First, let's write down the forward pass. The variables are:

- `xs` the input sequence, encoded using one-hot encoding. Denote it by x_t.
- `hs` the hidden state (a vector), at each time step. Denote it by $h_t = \tanh(W^{xh}x_t + W^{hh}h_{t-1})$
- `ys` the output layer. Denote it by $y_t = W^{hy}h_t + b^y$
- `ps` the output of the softmax. Denote it by $\hat{y}_t = softmax(y_t)$
- `loss` the cost/loss function. $Cost = -\sum_t \log(\sum_k \hat{y}_t^k x_t^k)$

Now, let's go line by line and interpret those. We will often use e.g. $\partial Cost_t/\partial h$ to denote the contribution from time-step t to the cost function, with $C = \sum_t C_t$ for

$$C_t = \log(\sum_k \hat{y}_t^k x_t^k).$$

```
dy = np.copy(ps[t])
dy[targets[t]] -= 1 # backprop into y
```

This is just the derivative of the softmax for $Cost_t$:

$$\frac{\partial Cost_t}{\partial y} = \hat{y} - x$$

Note that x is one-hot encoded, so that it's mostly zeros, with only a single 1 at coordinate `targets[t]`. That's why we first set `dy` to `ps` (i.e., the $\hat{y}$), and then subtract y.

```
dWhy += np.dot(dy, hs[t].T)
dby += dy
```

This corresponds to the t-th component of the derivatives wrt W^{hy} and b^y:

$$\partial Cost_t/\partial W^{hy} = \frac{\partial Cost_t}{\partial y_t}\frac{\partial y_t}{\partial W^{hy}} = \frac{\partial Cost_t}{\partial y_t}h_t^T$$

$$\partial Cost_t/\partial W^{hy} = \frac{\partial Cost_t}{\partial y_t}\frac{\partial y_t}{\partial b^y} = \frac{\partial Cost_t}{\partial y_t}1 = \frac{\partial Cost_t}{\partial y_t}$$

```
dh = np.dot(Why.T, dy) + dhnext
```

This is tricky. We want to account for the influence of h_t on both $Cost_t$ and $Cost_{(t+1):end}$.

$$\frac{\partial Cost_{t:end}}{\partial h_t} = \frac{\partial Cost_t}{\partial h_t} + \frac{\partial Cost_{(t+1):end}}{\partial h_t} = \frac{\partial Cost_t}{\partial y}\frac{\partial y}{\partial h_t} + dhnext$$

```
dhraw = (1 - hs[t] * hs[t]) * dh
```

$$\frac{\partial Cost_{t:end}}{\partial hraw_t} = (1 - h_t^2)\frac{\partial Cost_{t:end}}{\partial h_t}$$

The following:

```
dbh += dhraw
dWxh += np.dot(dhraw, xs[t].T)
dWhh += np.dot(dhraw, hs[t-1].T)
```

are similar to what we already had. Note that

$$\frac{\partial Cost_{t:end}}{\partial h_t} = \frac{\partial Cost}{\partial h_t}$$

since h_t cannot influence components of the cost that come before it in time.

Finally, we compute `dhnext`, which must be $\frac{\partial Cost_{t:end}}{\partial h_{t-1}}$ in order for our earlier definition to work. Now

$$\frac{\partial Cost_{t:end}}{\partial h_{t-1}} = \frac{\partial Cost_{t:end}}{\partial hraw_t}\frac{\partial hraw_t}{\partial h_{t-1}}$$

This is exactly what the following line does.

```
dhnext = np.dot(Whh.T, dhraw)
```

The following is self-explanatory:

```
for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
  np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
```

In the loop, we are adding up all the contributions to the gradients from all the time-steps t.