Code Review

Name: Kev Docherty
Project: Hangman
GitHub: https://github.com/KevDocherty/hangman699/tree/main

Overall Impression
File Structure
Documentation
    README
    Docstrings
    In-line Comments
Code

Overall Impression

Overall, the code is pretty well written and structured. The user experience is strong, and the logic of the game has been coded properly.

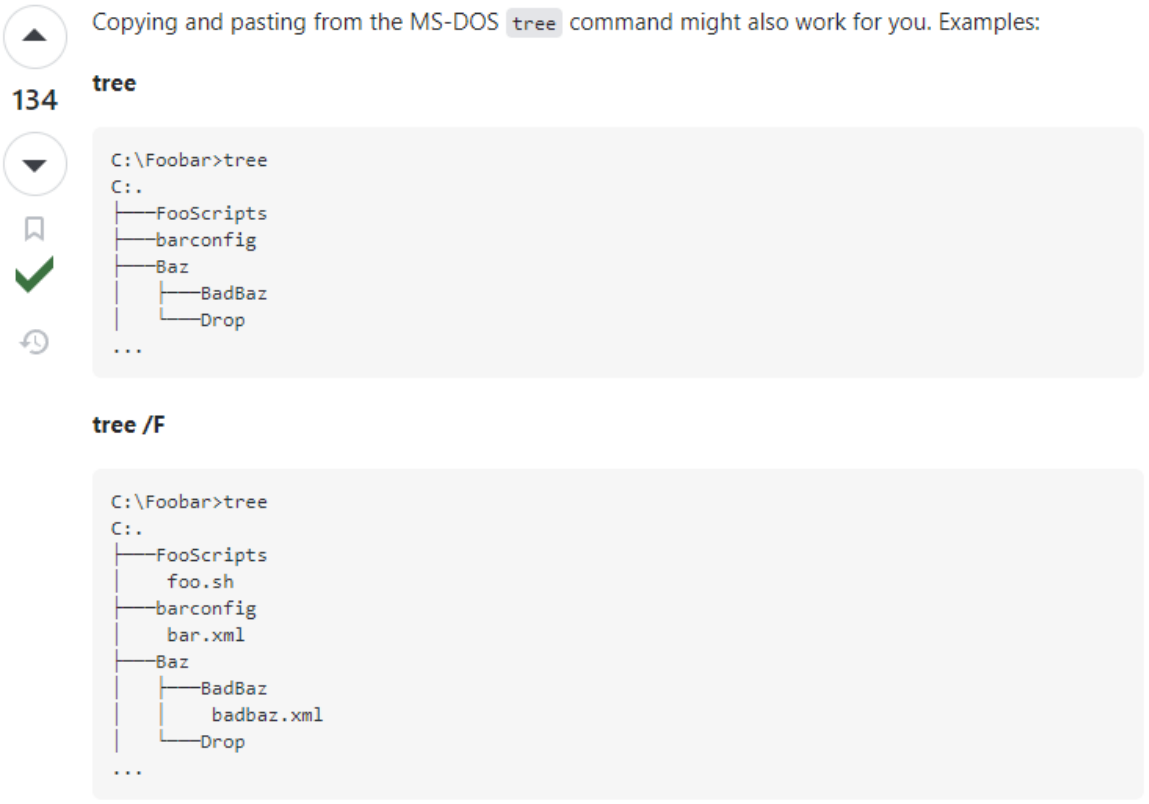
Documentation wise, the README could use some edits regarding the process of how to run the code. Some docstrings and in-line comments are missing too.

File Structure

The file structure provides a high level overview of the structure of your project.

For the file structure, I would suggest implementing a file structure diagram, which can be placed into your README like so:

A good example of a file structure diagram can be found in this stackoverflow link here:
https://stackoverflow.com/questions/347551/what-tool-to-use-to-draw-file-tree-diagram



134

Copying and pasting from the MS-DOS `tree` command might also work for you. Examples:

tree

```
C:\Foobar>tree
C:.
├───FooScripts
├───barconfig
├───Baz
│   ├───BadBaz
│   └───Drop
...
```

tree /F

```
C:\Foobar>tree
C:.
├───FooScripts
│    foo.sh
├───barconfig
│    bar.xml
├───Baz
│   ├───BadBaz
│   │    badbaz.xml
│   └───Drop
...
```

These file structures can be put in between backticks like these ``` within your README
Example

```
C:\Foobar>tree
C:.
├───FooScripts
├───barconfig
├───Baz
│   ├───BadBaz
│   └───Drop
```

...

Documentation

README

The README provides a high-level overview of the project, along with some instructions on how to go about running the project itself.

However, there are some points for improvement.

You should place instructions on how to go about running your project.

Within this section place some instructions on how to clone the repository and which python file should be run in order to play the game.

A file structure diagram is another improvement which will visually show your users how your project is structured.

Additionally, you can add a section regarding future improvements which can be made to your project.

For running your code and commands like git clone, I would suggest that put code blocks such as **python milestone_5.py** in backticks (``)

An example of each of this is shown below. These will help to enhance the presentation of the README file.

Code Blocks

```
python my_code.py
```

You can read up more about markdown syntax here:
https://www.markdownguide.org/basic-syntax/

Docstrings

The code lacks docstrings to explain what each method does.

Docstrings are a good way of documenting how a module, class, function or method works. This is good for providing a high level understanding to other programmers.

I would suggest reading up about docstrings here:

https://portal.theaicore.com/library/lesson/9de4031e-e4c8-4adc-845c-1222986607d5

https://www.geeksforgeeks.org/python-docstrings/

Using the **ask_for_input** method in your code as an example implementation.

```
34      def ask_for_input(self):
35          '''
36          Method to ask the input from an end user
37          The method will perform the following checks:
38
39          - If the letter is a single character
40          - If the letter entered is an alphanumeric character (i.e. from a to z)
41          - If the letter has been chosen already
42
43          Parameters:
44          None
45
46          Example Usage:
47
48          game = Hangman()
49
50          '''
```

In-line Comments

The code lacks in-line comments to display the functionality of how the code works. Especially during control flow statements.

Comments can be placed above the line or on the same line as the line of code. It is up to you how this is implemented.

Code

The code is well written. Most variables, methods, and classes are named appropriately and are descriptive, such as within check_guess and ask_for_input methods.

I do like the emphasis on the increased user experience by showing the progress on the users' word for when they guess a letter correctly or incorrectly.

The code currently shows the mystery word when playing the game, but I think that this is for debugging purposes. The code works as intended, so line 9 can be commented out.

I would also be vary of using **self** for variables inside your methods.

- The cases where referring to a variable using self is not needed are due to the following:
- when the variable is only used in 1 function, or is created inside a function/method and only used in that function/method
- when the variable doesn't need to be shared between methods
- when the variable doesn't need to be exposed to other classes/scopes/contexts

This can be problematic in your case, as **self.guess** is being shared between both **ask_for_input** and **check_guess** this can lead to some confusion later down the road for when the code is iterated and improved upon due to how dynamic the **self** variable changes throughout the code.

Furthermore when calling the main functionality of your code, I would suggest that you add an **if __name__=="__main__"** block.

When running your code within this block directly, only the play_game function, alongside anything else indented inside it will run.

For a more detailed explanation of this line of code, check out the following resource below.

https://portal.theaicore.com/library/lesson/28dc20c6-dc74-4e7c-b330-5f991779162d