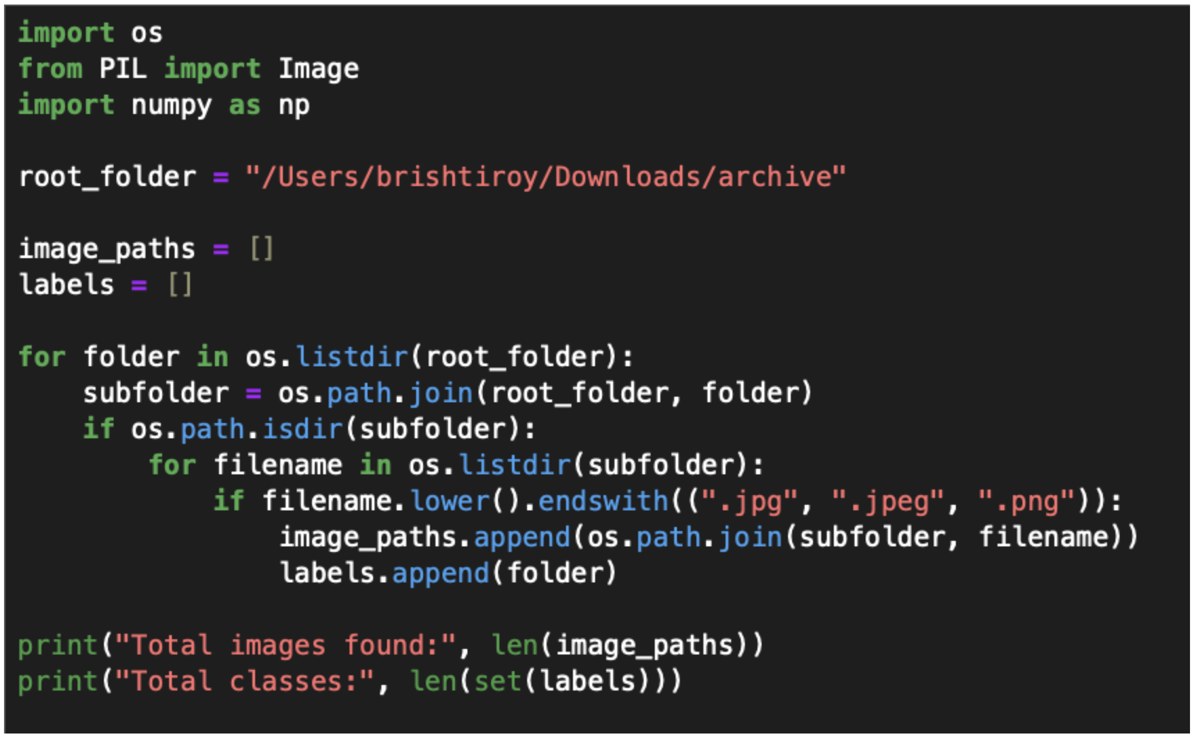
Brishti Roy
25MML0027

Assignment 1

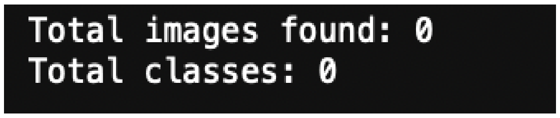
Cell 1 – Dataset Scanning & Label Collection

Code:



```python
import os
from PIL import Image
import numpy as np

root_folder = "/Users/brishtiroy/Downloads/archive"

image_paths = []
labels = []

for folder in os.listdir(root_folder):
    subfolder = os.path.join(root_folder, folder)
    if os.path.isdir(subfolder):
        for filename in os.listdir(subfolder):
            if filename.lower().endswith((".jpg", ".jpeg", ".png")):
                image_paths.append(os.path.join(subfolder, filename))
                labels.append(folder)

print("Total images found:", len(image_paths))
print("Total classes:", len(set(labels)))
```

Output:



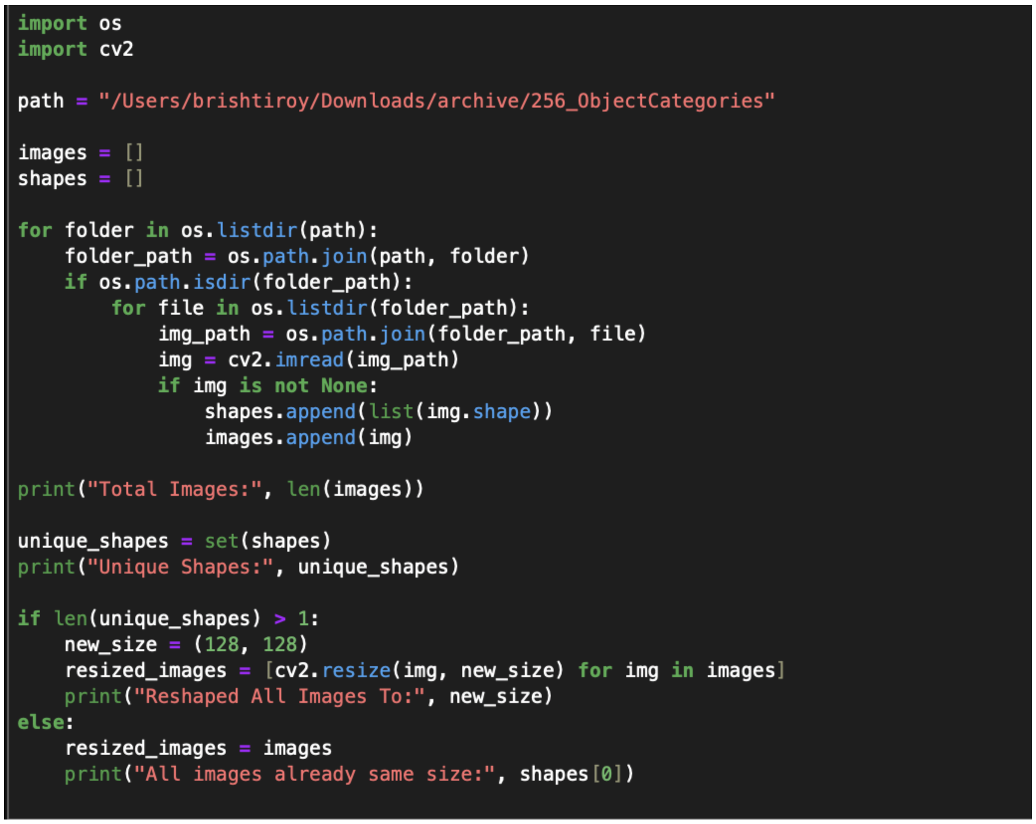
```
Total images found: 0
Total classes: 0
```

Reasoning:

- Scans the root dataset directory and treats each subfolder as a class label.
- Collects image file paths and assigns labels based on the folder name.
- Filters only valid image formats (.jpg, .jpeg, .png) to avoid non-image files.
- Prints a summary showing total images found and number of unique classes.
- Here it shows 0 because the file location access the main location instead if the nested ones.

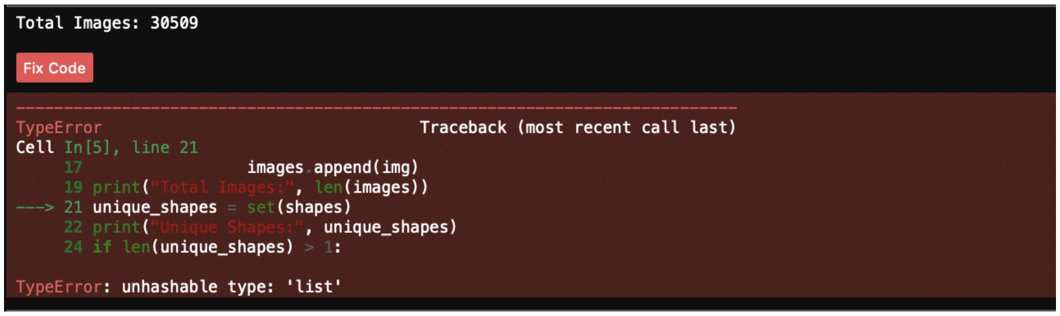
Cell 2 – Image Loading, Shape Check and Error

Code:



```
import os
import cv2

path = "/Users/brishtiroy/Downloads/archive/256_ObjectCategories"

images = []
shapes = []

for folder in os.listdir(path):
    folder_path = os.path.join(path, folder)
    if os.path.isdir(folder_path):
        for file in os.listdir(folder_path):
            img_path = os.path.join(folder_path, file)
            img = cv2.imread(img_path)
            if img is not None:
                shapes.append(list(img.shape))
                images.append(img)

print("Total Images:", len(images))

unique_shapes = set(shapes)
print("Unique Shapes:", unique_shapes)

if len(unique_shapes) > 1:
    new_size = (128, 128)
    resized_images = [cv2.resize(img, new_size) for img in images]
    print("Reshaped All Images To:", new_size)
else:
    resized_images = images
    print("All images already same size:", shapes[0])
```

Output:



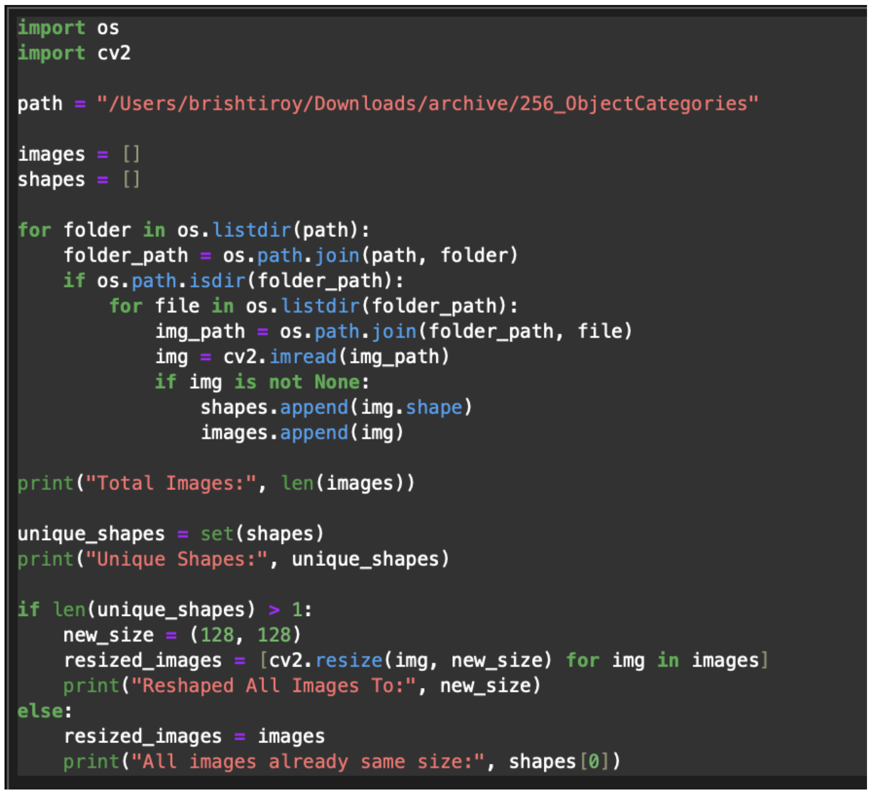
```
Total Images: 30509

Fix Code

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 21
     17                 images.append(img)
     19 print("Total Images:", len(images))
---> 21 unique_shapes = set(shapes)
     22 print("Unique Shapes:", unique_shapes)
     24 if len(unique_shapes) > 1:

TypeError: unhashable type: 'list'
```

Reasoning:

- This cell loads all images and stores their dimensions in the shapes list to check whether all images have the same size.
- An error occurs when converting shapes to a set because each shape is stored as a list, and lists are unhashable in Python.
- The error happens because set() only works with immutable data types, and lists are mutable.
- This issue is resolved by storing image shapes as tuples instead of lists.

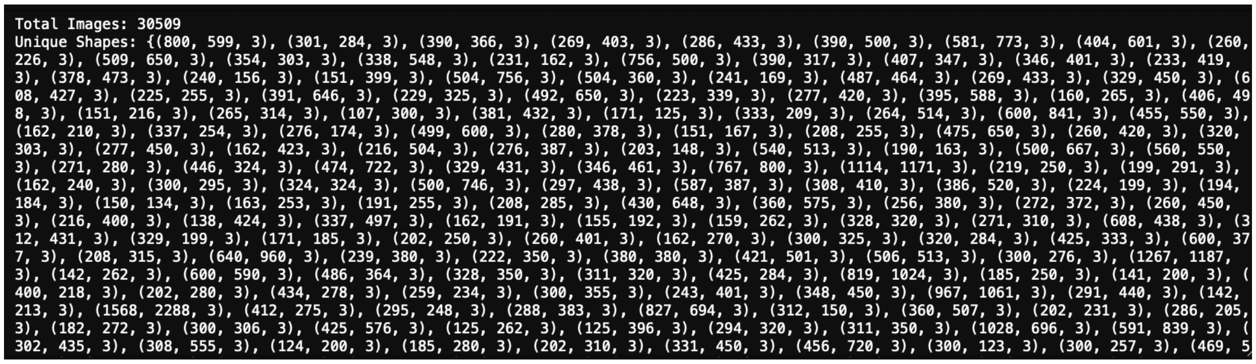
Cell 3 – Error Fix and Image Shape Analysis

Code:



```python
import os
import cv2

path = "/Users/brishtiroy/Downloads/archive/256_ObjectCategories"

images = []
shapes = []

for folder in os.listdir(path):
    folder_path = os.path.join(path, folder)
    if os.path.isdir(folder_path):
        for file in os.listdir(folder_path):
            img_path = os.path.join(folder_path, file)
            img = cv2.imread(img_path)
            if img is not None:
                shapes.append(img.shape)
                images.append(img)

print("Total Images:", len(images))

unique_shapes = set(shapes)
print("Unique Shapes:", unique_shapes)

if len(unique_shapes) > 1:
    new_size = (128, 128)
    resized_images = [cv2.resize(img, new_size) for img in images]
    print("Reshaped All Images To:", new_size)
else:
    resized_images = images
    print("All images already same size:", shapes[0])
```

Output:



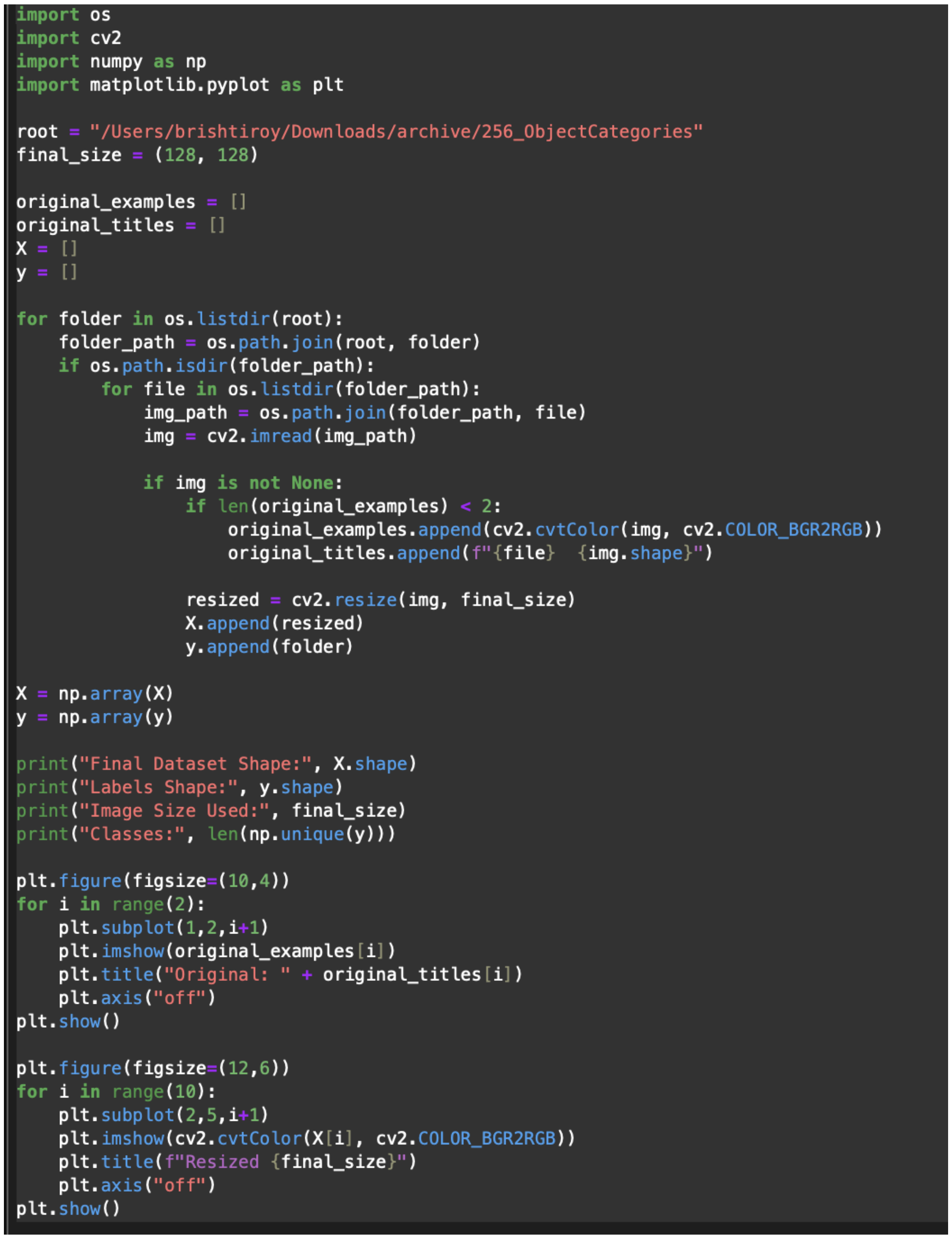
```
Total Images: 30509
Unique Shapes: {(800, 599, 3), (301, 284, 3), (390, 366, 3), (269, 403, 3), (286, 433, 3), (390, 500, 3), (581, 773, 3), (404, 601, 3), (260,
226, 3), (509, 650, 3), (354, 303, 3), (338, 548, 3), (231, 162, 3), (756, 500, 3), (390, 317, 3), (407, 347, 3), (346, 401, 3), (233, 419,
3), (378, 473, 3), (240, 156, 3), (151, 399, 3), (504, 756, 3), (504, 360, 3), (241, 169, 3), (487, 464, 3), (269, 433, 3), (329, 450, 3), (6
08, 427, 3), (225, 255, 3), (391, 646, 3), (229, 325, 3), (492, 650, 3), (223, 339, 3), (277, 420, 3), (395, 588, 3), (160, 265, 3), (406, 49
8, 3), (151, 216, 3), (265, 314, 3), (107, 300, 3), (381, 432, 3), (171, 125, 3), (333, 209, 3), (264, 514, 3), (600, 841, 3), (455, 550, 3),
(162, 210, 3), (337, 254, 3), (276, 174, 3), (499, 600, 3), (280, 378, 3), (151, 167, 3), (208, 255, 3), (475, 650, 3), (260, 420, 3), (320,
303, 3), (277, 450, 3), (162, 423, 3), (216, 504, 3), (276, 387, 3), (203, 148, 3), (540, 513, 3), (190, 163, 3), (500, 667, 3), (560, 550,
3), (271, 280, 3), (446, 324, 3), (474, 722, 3), (329, 431, 3), (346, 461, 3), (767, 800, 3), (1114, 1171, 3), (219, 250, 3), (199, 291, 3),
(162, 240, 3), (300, 295, 3), (324, 324, 3), (500, 746, 3), (297, 438, 3), (587, 387, 3), (308, 410, 3), (386, 520, 3), (224, 199, 3), (194,
184, 3), (150, 134, 3), (163, 253, 3), (191, 255, 3), (208, 285, 3), (430, 648, 3), (360, 575, 3), (256, 380, 3), (272, 372, 3), (260, 450,
3), (216, 400, 3), (138, 424, 3), (337, 497, 3), (162, 191, 3), (155, 192, 3), (159, 262, 3), (328, 320, 3), (271, 310, 3), (608, 438, 3), (3
12, 431, 3), (329, 199, 3), (171, 185, 3), (202, 250, 3), (260, 401, 3), (162, 270, 3), (300, 325, 3), (320, 284, 3), (425, 333, 3), (600, 37
7, 3), (208, 315, 3), (640, 960, 3), (239, 380, 3), (222, 350, 3), (380, 380, 3), (421, 501, 3), (506, 513, 3), (300, 276, 3), (1267, 1187,
3), (142, 262, 3), (600, 590, 3), (486, 364, 3), (328, 350, 3), (311, 320, 3), (425, 284, 3), (819, 1024, 3), (185, 250, 3), (141, 200, 3), (
400, 218, 3), (202, 280, 3), (434, 278, 3), (259, 234, 3), (300, 355, 3), (243, 401, 3), (348, 450, 3), (967, 1061, 3), (291, 440, 3), (142,
213, 3), (1568, 2288, 3), (412, 275, 3), (295, 248, 3), (288, 383, 3), (827, 694, 3), (312, 150, 3), (360, 507, 3), (202, 231, 3), (286, 205,
3), (182, 272, 3), (300, 306, 3), (425, 576, 3), (125, 262, 3), (125, 396, 3), (294, 320, 3), (311, 350, 3), (1028, 696, 3), (591, 839, 3), (
302, 435, 3), (308, 555, 3), (124, 200, 3), (185, 280, 3), (202, 310, 3), (331, 450, 3), (456, 720, 3), (300, 123, 3), (300, 257, 3), (469, 5
```

Reasoning:

- The earlier error is fixed in this cell by storing image shapes as tuples (img.shape) instead of lists, which allows them to be used inside a set.
- After fixing this, the set of unique shapes is successfully created, showing that the dataset contains images with many different dimensions.
- The output confirms that image sizes are inconsistent, which explains why resizing is required before further processing.

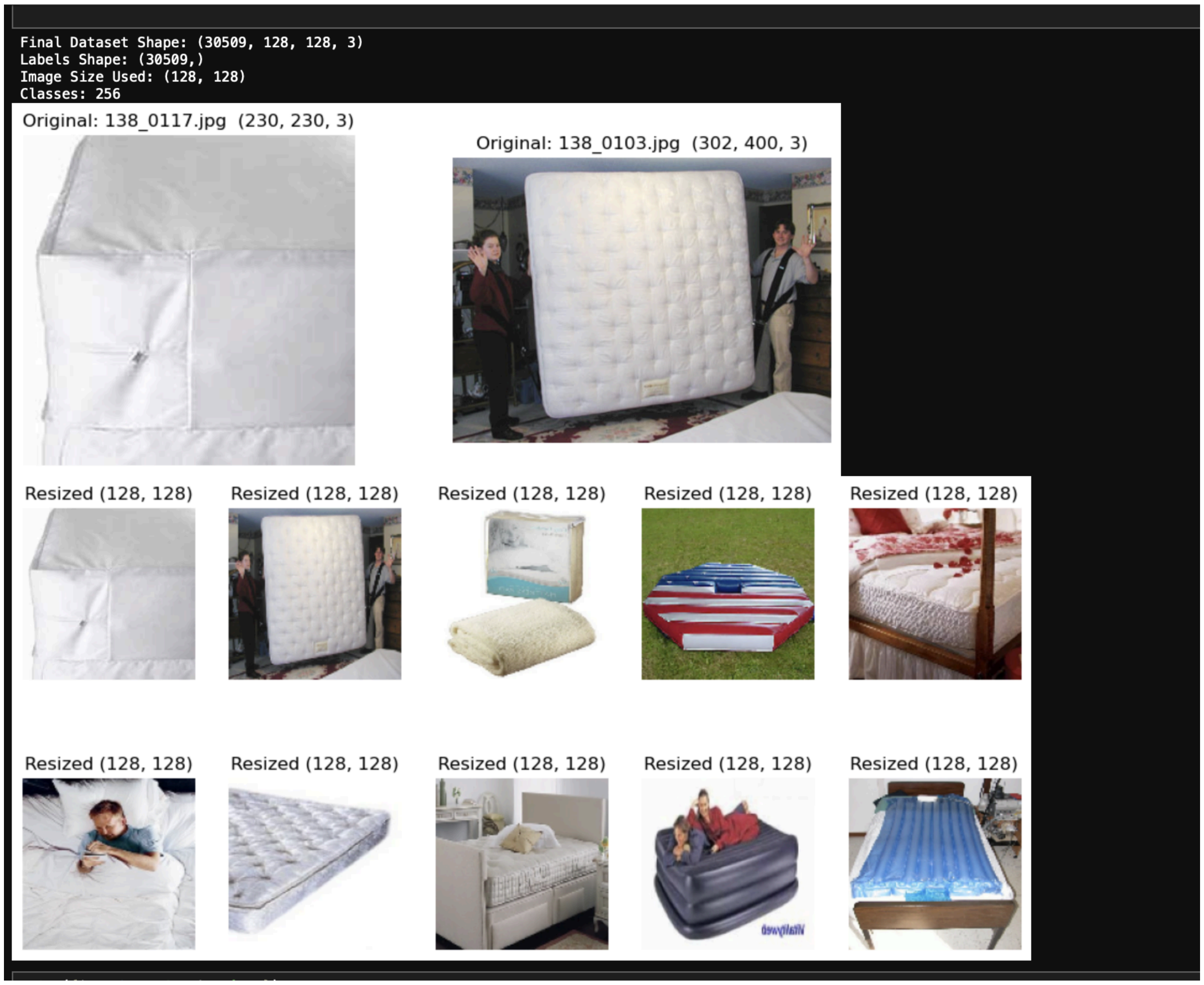
Cell 4 – Final Dataset Creation and Visualization

Code:



```python
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt

root = "/Users/brishtiroy/Downloads/archive/256_ObjectCategories"
final_size = (128, 128)

original_examples = []
original_titles = []
X = []
y = []

for folder in os.listdir(root):
    folder_path = os.path.join(root, folder)
    if os.path.isdir(folder_path):
        for file in os.listdir(folder_path):
            img_path = os.path.join(folder_path, file)
            img = cv2.imread(img_path)

            if img is not None:
                if len(original_examples) < 2:
                    original_examples.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
                    original_titles.append(f"{file}  {img.shape}")

                resized = cv2.resize(img, final_size)
                X.append(resized)
                y.append(folder)

X = np.array(X)
y = np.array(y)

print("Final Dataset Shape:", X.shape)
print("Labels Shape:", y.shape)
print("Image Size Used:", final_size)
print("Classes:", len(np.unique(y)))

plt.figure(figsize=(10,4))
for i in range(2):
    plt.subplot(1,2,i+1)
    plt.imshow(original_examples[i])
    plt.title("Original: " + original_titles[i])
    plt.axis("off")
plt.show()

plt.figure(figsize=(12,6))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(cv2.cvtColor(X[i], cv2.COLOR_BGR2RGB))
    plt.title(f"Resized {final_size}")
    plt.axis("off")
plt.show()
```

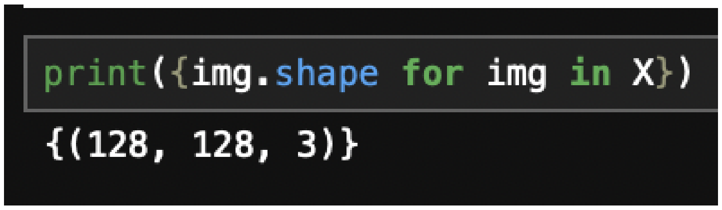
Output:



```
Final Dataset Shape: (30509, 128, 128, 3)
Labels Shape: (30509,)
Image Size Used: (128, 128)
Classes: 256
```

Reasoning:

- This cell loads all images, resizes them to **128 × 128**, and creates the final feature array X and label array y for the dataset.
- A few original images are stored before resizing to visually compare original sizes with resized images.
- The printed output confirms that the dataset contains **30,509 images**, each with a fixed shape, and **256 different classes**.
- Sample original and resized images are displayed to verify that resizing has been applied correctly without losing visual information.



```
print({img.shape for img in X})
{(128, 128, 3)}
```