1.1 Fundamentals of reinforcement learning

RL involves an agent learns through trial and error. An agent receives rewards for good decisions and penalties for bad decisions
The agent's goal is to devise a strategy to maximize positive feedback over time

Contrarily to supervised and unsupervised learning, there is no predefined training data and is suitable for **sequential decision-making tasks.** RL agents learns without any supervision.

RL has applications in robotics, autonomous vehicles, finance, and chatbot development.

1.2 Navigating the RL framework

5 key components

1. Agent - the learner
2. States - environment snapshot at a given time
3. Actions - agent's choice in response to state
4. Rewards - feedback for agent action
5. Environment - challenges to be solved

```
env = create_environment()
state = env.get_initial_state()
for i in range(n_iterations):
    action = choose_action(state)
    state, reward = env.execute(action)
    update_knowledge(state, action, reward)
```

Episodic tasks are tasks segmented in episodes, with a beginning and end, like an agent playing chess

Continuous tasks have continuous interaction without episodes, like adjusting traffic lights

The agent tries to maximize total reward over time, the **return**, the sum of expected rewards
The agent learns to anticipate return

However, immediate rewards are more valuable than future ones, **discounted return** gives more weight to nearer rewards

The discount factor gamma is the reward discount, ranging from 0 to 1. The lower the gamma, the more the agent prioritizes immediate gains.

$$\underbrace{\text{Discounted return}}_{\text{Sum of discounted expected rewards}} = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots + \gamma^{n-1} r_n$$

reward discounts

We can show this in a numerical example

```
expected_rewards = np.array([1,6,3])
discount_factor=0.9
discounts = np.array([discount_factor ** i for i in
range(len(expected_results))])
# prints [1, 0.9, 0.81] a decreasing gamma value

discounted_return = np.sum(expected_rewards*discounts)
# 1*1 + 0.9*6 + 0.81*3 = 8.83
```

1.3 Interacting with Gymnasium environments

Gymnasium is a standard library for RL tasks and provides RL environments like CartPole, MountainCar, FrozenLake, and Taxi

Offers a unified interface like functions and methods to initialize and visualize the environment, then executing actions

```
import gymnasium as gym
env = gym.make('CartPole', render_mode='rgb_array')
state, info = env.reset(seed=42) # seed arg is used to ensure reproducability
print(state) # array of numbers

# we can visualize using matplotlib

import matplotlib.pyplot as plt
def render():
    state_image = env.render()
    plt.imshow(state_image)
    plt.show()

render()

# left = 0, right = 1

action = 1
state, reward, terminated, truncated, info = env.step(action)
# we focuse on first three values
print(state, reward, terminated)
# reward = 1 because terminal state has not been reached
```

```python
while not terminated:
    action = 1 # right
    state, reward, terminated, _, _ = env.step(action)
    render()
```