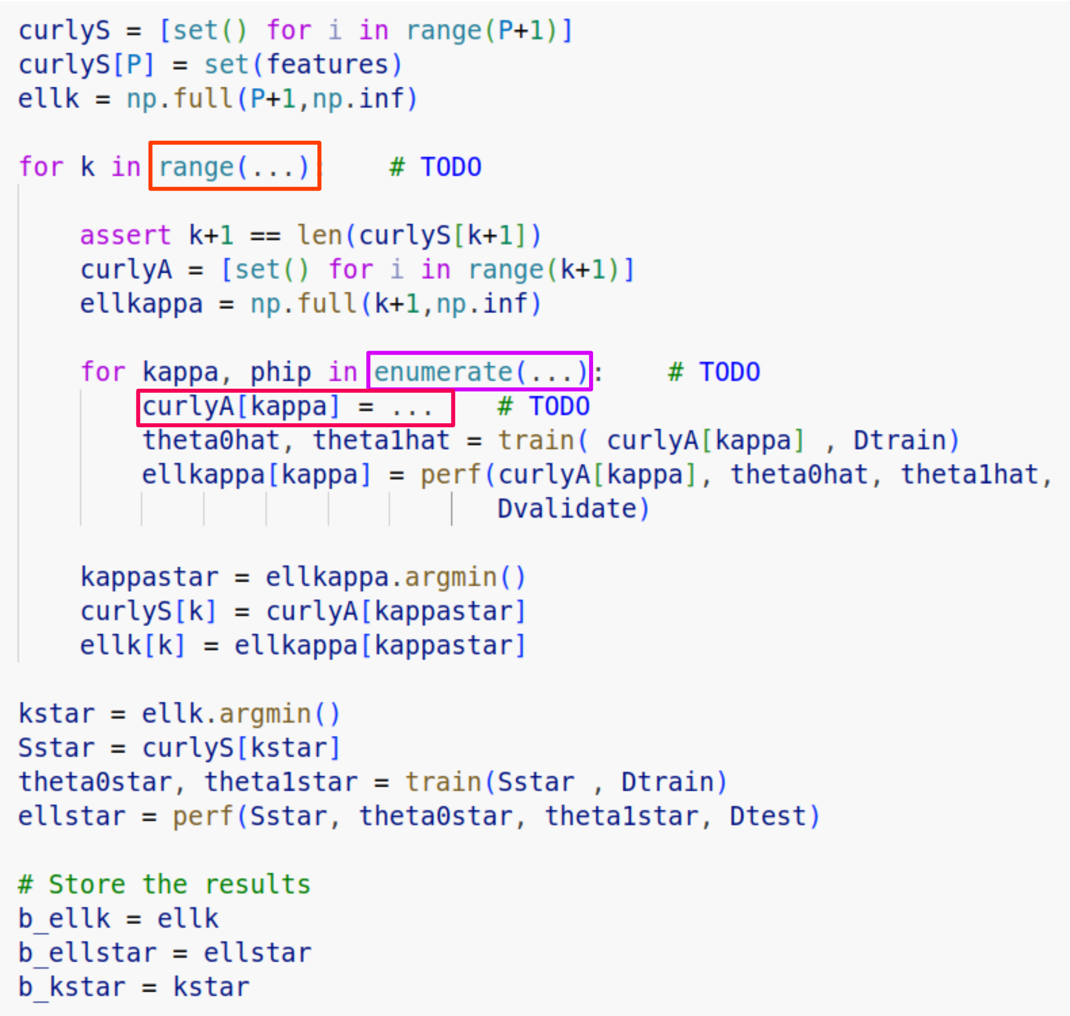
Backward stepwise selection

```
$\mathcal{S}_P = \mathcal{P}$
for $k = \boxed{P-1 \dots 1}$:
        for $\kappa, \phi_p \in \boxed{\text{enumerate}(\mathcal{S}_{k+1})}$
            $\boxed{\mathcal{A}_\kappa = \mathcal{S}_{k+1} \setminus \phi_p}$
            $\hat{\theta}_\kappa = \text{train}(\mathcal{A}_\kappa, \mathcal{D}_{\text{train}})$
            $\ell_\kappa = \text{perf}(\mathcal{A}_\kappa, \hat{\theta}_\kappa, \mathcal{D}_{\text{val}})$
        $\kappa^* = \text{argbest}(\{\ell_\kappa\})$
        $\mathcal{S}_k = \mathcal{A}_{\kappa^*}$
$\mathcal{S}^* = \text{best of } \{\mathcal{S}_k\}_P$
$\hat{\theta}^* = \text{train}(\mathcal{S}^*, \mathcal{D}_{\text{train}})$
$\ell^* = \text{perf}(\mathcal{S}^*, \hat{\theta}^*, \mathcal{D}_{\text{test}})$
```



```python
curlyS = [set() for i in range(P+1)]
curlyS[P] = set(features)
ellk = np.full(P+1,np.inf)

for k in range(...)      # TODO

    assert k+1 == len(curlyS[k+1])
    curlyA = [set() for i in range(k+1)]
    ellkappa = np.full(k+1,np.inf)

    for kappa, phip in enumerate(...):     # TODO
        curlyA[kappa] = ...      # TODO
        theta0hat, theta1hat = train( curlyA[kappa] , Dtrain)
        ellkappa[kappa] = perf(curlyA[kappa], theta0hat, theta1hat,
                               Dvalidate)

    kappastar = ellkappa.argmin()
    curlyS[k] = curlyA[kappastar]
    ellk[k] = ellkappa[kappastar]

kstar = ellk.argmin()
Sstar = curlyS[kstar]
theta0star, theta1star = train(Sstar , Dtrain)
ellstar = perf(Sstar, theta0star, theta1star, Dtest)

# Store the results
b_ellk = ellk
b_ellstar = ellstar
b_kstar = kstar
```