Display an image and extract feature using pre-trained convolutional neural network

In [1]:
```
%reset
```
```
Once deleted, variables cannot be recovered. Proceed (y/[n])?
Nothing done.
```

In [2]:
```
from PIL import Image
import torch
import matplotlib.pyplot as plt
import numpy as np
import torch.nn.functional as F
from torch import nn, optim
from torchvision import datasets, transforms
import torchvision.models as models
import os
import math
from time import time

os.chdir(r'/Users/jeremywan/Desktop/MMTech/lab1_cbir_student')
imgpath = r'./images'

from ai_pytorch_module import *
device= torch.device("cpu")
```

In [3]:
```
#%% Load model used to extract feature
model = models.vgg16(pretrained=True)
model.classifier = model.classifier[:4] # until linear (3)before relu (4)
model.to(device)
print(model)
```
```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
  )
)
```

In [4]:
```
pwd
```

Out[4]:
```
'/Users/jeremywan/Desktop/MMTech/lab1_cbir_student'
```

In [5]:
```
#%% Extract the feature from pre-trained convolutional neural network CNN

filename = '002.jpg'
start = time()
imgpath = os.path.join(imgpath, filename)
imgFeature = getCNNFeature(model, filename,  device, showImage = False,)
end = time()
print("The feature dimension = ", imgFeature.shape)
print(' Time taken to extract feature from one image (sec) = ", end - start) # 1.3 sec

im = Image.open(filename,)
plt.figure(figsize=(5,5))
plt.imshow(im) ; plt.axis('off')
titleStr = " Image {}".format(filename)
plt.title(titleStr,  fontsize=20)

# Extract the feature from colour histogram
# Your code for getColorHist(filename)

def getColorHist(filename):
#Extract the feature from pre-trained convolutional neural network CNN <br>

    im = np.array( Image.open(filename) )

    # create the histogram plot, with three lines, one for
    # each color
    colors = ("r", "g", "b")
    channel_ids = (0, 1, 2)
    plt.figure()
    plt.xlim([0, 256])

    hists = []
    for channel_id, c in zip(channel_ids, colors):
        histogram, bin_edges = np.histogram(
            im[:, :, channel_id], bins=256, range=(0, 256)
        )

        plt.plot(bin_edges[0:-1], histogram, color=c)
        histogram_n = histogram / np.sum(histogram)
        hists.extend(histogram_n)

    hist_feat = np.array(hists)  # hist is the histogram vector that represent the image im

    plt.xlabel("Color value")
    plt.ylabel("Pixels")
    plt.title(filename)
    plt.show()

    return hist_feat
```
```
/Users/jeremywan/opt/anaconda3/lib/python3.8/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental feature and subject to change. Please do not use them for anything important until they are released as stable. (Triggered internally at  /Users/distiller/project/conda/conda-bld/pytorch_1623459044885/work/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
The feature dimension =  (1, 4096)
 Time taken to extract feature from one image (sec) =  1.0924656906893331
```

Question 3:

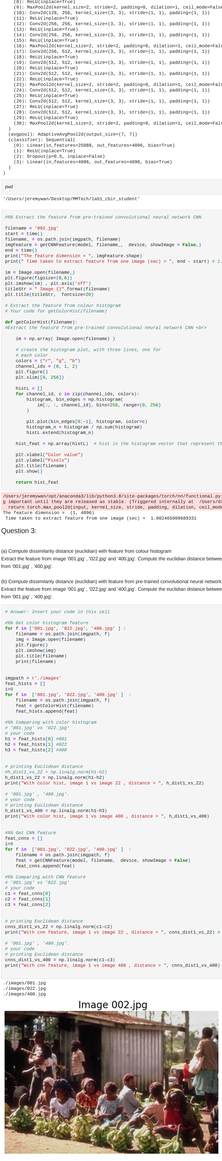
(a) Compute dissimilarity distance (euclidian) with feature from colour histogram

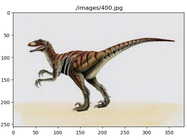
Extract the feature from image '001.jpg' , '022.jpg' and '400.jpg'. Compute the euclidian distance between the two feature vector. Compute the euclidian distance between the two feature vector from '001.jpg' , '022.jpg'. Then Compute the euclidian distance between the two feature vector from '001.jpg' , '400.jpg'.

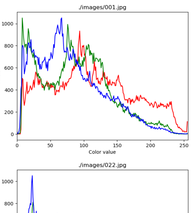
(b) Compute dissimilarity distance (euclidian) with feature from pre-trained convolutional neural network CNN

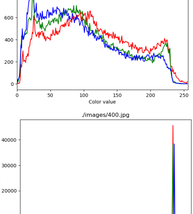
Extract the feature from image '001.jpg' , '022.jpg' and '400.jpg'. Compute the euclidian distance between the two feature vector. Compute the euclidian distance between the two feature vector from '001.jpg' , '022.jpg'. Then Compute the euclidian distance between the two feature vector from '001.jpg' , '400.jpg'.

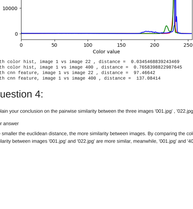
In [6]:
```
# Answer: Insert your code in this cell

#%% Get color histogram feature
for f in ['001.jpg', '022.jpg', '400.jpg' ] :
    filename = os.path.join(imgpath, f)
    img = Image.open(filename)
    img = np.array(img)
    plt.imshow(img)
    plt.title(filename)
    print(filename)


imgpath = r'./images'
feat_hists = []
ind
for f in ['001.jpg', '022.jpg', '400.jpg' ] :
    filename = os.path.join(imgpath, f)
    feat = getColorHist(filename)
    feat_hists.append(feat)

#%% Comparing with color histogram
# '001.jpg' vs '022.jpg'
# your code
h1 = feat_hists[0] #001
h2 = feat_hists[1] #022
h3 = feat_hists[2] #400


# printing Euclidean distance
h_dist1_vs_22 = np.linalg.norm(h1-h2)
h_dist1_vs_22 = np.linalg.norm(h1-h2)
print("With color hist, image 1 vs image 22 , distance = ", h_dist1_vs_22)

# '001.jpg' , '400.jpg'.
# your code
# printing Euclidean distance
h_dist1_vs_400 = np.linalg.norm(h1-h3)
print("With color hist, image 1 vs image 400 , distance = ", h_dist1_vs_400)


#%% Get CNN feature
feat_cnns = []
ind
for f in ['001.jpg', '022.jpg', '400.jpg' ] :
    filename = os.path.join(imgpath, f)
    feat = getCNNFeature(model, filename,  device, showImage = False)
    feat_cnns.append(feat)

#%% Comparing with CNN feature
# '001.jpg' vs '022.jpg'
# your code
c1 = feat_cnns[0]
c2 = feat_cnns[1]
c3 = feat_cnns[2]


# printing Euclidean distance
cnns_dist1_vs_22 = np.linalg.norm(c1-c2)
print("With cnn feature, image 1 vs image 22 , distance = ", cnns_dist1_vs_22) #

# '001.jpg' , '400.jpg'.
# your code
# printing Euclidean distance
cnns_dist1_vs_400 = np.linalg.norm(c1-c3)
print("With cnn feature, image 1 vs image 400 , distance = ", cnns_dist1_vs_400) #
```
```
./images/001.jpg
./images/022.jpg
./images/400.jpg
```












```
With color hist, image 1 vs image 22 , distance =  0.034546883934434469
With color hist, image 1 vs image 400 , distance =  0.7658288822987645
With cnn feature, image 1 vs image 22 , distance =  57.46564
With cnn feature, image 1 vs image 400 , distance =  137.08414
```

Question 4:

Explain your conclusion on the pairwise similarity between the three images '001.jpg' , '022.jpg' and '400.jpg' . Comment on the semantic similarity between the three images and the euclidian distance among their feature vectors. Compare colour histogram and CNN feature.

Your answer

The smaller the euclidean distance, the more similarity between images. By comparing the color histogram and CNN feature, the euclidean distance among their feature vector between '001.jpg' and '022.jpg' are smaller, however, between '001.jpg' and '400.jpg' are bigger. Hence, the similarity between images '001.jpg' and '022.jpg' are more similar, meanwhile, '001.jpg' and '400jpg' are less similar.