Exercise 10
Bandits, Bandits, Bandits

In this exercise we will implement some of the methods for multi-armed bandits you saw in the lecture. This will be one of the shorter exercises, so we encourage everyone to give it a try.

You will work with the `MultiArmedBandit` environment inside `env.py`. Based on an n-dimensional input array it constructs a n-armed bandit with win probabilities specified by the respective array entries. Your task is to implement the body of the `action_selection` function inside the child classes inheriting from `BaseAgent`. Please also get familiar with the `train` function inside the base class.

To really highlight the benefits and drawbacks of certain methods, we test our implementation on a 1000-armed bandit with only one arm having a win-probability $> 80\%$.

Programming Tasks

1. Your task is to implement:
   - A random agent (inside class `RandomAgent`)
   - An epsilon-greedy agent (`EpsilonGreedAgent`)
   - An agent using UCB1 (`UCB1Agent`), and
   - Probability matching via thompson sampling (`ProbabilityMatching`).
2. After you implemented the UCB1 agent, you might notice that it doesn't perform as great as expected. What might be the reason? Try to find a way to fix this.

Now, try to interpret the result graphs you are seeing after running the `bandit_agents.py` file and try to answer the following questions:

1. Why does the RMSE between utility/value estimate and true utility (win-probabilities) not correlate with the regret?
2. What is the reason that epsilon-greedy's end performance is lower compared to UCB1 and probability matching? What would you need to change? (Hint: we already did this in a past exercise)
3. Looking at these results, do you see a benefit of using probability matching compared to UCB1?