Лабораторная работа №6 “Кластеризация”

Долматович Алина, 858641

In [1]:
```
from scipy.io import loadmat
import matplotlib.pyplot as pyplot
from scipy.spatial.distance import cdist
import numpy as np
from scipy import misc
from scipy.cluster import hierarchy
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import fcluster
```

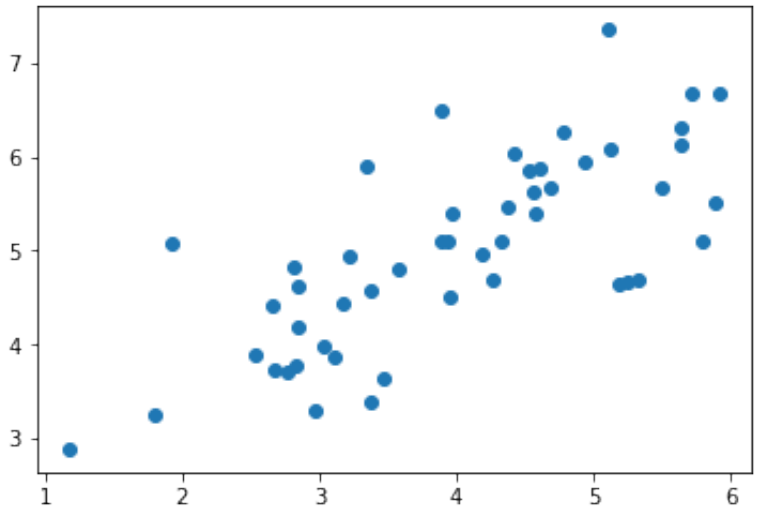
Загрузите данные ex6data1.mat из файла.

In [2]:
```
data = loadmat('ex6data1.mat')

x = data['X']
print(x.shape)
```

```
(50, 2)
```

In [3]:
```
pyplot.scatter(x[:, 0], x[:, 1])
pyplot.show()
```



Реализуйте функцию случайной инициализации K центров кластеров.

In [4]:
```
def initialize(k=3, n=2, low=3, hight=4):
    return np.random.uniform(low=low, high=hight, size=(k,n))

centroids = initialize()
print(centroids)
```

```
[[ 3.46934373  3.50080345]
 [ 3.34081346  3.80715562]
 [ 3.27849946  3.15128936]]
```

Реализуйте функцию определения принадлежности к кластерам.

In [5]:
```
def clusterMemberships(x, centroids):
    distances = cdist(x, centroids)
    memberships = distances.argmin(axis=1)
    return memberships

memberships = clusterMemberships(x, centroids)
print(memberships)
```

```
[0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 1 1 1 1 1 2 1 2 1 1 1]
```

Реализуйте функцию пересчета центров кластеров.

In [6]:
```
def recalculateCenters(x, centroids, memberships, k=3):
    centroids = centroids.copy()
    for index in range(k):
        centroids[index, :] = np.mean(x[memberships == index, :], a

    return centroids

centroids = recalculateCenters(x, centroids, memberships)
print(centroids)
```

```
[[ 3.428684    3.51098061]
 [ 4.14810562  5.19315195]
 [ 1.98037302  3.14216042]]
```

Реализуйте алгоритм K-средних.

In [7]:
```
def kMeans(x, centroids, k, n=2):
    centroidsHistory = []
    centroidsHistory.append(centroids)

    iterations = 3

    for i in range(iterations):
        memberships = clusterMemberships(x, centroids)
        centroids = recalculateCenters(x, centroids, memberships, 
        centroidsHistory.append(centroids)

    return centroidsHistory

centroids = initialize(3, 2)
centroidsHistory = kMeans(x, centroids, 3)
print(centroidsHistory)
```

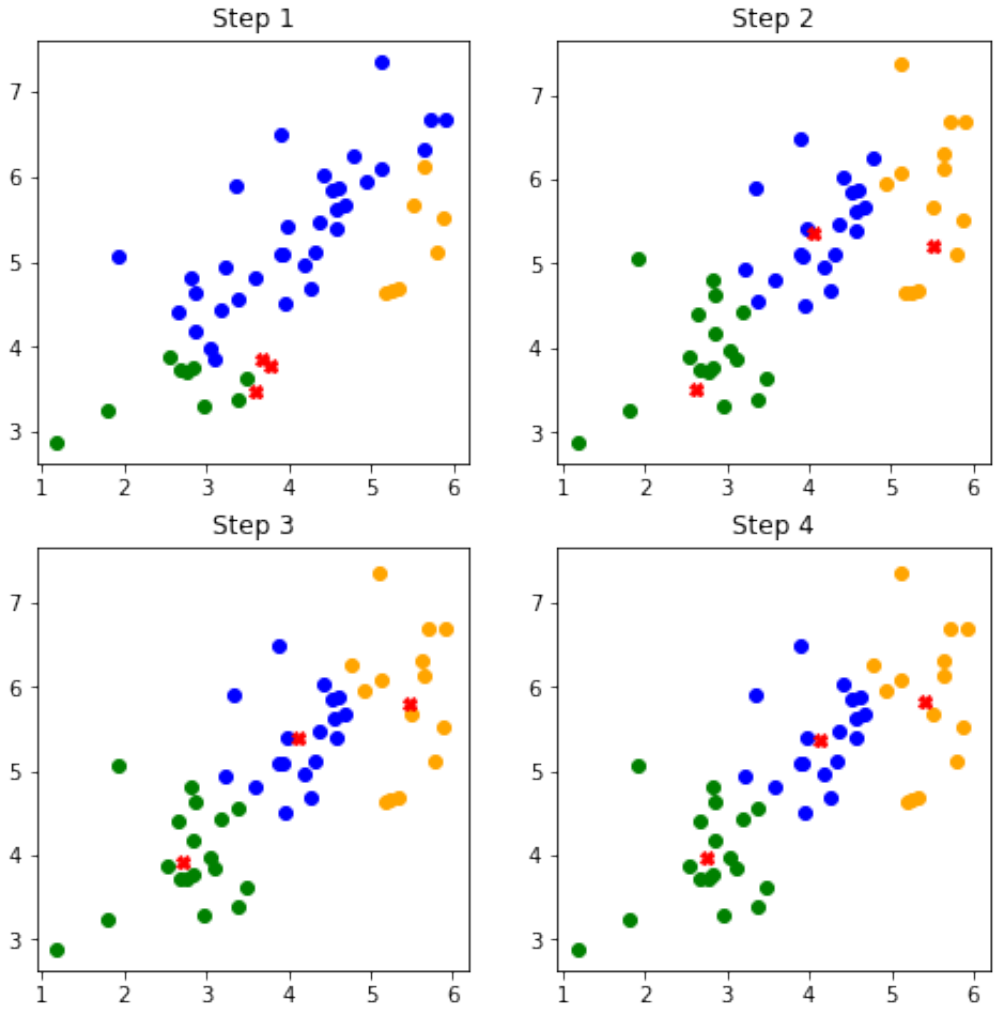
```
[array([[ 3.57640242,  3.48462973],
       [ 3.67674903,  3.87124636],
       [ 3.77520516,  3.79314668]]), array([[ 2.62353611,  3.5052595
1],
       [ 4.03767443,  5.35807228],
       [ 5.51007175,  5.20263767]]), array([[ 2.70689034,  3.9367106
],
       [ 4.12320662,  5.38602638],
       [ 5.46015406,  5.80736017]]), array([[ 2.74403119,  3.9716365
3],
       [ 4.1284461 ,  5.38337803],
       [ 5.41133404,  5.83928776]])]
```

Постройте график, на котором данные разделены на K=3 кластеров (при помощи различных маркеров или цветов), а также траекторию движения центров кластеров в процессе работы алгоритма

In [8]:
```
def plotCentroidsHistory(centroidsHistory, k=3):
    pyplot.figure(figsize=(8, 8))
    for i in range(len(centroidsHistory)):
        memberships = clusterMemberships(x, centroidsHistory[i])
        pyplot.subplot(2, 2, i + 1)
        colors = ["green", "blue", "orange"]
        for kIndex, color in zip(range(k), colors):
            pyplot.scatter(x[memberships == kIndex, 0], x[membersh
        pyplot.plot(centroidsHistory[i][:, 0], centroidsHistory[i]
        pyplot.title('Step {:}'.format(i + 1));

    pyplot.show()

plotCentroidsHistory(centroidsHistory)
```



Загрузите данные bird_small.mat из файла.

In [9]:
```
1 birdData = loadmat('bird_small.mat')
2
3 aOriginal = birdData['A']
4 print(aOriginal.shape)
```

```
(128, 128, 3)
```

С помощью алгоритма K-средних используйте 16 цветов для кодирования пикселей.

In [10]:

```
def distance(x1, y1, x2, y2):
    dist = np.square(x1 - x2) + np.square(y1 - y2)
    dist = np.sqrt(dist)
    return dist

def kMeansFor(image, clusters):
    iterations = 10
    points = np.reshape(image, (image.shape[0] * image.shape[1], i
    m, n = points.shape
    index = np.zeros(m)

    means = initialize(k=clusters, n=image.shape[2], low=0, hight=

    while(iterations > 0):
        for j in range(len(points)):
            minv = 1000
            temp = None

            for k in range(clusters):
                x1 = points[j, 0]
                y1 = points[j, 1]
                x2 = means[k, 0]
                y2 = means[k, 1]

                if(distance(x1, y1, x2, y2) < minv):
                    minv = distance(x1, y1, x2, y2)
                    temp = k
                    index[j] = k

        for k in range(clusters):
            sumx = 0
            sumy = 0
            count = 0

            for j in range(len(points)):
                if(index[j] == k):
                    sumx += points[j, 0]
                    sumy += points[j, 1]
                    count += 1

            if(count == 0):
                count = 1

            means[k, 0] = float(sumx / count)
            means[k, 1] = float(sumy / count)

        iterations -= 1

    return means, index
```

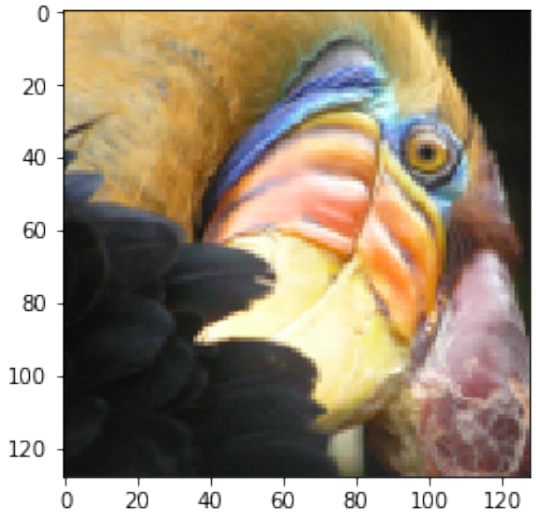
Насколько уменьшился размер изображения? Как это сказалось на качестве?

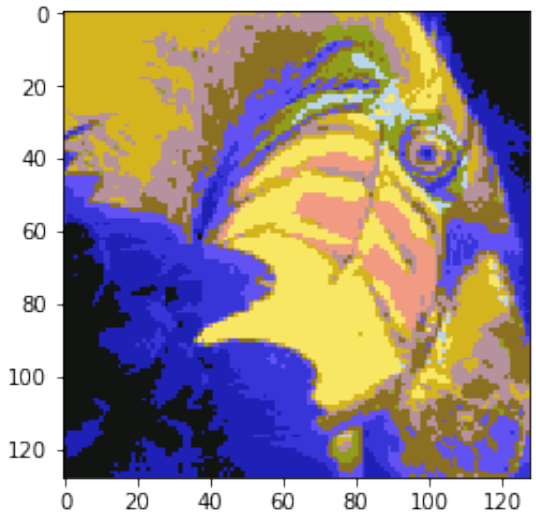
In [11]:
```
def compress_image(means, index, size):
    centroid = np.array(means)
    recovered = centroid[index.astype(int), :]
    recovered = np.reshape(recovered, (128, 128, 3))
    pyplot.imshow(recovered)
    pyplot.show()

k = 16
image = aOriginal / 255.
means, index = kMeansFor(image, k)
```

In [12]:
```
pyplot.imshow(aOriginal)
pyplot.show()

compress_image(means, index, image)
```



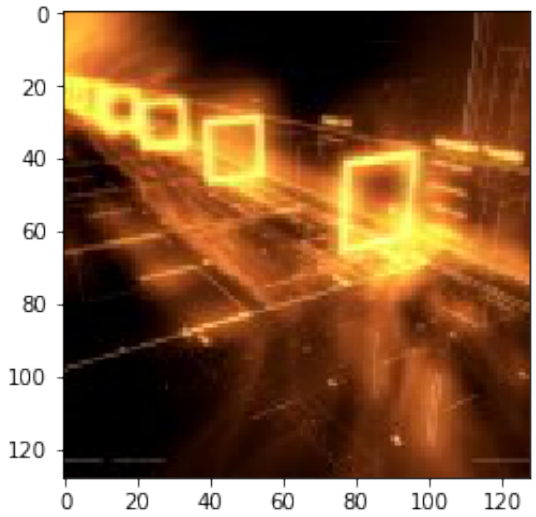


Реализуйте алгоритм K-средних на другом изображении.

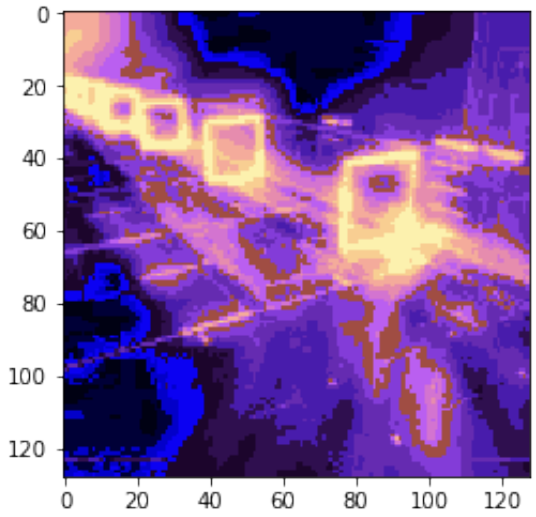
In [13]:
```
testImage = misc.imread('testImage.jpg')
pyplot.imshow(testImage)
pyplot.show()

testImage = testImage / 255.
print(testImage.shape)

means, index = kMeansFor(testImage, k)
compress_image(means, index, testImage)
```




```
(128, 128, 3)
```



Реализуйте алгоритм иерархической кластеризации на том же изображении. Сравните полученные результаты.

In [14]:
```
points = np.reshape(testImage, (testImage.shape[0] * testImage.sha
distance_mat = pdist(points)
Z = hierarchy.linkage(distance_mat, 'single')
max_d = .1
while max_d > 0.005:
    max_d *= .5
    clusters = fcluster(Z, max_d, criterion='distance')
    meshx, meshy = np.meshgrid(np.arange(128), np.arange(128))
pyplot.axis('equal')
pyplot.axis('off')
pyplot.scatter(meshx, -(meshy - 128), c=clusters.reshape(128, 128)
pyplot.show()
```