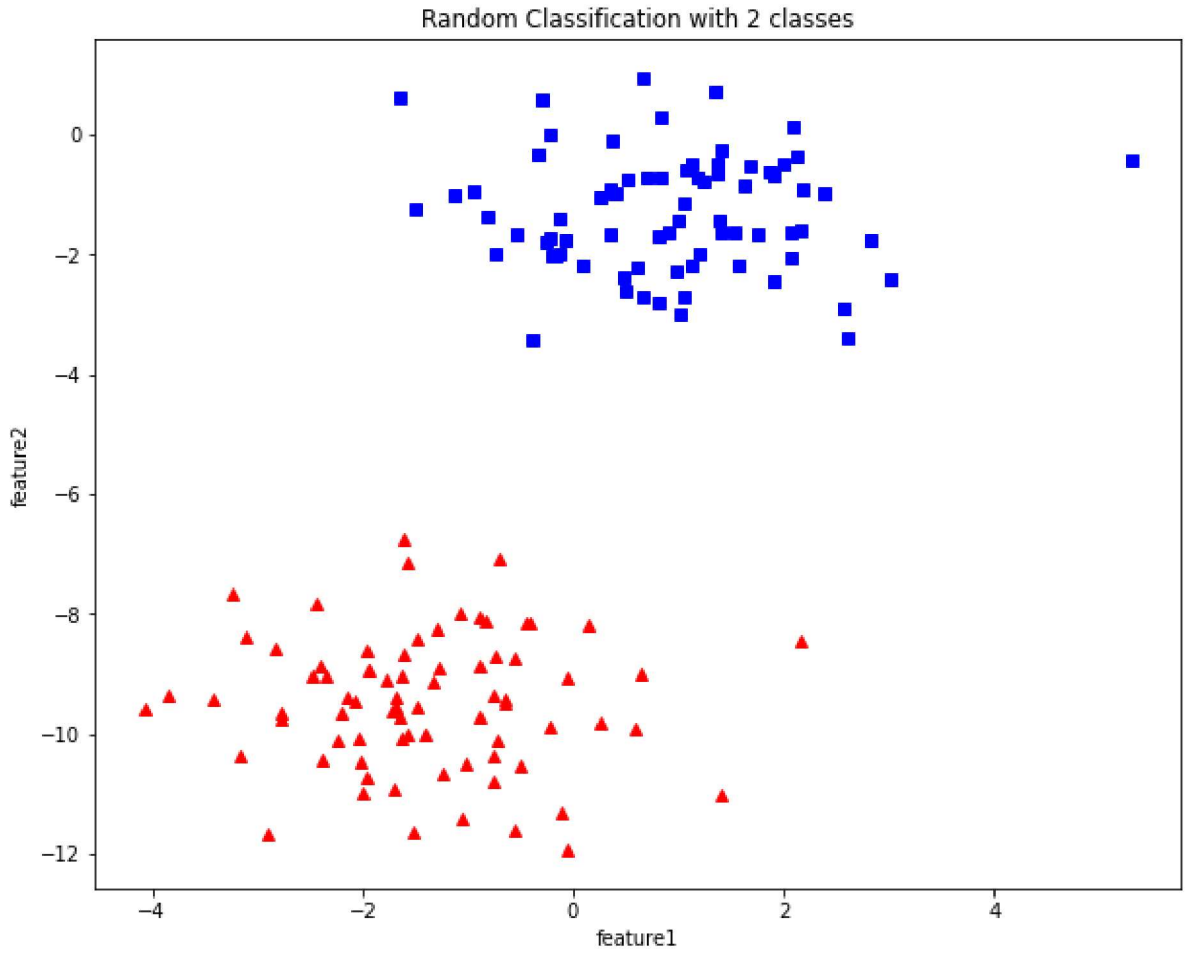
In [1]:
```
# imports
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
```

In [2]:
```
X, y = datasets.make_blobs(n_samples=150, n_features=2, centers=2, cluster_std
```

In [3]:
```
fig = plt.figure(figsize=(10,8))
plt.plot(X[:,0][y==0], X[:,1][y==0],'r^')
plt.plot(X[:,0][y==1], X[:,1][y==1],'bs')
plt.xlabel("feature1")
plt.ylabel("feature2")
plt.title("Random Classification with 2 classes")
```

Out[3]:
```
Text(0.5, 1.0, 'Random Classification with 2 classes')
```



In [4]:
```
def step_func(z):
  return 1.0 if z>0 else 0.0
```

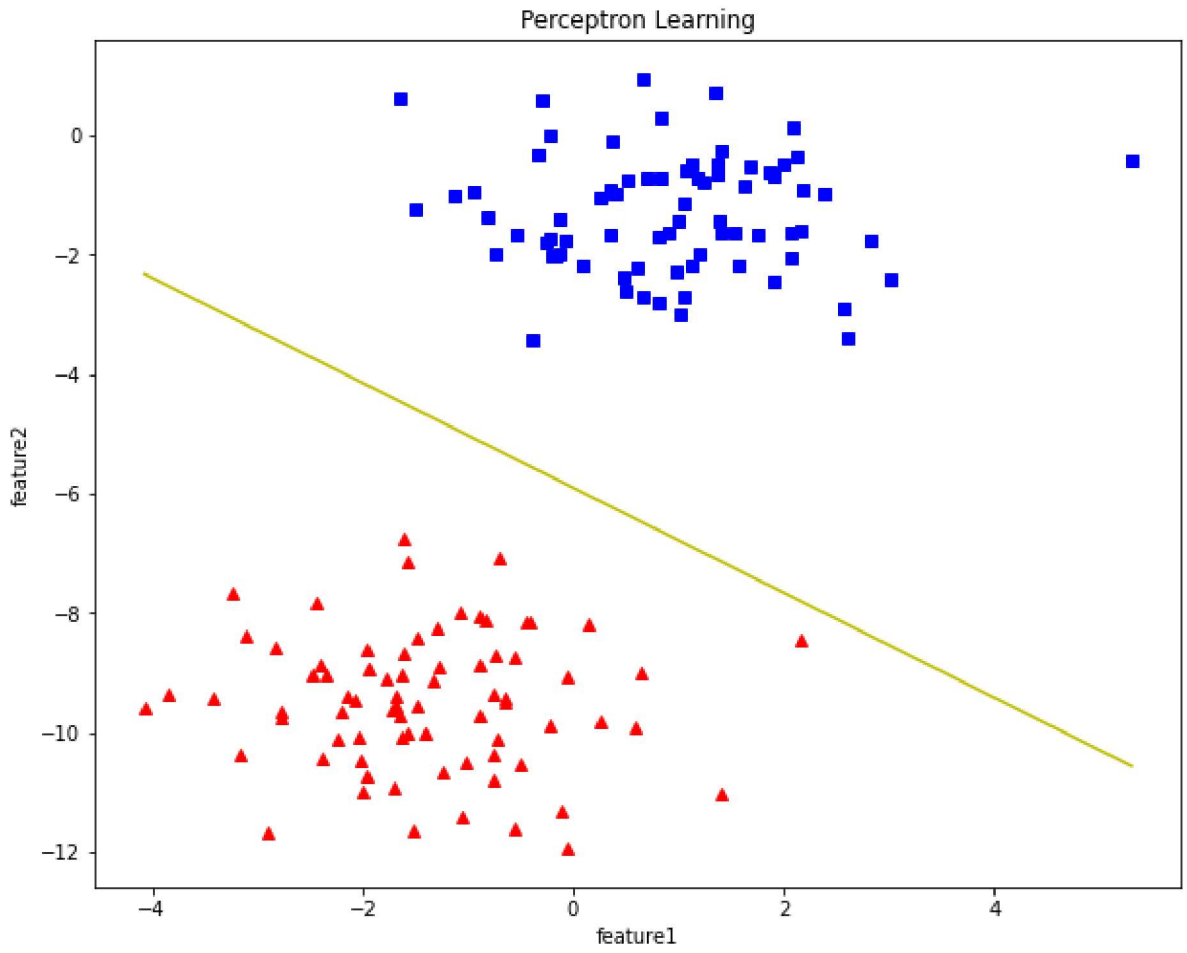
- This function (perceptron) is used to deploy perceptron learning
- the logic of weight updation rule will be implemented in the body of the function
- the data variables used will be as follows:
    1. X - matrix of input features
    2. y - column of labels/target

3. lr - learning rate
4. epochs - number of iterations
5. m - no. of training samples
6. n - no. of features
7. weights - connection weights
8. n_miss_list - array of miss classified examples

In [5]:
```
def perceptron(X, y, lr, epochs):
  m, n = X.shape
  weights = np.zeros((n+1, 1))
  n_miss_list = []
  for epoch in range(epochs):
    n_miss = 0
    for idx, x_i in enumerate(X):
      x_i = np.insert(x_i, 0, 1).reshape(-1,1)
      y_hat = step_func(np.dot(x_i.T, weights))
      if (np.squeeze(y_hat) - y[idx]) != 0 :
        weights += lr * ((y[idx] - y_hat) * x_i)
        n_miss += 1
    n_miss_list.append(n_miss)
  return weights, n_miss_list
```

In [6]:
```
def plot_decision_boundary(X, weights):
  x1 = [min(X[:,0]), max(X[:,0])]
  m = -weights[1]/ weights[2]
  c = -weights[0]/ weights[2]
  x2 = m * x1 + c
  fig = plt.figure(figsize=(10,8))
  plt.plot(X[:,0][y==0], X[:,1][y==0],'r^')
  plt.plot(X[:,0][y==1], X[:,1][y==1],'bs')
  plt.xlabel("feature1")
  plt.ylabel("feature2")
  plt.title("Perceptron Learning")
  plt.plot(x1, x2, 'y-')
```

In [7]:
```
weights, miss_l = perceptron(X, y ,0.5, 100)
plot_decision_boundary(X, weights)
```



In [ ]:

In [ ]: