```
# %%

# ================================================================================
# EXPERIMENT NO 10
# ================================================================================


#%%
import numpy as np

#%%

# Create data
X = np.random.rand(100,1)
y = 4 + 3*X +np.random.randn(100,1)

#%%

# concatenate for w0
Xwb = np.c_[np.ones((100,1)), X]

#%%

# Compute w directly
W = (np.linalg.inv(X.T.dot(X)).dot(X.T)).dot(y)

#%%

import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
from numpy import linalg as LA

#%%

mpl.use('PDF')

#%%

def func(x,y):
    return (0.75*x-1.5)**2+(y-2.0)**2 + 0.25*x*y

#%%

def func_grad(vx,vy):
    dfdx = 1.125*vx - 2.25 + 0.25*vy
    dfdy = 2.0*vy - 4.0 + 0.25*vx
    return np.array([dfdx,dfdy])
```

```
#%%

#prepare for contour plot

xlist = np.linspace(0, 5, 26)
ylist = np.linspace(0, 5, 26)

x, y = np.meshgrid(xlist, ylist)
z = func(x,y)
lev = np.linspace(0,20,21)

#%%

#iterate location

v_init = np.array([5,4])
num_iter = 10
values = np.zeros([num_iter,2])

for gamma in [0.01, 0.1, 0.2, 0.3, 0.5, 0.75]:
    values[0,:] = v_init
    v = v_init

#%%

# actual gradient descent algorithm

for i in range(1,num_iter):
    v = v - gamma * func_grad(v[0],v[1])
    values[i,:] = v

#%%

#plotting

plt.contour(x,y,z,levels=lev)
plt.plot(values[:,0],values[:,1],'r-')
plt.plot(values[:,0],values[:,1],'bo')

#%%

grad_norm = LA.norm(func_grad(v[0],v[1]))

title = "gamma %0.2f | final grad %0.3f" % (gamma,grad_norm)

plt.title(title)
```

```
file = "RAHUL DHANOLA_R177219139_500075154_B4_EXPERIMENT NO 10.pdf" % (gamma*100)

plt.savefig(file, bbox_inches='tight')

#%%

plt.clf()

plt.cla()

# %%

# ============================================================================
# THE END
# ============================================================================
```