N97071133 黃佳偉

```
def computeCost(X,Y,theta):
    inner=np.power((( X * theta.T )- Y ),2)
    suminner=np.sum(inner)
    return suminner/(2*len(X))
```

```
def gradecent(X,Y,theta,alpha,iters):
    temp=np.matrix(np.zeros(theta.shape))
    parameters=int(theta.shape[1])
    cost=np.zeros(iters)
    for i in range(iters):
        error=X*theta.T-Y
        for j in range(parameters):
            term=np.multiply(error,X[:,j])
            temp[0,j]=theta[0,j]-(alpha/len(X))*np.sum(term)

        theta=temp
        cost[i]=computeCost(X,Y,theta)
    return theta , cost
```

```
def normalize(X):
    f_standValue = X.std()
    Xnor = (X) / f_standValue
    return Xnor,f_standValue
```

```
def predict(X,theta):
    result = X * theta.T
    return result
```

Function 簡介:

1. 計算 cost 算法偏向平均平方差的做法。
2. 梯度下降法，也另外包成一個 function，通過參數 alpha 控制其學習率，iters 決定訓練的次數。
3. 正規化的部分，採用資料 / 其欄位的標準差的做法。
4. 預測的部分，則把訓練完的最佳係數帶回原式，推出 Y，但需要在乘上正規化係數。

一. 數據前處理

一開始會把數值讀入後，把年齡和身高和體重做正規化，並把各個欄位的標準差係數存下來。

二. 定義輸入輸出

再來則把年齡和身高切出為 X 訓練輸入，體重則為 Y 訓練輸出。

三. 訓練

然後定義一個線性方乘的係數，和計算損失，利用梯度下降法，把損失求到最小，此時的係數結果，則拿來當訓練的結果，可以用來預測未知的體重。

四. 預測

先將待求的輸入端，一樣做跟訓練 X 的數據前置處理，處理完成後，帶入係數 theta 求解，並乘回標準差係數，求得體重四捨五入為 59。