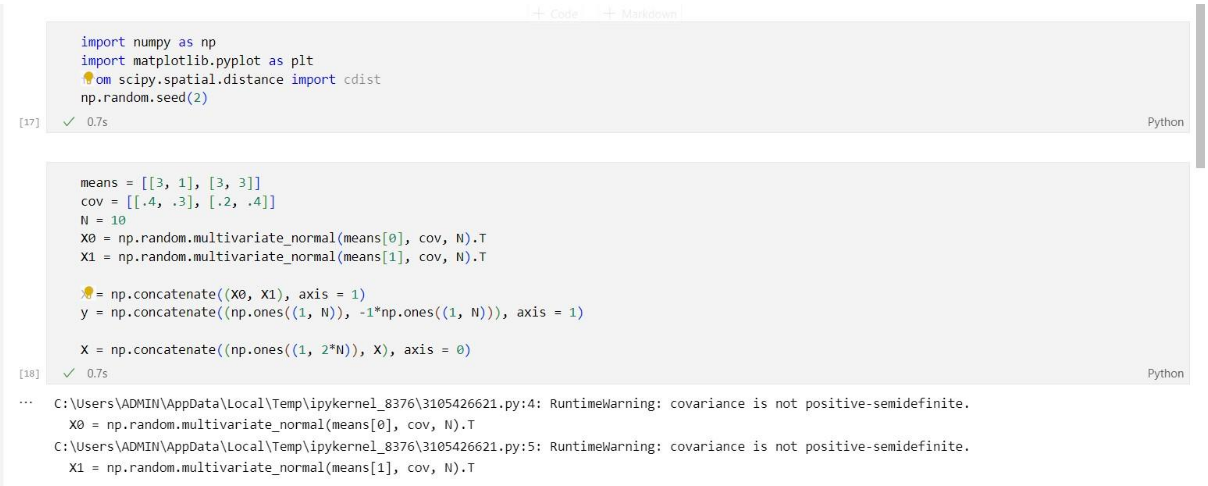
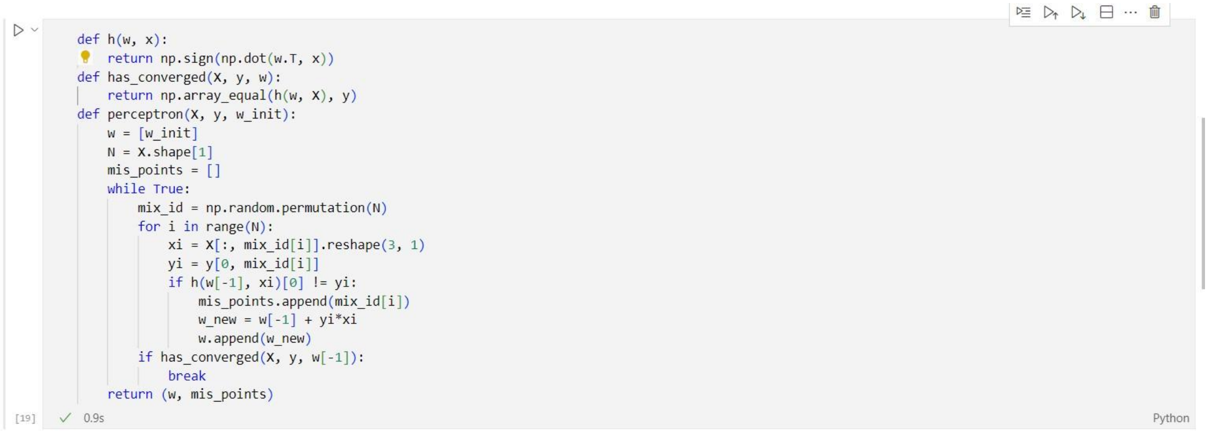


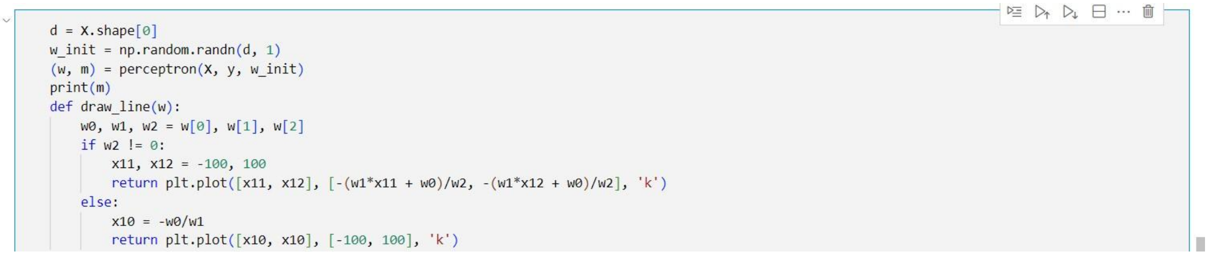
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
np.random.seed(2)
```

```python
means = [[3, 1], [3, 3]]
cov = [[.4, .3], [.2, .4]]
N = 10
X0 = np.random.multivariate_normal(means[0], cov, N).T
X1 = np.random.multivariate_normal(means[1], cov, N).T

X = np.concatenate((X0, X1), axis = 1)
y = np.concatenate((np.ones((1, N)), -1*np.ones((1, N))), axis = 1)

X = np.concatenate((np.ones((1, 2*N)), X), axis = 0)
```

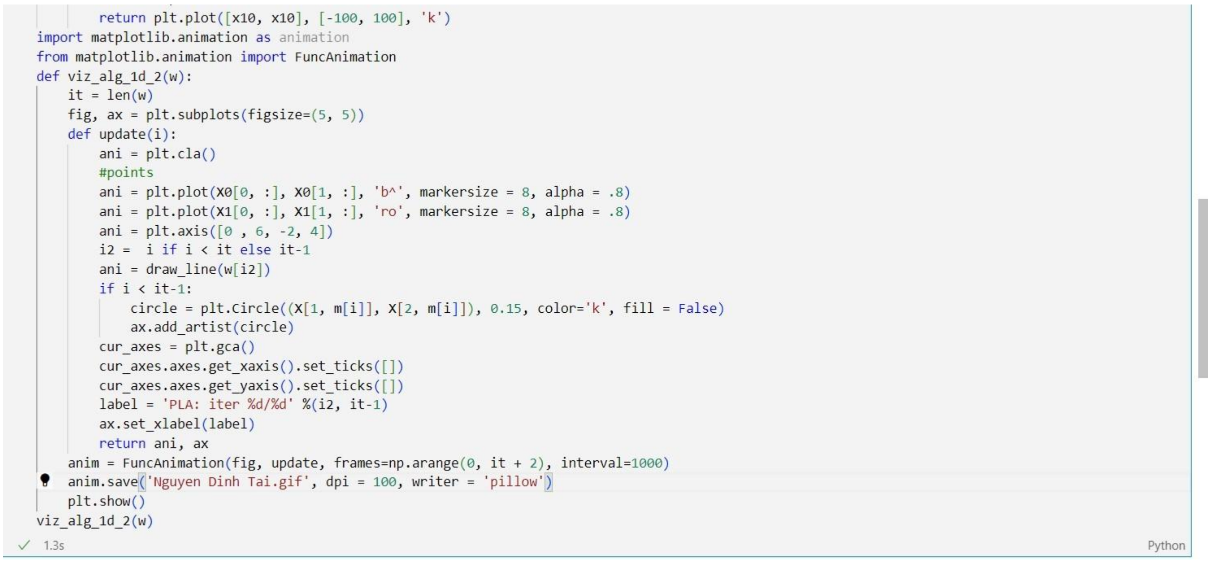
```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_8376\3105426621.py:4: RuntimeWarning: covariance is not positive-semidefinite.
  X0 = np.random.multivariate_normal(means[0], cov, N).T
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_8376\3105426621.py:5: RuntimeWarning: covariance is not positive-semidefinite.
  X1 = np.random.multivariate_normal(means[1], cov, N).T
```



```python
def h(w, x):
    return np.sign(np.dot(w.T, x))
def has_converged(X, y, w):
    return np.array_equal(h(w, X), y)
def perceptron(X, y, w_init):
    w = [w_init]
    N = X.shape[1]
    mis_points = []
    while True:
        mix_id = np.random.permutation(N)
        for i in range(N):
            xi = X[:, mix_id[i]].reshape(3, 1)
            yi = y[0, mix_id[i]]
            if h(w[-1], xi)[0] != yi:
                mis_points.append(mix_id[i])
                w_new = w[-1] + yi*xi
                w.append(w_new)
        if has_converged(X, y, w[-1]):
            break
    return (w, mis_points)
```



```python
d = X.shape[0]
w_init = np.random.randn(d, 1)
(w, m) = perceptron(X, y, w_init)
print(m)
def draw_line(w):
    w0, w1, w2 = w[0], w[1], w[2]
    if w2 != 0:
        x11, x12 = -100, 100
        return plt.plot([x11, x12], [-(w1*x11 + w0)/w2, -(w1*x12 + w0)/w2], 'k')
    else:
        x10 = -w0/w1
        return plt.plot([x10, x10], [-100, 100], 'k')
```



```python
        return plt.plot([x10, x10], [-100, 100], 'k')
import matplotlib.animation as animation
from matplotlib.animation import FuncAnimation
def viz_alg_1d_2(w):
    it = len(w)
    fig, ax = plt.subplots(figsize=(5, 5))
    def update(i):
        ani = plt.cla()
        #points
        ani = plt.plot(X0[0, :], X0[1, :], 'b^', markersize = 8, alpha = .8)
        ani = plt.plot(X1[0, :], X1[1, :], 'ro', markersize = 8, alpha = .8)
        ani = plt.axis([0 , 6, -2, 4])
        i2 =  i if i < it else it-1
        ani = draw_line(w[i2])
        if i < it-1:
            circle = plt.Circle((X[1, m[i]], X[2, m[i]]), 0.15, color='k', fill = False)
            ax.add_artist(circle)
        cur_axes = plt.gca()
        cur_axes.axes.get_xaxis().set_ticks([])
        cur_axes.axes.get_yaxis().set_ticks([])
        label = 'PLA: iter %d/%d' %(i2, it-1)
        ax.set_xlabel(label)
        return ani, ax
    anim = FuncAnimation(fig, update, frames=np.arange(0, it + 2), interval=1000)
    anim.save('Nguyen Dinh Tai.gif', dpi = 100, writer = 'pillow')
    plt.show()
viz_alg_1d_2(w)
```

```
[0, 18, 5, 14, 9, 1, 11, 11, 4]
```