Project 1

Purpose: Create an Image Manipulation App

Overview:

In this project, you will learn how to create an API to perform various tasks related to image manipulation, such as Cropping, Resizing, Watermarking etc.

Part 1: Design a parser

In the first part, we will simply design a parser that will take various arguments from the user. If we can design the parser, we can easily connect it with some interfaces. Our parser will looks like as follows:

```
(base) MacBook-Pro-2:pythonProject1 noors2$ python3 main.py --help
usage: We are designing an image editing app [-h] [--resize RESIZE RESIZE]
                                             [--crop CROP CROP] [--sign SIGN]
                                             [--rotate ROTATE]
                                             [--upload UPLOAD]
                                             select

positional arguments:
  select                Enter a file or folder

optional arguments:
  -h, --help            show this help message and exit
  --resize RESIZE RESIZE, -r RESIZE RESIZE
                        provide a vertical and horizontal dimension to crop
  --crop CROP CROP, -c CROP CROP
                        provide a vertical and horizontal dimension to crop
  --sign SIGN, -s SIGN  provide the file containing signature of the owner
  --rotate ROTATE, -rt ROTATE
                        provide an angle of rotation
  --upload UPLOAD, -u UPLOAD
                        provide a link where the image will be uploaded in
                        server
```

Part 2: Design Functions:

You need to create the following function:

def operation_resize(image, resize_val): This function will take an image and a 2D list resize_val (sent using parser optional argument) and return the resized image

def operation_crop(image,crop_val): This function will take an image and a 2D list crop_val (sent using parser optional argument) and return the cropped image

def operation_signature(image,sign): This function will take the orginal image and a signature as image format, and return the signed image

def operation_rotate(image, angle): This function will take an angle and rotate the image

Part 3: Working with Folders

From the parser, you can see that someone might like to provide a folder or file as input.

You can use os.path.isdir(name) to check if the provided name is a file or folder. If it returns true that means name is a folder, otherwise, it's a file

However, we can assume that the folder might be either zipped or unzipped. The zip folder normally ends with either .zip, .rar etc. Therefore, we might want to unzip the folder before doing any operations on the pictures inside the file. Depending on your operating system, you have to find the appropriate command for unzipping. For example, in MAC os, we can use the following commands to unzip:

#unzip name (if its .zip file)

#tar -xvf name (if it's .tar file)

However, first, we will check if the folder is in compressed format (.zip). If the folder is in zip, then we will unzip it first. Next, we can use glob module to get all the image files from the unzipped directory.

We will create a directory named 'processed' using our script.

Next, we will do the one of the operations specified by the user as a command line arguments and save all the edited image/images one by one in the processed directory.