Data Scientist @ Babbel: Take-home Challenge

While we do a lot of classic supervised machine learning in Babbel's Data Solutions team, we find that supervised ML does not make a good take home challenge to show your true data science skills. Everybody can call Keras, xgboost or a scikit-learn algorithm. Engineering good features requires intimate knowledge of the data set and its source, and meaningfully tuning a model requires knowledge of its intended application context. We therefore look at another kind of mathematical algorithm for this challenge – it is one that's actually used at Babbel. Here we go...

> "Thompson Sampling is a Bayesian approach to the Multi-Armed Bandit problem based on the Beta-Bernoulli model."

Task 1 (of 3):

Familiarise yourself with the Thompson sampling algorithm if you don't know it.

Explain, in your own words, the four concepts underlined above in one to two pages (for all four concepts, not for each one), in a meaningful order that ties them together. Your explanation should be aimed at a reader with a basic understanding of data science – think of a junior data scientist. Try to use words rather than equations.

Task 2 (of 3):

Implement Thompson Sampling for the Beta-Bernoulli case in (modern!) Python, using only the Python standard library, as well as numpy and scipy if you see fit. Make a nice, reusable component that could be used in a system that needs to do Thompson Sampling.

- Submit a clean Python script for this task, **not a notebook**.
- Use Python 3.6 or above.
- Comment your code!

Task 3 (of 3):

Using your implementation, simulate an experiment optimising over five actions (or “slot machines”) with success probabilities 0.1, 0.4, 0.45, 0.6 and 0.61.

For your experiment, visualise the following quantities, using a plotting library of your choice:

1. The evolution of the posterior distributions of the estimated success probabilities over the iterations of the Bandit.
2. The evolution of the Bandit’s best (point) estimate of the success probabilities over the iterations.
3. The evolution of regret over time.

Answer the following questions:

1. How well does the algorithm discover the true individual success probabilities?
2. How does this relate to the probabilities themselves, and why?
3. Can you think of a simple modification of the algorithm to improve the accuracy of the estimation of the success probabilities? (Consider this a bonus question.)

You can use a Jupyter or Jupyterlab notebook, or a Python script with exported graphs for this task, whatever you prefer. **If you use a notebook, please submit the notebook itself as well as a PDF export.** Document your code and your findings.