In [1]:

```
def naive_relu(x):
    assert len(x.shape)

    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i,j] = max(x[i,j],0)
    return x
```

In [2]:

```
def naive_add(x):
    assert len(x.shape) == 2
    assert x.shape == y.shape

    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i,j] = y[i,j]
    return x
```

In [ ]:

```
import numpy as np

z = x + y
z = np.maximum(z, 0.)
```

In [3]:

```
def naive_add(x):
    assert len(x.shape) == 2
    assert len(y.shape) == 1
    assert x.shape[1] == y.shape[0]

    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i,j] = y[j]
    return x
```

In [5]:

```
import numpy as np

x = np.random.random((64, 3, 32, 10))
y = np.random.random((32, 10))
z = np.maximum(x,y)
```

In [ ]:

```
z = np.dot(x, y)
z = xy
```

In [6]:

```
def naive_vector_dot(x, y):
    assert len(x.shape) == 1
    assert len(y.shape) == 1
    assert x.shape[0] == y.shape[0]

    z = 0.
    for i in range(x.shape[0]):
        z += x[i] * y[i]
    return z
```

In [7]:

```
def naive_matrix_vector_dot(x, y):
    assert len(x.shape) == 2
    assert len(y.shape) == 1
    assert x.shape[1] == y.shape[0]

    z = np.zeros(x.shape[0])
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            z[i] += x[i,j] * y[j]
    return z
```

In [8]:

```
def naive_matrix_vector_dot(x, y):
    z = np.zeros(x.shape[0])
    for i in range(x.shape[0]):
        z[i] = naive_vector_dot(x[i, :], y)
    return z
```

In [9]:

```
def naive_matrix_dot(x, y):
    assert len(x.shape) == 2
    assert len(y.shape) == 2
    assert x.shape[1] == y.shape[0]

    z = np.zeros((x.shape[0], y.shape[1]))
    for i in range(x.shape[0]):
        for j in range(y.shape[1]):
            row_x = x[i, :]
            column_y = y[:, j]
            z[i, j] = naive_vector_dot(row_x, column_y)
    return z
```

In [11]:

```
x = np.array([[0., 1.],
              [2., 3.],
              [4., 5.]])

print(x.shape)
```

(3, 2)

In [13]:

```
x = x.reshape((2,3))
x
```

Out[13]:

```
array([[0., 1., 2.],
       [3., 4., 5.]])
```

In [14]:

```
x = np.zeros((300, 20))
x = np.transpose(x)
x.shape
```

Out[14]:

```
(20, 300)
```