# OpenCV Image Capture, Display, and Manipulation Script

1. Import the OpenCV library:
   ```python
   import cv2
   ```

2. Create a VideoCapture object for the default camera (camera index 0):
   ```python
   cap = cv2.VideoCapture(0)
   ```

3. Read a frame from the camera:
   ```python
   status, photo = cap.read()
   ```

4. Check the status of the read operation:
   ```python
   status
   ```

5. Display the captured frame in a window named "ayush":
   ```python
   cv2.imshow("ayush", photo)
   ```

6. Wait for any key press:
   ```python
   cv2.waitKey()
   ```

7. Close all OpenCV windows:
   ```python
   cv2.destroyAllWindows()
   ```

8. Release the VideoCapture object:
   ```python
   cap.release()
   ```

```

9. Access the shape of the captured frame:
```python
photo.shape
```

10. Create a cropped copy of the image:
```python
crop = photo[200:500, 200:500].copy()
```

11. Display the original image with the OpenCV window named "ayush":
```python
cv2.imshow("ayush", photo)
```

12. Wait for any key press:
```python
cv2.waitKey()
```

13. Close all OpenCV windows:
```python
cv2.destroyAllWindows()
```

14. Resize the cropped image to (100, 100) pixels:
```python
crop_resize = cv2.resize(crop, (100, 100))
```

15. Resize the cropped image again (redundant line, can be removed):
```python
crop_resize = cv2.resize(crop, (100, 100))
```

16. Display the original image with the OpenCV window named "ayush":
```python
cv2.imshow("ayush", photo)
```

17. Wait for any key press:
```python

```
cv2.waitKey()
```

18. Close all OpenCV windows:
```python
cv2.destroyAllWindows()
```

19. Save the original image to a file named "my.png":
```python
cv2.imwrite("my.png", photo)
```

# Linear Regression Analysis for Salary Prediction using scikit-learn

1. Import the Pandas library and read a CSV file named "salary_data.csv" into a DataFrame:
```python
import pandas as pd
dataset = pd.read_csv("salary_data.csv")
```

2. Display the content of the DataFrame:
```python
dataset
```

3. Extract the dependent variable (target) 'Salary' and the independent variable 'YearsExperience':
```python
y = dataset["Salary"]
x = dataset["YearsExperience"]
```

4. Import the `train_test_split` function from scikit-learn and split the dataset into training and testing sets:
```python
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42)
```

5. Check the shape of the independent variable 'x':
```python
x.shape
```

6. Reshape the independent variable 'x' to be a 2D array:
```python
X = x.values.reshape(30, 1)
```

7. Check the number of dimensions of the reshaped 'X':
```python
X.ndim
```

8. Import the `LinearRegression` class from scikit-learn and create a linear regression model:
```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

9. Fit the linear regression model using the training data:
```python
model.fit(X, y)
```

10. Predict the salary for a given years of experience (e.g., 4.7 years):
```python
model.predict([[4.7]])
```

11. Access the coefficient(s) of the linear regression model:
```python
model.coef_
```

12. Access the intercept of the linear regression model:
```python
model.intercept_
```

13. Predict the salaries for the entire dataset and store the predictions in 'y_pred':
   ```python
   y_pred = model.predict(X)
   ```

14. Import the `mean_absolute_error` function from scikit-learn and calculate the mean absolute error between the actual 'y' and predicted 'y_pred':
   ```python
   from sklearn import metrics
   metrics.mean_absolute_error(y, y_pred)
   ```



   ```
   ```