Tema 03. Redes de Neuronas: Ejemplo

Autor: Ismael Sagredo Olivenza

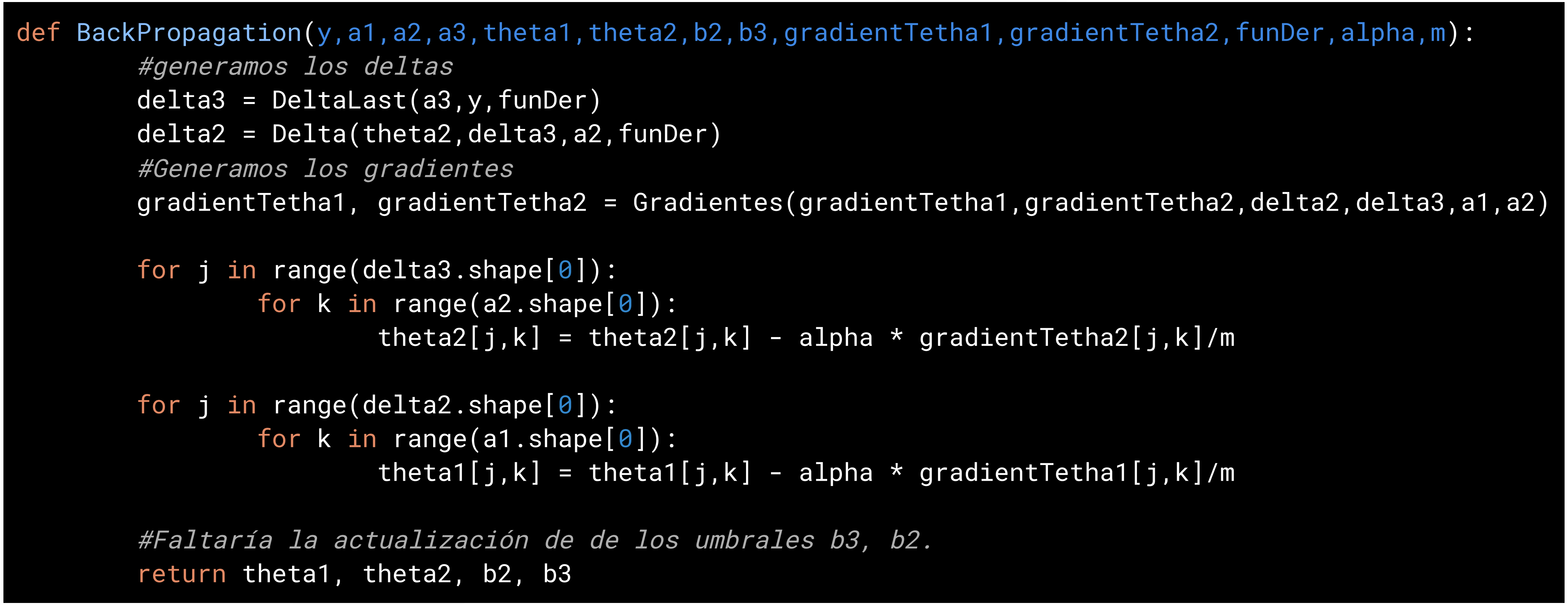
Example MLP

```python
def MLP(X,Y,hidden, fun, funDer, alpha, epochs):
        theta1 = np.random.rand(hidden, X.shape[1])
        b2 = np.random.rand(hidden)
        outPuts = Y.shape[1]
        theta2 = np.random.rand(outPuts, hidden)
        b3 = np.random.rand(outPuts)
        m=X.shape[0]
        for epoch in range(epochs):
                c = 0
                gradientTetha2 = np.zeros(theta2.shape)
                gradientTetha1 = np.zeros(theta1.shape)
                for i in range(m):
                        a1,a2,a3 = ForwardProp(X[i],fun,theta1,theta2,b2,b3)
                        y_ = Y[i]
                        e = cost(y_,a3)
                        c += np.sum(e)
                        theta1, theta2, b2, b3 = BackPropagation(y_,a1,a2,a3,theta1,theta2,b2,b3,
                        gradientTetha1,gradientTetha2,funDer,alpha,m)
                J = - c / m

        return histoy, theta1, theta2
```

ForwardPropagation

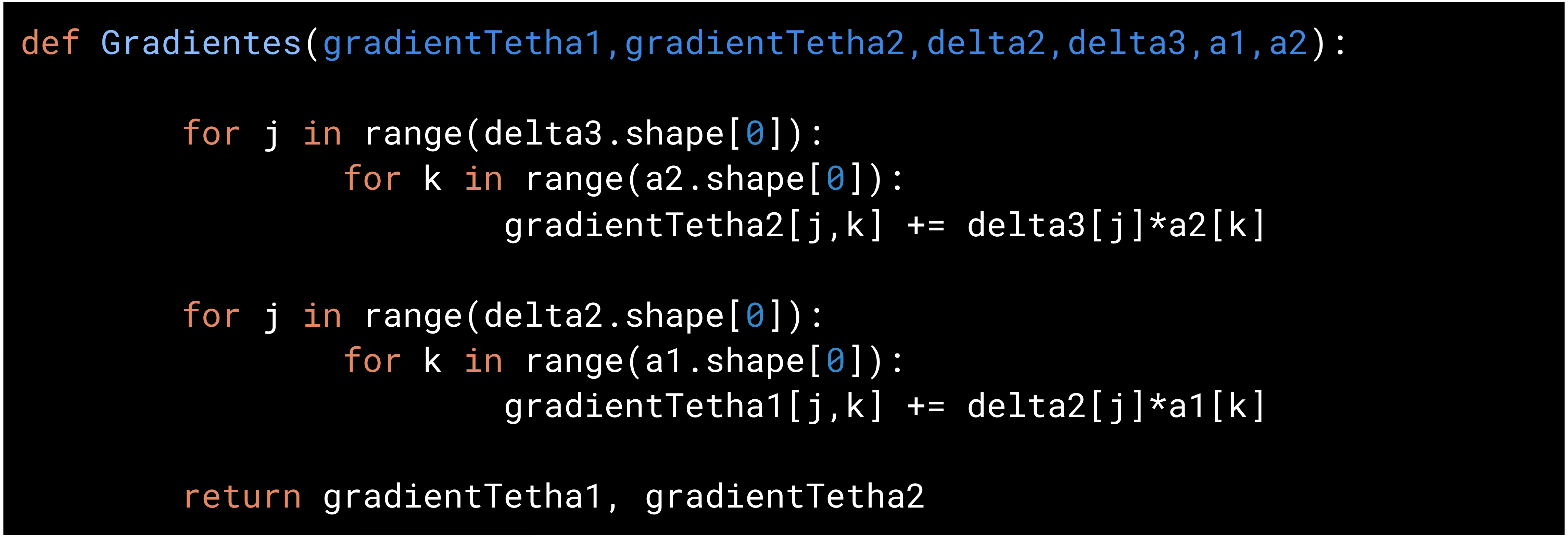
```
def ForwardProp(x,fun,theta1,theta2,b2,b3):
        a1= np.array(x)
        z2 = np.dot(a1, theta1.T)+b2
        a2 = fun(z2)

        z3 = np.dot(a2, theta2.T) + b3
        a3 = fun(z3)
        return a1,a2,a3
```

BackPropagation



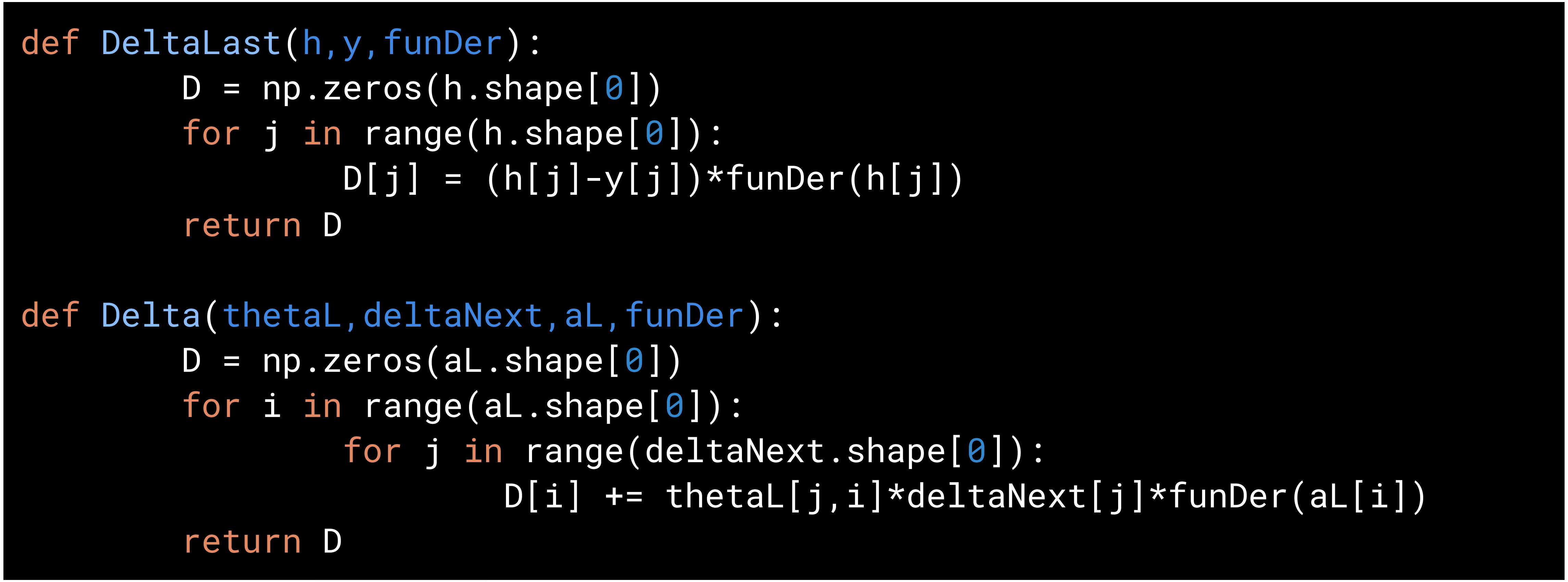
```
def BackPropagation(y,a1,a2,a3,theta1,theta2,b2,b3,gradientTetha1,gradientTetha2,funDer,alpha,m):
        #generamos los deltas
        delta3 = DeltaLast(a3,y,funDer)
        delta2 = Delta(theta2,delta3,a2,funDer)
        #Generamos los gradientes
        gradientTetha1, gradientTetha2 = Gradientes(gradientTetha1,gradientTetha2,delta2,delta3,a1,a2)

        for j in range(delta3.shape[0]):
                for k in range(a2.shape[0]):
                        theta2[j,k] = theta2[j,k] - alpha * gradientTetha2[j,k]/m

        for j in range(delta2.shape[0]):
                for k in range(a1.shape[0]):
                        theta1[j,k] = theta1[j,k] - alpha * gradientTetha1[j,k]/m

        #Faltaría la actualización de de los umbrales b3, b2.
        return theta1, theta2, b2, b3
```

Gradients



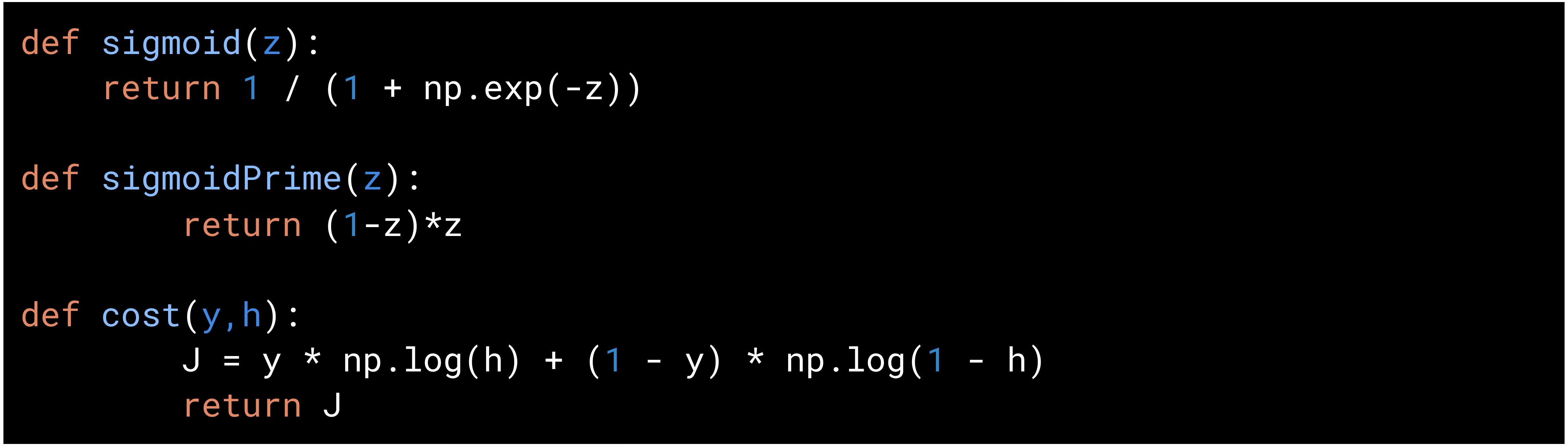
```
def Gradientes(gradientTetha1,gradientTetha2,delta2,delta3,a1,a2):

        for j in range(delta3.shape[0]):
                for k in range(a2.shape[0]):
                        gradientTetha2[j,k] += delta3[j]*a2[k]

        for j in range(delta2.shape[0]):
                for k in range(a1.shape[0]):
                        gradientTetha1[j,k] += delta2[j]*a1[k]

        return gradientTetha1, gradientTetha2
```

Delta rules



```python
def DeltaLast(h,y,funDer):
        D = np.zeros(h.shape[0])
        for j in range(h.shape[0]):
                D[j] = (h[j]-y[j])*funDer(h[j])
        return D

def Delta(thetaL,deltaNext,aL,funDer):
        D = np.zeros(aL.shape[0])
        for i in range(aL.shape[0]):
                for j in range(deltaNext.shape[0]):
                        D[i] += thetaL[j,i]*deltaNext[j]*funDer(aL[i])
        return D
```

Miscelanea



```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def sigmoidPrime(z):
        return (1-z)*z

def cost(y,h):
        J = y * np.log(h) + (1 - y) * np.log(1 - h)
        return J
```