Lab 01 – Computer Vision

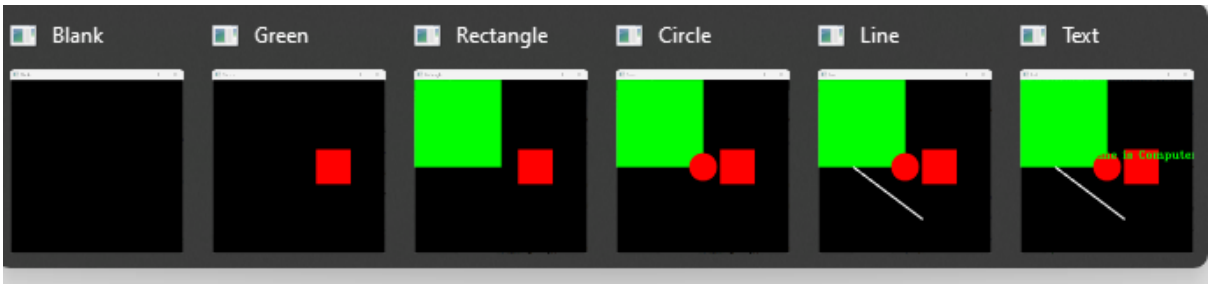
BSDSF21M020 – BUSHRA SHAHBAZ

1st October, 2024

Computer Vision **1st October, 2024**

Lab Tasks: **Submit by the end of the Lecture**

Task#1: Run 1draw.py. State your observations.



1. **Blank Image**: A black canvas of size 500x500 pixels is created using np.zeros(). This blank image is shown in a window labeled "Blank," with all pixel values initially set to zero (black).
2. **Red Square**: A section of the image, from coordinates (200, 300) to (300, 400), is filled with red ((0, 0, 255) in BGR format). This small red square is added to the canvas, and the updated image is shown in a window labeled "Green."
3. **Green Rectangle**: A larger green rectangle is drawn, starting at the top-left corner (0, 0) and covering half of the canvas (the top-left quadrant). This green rectangle ((0, 255, 0)) is filled in completely due to the thickness being set to -1.
4. **Red Circle**: A red circle is drawn at the center of the canvas, with a radius of 40 pixels. It overlaps the green rectangle because both are placed at the center of the image.
5. **White Line**: A white line is drawn diagonally across the canvas, starting from coordinates (100, 250) and ending at (300, 400), with a thickness of 3 pixels. The line cuts across the existing shapes on the image.
6. **Text**: Green text, "Hello, my name is Computer Vision!!!", is added near the center of the image. It uses the FONT_HERSHEY_TRIPLEX font and is positioned at coordinates (0, 225).

Each element is drawn on the same canvas, and the updates are displayed progressively using cv.imshow(). The waitKey(0) function keeps the windows open until a key is pressed.

Task #2: Run 2imageSize.py. State your observations.

1. **Opening the Image**: The script opens the image file "cat_hat.jpg" using Image.open(). The image is loaded and stored in the variable image.
2. **Image Dimensions**: The size of the image is retrieved using the image.size attribute. This returns a 2-tuple containing the width and height of the image in pixels.
3. **Displaying Size and Dimensions**: The width and height are then printed individually using print(width, height), and the full size (tuple) is also printed via print(image.size).

In summary, the script loads the image, retrieves its dimensions, and prints them. This demonstrates how Pillow allows for easy manipulation and access to image properties.

In [2]:
```
"""
Created on Fri Sep 27 22:59:36 2024

@author: HP
"""

from PIL import Image

filename = "cat_hat.jpg"
with Image.open(filename) as image:
    width, height = image.size
print(image.size)
print(width,height)
#Image.size gives  2-tuple and the width, height can be obtained
```

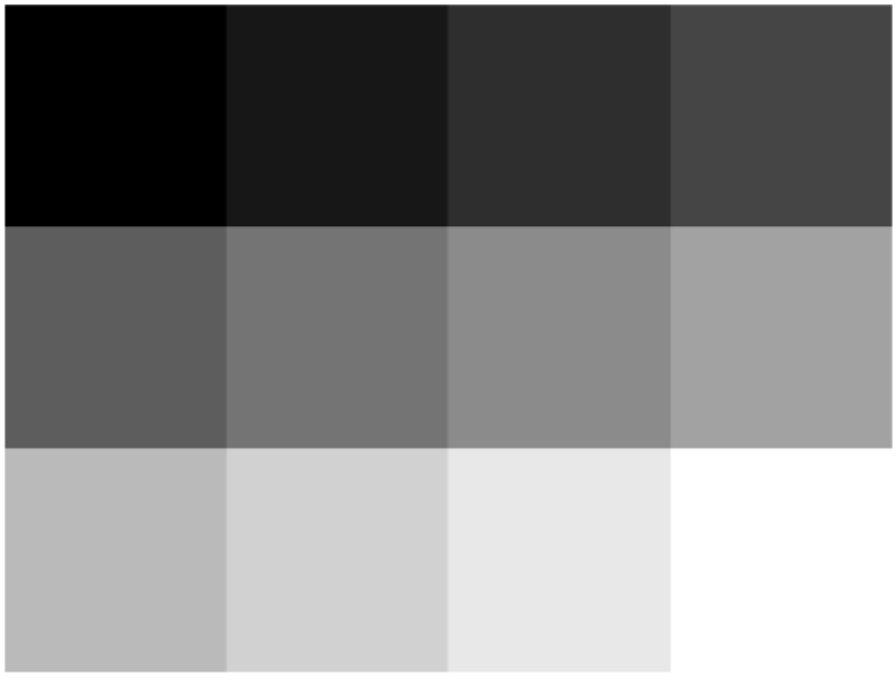
```
(300, 168)
300 168
```

Task #3: Run 3MyFirst.py. State your observations.

- **Prints a message**: "My First Python Program".
- **Creates a NumPy array**: A 3x4 matrix using np.arange(12).reshape(3,4).
- **Displays the array as an image**: Using plt.imshow() with a grayscale color map.
- **Removes axes**: Using plt.axis('off').
- **Shows the image**: With plt.show(), visualizing the matrix as a grayscale image where pixel intensity reflects array values.

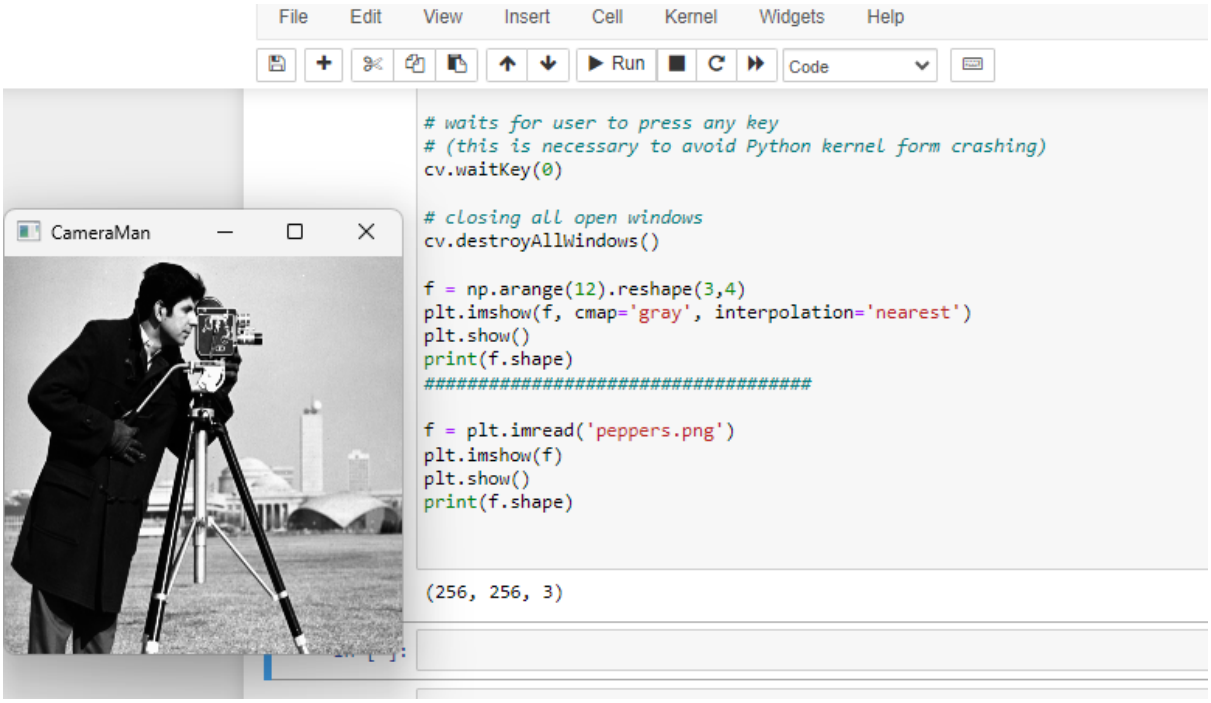
```
plt.axis('off')
plt.show()
```

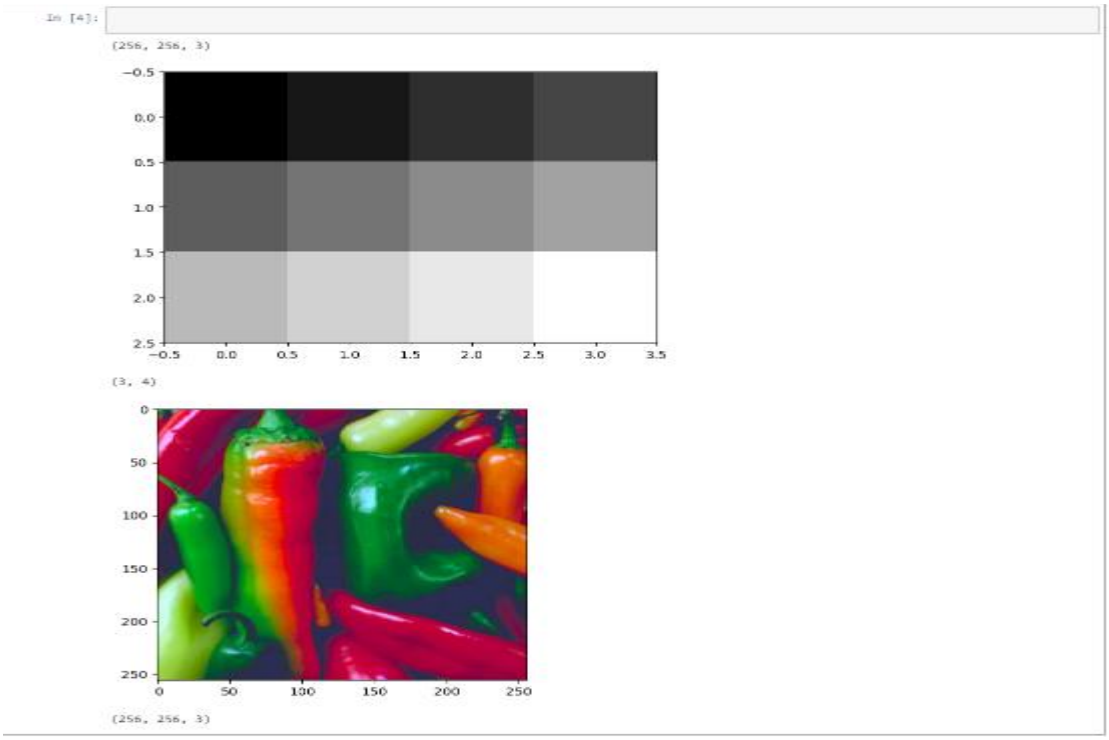
```
My First Python Program
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```



Task #4: Run 4Read_write.py. State your observations.

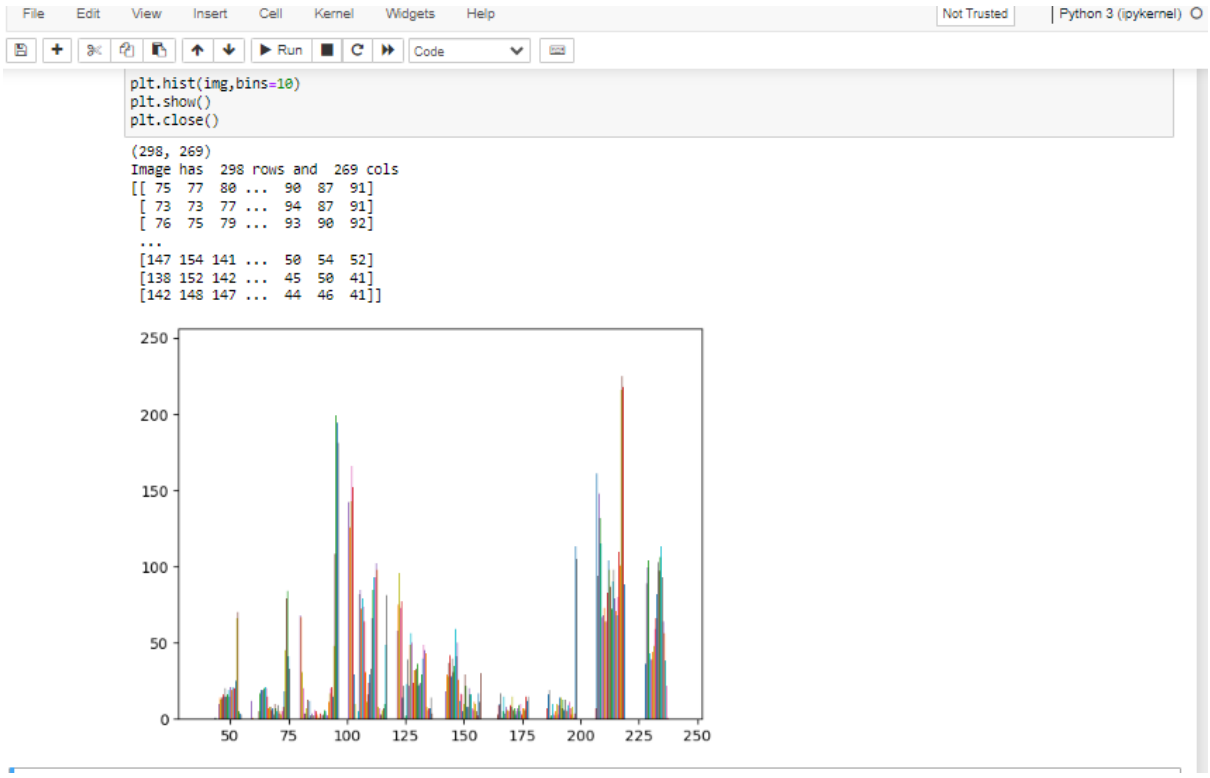
- **Reads and Displays an Image**:
  - The image "cameraman.png" is loaded using cv.imread().
  - It is displayed in a window named 'CameraMan' using cv.imshow().
  - The dimensions of the image (height, width, channels) are printed using image.shape.
  - The script waits for a key press with cv.waitKey(0) and then closes all windows with cv.destroyAllWindows().
- **NumPy Array Visualization**:
  - A 3x4 matrix f is created using np.arange(12).reshape(3,4) and visualized as a grayscale image using plt.imshow(). The matrix dimensions are printed via f.shape.
- **Reads and Displays Another Image**:
  - The image "peppers.png" is loaded using plt.imread() and displayed using plt.imshow().
  - The dimensions of the image are printed using f.shape.





Task #5: Run 5row_col.py. State your observations.

- **Reads an Image**:
  - The image file 'gray_output.jpg' is read using io.imread(), and its shape (dimensions) is printed. This shape indicates the number of rows and columns in the image.
- **Extracts Dimensions**:
  - The number of rows (r) and columns (c) are extracted from img.shape, where r represents the height and c represents the width of the image.
  - It then prints the dimensions, indicating how many rows and columns the image has.
- **Prints Image Values**:
  - The script prints all pixel values of the image stored in the img array. This will display the grayscale intensity values if the image is in grayscale.
- **Histogram of Pixel Values**:
  - A histogram of the pixel values is created using plt.hist(img, bins=10), which shows the distribution of pixel intensities across the image.
  - Finally, plt.show() displays the histogram, and plt.close() closes the plotting window.



Task #6: Run 6run_tell.py. State your observations.

1. **Imports Libraries**: It imports matplotlib.pyplot for plotting and skimage.data for sample images.
2. **Loads and Converts Image**: It loads a sample RGB image of a cat and converts it to grayscale using rgb2gray.
3. **Creates Subplots**: It sets up a figure with two subplots to display the original RGB image and the grayscale image side by side.
4. **Displays Images**: Each image is shown with appropriate titles ("RGB" for the original and "Grayscale" for the converted image).
5. **Shows the Plot**: The layout is adjusted for clarity, and the final plot is displayed using plt.show().