Ex5: Gradient Descent

Cho dữ liệu chiều cao heights (heights_1.txt), weights (weights_1.txt)

1. Tạo 2 numpy array heights và weights chứa 2 danh sách từ 2 tập tin trên.
2. Chuyển heights sang mét (heights*0.0254) và weights sang kg (weights * 0.453592)
3. Trực quan hóa dữ liệu theo heights, weights
4. X = heights đã chuyển theo định dạng chuẩn, y = weights
5. Với y = mx + b (weights = m*heights + b), gọi hàm tính m, b: theta = gradient_descent_2(alpha, X, y, 1000)
6. Từ m, b (m = theta[1], b = theta[0]) => dự đoán weights_predict theo m, b
7. Trực quan hóa dữ liệu
8. Với chiều cao là 1.8, 1.9, 2.0 thì cân nặng lần lượt là bao nhiêu?

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
from sklearn.datasets.samples_generator import make_regression
from scipy import stats
```

In [2]:
```
h = [74, 74, 72, 72, 73, 69, 69, 71, 76, 71, 73, 73, 74, 74, 69, 70, 73, 75, 78, 7
```

In [3]:
```
heights = np.array(h)
```

In [4]:
```
heights.size
```

Out[4]: 1015

In [5]:
```
w = [180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185, 160, 180, 18
```

In [6]:
```
weights = np.array(w)
```

In [7]:
```
weights.size
```

Out[7]: 1015

In [8]:
```
heights = heights*0.0254
```

In [9]:
```
heights[0:5]
```

Out[9]: array([1.8796, 1.8796, 1.8288, 1.8288, 1.8542])

In [10]:
```
weights = weights * 0.453592
```

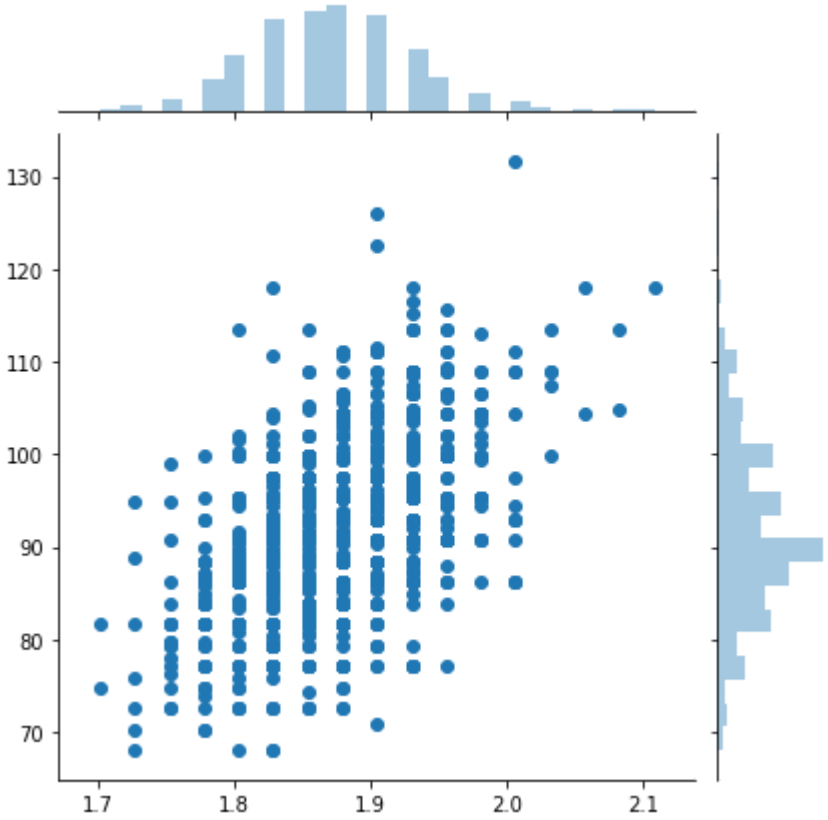
In [11]:
```
weights[0:5]
```

Out[11]:
```
array([81.64656 , 97.52228 , 95.25432 , 95.25432 , 85.275296])
```

In [12]:
```
import seaborn as sns
```

In [13]:
```
plt.figure(figsize=(12,8))
sns.jointplot(x=heights, y = weights)
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```



In [14]:
```
from chapter4_lib import *
```

In [15]:
```
# y = mx + b
m = heights.size
X = np.c_[ np.ones(m), heights] # insert column
y = weights
alpha = 0.01 # learning rate
theta = gradient_descent_2(alpha, X, y, 1000)
```

```
iter 0 | J: 3956.720
iter 1 | J: 3611.334
iter 2 | J: 3296.374
iter 3 | J: 3009.158
iter 4 | J: 2747.243
iter 5 | J: 2508.400
iter 6 | J: 2290.597
iter 7 | J: 2091.980
iter 8 | J: 1910.859
iter 9 | J: 1745.693
iter 10 | J: 1595.076
iter 11 | J: 1457.728
iter 12 | J: 1332.478
iter 13 | J: 1218.261
iter 14 | J: 1114.106
iter 15 | J: 1019.126
iter 16 | J: 932.512
iter 17 | J: 853.529
iter 18 | J: 781.502
```

In [16]:
```
X[0:5]
```

Out[16]:
```
array([[1.    , 1.8796],
       [1.    , 1.8796],
       [1.    , 1.8288],
       [1.    , 1.8288],
       [1.    , 1.8542]])
```

In [17]:
```
print("m = ", theta[1], "b = ", theta[0])
```

```
m =  38.16395791358294 b =  19.96584532847684
```

In [18]:
```
for i in range(X.shape[1]):
    weights_predict = theta[1]* X + theta[0]
```

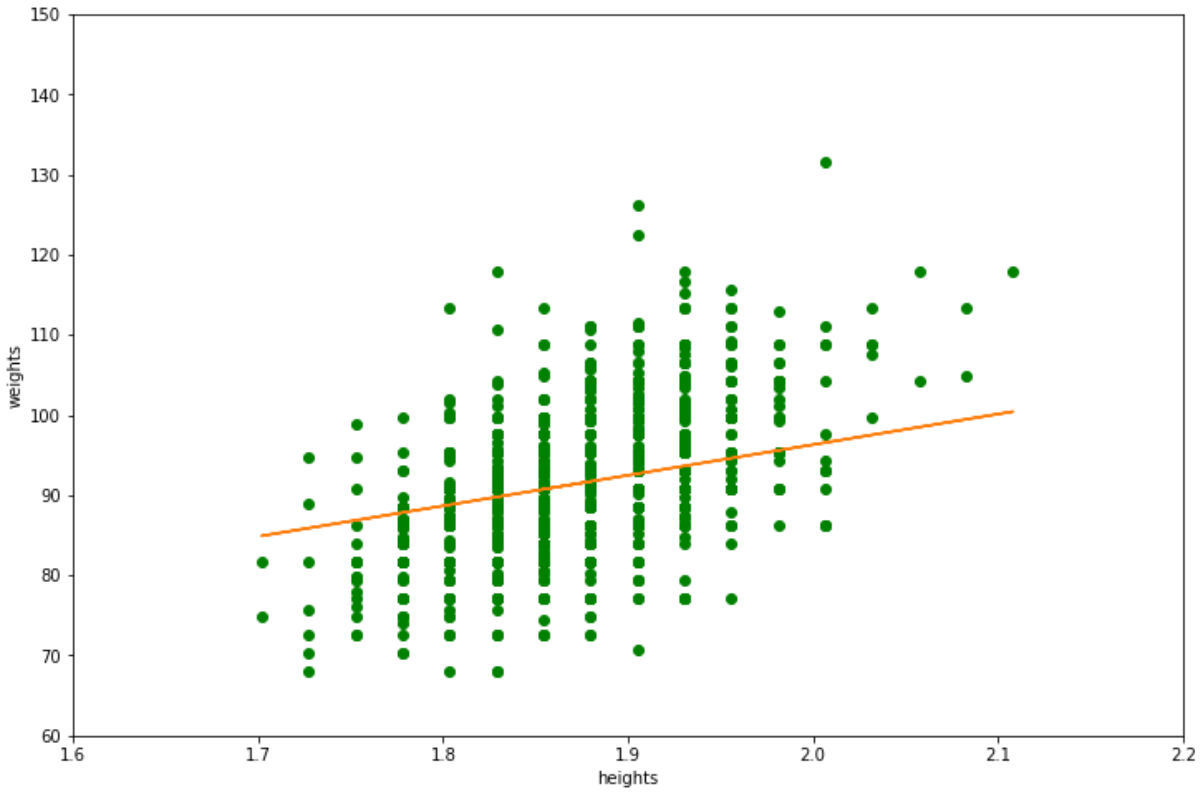
In [19]:
```
heights.size
```

Out[19]: 1015

In [20]:
```
weights.size
```

Out[20]: 1015

In [21]:
```
plt.figure(figsize=(12,8))
plt.xlim(1.6,2.2)
plt.ylim(60,150)
plt.scatter(X[:,1], weights, color="green")
plt.plot(X, weights_predict)
plt.xlabel("heights")
plt.ylabel("weights")
plt.show()
```



In [22]:
```
height = np.array([1.8,1.9,2.0])
weight = theta[1]*height + theta[0]
weight
```

Out[22]:
```
array([88.66096957, 92.47736536, 96.29376116])
```

In [ ]:
```

```