DSN 2098

TEAM NAME: PYCAKE
TEAM NO: 63

Guide- Mr Kailash

PICZAP

Objective
How to add watermark on images using OpenCV in Python.

Group members-

1. Parmeet Singh Banwait - 20BAI10141
2. MVN Rajesh Reddy - 20BAI10249
3. Sparsh Handa - 20BAI10295
4. Mohd Mohsin Khan - 20BAI10340

Objective/ Problem Statement

What is the main objective, we are into??

How to add watermark on images using OpenCV in Python.

We will also look into-

1. How we can resize an image
2. Cropping an image
3. Compressing an image

Roadmap

What we are gonna going to discuss:

- Introduction to the project.
- Overall architecture diagram and flow diagram
- Complete Module split-up and Explanation
- 50% Module Implementation with demo
- Implementation Screenshot
- Refrences

Introduction to Piczap

Piczap is just a simple solution which makes our work easier

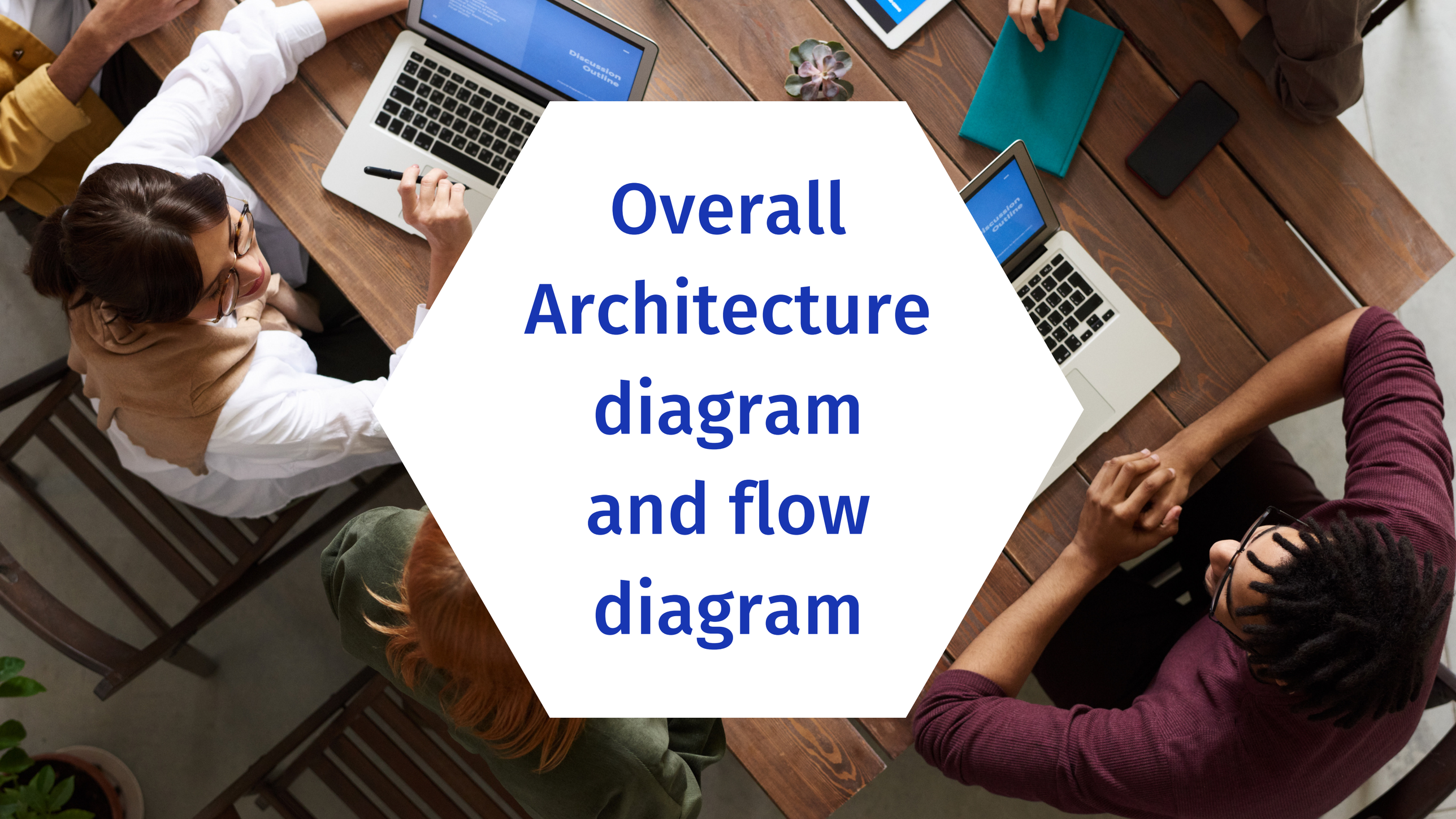
What makes "Software different"

Watermarks

Cropping

Resizing

Compression



Overall Architecture diagram and flow diagram

An image (which need to be edited)

1. To add watermark
2. Resize of an image
3. Compressing of an image
4. Cropping of an image
5. Exit

Adding watermark

- User will give the watermark to be added in the image

Resizing an image

- User will give the length and width of the image in which it need to be resized

Compressing an image

- User will give at what resolution the image he/she wants to resolve

Cropping an Image

- User give the length and breadth of the portion that need to be cropped

This part constitutes of six loop

1st loop

It will add watermark to the image.

2nd loop

It will remove watermark to the image.

3rd loop

It will resize the image.

4th loop

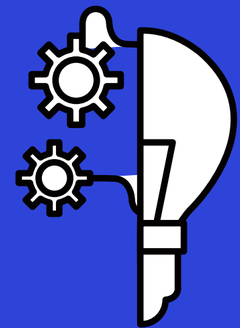
It will compress the image.

5th loop

It will crop the image.

6th loop

It will break the loop.



PROCESSING

Output

It displays the final image of after the processing.

Complete Module split-up and Explanation

Library used in the project-

CV2

OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.

PIL, Image

PIL is an additional, free, open-source library for the Python programming language that provides support for opening, manipulating, and saving many different image file formats.

OS

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

sys

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

random

Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random.

skimage

Scikit-image, or skimage, is open source Python package designed for image preprocessing. If you have previously worked with sklearn, getting started with skimage will be a piece of cake. Even if you are completely new to Python, skimage is fairly easy to learn and use.

Piczap

Watermark

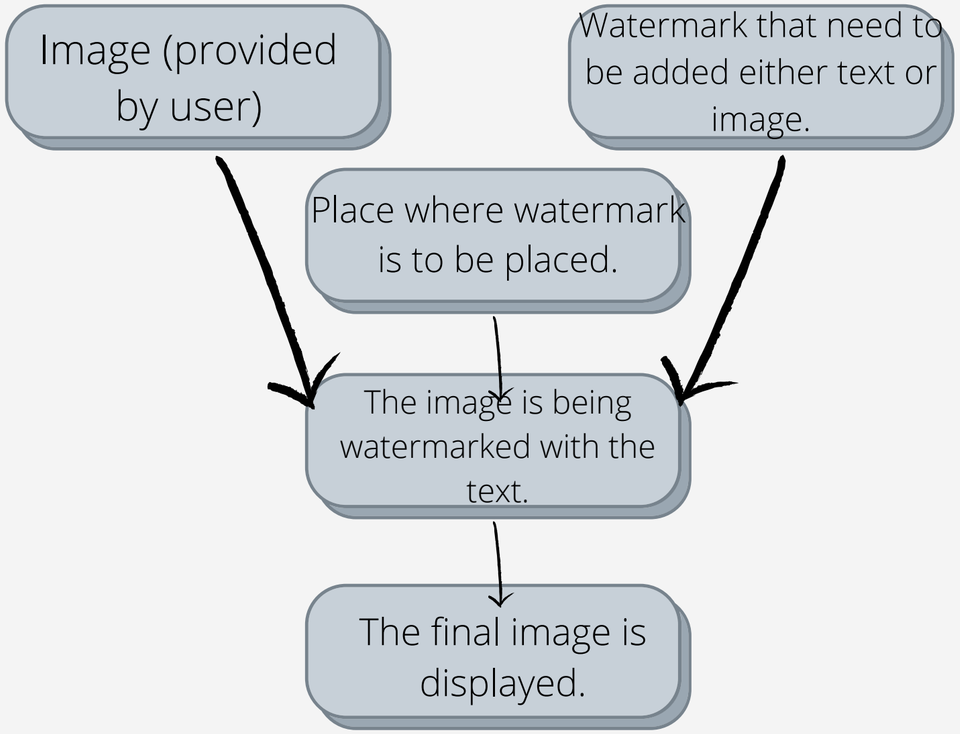
Cropping

Compressing

Resizing

Adding Watermark:-

These function of the app enables us to add watermark to the image.



Cropping an Image

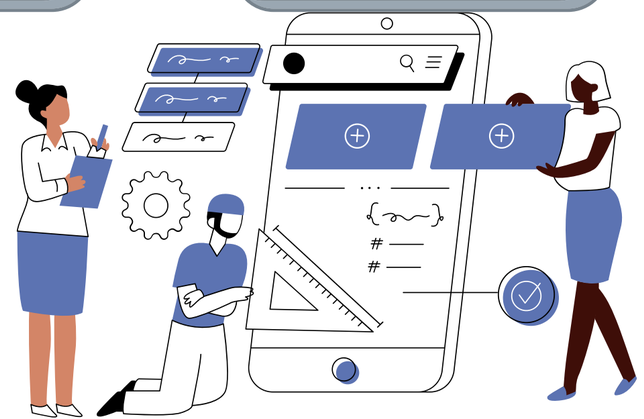
It enables us to crop an image by giving the exact length and width of the image.

The image that need to be cropped.

The length & breadth of the portion to be cropped.

The image is being cropped.

The final image is displayed.



Resizing an Image

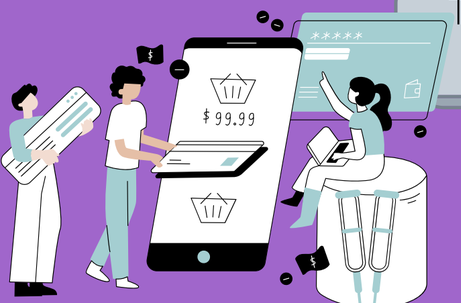
It is used to resize the image according to users will.

The image that need to be resized.

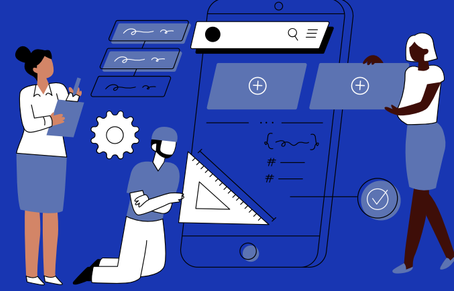
The length and breadth at which the image need to be resized.

The image is being resized.

The final image is displayed.



Compressing an image



It is used to compress the image according to users choice.

The image that need to be resized.

The resolution at which we want to compress.

The image is being compressed.

The final image is displayed.


50% Module Implementation
Screenshots

Image used->

<- watermark used

```python
def watermark_image():
    logo = cv2.imread("PHOTO.jpg")

    # importing image on which we are going to apply watermark
    img = cv2.imread("Watermark.jpg")

    # calculating dimensions
    # height and width of the logo
    h_logo, w_logo, _ = logo.shape

    # height and width of the image
    h_img, w_img, _ = img.shape

    # calculating coordinates of center
    # calculating center, where we are going to
    # place our watermark
    center_y = int(h_img/2)
    center_x = int(w_img/2)

    # calculating from top, bottom, right and left
    top_y = center_y - int(h_logo/2)
    left_x = center_x - int(w_logo/2)
    bottom_y = top_y + h_logo
    right_x = left_x + w_logo

    # adding watermark to the image
    destination = img[top_y:bottom_y, left_x:right_x]
    result = cv2.addWeighted(destination, 1, logo, 0.5, 0)

    # displaying and saving image
    img[top_y:bottom_y, left_x:right_x] = result
    cv2.imwrite("watermarked.jpg", img)
    cv2.imshow("Watermarked Image", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Watermarking an Image

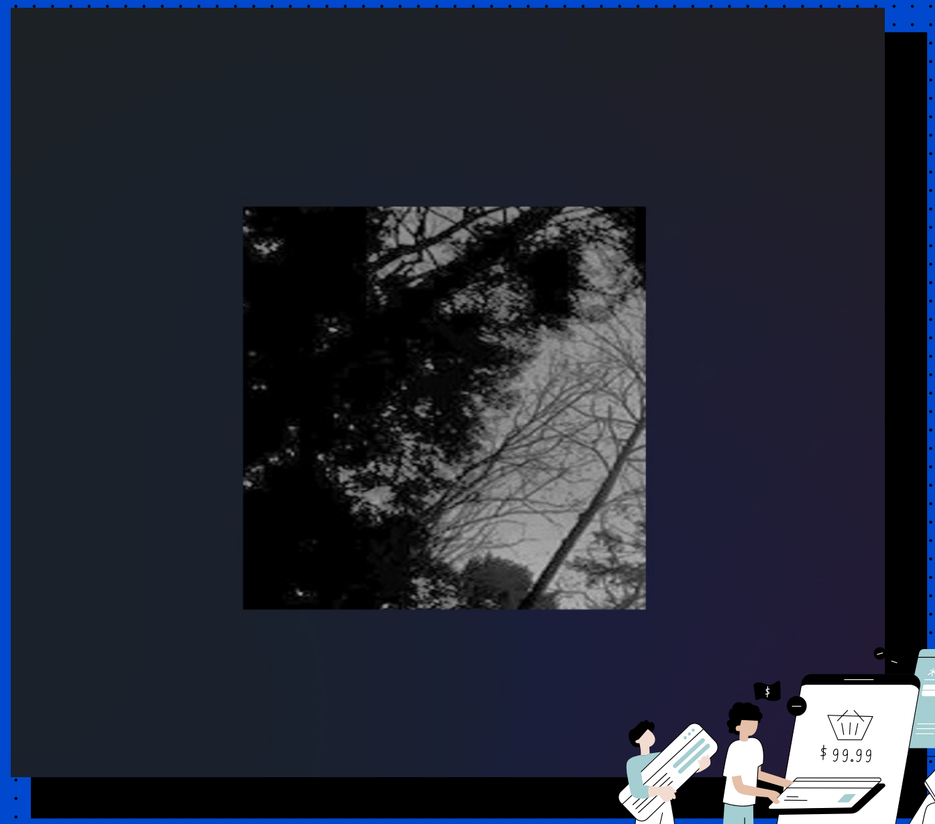
Cropping an Image

```python
def crop_image():
    im = Image.open("PHOTO.jpg")

    # Size of the image in pixels (size of original image)
    width, height = im.size

    # Setting the points for cropped image
    #left = 4
    left=(int(input("Enter left side value for crop image: ")))
    top = height / 5
    #right = 154
    right=(int(input("Enter right side value for crop image: ")))
    bottom = 3 * height / 5


        # Cropped image of above dimension
    im1 = im.crop((left, top, right, bottom))
    newsize = (300, 300)
    im1 = im1.resize(newsize)
    # Shows the image in image viewer
    im1.show()
```



Compressing an Image

```python
def compress_image():
    def initialize_K_centroids(X, K):
        m = len(X)
        return X[np.random.choice(m, K, replace=False), :]

    def find_closest_centroids(X, centroids):
        m = len(X)
        c = np.zeros(m)
        for i in range(m):
        # Find distances
            distances = np.linalg.norm(X[i] - centroids, axis=1)
```

18:37

```python

        # Assign closest cluster to c[i]
            c[i] = np.argmin(distances)

        return c

    def compute_means(X, idx, K):
        _, n = X.shape
        centroids = np.zeros((K, n))
        for k in range(K):
            examples = X[np.where(idx == k)]
            mean = [np.mean(column) for column in examples.T]
            centroids[k] = mean
        return centroids

    def find_k_means(X, K, max_iters=10):
        centroids = initialize_K_centroids(X, K)
        previous_centroids = centroids
        for _ in range(max_iters):
            image = load_image(image_path)
```

18:37

```python
        w, h, d = image.shape
        print('Image found with width: {}, height: {}, depth: {}'.format(w, h, d))

        X = image.reshape((w * h, d))

        idx = find_closest_centroids(X, centroids)
        centroids = compute_means(X, idx, K)
        if (centroids == previous_centroids).all():
            return centroids
        else:
            previous_centroids = centroids

        return centroids, idx
```

```python
        else:
            previous_centroids = centroids

        return centroids, idx

    try:
        image_path = sys.argv[1]
        assert os.path.isfile(image_path)
    except (IndexError, AssertionError):
        print('Please specify an image')
```

18:37

```python

    def load_image(path):
        """ Load image from path. Return a numpy array """
        image = Image.open('PHOTO.jpg')
        return np.asarray(image) / 255

    image = load_image(image_path)
    w, h, d = image.shape
    print('Image found with width: {}, height: {}, depth: {}'.format(w, h, d))

    X = image.reshape((w * h, d))
    K=int(input("enter your desired number of colors in the compressed image"))

    #K = 20, the desired number of colors in the compressed image

    colors, _ = find_k_means(X, K, max_iters=20)

    idx = find_closest_centroids(X, colors)
```

18:37

```python
    idx = np.array(idx, dtype=np.uint8)
    X_reconstructed = np.array(colors[idx, :] * 255, dtype=np.uint8).reshape((w, h, d))
    compressed_image = Image.fromarray(X_reconstructed)

    compressed_image.save('out.png')
```

References

https://pypi.org/project/opencv-python/

https://www.edureka.co/blog/python-libraries/

https://pillow.readthedocs.io/en/stable/

https://scikit-image.org/

https://docs.python.org/3/library/random.html

Thank you!

Any questions for us??