff too, and add a “probably ” button that helps sort the list even when the user does not know a fact about the pokemon.

Resources

Dataset of 32000 Pokemon Images & CSV, JSON

random pokespreadsheet - Google Sheets