Algorithm 1: Proximal Policy Optimization (PPO)

Input: Initial policy parameters θ, value function parameters ϕ, clipping threshold ϵ, total batch size B, minibatch size M, number of update epochs K, trajectory horizon T, learning rates λ_π, λ_V

for *each* ***epoch*** **do**
- Initialize buffer $\mathcal{D} \leftarrow \emptyset$;
- Reset environment and observe initial state s_0;
- **while** *buffer* $\mathcal{D}$ *not full* **do**
  - Sample action $a_t \sim \pi_\theta(\cdot|s_t)$ and store $\log \pi_\theta(a_t|s_t)$;
  - Execute a_t in environment;
  - Observe r_t, s_{t+1}, done signal d_t;
  - Store $(s_t, a_t, r_t, \log \pi_\theta(a_t|s_t), d_t)$ in $\mathcal{D}$;
  - **if** d_t *is True* **then**
    - Reset environment and observe new s_{t+1};
  - **end**
  - $s_t \leftarrow s_{t+1}$;
- **end**
- Compute advantage estimates $\hat{A}_t$ and returns $\hat{R}_t$ using GAE or TD;
- **for** $k = 1$ **to** K **do**
  - Shuffle $\mathcal{D}$ and split into minibatches of size M;
  - **for** *each minibatch* **do**
    - Compute importance ratio:
    $$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$
    - Compute clipped objective:
    $$L^{\text{CLIP}}(\theta) = \frac{1}{M} \sum \min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)$$
    - Update policy:
    $$\theta \leftarrow \theta + \lambda_\pi \nabla_\theta L^{\text{CLIP}}(\theta)$$
    - Compute value loss:
    $$L^V(\phi) = \frac{1}{M} \sum \Big(V_\phi(s_t) - \hat{R}_t\Big)^2$$
    - Update value function:
    $$\phi \leftarrow \phi - \lambda_V \nabla_\phi L^V(\phi)$$
  - **end**
- **end**

end