Warm-up B — Reinforcement learning

Goal: Learn state / action / reward, then run a simple Q-learning loop on email data.

Data: `cyber-datasets/`
Library: `seas8414-teach` (plus plain numpy/pandas in the Q-learning cell)

Run cells top to bottom. Compare with Warm-up A: *what is this row?* vs *what should I do next?*

In [ ]:
```
import seas8414_teach as teach
teach.setup()
print(teach.__version__)
teach.story_title_card(
    "Warm-up B · Reinforcement learning",
    "Actions earn rewards over time — labels grade decisions.",
    ["Policy demo", "Tabular Q sketch", "Bridge to Acts 4–5"],
)
```

In [ ]:
```
teach.list_tables()
```

---

§0 · RL basics (vocabulary + one simple loop)

SL question: what is this row?
RL question: what should I do next?

Term	Cyber English
State	What the agent sees right now (e.g. mail features, attacker depth)
Action	A control we choose (deliver / quarantine, allow / block)
Reward	Score for that choice (+1 right, −1 wrong in the toy)
Policy	Habit: “in this state, take that action”
Return	Total reward over a shift / episode
Explore	Sometimes try a random action to learn
Exploit	Take the action that currently looks best

Diagram 1 · Agent–environment loop

Diagram 2 · Mail quarantine MDP

Diagram 3 · One Q-learning step

Diagram 4 · Right tool for the shape

In [ ]:

```python
import numpy as np
import pandas as pd

# Load local cyber data (same folder as the SL warm-up)
mail = pd.read_csv("cyber-datasets/email_spam.csv")

# Truth: 1 = spam, 0 = ham (legitimate mail)
is_spam = (mail["label"].astype(str) == "spam").astype(int).to_numpy()

# State: one easy-to-see number, cut into 5 buckets (0..4)
# Higher capital-run length often co-occurs with spammy mail in this table
score = mail["capital_run_length_total"]
state = pd.qcut(score, q=5, labels=False, duplicates="drop").to_numpy()
n_states = int(state.max()) + 1

# Q-table: how good each action is in each state
# column 0 = deliver, column 1 = quarantine
Q = np.zeros((n_states, 2))
learning_rate = 0.1   # size of each learning step
explore_rate = 0.1    # chance of trying a random action
discount = 0.9        # how much the next state still counts
rng = np.random.default_rng(8414)

for t in range(len(is_spam)):
    s = int(state[t])

    # Explore sometimes; otherwise take the best action so far
    if rng.random() < explore_rate:
        action = int(rng.integers(2))   # 0 or 1
    else:
        action = int(Q[s].argmax())     # best known action in this state

    # Grade the decision against the truth
    if action == is_spam[t]:
        reward = 1.0    # right call
    else:
        reward = -1.0   # wrong call (missed spam or blocked ham)

    next_s = int(state[min(t + 1, len(is_spam) - 1)])
    best_future = Q[next_s].max()
    target = reward + discount * best_future
    Q[s, action] = Q[s, action] + learning_rate * (target - Q[s, action])

# Score the learned policy on the second half of the stream
half = len(is_spam) // 2
late_rewards = []
for t in range(half, len(is_spam)):
    action = int(Q[int(state[t])].argmax())
    late_rewards.append(1.0 if action == is_spam[t] else -1.0)
```

```
print("mean reward on later emails:", round(float(np.mean(late_rewards)), 3))
print("Q-table  rows = state buckets  cols = [deliver, quarantine]")
print(Q.round(2))
```

In [ ]:
```
teach.story_rl_warmup()
```

In [ ]:
```
X, y, cols = teach.cyber_table("dns_tunnel.csv", "dns_tunnel")
print(X.shape, cols[:4])
```

In [ ]:
```
# Optional: needs OPENAI_API_KEY in ~/.env — skip if unset
teach.ask(
    "Define state, action, reward for a mail quarantine agent in one minute.
    mean_reward="see prior cell",
)
```

Exit ticket

1. SL question vs RL question?
2. In the mail toy, what is state / action / reward?
3. What does the Q-table store?
4. Why is “always quarantine” weak long-run?