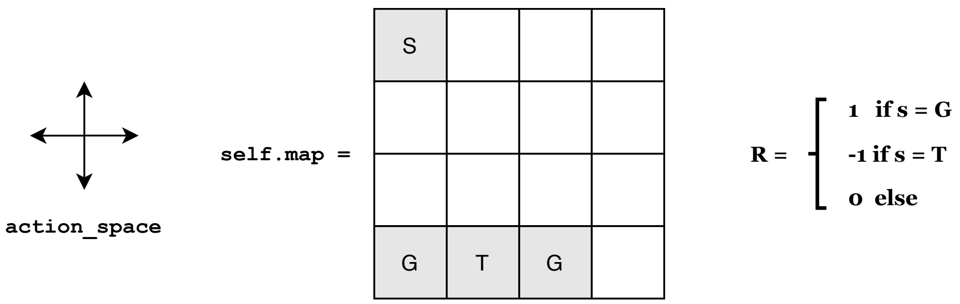
Exercise 4
Model-free Prediction

1 Temporal Difference Learning

TD-Learning is a technique that allows us to solve MDPs without access to the state transitions $\mathcal{P}$ and reward function $\mathcal{R}$. Your task is to implement the TD(0) learning algorithm to evaluate the value function for the environment from the last exercise. As a short reminder, here is what it looks like:



An implementation of the environment is provided in `gridworld.py`. The file `util.py` contains a helper function for plotting the associated value function.

The skeleton code for this exercise is contained in `td_agent.py`. The dependencies are listed in the requirements file, and can be installed via `pip install -r requirements.txt`. We recommend using `python 3.12`, but older versions should also work.

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Programming Tasks

1. `learn(n_timesteps)`. This method should implement `n_timesteps` of environment interaction via selecting a random action and deploying it in the environment. Implement TD(0)-Learning and update the array `self.V` holding the current approximation at every time step.
2. `action(state)`. Currently, this method randomly selects an action. Implement action selection based on `self.policy`. Note that by default this policy is also defined to be random, so the results should be the same.
3. `self.policy`. Experiment with different (not fully random) policies. Run the script and examine how the TD estimate of V changes visually.