In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load and Visualise data

- **Load**
- **Visualise**
- **Normalisation**

Load the data

In [2]:

```
dfX = pd.read_csv('./Dataset/Linear_X_Train.csv')
dfY = pd.read_csv('./Dataset/Linear_Y_Train.csv')
```

In [3]:

```
print(dfX.columns, dfX.shape)
dfX.head()
```

```
Index(['x'], dtype='object') (3750, 1)
```

Out[3]:

	x
0	-0.289307
1	-0.588810
2	1.027507
3	-0.259013
4	0.782043

In [4]:

```
print(dfY.columns, dfY.shape)
dfY.head()
```

```
Index(['y'], dtype='object') (3750, 1)
```

Out[4]:

	y
0	-0.091101
1	-53.467721
2	75.457009
3	-12.025286
4	57.414187

In [5]:

```
X = dfX.values.reshape((-1,))
Y = dfY.values.reshape((-1,))
```

In [10]:

```
print(X.shape, Y.shape)
print(type(X))
print(type(Y))
print(X)
print(Y)
```

```
(3750,) (3750,)
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
[-0.28930742 -0.58880963  1.02750749 ... -0.30516366  1.67583291
 -0.49175724]
[-9.11011171e-02 -5.34677208e+01  7.54570088e+01 ... -3.49832749e+01
  1.45206092e+02 -1.71244939e+01]
```

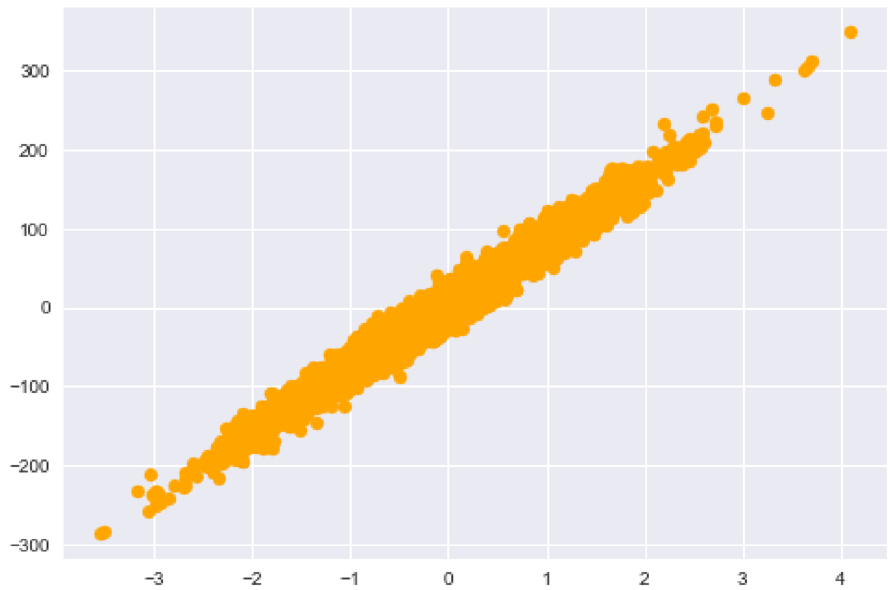
Visualise the data

In [11]:

```
plt.style.use('seaborn')
plt.scatter(X, Y, color='Orange')
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x1069cc58>
```



Normalize the data

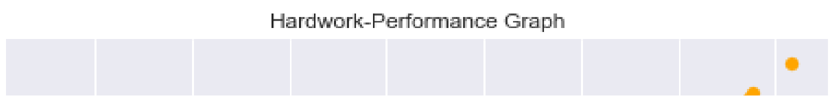
In [12]:

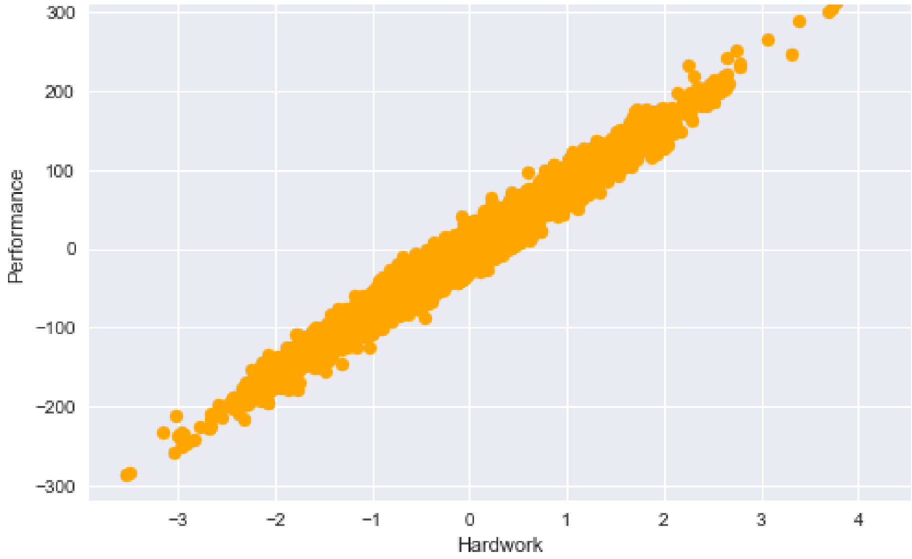
```
u = X.mean()
std = X.std()
print(u, std)

X = (X - u) / std

plt.style.use('seaborn')
plt.scatter(X, Y, color='Orange')
plt.title('Hardwork-Performance Graph')
plt.xlabel('Hardwork')
plt.ylabel('Performance')
```

```
-0.03779466168006855 0.9920801322508517
```

Out[12]:

```
Text(0, 0.5, 'Performance')
```





In [13]:

```
def hypothesis(x, theta):
    return theta[0] + theta[1] * x
```

In [14]:

```
def gradient(X, Y, theta):
    grad = np.zeros((2, ))
    m = X.shape[0]

    for i in range(m):
        x = X[i]
        y_ = hypothesis(x, theta)
        y = Y[i]
        grad[0] += (y_ - y)
        grad[1] += (y_ - y) * x

    return grad / m
```

In [15]:

```
def error(X, Y, theta):
    total_error = 0.0
    m = X.shape[0]
    for i in range(m):
        x = X[i]
        y_ = hypothesis(x, theta)
        y = Y[i]
        total_error += (y_ - y) ** 2
    return total_error / m
```

In [16]:

```
def gradientDescent(X, Y, max_steps = 100,lr = 0.1):
    theta = np.zeros((2, ))
    error_list = []
    for i in range(max_steps):
        # Compute gradient
        grad = gradient(X, Y, theta)
        error_list.append(error(X, Y, theta))
        # Update theta
        theta[0] = theta[0] - lr * grad[0]
        theta[1] = theta[1] - lr * grad[1]
    return theta, error_list
```

In [17]:

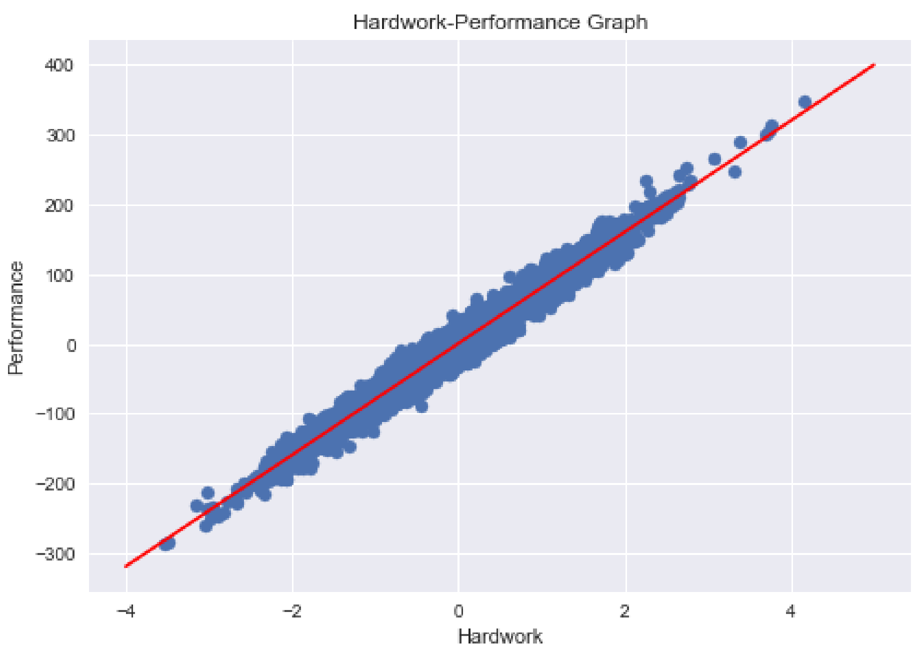
```
theta, error_list = gradientDescent(X, Y)
print(theta)
print(error_list)
```

```
plt.style.use('seaborn')
plt.scatter(X, Y)
plt.title('Hardwork-Performance Graph')
plt.xlabel('Hardwork')
plt.ylabel('Performance')

xl = [-4, 5]
yl = [theta[0] + theta[1] * (-4), theta[0] + theta[1] * (5)]
plt.plot(xl, yl, color='Red')
```

```
[ 0.6838508  79.90361453]
[6576.35005246196, 5363.125102574931, 4380.412893166487, 3584.4160035456366, 2939.6585229
52752, 2417.404963672508, 1994.3795806555281, 1651.7290204117544, 1374.1820666143055, 114
9.3690340383696, 967.2704776518652, 819.770646978796, 700.2957841336078, 603.521145229006
7, 525.1336877162773, 461.6398471309694, 410.20983625687, 368.5515274488485 6, 334.8082973
1435097, 307.47628090540843, 285.33734761416457, 267.4048116482566, 252.87945751587242, 2
41.11392066864, 231.5838358223823, 223.8644670969137 3, 217.61177842928484, 212.5471006085
0397, 208.4447115736724, 205.1217764554589, 202.43019900970543, 200.2500212786460 2, 198.4
8407731648712, 197.05366270713802, 195.8950268735658, 194.95653184837263, 194.19635087796
493, 193.58060429193594, 193.08184955725244, 192.67785822215873, 192.35062524073192, 192.
08556652577707, 191.87086896666273, 191.69696394378104, 191.55610087524585, 191.442001789
7342, 191.34958153046787, 191.2747211204641, 191.21408418835912, 191.16496827335493, 191.
12518438220116, 191.09295943036707, 191.06685721938175, 191.04571442848274, 191.028588767
85533, 191.01471698274722, 191.00348083680868, 190.99437955859895, 190.98700752324964, 19
0.9810361746153, 190.97619938222232, 190.9722815803843, 190.96910816089553, 190.966537691
1093, 190.9644556105823 2, 190.9627691253558, 190.9614030723223, 190.9602965693647, 190.95
940030196982, 190.95867432537915, 190.9580862843408, 190.95760997109994, 190.957224157375
12, 190.95691164825783, 190.95665851587307, 190.9564534786412, 190.95628739848314, 190.95
615287355545, 190.95604390836414, 190.95595564655886, 190.95588415449666, 190.9558262459 2
575, 190.9557793399843, 190.95574134617098, 190.95571057118218, 190.95568564344168, 190.9
5566545197224, 190.95564909688107, 190.9556358492579, 190.95562511868232, 190.95561642691
717, 190.95560938658627, 190.9556036839193, 190.95559906475864, 190.95559532323833, 190.9
559229260724, 190.95558983779603, 190.95558784939806, 190.95558623879643, 190.9555849342
0947]
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x49131a8>]
```

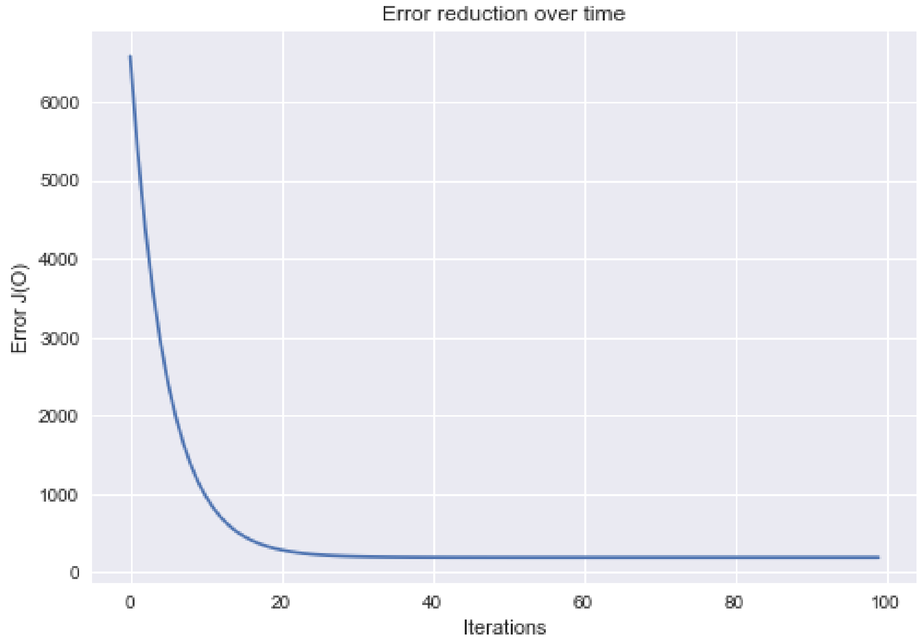


In [18]:

```
plt.plot(error_list)
plt.xlabel('Iterations')
plt.ylabel('Error J(O)')
plt.title('Error reduction over time')
```

Out[18]:

```
Text(0.5, 1.0, 'Error reduction over time')
```

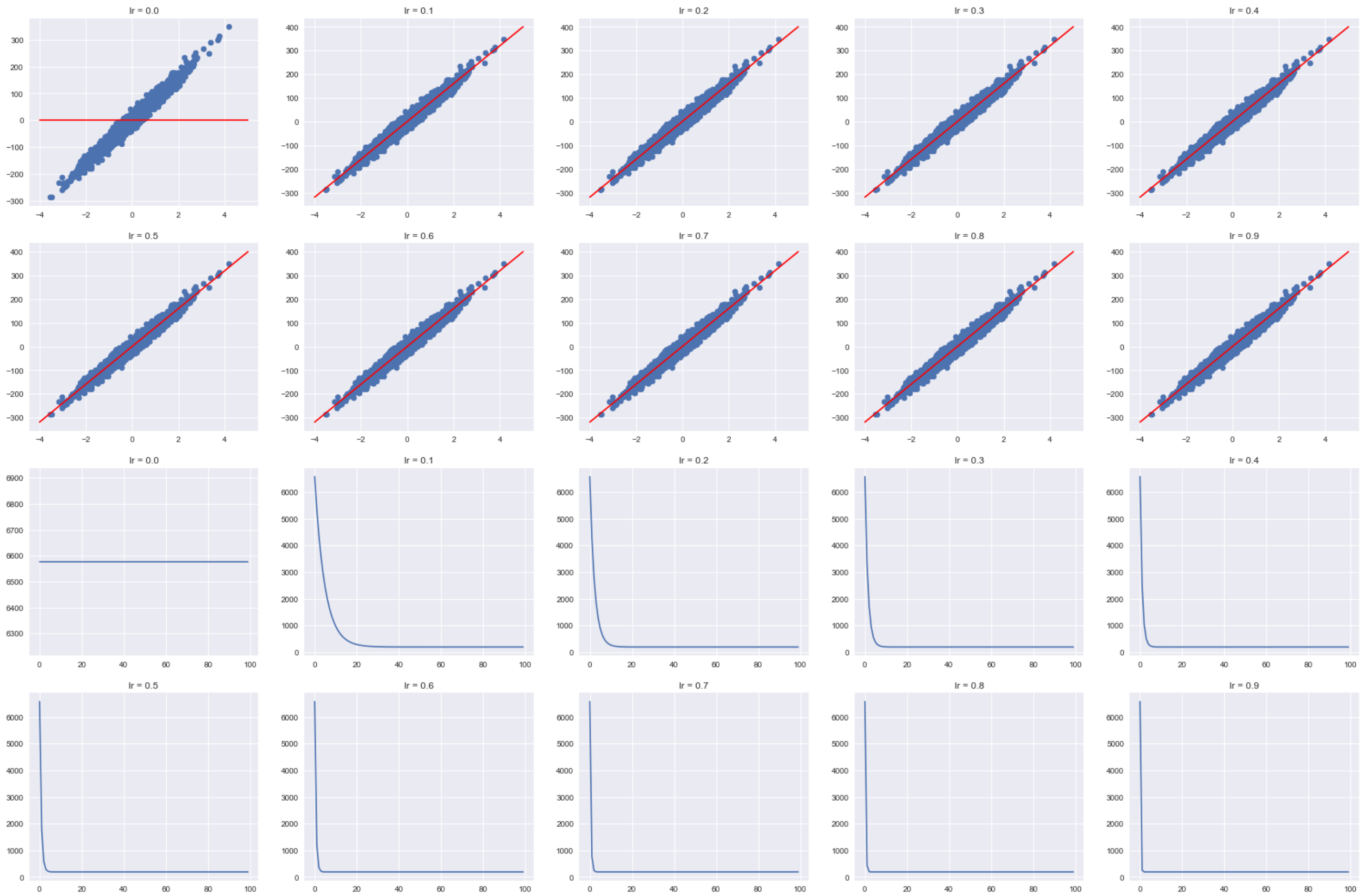


In [20]:

```
i = 1
plt.figure(figsize=(30, 20))

for j in range(0, 10, 1):

    # DATA PREPARATION
    X = dfX.values.reshape((-1,))
    Y = dfY.values.reshape((-1,))

    u = X.mean()
    std = X.std()
    print(u, std)
    X = (X - u) / std

    # ALGORITHM
    lr = j / 10
    plt.subplot(4, 5, j + 1)

    theta, error_list = gradientDescent(X, Y, 100, lr)
    print(theta)
#     print(error_list)

    plt.style.use('seaborn')
    plt.scatter(X, Y)
    plt.title('lr = ' + str(lr))


    xl = [-4, 5]
    yl = [theta[0] + theta[1] * (-4), theta[0] + theta[1] * (5)]
    plt.plot(xl, yl, color='Red')

    plt.subplot(4, 5, j + 11)
    plt.plot(error_list)
    plt.title('lr = ' + str(lr))

    i += 1
plt.show()
```

```
-0.03779466168006855 0.9920801322508517
[0. 0.]
-0.03779466168006855 0.9920801322508517
[ 0.6838508  79.90361453]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573693]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
```

```
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
-0.03779466168006855 0.9920801322508517
[ 0.68386897 79.90573694]
```



In [ ]:

```

```

SUBMISSION

In [92]:

```
theta
```

Out[92]:

```
array([ 0.6838508 , 79.90361453])
```

In [20]:

```
theta = [0.0, 0.0]
theta[0] = 0.6838508
theta[1] = 79.90361453
```

In [21]:

```
print(theta[0], theta[1])
```

```
0.6838508 79.90361453
```

In [71]:

```
a = np.array([1,2])
print(a)
```

```
[1 2]
```

In [22]:

```
X = pd.read_csv('./Dataset/Test Cases/Linear_X_Test.csv')
```

In [23]:

```
X = X.values.reshape((-1, ))
```

In [24]:

```
print(X)
print(X.shape)
```

```
[-1.87794441 -0.86903192 -2.53018242 ...  0.12800782 -0.27803759
 -0.68042543]
(1250,)
```

In [18]:

```
u
```

Out[18]:

```
-0.03779466168006855
```

In [19]:

```
std
```

Out[19]:

```
0.9920801322508517
```

In [25]:

```
preds = []
for i in X:
    pred = theta[0] + theta[1] * i
    pred = pred * std + u
    preds.append(pred)

preds = np.array(preds)
print(type(preds))
```

```
<class 'numpy.ndarray'>
```

In [26]:

```
print(preds.shape)
```

```
(1250,)
```

In [27]:

```
dfPreds = pd.DataFrame(preds, columns=['y'])
```

In [28]:

```
print(dfPreds.shape)
print(type(dfPreds))
dfPreds.head()
```

```
(1250, 1)
<class 'pandas.core.frame.DataFrame'>
```

Out[28]:

	y
0	-148.225494
1	-68.248205
2	-199.928916
3	219.444302
4	47.489686

In [29]:

```
dfPreds.to_csv('sol_new.csv', index=False)
```

In [ ]:

```
```