<u>Question 1:</u> Explain the convergence method and why you picked it. There is no wrong answer. You will get credit for any method you pick as long as it converges and you provide a reasonable explanation of why you picked it.

In Policy Iteration, I used the **maximum absolute difference between successive value function estimates**, defined as:

$$\Delta_k = \max_s |V_k(s) - V_{k-1}(s)| \,.$$

This quantity measures how much the value function changed during the most recent policy evaluation sweep. If Δ_k becomes smaller than a threshold $\varepsilon = 10^{-4}$, we treat the evaluation step as having converged.

This method is the standard stopping criterion presented in class and in Sutton & Barto. It makes sense because policy evaluation is a contraction mapping: each Bellman expectation update brings V closer to the true V^π. Therefore, when the updates become negligibly small, additional sweeps would not meaningfully change the estimate.

I also included a **safety cutoff of 100 iterations** to prevent infinite loops when the initial policy does not lead to terminal states. This ensures robustness even when a random initial policy is used.

So to summarize, I chose this method because:

1. It is widely used and theoretically justified.
2. It provides a clear numerical stopping point.
3. It aligns directly with the assignment prompt (|V□ – V□-1| vs. t).
4. It works reliably for all policies, convergent or not.

<u>Question 2:</u> Explain the convergence method and why you picked it. There is no wrong answer. You will get credit for any method you pick as long as it converges and you provide a reasonable explanation of why you picked it.

For Value Iteration, I used the **same convergence metric**:

$$\Delta_k = \max_s |V_k(s) - V_{k-1}(s)| \,,$$

but applied after **optimality backups** instead of policy evaluation.

Value Iteration repeatedly applies the Bellman optimality operator:

$$V_{k+1}(s) = \max_a \Big(R(s,a,s') + \gamma V_k(s') \Big).$$

This operator is *also* a contraction mapping, meaning repeated application makes the value function shrink toward the optimal value function V^*. Thus, using the maximum update difference Δ_k is a direct and mathematically justified way to detect when the algorithm has essentially converged.

I used a convergence tolerance of $\varepsilon = 10^{-4}$. Once $\Delta_k < \varepsilon$, further iterations would produce negligible changes and the policy extracted from V would remain the same.

This method was chosen because:

1. It is the standard and most stable convergence check for Value Iteration.
2. It matches the assignment requirement exactly.
3. It offers a clear interpretation: once changes are small, the greedy policy is optimal.
4. It is inexpensive to compute and easy to visualize.

Your Value Iteration convergence plot shows Δ rapidly falling to 0 within a few iterations, confirming fast convergence.

Question 3: Compare the convergence times between using Policy Iteration and Value Iteration and give reasons for why you would ever use Policy Iteration.

From the convergence plots, we can deduce the following for each approach:

- **Policy Iteration**
  - Policy Iteration required several outer loops, and each outer loop required many policy evaluation sweeps before convergence.
  - The total number of backups performed is large, even though the number of outer iterations (policy improvements) is small.
  - The errors occasionally spike because each new policy resets the evaluation process.
- **Value Iteration**
  - Value Iteration converged much faster in terms of iterations.
  - The value function reached stability in only a few passes over the entire grid.
  - This aligns with theory: VI quickly propagates optimal values from the terminal states outward.

Why ever use Policy Iteration?

Even though Policy Iteration may take more computation per iteration, it has several advantages:

1. Guaranteed monotonic policy improvement.
   Each improvement step produces a strictly better (or equal) policy.
2. Often converges in very few policy updates.
   For small or structured MDPs, PI can converge in 2–3 outer loops.
3. More stable with respect to stochastic policies.
   PI evaluates a full policy, so it handles soft/deterministic policies naturally.
4. Easier to interpret conceptually.
   PI separates “evaluation” from “improvement,” which is pedagogically useful and easier to analyze.
5. In large state spaces (with approximations), PI variants (like PPO or actor–critic) are widely used in practice