INDU report for ASCII Art Studio - Maximus Österman

Overview

The program in question is the ASCII art with task 2. In essence the user will be able to load in an image, located in the project directory, and see it being displayed in ASCII art. The program is also able to change height, width, contrast and brightness. If you feel like quitting the program, you are able to save your sessions and load them in at a later moment. Finally you are also able to store the files with the changes on your machine.

Usage

Before using the program:
Firstly, to be able to run the program some python modules outside the standard library need to be installed. Depending on what OS you are running you need to use pip/pip3. However this can be done by using pip/pip3 and the “REQUIREMENTS.txt" file provided in the project directory. For example → `$ pip3 install -r requirements.txt`

Program is started by issuing main.py. The user will be welcomed and prompted to enter a command. In order to use a file in the program it needs to be located in the same directory as the program. The program revolves around the “current file” which is the latest file/image to have been used. It is also the standard file which is to be manipulated when using a command without specifying any file.
These are the following commands with arguments to be used:

- load image *filename*
  Loading a file and saving it in the session as *filename*. Loading a file makes it current.

- load image *filename* as *alias*
  As above with a difference being loading the file with an alias which is to be used *instead* of the filename.

- info
  Displays a list of all images in the current session with its properties. The current file is also listed here.

- render *img*
  Renders *img* in ascii-art. Keep in mind, when rendering the file is not resized automatically.

- render *img* to *filename* as alias
  This renders *alias* in ascii-art then saving it in a text file *filename*

- set *img* width *num*
  Changing *img* width to *num* pixels

- set *img* height *num*
  Changing *img* height to *num* pixels

- set *img* brightness *num*
  Changing *img* brightness to *num* value

- set *img* contrast *num*
  Changing *img* contrast to *num* value

- save session as *filename*
  Enter *filename* without the file extension. You can see all your saved sessions under the directory “sessions” in the project directory.

- load *session* as filename
  Load the *session* WITH its file extension. When loading a session the current_file and all the images with its properties will be loaded.

Code testing

Basic test coverage is run with pytest. Tests that are written are for checking that the parse method is working correctly and that corresponding for the functions is called. →

```
$ pytest test.py
```

Code Structure

There are four different modules in the project. “main.py” which is the module to be executed. The module only imports functionality. “helpers.py” only has some utility functions which are being used by the other modules.

“image_file.py” contains a class which is handling the images as objects. Attributes as images for the Pillow module, width, height, filename and size ratio. Methods in the class perform actions on the images. Getting its info, changing to gray, converting to ASCII, changing brightness, changing contrast and rendering the image. It is also in this module where the ASCII characters are defined in a list.

The algorithm being used in the program is when converting the image pixels to ASCII. Using a method on the Image class from the PIL lib, data from every pixel can be retrieved. Depending on the darkness of each pixel it is being paired to one character from the constant ASCII-char list. This is by floor dividing the pixel data by 25 and using it as an index in the ASCII-chars list.

“ascii_studios.py” contains a class which has the program functionality. A parse method takes in a user input and runs a method based on what command was entered. The benefit of factoring the code this way shows itself when the code is being tested. The parse function is being mocked with an input of a command. Assertions are then being made, checking that the right method is being called. Another benefit of structuring it by object orientation is using the “current file” as an attribute. The alternative is using a global variable which can make things a bit more complicated.

Considering that the code is to be tested, it is of the highest importance that the functions do one thing each. When writing the tests, some functions are to be mocked and therefore the code will be easier to test if just one function does one thing. One note to add here as well is that writing tests is often easier to do when a code is written in a maintainable and readable way. This can be done by using dependency injection, variable names and documenting the code, just to name a few.

The benefits of writing tests are many. If the code is to be maintained and expandable you will be able to make changes to your code with confidence. To make sure that your tests are reliable they need code coverage (testing as many scenarios as possible) and making sure that the tests fail before they are working, this to make sure that you actually test what you want to test.