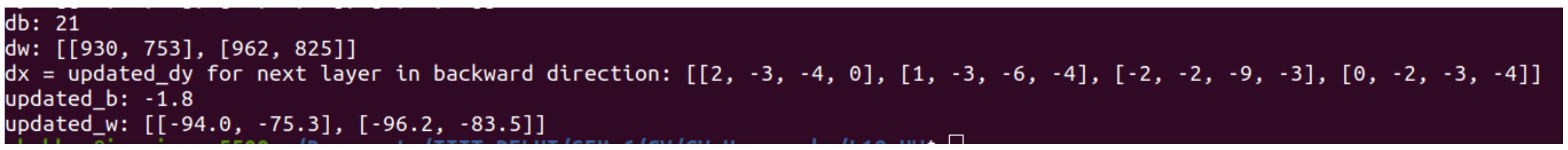
Code

```python
import numpy as np


dy = [[-2, 3, 4], [-1, 5, 3], [2, 3, 4]]
x = [[12, 14, 15, 17], [34, 56, 73, 32], [67, 43, 31, 21], [32, 31, 43, 56]]
w = [[-1, 0], [0, -1]]
b = 0.3


def convolute(a, b):
    l = len(a) - len(b) + 1
    ans = [[0]*l for i in range(l)]
    for i in range(l):
       for j in range(l):
            s = 0
            for p in range(len(b)):
                  for q in range(len(b[0])):
                        s += b[p][q] * a[i + p][j + q]
            ans[i][j] = s
    return ans


db = 0
db = sum([sum(i) for i in dy])
print("db:", db)


dw = convolute(x, dy)
print("dw:", dw)


def pad_with(vector, pad_width, iaxis, kwargs):
    pad_value = kwargs.get('padder', 10)
    vector[:pad_width[0]] = pad_value
    vector[-pad_width[1]:] = pad_value


dy_0 = np.pad(dy, 1, pad_with, padder=0)
w_dash = np.rot90(w, 2)
dx = convolute(dy_0, w_dash)
print("dx = updated_dy for next layer in backward direction:", dx)


learning_rate = 0.1
updated_b = b - learning_rate * db
print("updated_b:", updated_b)


updated_w = [[w[i][j] - learning_rate * dw[i][j] for j in range(len(w[i]))] for i in range(len(w))]
print("updated_w:", updated_w)
```

Output:



```
db: 21
dw: [[930, 753], [962, 825]]
dx = updated_dy for next layer in backward direction: [[2, -3, -4, 0], [1, -3, -6, -4], [-2, -2, -9, -3], [0, -2, -3, -4]]
updated_b: -1.8
updated_w: [[-94.0, -75.3], [-96.2, -83.5]]
```

updated_dy for next layer in backward direction: **[[2, -3, -4, 0], [1, -3, -6, -4], [-2, -2, -9, -3], [0, -2, -3, -4]]**
updated_b: **-1.8**
updated_w: **[[-94.0, -75.3], [-96.2, -83.5]]**