```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 14, 'figure.figsize': [12.0, 6.0]})
```

```
X = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],   # для умножения на intercept
              [1, 1, 2, 1, 3, 0, 5, 10, 1, 2]]) # стаж
y = [45, 55, 50, 59, 65, 35, 75, 80, 50, 60]
X.shape
```

```
(2, 10)
```

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_{pred_i} - y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left((w_0 \cdot x_{i0} + w_1 \cdot x_{i1}) - y_i\right)^2 \rightarrow \min_{w}$$

$$w_j \leftarrow w_j - \alpha\frac{\partial}{\partial w_j}L(w)$$

$$\frac{\partial L(w)}{\partial w_j} = \frac{1}{n}2\sum_{i=1}^{n}x_{ij}\Big(\sum_{j=0}^{m}(w_j x_{ij}) - y_i\Big)$$

$$\vec{w} = \vec{w} - \alpha\frac{2}{n}X^T(X\vec{w} - \vec{y})$$

```
def calc_mse(y, y_pred):
    err = np.mean((y - y_pred)**2)
    return err
```

1. Подберите скорость обучения (alpha) и количество итераций:

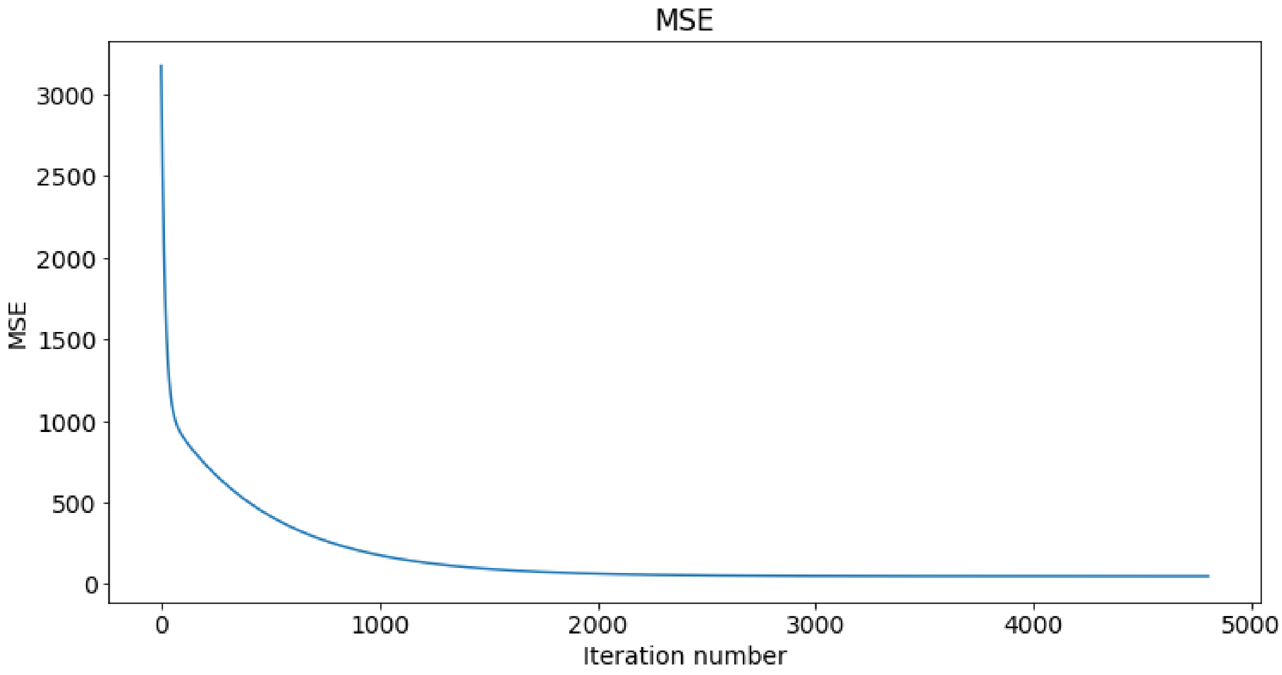
```
n = X.shape[1]
alpha = 1e-3
w = np.array([1, 0.5])
errors = []

for i in range(int(4801)):
    y_pred = np.dot(w, X)
    err = calc_mse(y, y_pred)
    errors.append(err)
    for j in range(w.shape[0]):
        w[j] -= alpha * (1/n * 2 * np.sum(X[j] * (y_pred - y)))
    if i % 400 == 0:
        print(i, w, err)
```

```
0 [1.1102 0.84  ] 3173.15
400 [18.17511588  9.27626113] 500.7989288428639
800 [28.06626432  7.44984037] 243.83236130353217
1200 [34.59041507  6.24510603] 132.03487746741158
1600 [38.89371573  5.45046864] 83.39556413536022
2000 [41.73215361  4.92632937] 62.2342399245102
2400 [43.60437454  4.58060946] 53.027661712772876
```

```
2800 [44.8392831   4.35257415] 49.02219014944904
3200 [45.65382326  4.20216307] 47.27954461105105
3600 [46.19109031  4.1029526 ] 46.52137833162843
4000 [46.54546925  4.03751382] 46.19152576288303
4400 [46.77921602  3.99435069] 46.048018038969396
4800 [46.93339434  3.96588049] 45.985582670037175
```

```
plt.plot(range(len(errors)), errors)
plt.title('MSE')
plt.xlabel('Iteration number')
plt.ylabel('MSE');
```



2. В этом коде мы избавляемся от итераций по весам, но тут есть ошибка, исправьте ее:

```
w = np.array([1, 0.5])
alpha = 1e-2

for i in range(501):
    y_pred = np.dot(w, X)
    err = calc_mse(y, y_pred)

    w -= (alpha * (1/n * 2 * np.sum(X * (y_pred - y), axis=1)))
    if i % 100 == 0:
        print(i, w, err)
```

```
0 [2.102 3.9  ] 3173.15
100 [31.88770806  6.74418155] 175.19445858001853
200 [41.83683774  4.90699865] 61.9177717428135
300 [45.33508261  4.26102097] 47.913169919666785
```

```
400 [46.56511152  4.03388672] 46.181755648107604
500 [46.99760587  3.95402334] 45.96769776787538
```

3. Вместо того, чтобы задавать количество итераций, задайте условие остановки алгоритма - когда ошибка за итерацию начинает изменяться ниже определенного порога

```
w = np.array([1, 0.5])
min_mse = 0.45
curr_mse = 0
err = np.inf
i = 0

while True:
    y_pred = np.dot(w, X)
    curr_mse = err
    err = calc_mse(y, y_pred)
    if (curr_mse - err) < min_mse:
        break
    w -= (alpha * (1/n * 2 * np.sum(X * (y_pred - y), axis=1)))
    if i % 6 == 0:
        print(i, w, err)
    i += 1
```

```
0 [2.102 3.9  ] 3173.15
6 [ 6.07189929 10.54296976] 996.2256404018397
12 [ 8.71625632 10.91060358] 859.8557658274829
18 [11.07405383 10.57454305] 763.5834236832634
24 [13.27392695 10.17984555] 678.9831179277083
30 [15.33838126  9.79996576] 604.3598272772635
36 [17.27714615  9.4421132 ] 538.5331668304805
42 [19.09803573  9.10589002] 480.4661345401807
48 [20.80823551  8.79009064] 429.2440269430093
54 [22.41447637  8.49348622] 384.05996631949137
60 [23.92307798  8.21491142] 344.2021894686833
66 [25.3399756   7.95327044] 309.04283562650113
72 [26.67074369  7.70753393] 278.0280560710098
78 [27.92061783  7.47673511] 250.66928960466856
84 [29.09451537  7.25996595] 226.5355664801635
90 [30.19705472  7.05637362] 205.24671953719772
96 [31.23257356  6.86515715] 186.46739560848204
102 [32.2051459   6.68556423] 169.90177285942266
108 [33.11859808  6.51688829] 155.2889008462543
114 [33.97652388  6.35846573] 142.39858988674843
120 [34.78229859  6.20967327] 131.0277849906938
126 [35.53909234  6.06992551] 120.9973672303842
132 [36.24988257  5.93867266] 112.14933216461309
138 [36.9174657   5.81539833] 104.3443008692165
144 [37.54446818  5.69961752] 97.45932436660489
150 [38.13335681  5.59087473] 91.38594686849362
156 [38.68644842  5.48874214] 86.0284973230128
162 [39.20591903  5.39281791] 81.30258235373915
168 [39.69381237  5.30272468] 77.13375685063328
174 [40.15204793  5.21810797] 73.45635127132434
```

```
180 [40.58242855  5.1386349 ] 70.21243717977558
186 [40.98664744  5.06399279] 67.35091472694727
```

✓ 0 сек.   выполнено в 12:41   ● ✕