1. Creating a list with the names called image_files

In [3]:

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from sklearn import neighbors

image_files = ['farm1.jpg', 'farm2.jpg', 'farm3.jpg', 'farm4.jpg', 'farm5.jpg', '
 'city1.jpg', 'city2.jpg', 'city3.jpg', 'city4.jpg', 'city5.jpg', 'city6.jpg', 'c
 'desert1.jpg', 'desert2.jpg', 'desert3.jpg', 'desert4.jpg', 'desert5.jpg', 'dese
print("Creating a list with the names called image_files:")

for x in image_files:
    print('\n',x)
```

```
Creating a list with the names called image_files:

 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
```

2. Create the scatter plot in the first page

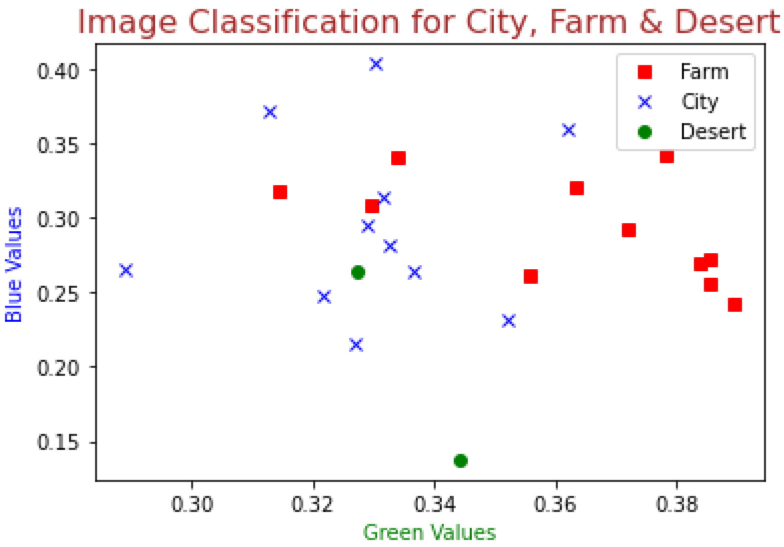
In [4]:

```
# percentage_of_BlueGreen returns % of Green and Blue of an Image

def percentage_of_BlueGreen(image):

    BlueGreen = np.array(image).mean(axis=(0,1))
    R = BlueGreen[0]
    G = BlueGreen[1]
    B = BlueGreen[2]

    Sum = BlueGreen[0] + BlueGreen[1] + BlueGreen[2]

    percentage_of_Green = BlueGreen[1]/Sum
    percentage_of_Blue = BlueGreen[2]/Sum

    return percentage_of_Green, percentage_of_Blue
```

In [5]:

```
#scatter plot

PercentageBlueGreen = []

# getting percent of green and percent blue.
for x in image_files:

    image = Image.open('images2/' + x)
    PercentageBlueGreen.append(percentage_of_BlueGreen(image))


#Green
Green = [x for x, y in PercentageBlueGreen]
GreenArray = np.array(Green)
print("Percentage values of Green ")
print(GreenArray)


#Blue values
Blue = [y for x, y in PercentageBlueGreen]
BlueArray = np.array(Blue)
print("Percentage values of Blue")
print(BlueArray)

from matplotlib.pyplot import *

%matplotlib inline


plot(GreenArray[0:11],BlueArray[0:11],'rs',label='Farm')

plot(GreenArray[11:22],BlueArray[11:22],'bx',label='City')

plot(GreenArray[22:44],BlueArray[22:44],'go',label='Desert')

xlabel('Green Values',fontsize=10, color = 'green')
ylabel('Blue Values',fontsize=10, color = 'blue')
title('Image Classification for City, Farm & Desert',fontsize=16, color = 'brown'

legend(loc='best')
show()
```

```
Percentage values of Green
[0.38537916 0.38947877 0.37176749 0.38534941 0.38368854 0.37822351
 0.35577841 0.36318264 0.33384679 0.31457989 0.32982159 0.33021422
 0.31267745 0.3620055  0.33263931 0.33155648 0.28899154 0.32887465
 0.32171351 0.35209261 0.32718513 0.33655681 0.34419192 0.32732039]
Percentage values of Blue
[0.27250258 0.2416675  0.2923693  0.25567274 0.26974449 0.34243724
 0.26138973 0.32079251 0.33987008 0.31740955 0.30761097 0.40329483
 0.37068047 0.35922372 0.28122414 0.31387494 0.26478622 0.29461288
 0.24749944 0.23171261 0.21564911 0.2638719  0.13749538 0.26438328]
```



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

```
In [6]: training_target = np.array(['farm', 'farm', 'farm', 'farm', 'farm', 'farm', 'farm
        print("image_files list: \n")
        for x in training_target:
            print('\n',x)
```

```
image_files list: 


 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 [8]:

```
training_data = []
print('Empty array of zeros called training_data:',training_data)
```

```
Empty array of zeros called training_data: []
```

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 [9]:
```
for x in image_files:
    image = Image.open('images2/' + x)
    training_data.append(percentage_of_BlueGreen(image))


print("Percentage values of Green and Blue colors filled in the training_data arr

for x in training_data:
    print(x)
```

```
Percentage values of Green and Blue colors filled in the training_data array:
(0.38537916213835416, 0.2725025827290944)
(0.38947876516901914, 0.24166749580794727)
(0.37176749098686257, 0.29236929740095713)
(0.3853494059331435, 0.25567274038089727)
(0.3836885427597768, 0.2697444869452292)
(0.3782235141367888, 0.3424372370985558)
(0.3557784135089085, 0.2613897337397366)
(0.36318263603850426, 0.3207925148928169)
(0.3338467930412881, 0.33987007505544775)
(0.3145798947161084, 0.31740954537386984)
(0.32982159222616164, 0.30761097231014695)
(0.3302142216023482, 0.4032948263728943)
(0.3126774452579913, 0.3706804693524618)
(0.36200550003320575, 0.3592237167477091)
(0.3326393074627567, 0.2812241449923416)
(0.33155647847549335, 0.3138749350290284)
(0.2889915365854203, 0.2647862205478914)
(0.3288746497784961, 0.2946128831876114)
(0.32171351112006713, 0.24749944089149414)
(0.3520926067264411, 0.2317126103798501)
(0.32718512631637453, 0.2156491053354232)
(0.33655681001293364, 0.26387190302313 27)
(0.344191920645267 6, 0.1374953806468185)
(0.32732039192104917, 0.26438328280357887)
```

6. Create your classifier.

In [10]:
```
k = neighbors.KNeighborsClassifier(1,weights='distance')
print('Created classifier:',k)
```

```
Created classifier: KNeighborsClassifier(n_neighbors=1, weights='distance')
```

7. Train your classifier.

In [11]:

```
trainingArray = np.column_stack((GreenArray,BlueArray))
print('Training Array:',trainingArray)

print('\nTraining classifier:')
k.fit(trainingArray, training_target)
```

```
Training Array: [[0.38537916 0.27250258]
 [0.38947877 0.2416675 ]
 [0.37176749 0.2923693 ]
 [0.38534941 0.25567274]
 [0.38368854 0.26974449]
 [0.37822351 0.34243724]
 [0.35577841 0.26138973]
 [0.36318264 0.32079251]
 [0.33384679 0.33987008]
 [0.31457989 0.31740955]
 [0.32982159 0.30761097]
 [0.33021422 0.40329483]
 [0.31267745 0.37068047]
 [0.3620055  0.35922372]
 [0.33263931 0.28122414]
 [0.33155648 0.31387494]
 [0.28899154 0.26478622]
 [0.32887465 0.29461288]
 [0.32171351 0.24749944]
 [0.35209261 0.23171261]
 [0.32718513 0.21564911]
 [0.33655681 0.2638719 ]
 [0.34419192 0.13749538]
 [0.32732039 0.26438328]]

Training classifier:
```

Out[11]:

```
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

In [13]:

```python
test_images = ['test1.jpg', 'test2.jpg', 'test3.jpg']
test_data = []
for x in test_images:
    path = ('images2/' + x)
    image = Image.open(path)
    test_data.append(percentage_of_BlueGreen(image))

print("Percentage of Green and Blue for each image in the test_data")
for x in test_data:
    print(x)

#Percentage values of Green color from Test data
Test_Percent_of_Green = [x for x, y in test_data]
Test_Percent_of_Green_Array = np.array(Test_Percent_of_Green)

#Percentage values of Blue color from Test data
Test_Percent_of_Blue = [y for x, y in test_data]
Test_Percent_of_Blue_Array = np.array(Test_Percent_of_Blue)
```

```
Percentage of Green and Blue for each image in the test_data
(0.32695920083037133, 0.3268851262195992)
(0.3342938446981946, 0.17936788871306228)
(0.35004008017770316, 0.24578861396084875)
```

9. Predict the class of the test images

In [15]:

```python
test_array = np.column_stack((Test_Percent_of_Green_Array,Test_Percent_of_Blue_Ar

print("Percentage Green and Blue in predicting the class of test images:")
print(test_array)

predict_classifier = k.predict(test_array)
```

```
Percentage Green and Blue in predicting the class of test images:
[[0.3269592  0.32688513]
 [0.33429384 0.17936789]
 [0.35004008 0.24578861]]
```

10. Print the prediction from the test images and compare with the actual images shown below. Make this comparison clear in the output of your code (e.g. prepend with ‘predicted:’ and ‘actual:’). Try to explain any errors if you note any.

In [19]:

```
print("Predicted results from the test images:")
print(predict_classifier)
print("\nActual results from the test images:")
print("['city', 'desert', 'farm']")
```

```
Predicted results from the test images:
['city' 'desert' 'desert']

Actual results from the test images:
['city', 'desert', 'farm']
```

The predicted and actual values for the first two images are correct (i.e. for the City and Desert).

But the prediction for the third image the prediction is wrong (for the farm), because the image has dry grass in brown color, just like the colors in desert images. Hence it is being considered as desert.

In [ ]:

```

```

In [ ]:

```

```