In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
from sklearn import neighbors
from PIL import Image
```

1. Create a list with the names called images

In [2]:
```
images= ['farm1.jpg', 'farm2.jpg', 'farm3.jpg', 'farm4.jpg',
 'farm5.jpg', 'farm6.jpg', 'farm7.jpg', 'farm8.jpg',
 'city1.jpg', 'city2.jpg', 'city3.jpg', 'city4.jpg',
 'city5.jpg', 'city6.jpg', 'city7.jpg', 'city8.jpg',
 'desert1.jpg', 'desert2.jpg', 'desert3.jpg', 'desert4.jpg',
 'desert5.jpg', 'desert6.jpg', 'desert7.jpg', 'desert8.jpg']
```

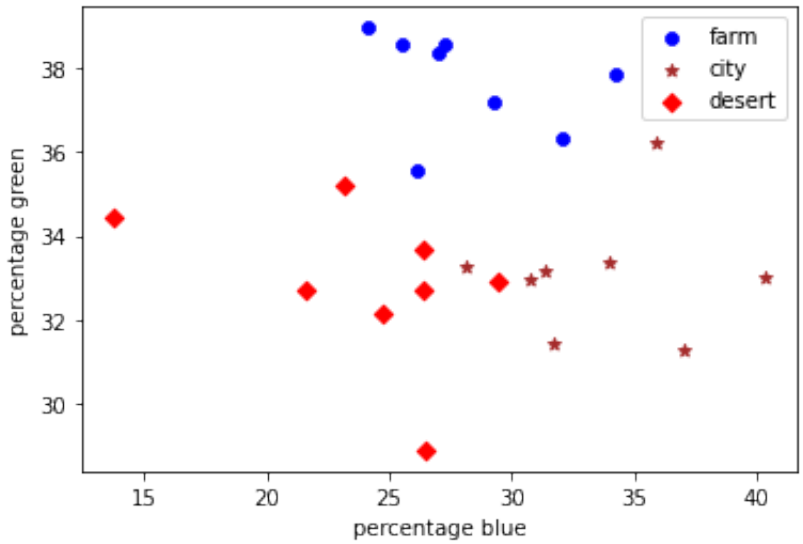
In [3]:
```
def percentage_green(image_f):
    red = 0
    green = 0
    blue = 0

    img = Image.open(image_f)
    pixels = img.load()
    width, height = img.size
    for x in range(width):
        for y in range(height):
            rgb = pixels[x, y]
            red += rgb[0]
            green += rgb[1]
            blue += rgb[2]

    percent = green / (red + blue + green)
    return percent * 100


def percentage_red(image_f):
    red = 0
    green = 0
    blue = 0

    img = Image.open(image_f)
    pixels = img.load()
    width, height = img.size
    for x in range(width):
        for y in range(height):
```

```
            rgb = pixels[x, y]
            red += rgb[0]
            green += rgb[1]
            blue += rgb[2]

    percent = red / (red + blue + green)
    return percent * 100


def percentage_blue(image_f):
    red = 0
    green = 0
    blue = 0

    img = Image.open(image_f)
    pixels = img.load()
    width, height = img.size
    for x in range(width):
        for y in range(height):
            rgb = pixels[x, y]
            red += rgb[0]
            green += rgb[1]
            blue += rgb[2]

    percent = blue / (red + blue + green)
    return percent * 100
```

In [4]:
```
perB_farm=[]
perG_farm=[]

for i in range (0,8):
    perB_farm.append(percentage_blue(images[i]))
for j in range(0,8):
    perG_farm.append(percentage_green(images[j]))


perB_city=[]
perG_city=[]

for i in range (8,16):
    perB_city.append(percentage_blue(images[i]))
for j in range(8,16):
    perG_city.append(percentage_green(images[j]))

perB_desert=[]
perG_desert=[]

for i in range (16,24):
    perB_desert.append(percentage_blue(images[i]))
for j in range(16,24):
    perG_desert.append(percentage_green(images[j]))
```

2. Create the scatter plot in the first page

In [19]:
```
plt.scatter(perB_farm,perG_farm,marker='8',color='blue')
plt.scatter(perB_city,perG_city,marker='*',color='brown')
plt.scatter(perB_desert,perG_desert,marker='D',color='red')
plt.xlabel('percentage blue')
plt.ylabel('percentage green')
plt.legend(['farm','city','desert'])
plt.show()
```



3.Now create an array of strings called training_target with the category of each

In [6]:
```
training_target=['farm', 'farm', 'farm', 'farm',
 'farm', 'farm', 'farm', 'farm',
 'city', 'city', 'city', 'city',
 'city', 'city', 'city', 'city',
 'desert', 'desert', 'desert', 'desert',
 'desert', 'desert', 'desert', 'desert']
```

4. Create an empty array of zeros called training_data that will eventually store the percent green and percent blue values

In [7]:
```
training_data=np.zeros((24,2))
training_data
```

Out[7]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

5. Now fill the training_data array with the proper values for each image, and observe the values in the array after it is finished.

In [8]:
```
for i in range(0,8):
    training_data[i,0]=perB_farm[i]
    training_data[8+i,0]=perB_city[i]
    training_data[16+i,0]=perB_desert[i]
    training_data[i,1]=perG_farm[i]
    training_data[8+i,1]=perG_city[i]
    training_data[16+i,1]=perG_desert[i]

print(training_data)
```

```
[[27.25025827 38.53791621]
 [24.16674958 38.94787652]
 [29.23692974 37.1767491 ]
 [25.56727404 38.53494059]
 [26.97444869 38.36885428]
 [34.24372371 37.82235141]
 [26.13897337 35.57784135]
 [32.07925149 36.3182636 ]
 [33.98700751 33.3846793 ]
 [31.74095454 31.45798947]
 [30.76109723 32.98215922]
 [40.32948264 33.02142216]
 [37.06804694 31.26774453]
 [35.92237167 36.20055   ]
 [28.1224145  33.26393075]
 [31.3874935  33.15564785]
 [26.47862205 28.89915366]
 [29.46128832 32.88746498]
 [24.74994409 32.17135111]
 [23.17126104 35.20926067]
 [21.56491053 32.71851263]
 [26.3871903  33.655681  ]
 [13.74953806 34.41919206]
 [26.43832828 32.73203919]]
```

6. Create your classifier.

In [9]:
```
knnClsfier= neighbors.KNeighborsClassifier(n_neighbors = 1, weights='d
```

7. Train your classifier.

In [10]:
```
knnClsfier.fit(training_data, training_target)
```

Out[10]:
```
KNeighborsClassifier(n_neighbors=1, weights='distance')
```

8. Now create an empty test_data array and fill it with the proper values for each test image, and observe the filled array and consider if it matches your expectations based on your observations of the images.

In [11]:
```
test1 = mpimg.imread('test1.jpg')
test2 = mpimg.imread('test2.jpg')
test3 = mpimg.imread('test3.jpg')

test = ['test1.jpg','test2.jpg','test3.jpg']
```

In [12]:
```
perB_t=[]
perG_t=[]

for i in range(3):
    perB_t.append(percentage_blue(test[i]))

for j in range(3):
    perG_t.append(percentage_green(test[j]))

print(perB_t)
print(perG_t)
```

```
[32.68851262195992, 17.936788871306227, 24.578861396084875]
[32.69592008303714, 33.42938446981946, 35.00400801777032]
```

In [13]:
```
test_d = np.zeros((3,2))

for i in range(3):
    test_d[i,0]=perB_t[i]
    test_d[i,1]=perG_t[i]

print(test_d)
```

```
[[32.68851262 32.69592008]
 [17.93678887 33.42938447]
 [24.5788614  35.00400802]]
```

9. Predict the class of the test images.

In [14]:
```
predict = knnClsfier.predict(test_d)
predict
```

Out[14]:
```
array(['city', 'desert', 'desert'], dtype='<U6')
```

In [20]:
```
print('Pridicted:',predict)
```

```
Pridicted: ['city' 'desert' 'desert']
```

In [16]:
```
print('Actual:[city,desert,farm]')
```

```
Actual:[city,desert,farm]
```

In [17]:
```
fig, axes = plt.subplots(figsize=(10,7),nrows=1,ncols=3)

plt.subplot(1,3,1)
plt.imshow(test1)
plt.axis('off')
plt.title(predict[0])

plt.subplot(1,3,2)
plt.imshow(test2)
plt.axis('off')
plt.title(predict[1])

plt.subplot(1,3,3)
plt.imshow(test3)
plt.axis('off')
plt.title(predict[2])
```

Out[17]:
```
Text(0.5, 1.0, 'desert')
```

Prediction results: first image city and second desert had predicted correctly but test3.jpg is farm, our model predicted as dessert ,this is an error