Farmer Assistant

Team Members

Ahmed Selim Mahmoud	MI
Arwa Mohammed Hamed	DS
Hager Mohammed Hegazy	MI
Mariam Hassan Elsaka	DS
Nour Ali Saif	MI

Agenda

1. Project Name

2. Problem Definition
   1. description& images
   2. environment
   3. states
   4. actions
   5. rewards
   6. policy(by text,symols)
   7. the algorithm and it's mechnism
   8. input and output(screen after the run).

3. Summary

4. Team member tasks

5. code and declaration

6. Colab link

Project Name

Farmer Assistant

Problem Definition:-

1.Description

The project aims to develop an agent using PPO algorithm to assist farmers in real-time crop management, monitoring growth stages, watering, and pest control, optimizing yield and health, thus enhancing agricultural efficiency and productivity.

Clarification the scenarios of our agent

Problem Definition:-

2.Environment

The environment we created simulates a three different plants growth scenario in a one plot field where the agent endeavors to optimize plant growth and health through planting, watering, and pest management amidst fluctuating weather conditions. It offers a platform for experimenting with agricultural strategies to maximize yield and resilience.

Problem Definition:-

3.States

Our 4 observations:

- **Soil Condition:** Represents whether the soil is empty (0) or planted (1).
- **Plant Stage:** Indicates the growth stage of the plant, ranging from small (0) to fully grown (varies according to plant type).
- **Presence of Bugs:** Indicates whether there are pests present (1) or not (0).
- **WeatherCondition**: Represents the current weather condition, categorized as sunny (0), normal (1), or rainy (2).

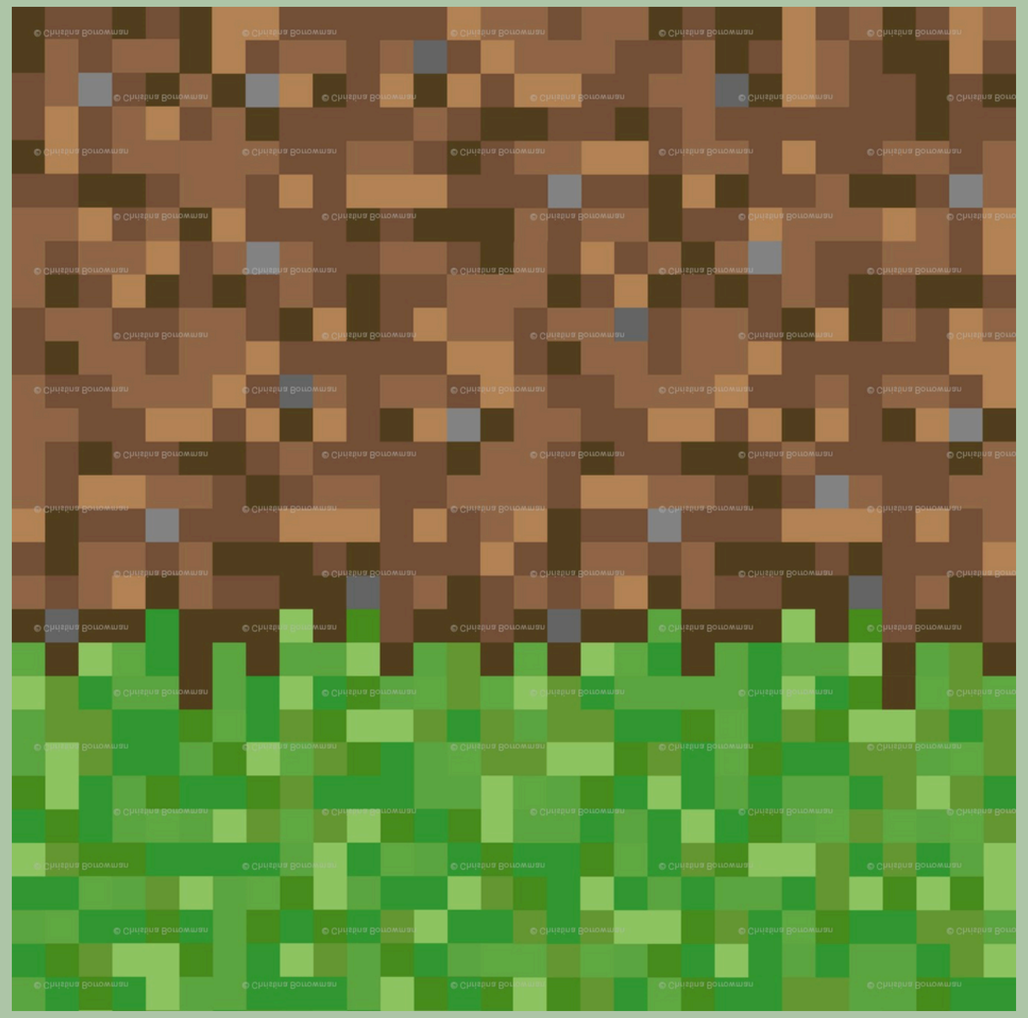
Problem Definition:-

3.States

- **Boundaries of Plant Stage according to Plant Type:**

1. **For tomatoes: Plant stage ranges from 0 to 2.**
2. **For corn: Plant stage ranges from 0 to 3.**
3. **For beans (assumed as the default): Plant stage ranges from 0 to 4.**

Soil is a constant state for all plants

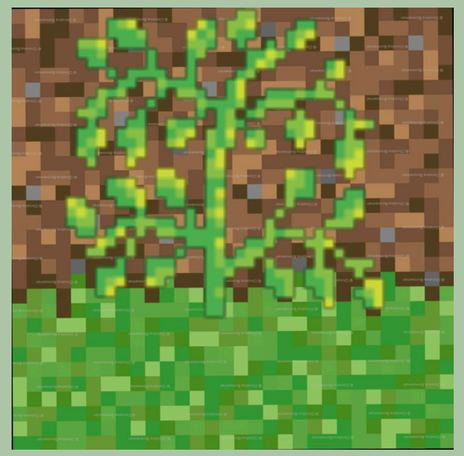


Soil

For Tomato

All these states are repeated with different weather conditions “sunny, normal and runny” .

So, the **number of tomato states** is **19**.



so_small_tomato

so_small_bad_tomato

gesture_tomato

gesture_bad_tomato

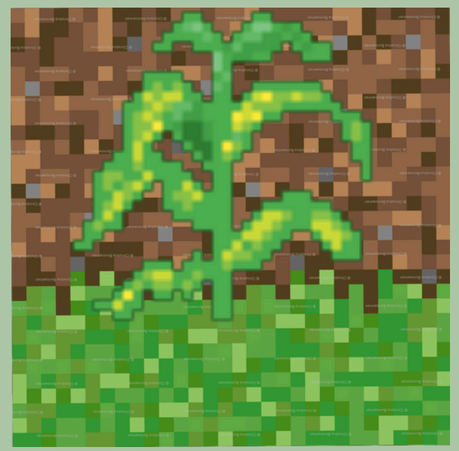
full_grown_tomato

full_grown_bad_tomato

For Corn

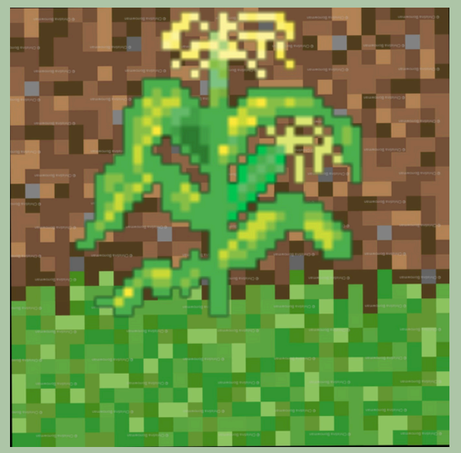
All these states are repeated with different weather conditions “sunny, normal and runny”

So, the **number of corn states** is **25**.



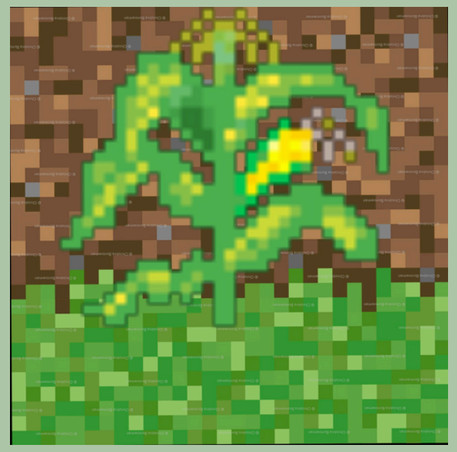
so_small_corn

so_small_bad_corn



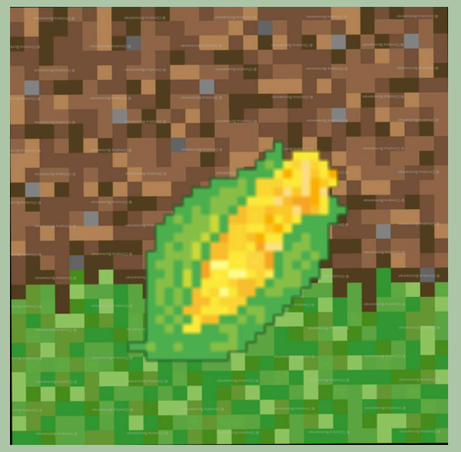
small_corn

small_bad_corn



gesture_corn

gesture_bad_corn



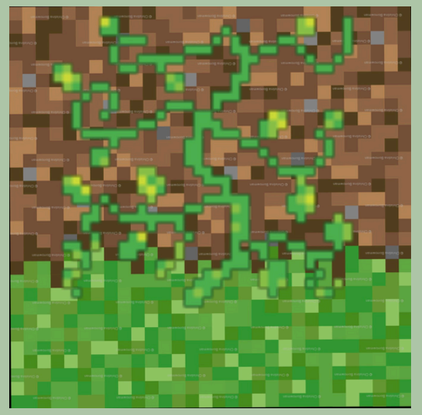
fully_corn

fully_bad_corn

For Bean

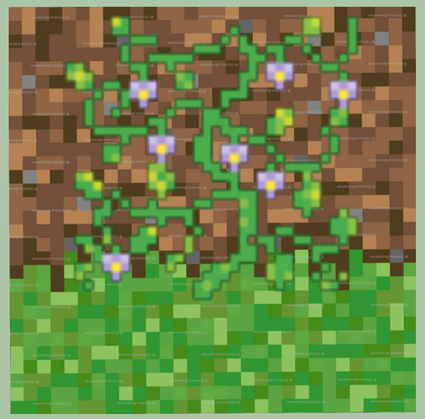
All these states are repeated with different weather conditions “sunny, normal and runny” .

So, the **number of corn states** is **31**.

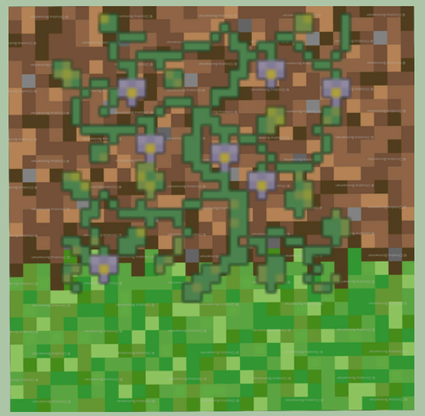


so_small_bean

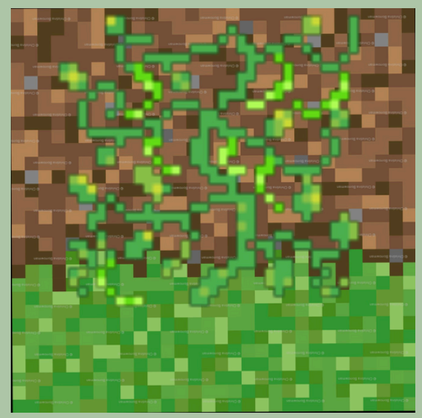
so_small_bad_bean



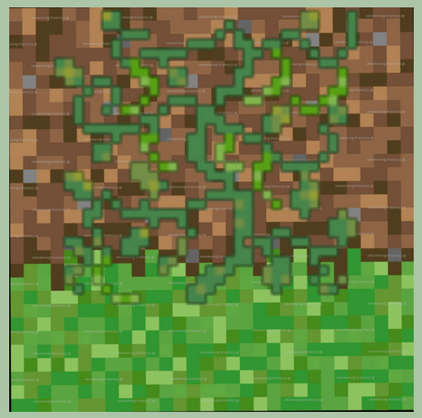
small_bean



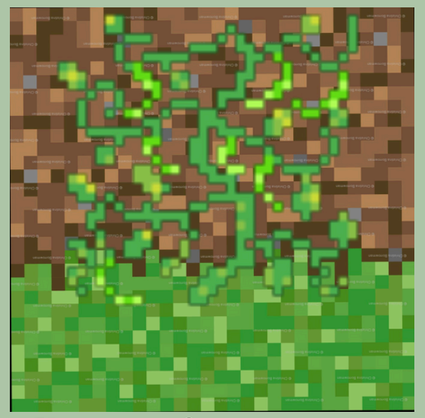
small_bad_bean



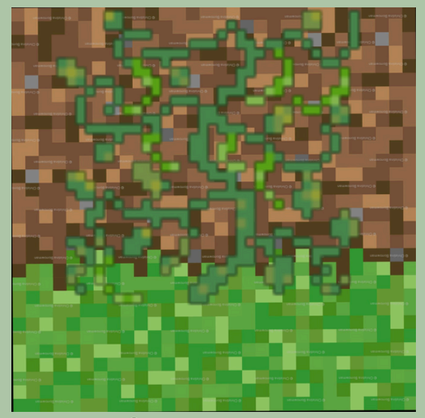
gesture_bean



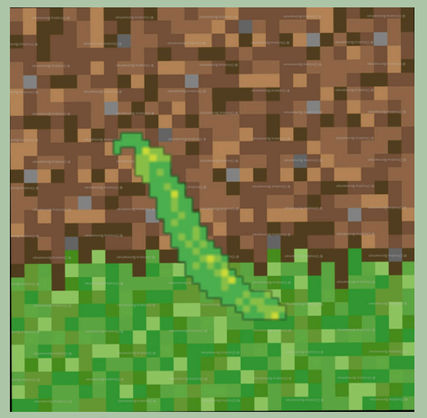
gesture_bad_bean



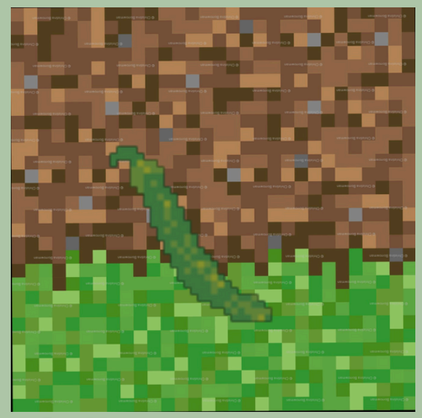
flowering_bean



flowering_bad_bean



fully_bean



fully_bad_bean

Problem Definition:-

4.Our 6 actions

- **Plant: Planting in empty soil.**
- **Kill Bug: Eliminating pests from the plant.**
- **Water 1L: Watering the plant with 1L of water.**
- **Water 2L: Watering the plant with 2L of water.**
- **Water 3L: Watering the plant with 3L of water.**
- **Harvest: Collecting the fully grown plant.**

Problem Definition:-

5.Rewards given actions

Planting:

1. **Reward: +20 for successful planting.**
2. **Penalty: 0 for attempting to plant in non-empty soil.**

Killing Bug:

1. **Reward: +10 for successfully eliminating pests.**
2. **Penalty: 0 for attempting to kill non-existent bugs.**

Problem Definition:-

5.Rewards given actions

Watering:

Reward:

- **1 L** of water:

1. **+10** if it rainy

penality:

1. **-3** if it sunny or normal
2. **-1** for watering the bad plant or watering unplanted soil or fully grown plant

Problem Definition:-

5.Rewards given actions

Watering:

Reward:

- **2 L** of water:

1. **+10** if it normal

penality:

1. **-3** if it sunny
2. **-1** if it rainy or for watering the bad plant or watering unplanted soil or fully grown plant

Problem Definition:-

5.Rewards given actions

Watering:

Reward:

- **3 L** of water:

1. **+10** if it sunny

penality :

1. **-3** if it rainy or normal
2. **-1** for watering the bad plant or watering unplanted soil or fully grown plant

Problem Definition:-

5.Rewards given actions

Harvesting:

Reward:

1. **+100 for harvesting a fully grown plant.**
2. **-10 for harvesting a fully grown plant infested with pests.**

penality:

1. **0 for attempting to harvest before the plant is fully grown.**

Problem Definition:-

6.Policy:

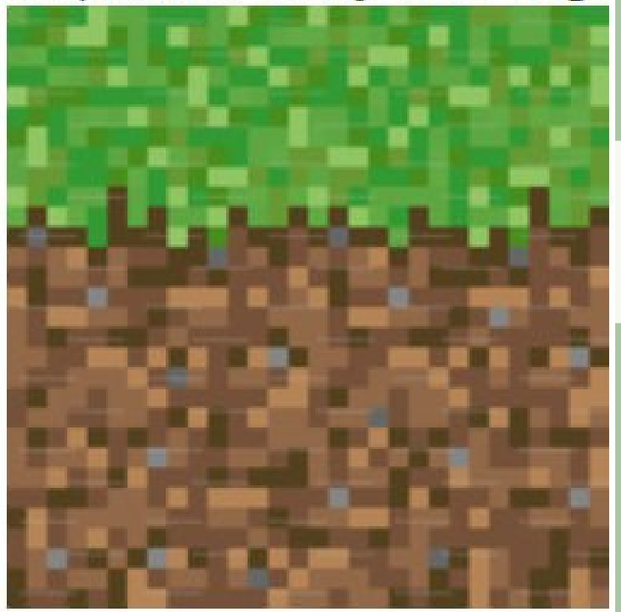
at first agent plant the plant, after that water it multiple times by suitable quantity based on weather,kill bug if we have.. finally when plant is ready to harvest it harvest it and out done.

Now we will look at an example of tomato policy
note: tomato age is 2 that means it will harvest after 3 times of watering

For Tomato

1.

obs: (0, 0, 0, 0)
No plant & Sunny & No bugs



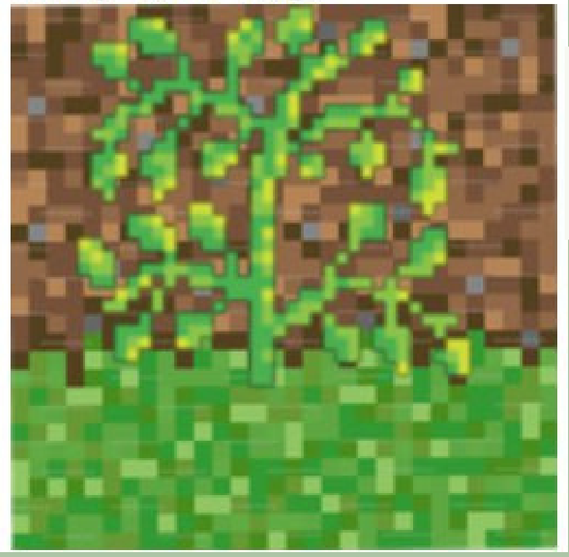
(A:0) plant

Observations :

**(NO plants,
Growth Stage 0 ,
NO Bug ,
Sunny)**

2.

action: 0
obs: (1, 0, 0, 0)
reward: 20
Sunny & No bugs



(A:4) water3L

Observations :

**(planted,
Growth Stage 0 ,
NO Bug ,
Sunny)**

3.

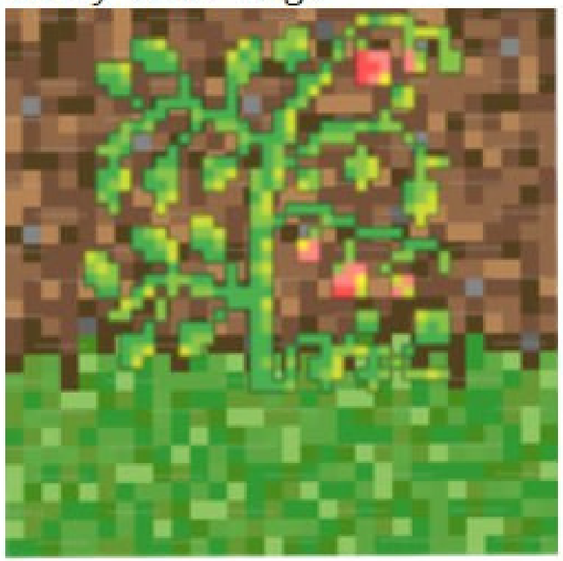
action: 4
obs: (1, 1, 1, 2)
reward: 10
Rainy & Bugs!!

Observations :

**(planted,
Growth Stage 1
,Bug ,
Rainy)**

4.

```
action:  1
obs:  (1, 1, 0, 2)
reward:  10
Rainy & No bugs
```



(A:1) kill bug

(A:2) water2L

Observations :

**(planted,
Growth Stage 1 ,
NO Bug ,
Rainy)**

5.

```
action:  2
obs:  (1, 2, 0, 1)
reward:  10
Normal & No bugs
```

(A:5) Harvest

Observations :

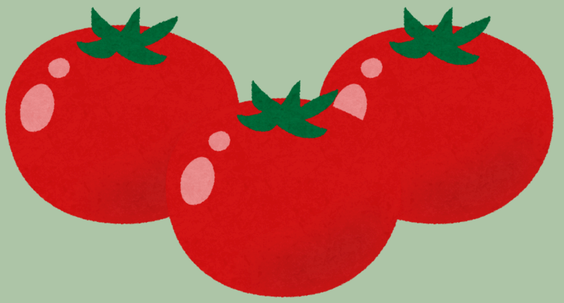
**(planted,
Growth Stage 2 ,
NO Bug ,
Normal)**

Done.

Then the soil returns empty

```
action:  5
obs:  (0, 0, 0, 1)
reward:  100
Episode done!
```

Fruits!!



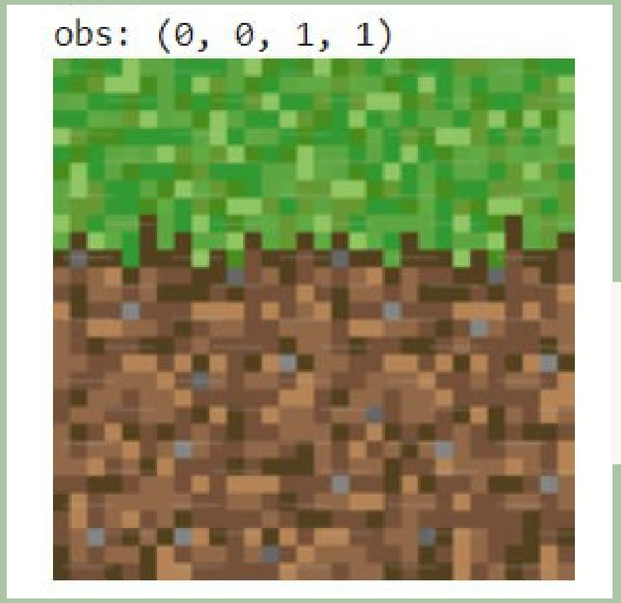
Observations :

(No plants,
Growth Stage 0 ,
NO Bug ,
Normal)

action(t+1) / obs(t)	Plant(a0)	Kill Bug(a1)	Water 1L(a2)	Water 2L(a3)	water 3L(a4)	Harvest(a5)
(0, 0, 0,0)	✓					
(1, 0, 0, 0)					✓	
(1, 1, 1, 2)		✓				
(1, 1, 0, 2)			✓			
(1, 2, 0, 1)						✓

For Corn

1.

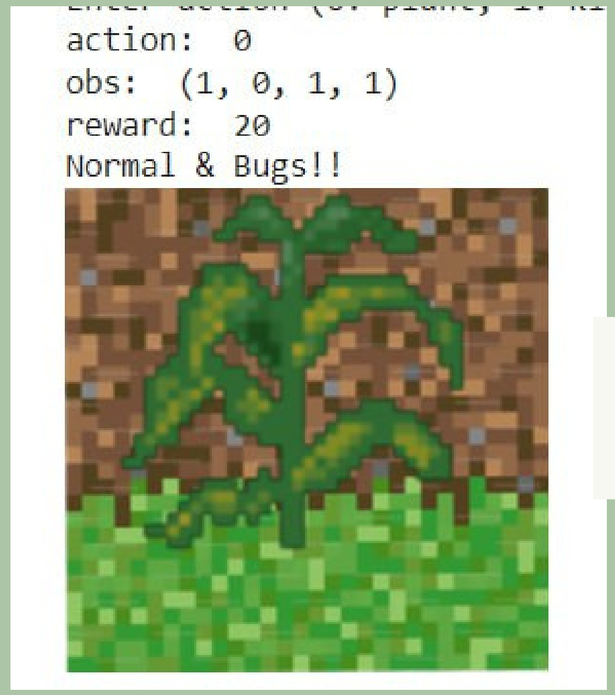


Observations :

**(NO planted,
Growth Stage 0 ,
Bug ,
Rainy)**

(A:0) plant

2.

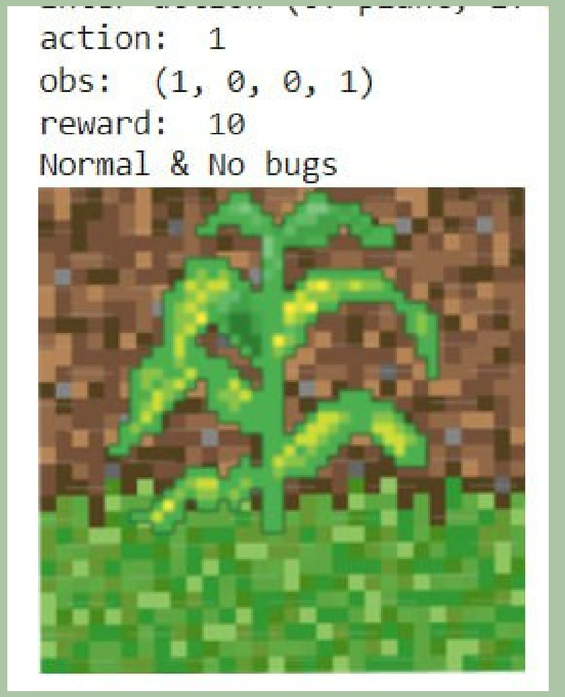


Observations :

**(planted,
Growth Stage 0 ,
Bug ,
Normal)**

(A:1) kill bug

3.



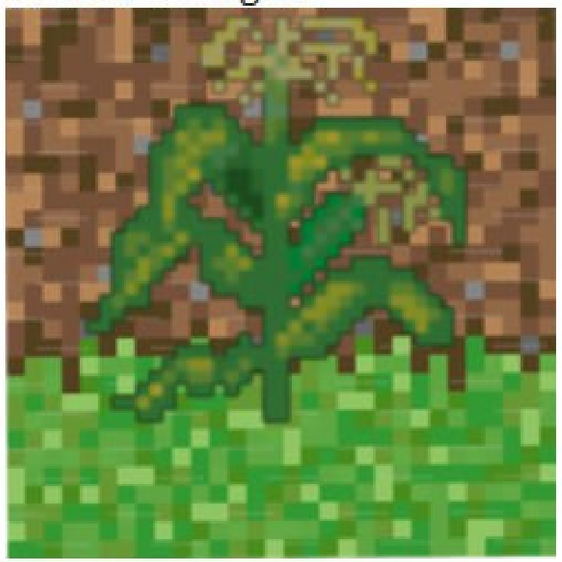
Observations :

**(planted,
Growth Stage 0 ,
NO Bug ,
Normal)**

(A:3) water 2L

4.

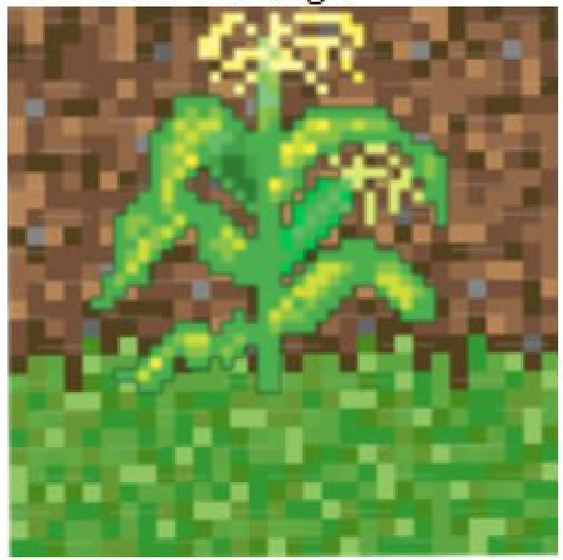
```
action:  3
obs:  (1, 1, 1, 1)
reward:  10
Normal & Bugs!!
```



(A:1) kill bug

5.

```
action:  1
obs:  (1, 1, 0, 1)
reward:  10
Normal & No bugs
```



(A:3)water 2L

Observations :

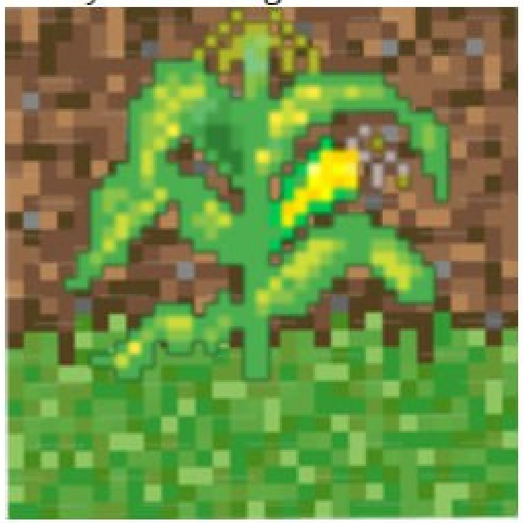
(planted,
Growth Stage 1 ,
Bug ,
Normal)

Observations :

(planted,
Growth Stage 1 ,
NO Bug ,
Normal)

6.

```
action:  3
obs:  (1, 2, 0, 0)
reward:  10
Sunny & No bugs
```



(A:3)water 2L

(A:4) water 3L

Observations :

(planted,
Growth Stage 2 ,
No Bug ,
Sunny)

7.

```
action:  4
obs:  (1, 3, 1, 2)
reward:  10
Rainy & Bugs!!
```

(A:1) kill bug

Observations :

(planted,
Growth Stage 3 ,
Bug ,
Rainy)

8.

(A:1) kill bug

```
action:  1
obs:  (1, 3, 0, 2)
reward:  10
Rainy & No bugs
```

Observations :

**(planted,
Growth Stage 3 ,
NO Bug ,
Rainy)**

(A:5) Harvest and return the Soil

Done.

```
action:  5
obs:  (0, 0, 0, 2)
reward:  100
Episode done!
```

Observations :

**(No plants,
Growth Stage 0 ,
NO Bug ,
Rainy)**

Fruits!!

obs(t) \ action(t+1)	Plant(a0)	Kill Bug(a1)	Water 1L(a2)	Water 2L(a3)	water 3L(a4)	Harvest(a5)
(0, 0, 1 , 1)	✓					
(1, 0, 1, 1)		✓				
(1, 0,0, 1)				✓		
(1 , 1 , 1 ,1)		✓				
(1, 1 , 0 , 1)				✓		
(1, 2, 0, 0)					✓	
(1 , 3 , 1 , 2)		✓				
(1 , 3 , 0 , 2)						

For Bean

1.

obs: (0, 0, 1, 1)

(A:0) plant

2.

action: 0
obs: (1, 0, 1, 1)
reward: 20
Normal & Bugs!!

(A:1) kill bug

3.

action: 1
obs: (1, 0, 0, 1)
reward: 10
Normal & No bugs

Observations :

**(NO planted,
Growth Stage 0 ,
Bug ,
Rainy)**

Observations :

**(planted,
Growth Stage 0 ,
Bug ,
Normal)**

Observations :

**(planted,
Growth Stage 0 ,
NO Bug ,
Normal)**

4.

```
action:  3
obs:  (1, 1, 0, 1)
reward:  10
Normal & No bugs
```

(A:3) water 2L

(A:3) water 2L

Observations :

(planted,
Growth Stage 1 ,
Bug ,
Normal)

5.

```
action:  3
obs:  (1, 2, 0, 2)
reward:  10
Rainy & No bugs
```

(A:2)water 1L

Observations :

(planted,
Growth Stage 2 ,
No Bug ,
Rainy)

(A:2)water 1L

6.

```
action:  2
obs:  (1, 3, 1, 0)
reward:  10
Sunny & Bugs!!
```

Observations :

(planted,
Growth Stage 3 ,
Bug ,
Sunny)

(A:1) Kill Bug

7.

```
action:  1
obs:  (1, 3, 0, 0)
reward:  10
Sunny & No bugs
```

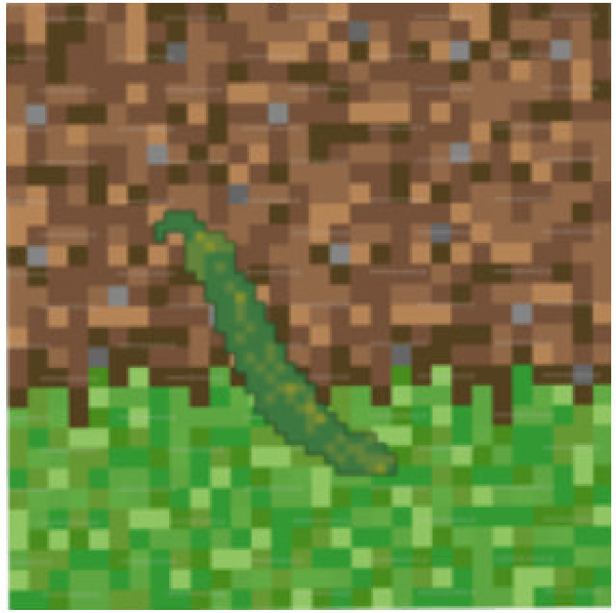
Observations :

(planted,
Growth Stage 3 ,
No Bug ,
Sunny)

(A:4) water 3L

8.

```
action:  4
obs:  (1, 4, 1, 1)
reward:  10
Normal & Bugs!!
```



(A:4) water 3L

(A:3) water 2L

Observations :

(planted,
Growth Stage 4 ,
Bug ,
Normal)

9.

```
action:  1
obs:  (1, 4, 0, 1)
reward:  10
Normal & No bugs
```

(A:5)

Observations :

(planted,
Growth Stage 4 ,
No Bug ,
Normal)

(A:5) Harvest and return the Soil

Done.

```
action:  5
obs:  (0, 0, 0, 1)
reward:  100
Episode done!
```

Fruits!!

<u>Observations :</u>

(No plants,
Growth Stage 0 ,
NO Bug ,
Normal)

actions(t+1) / obs(t)	Plant(a0)	Kill Bug(a1)	Water 1L(a2)	Water 2L(a3)	water 3L(a4)	Harvest(a5)
(0, 0, 1 , 1)	✓					
(1, 0, 1, 1)		✓				
(1, 0,0, 1)				✓		
(1 , 0 , 1 ,1)				✓		
(1, 2 , 0 , 2)			✓			
(1, 3 , 1 , 0)		✓				
(1, 3, 0, 0)					✓	
(1 , 4 , 1 , 1)				✓		
(1 , 4 , 0 , 1)						✓

Problem Definition:-

7.The PPO Algorithm:

- the (PPO) algorithm to train an actor-critic model. This policy optimizes the actor's policy using a clipped surrogate objective to improve stability and performance in reinforcement learning tasks. The actor network learns to choose actions from our actions(plant,kill pests,water1l or 2l or 3l and harvest), while the critic network estimates the value function to guide the learning process. The policy training iterates to maximize rewards while ensuring a controlled policy update through the PPO clipping mechanism.

Problem Definition:-

8.Input and Output:

- **Input**

the input to the algorithm consists of observations of the environment, such as soil condition, plant stage, presence of bugs, and weather condition. These observations are used by the neural network model (ActorCriticNetwork) to predict the policy logits (probability distribution over actions) and the value function (estimate of expected return) for each observation.

Problem Definition:-

8.Input and Output:

- **The predicted output:**

1. **Policy Logits: The probability distribution over possible actions given the current state of the environment.**
2. **Value Function: An estimate of the expected return or future rewards associated with being in the current state.**

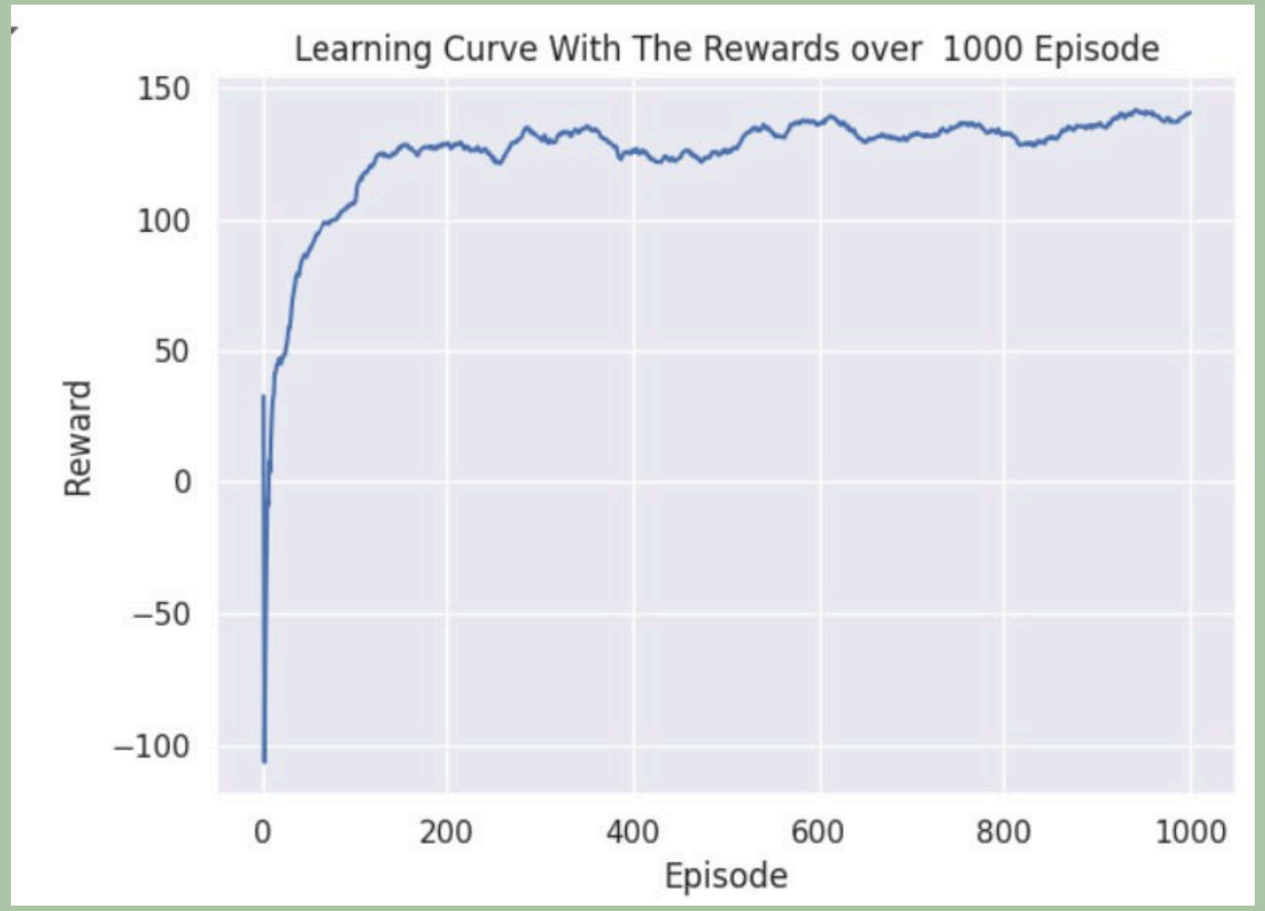
During training, the PPO algorithm aims to improve both the policy and value function by iteratively updating the neural network parameters based on the observed rewards and the advantage estimates (GAEs). This process involves maximizing the likelihood of actions with higher rewards (policy optimization) while minimizing the difference between predicted and observed returns (value function optimization).

Problem Definition:-

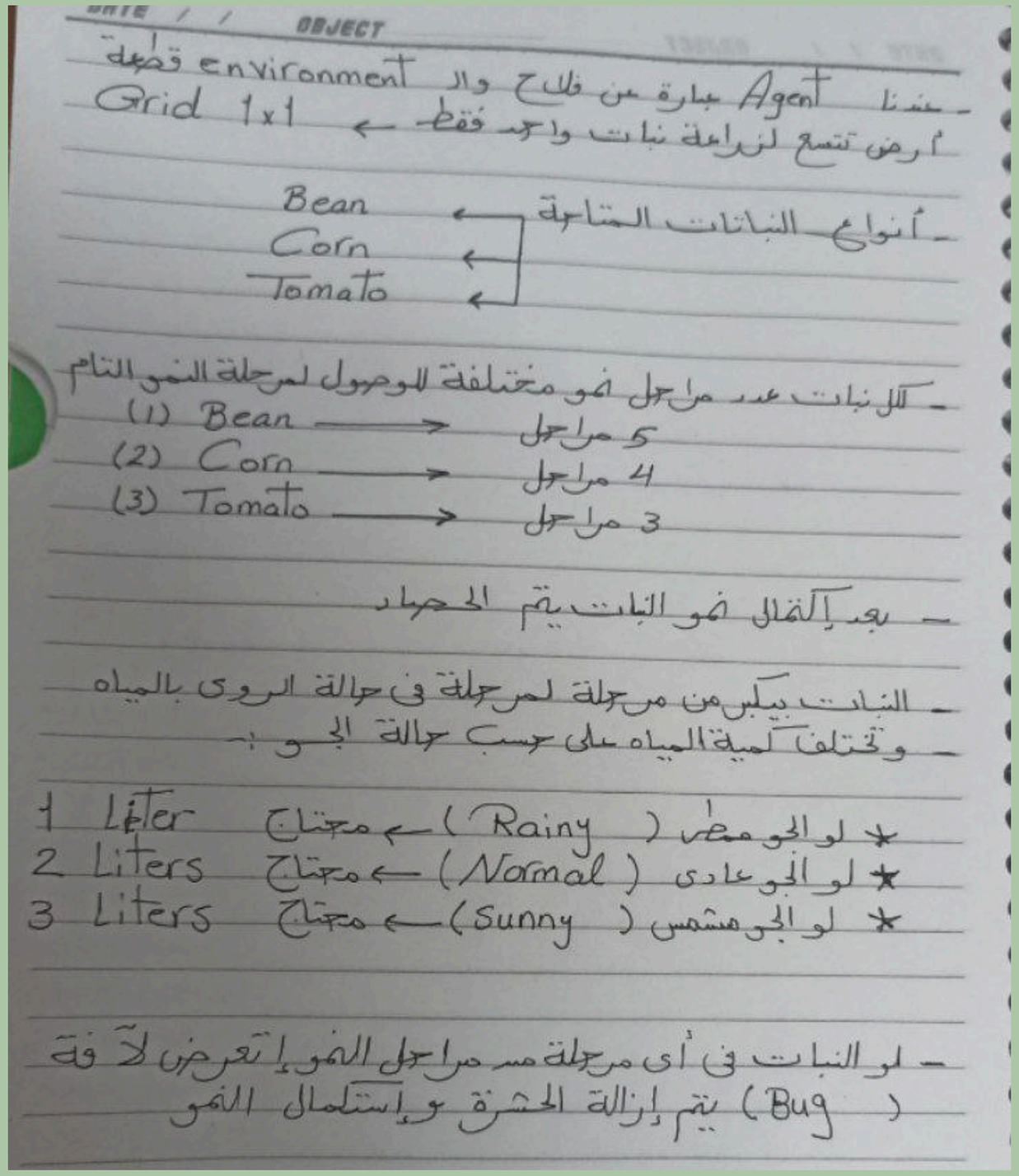
8.Input and Output:

- **The predicted output:**

3.The learning curve plotted at the end of the code provides a visualization of how the agent's performance (reward) evolves over the training episodes.



Summary


DATE / / OBJECT
- عندنا Agent عبارة عن فلاح والـ environment قطعة أرض تتسع لزراعة نبات واحد فقط ← Grid 1x1
- أنواع النباتات المتاحة
Bean
Corn
Tomato
- كل نبات عدد مراحل نمو مختلفة للوصول لمرحلة النمو التام
(1) Bean ← 5 مراحل
(2) Corn ← 4 مراحل
(3) Tomato ← 3 مراحل
- بعد اكتمال نمو النبات يتم الحصاد
- النبات بيكبر من مرحلة لمرحلة في حالة الروى بالمياه
- وتختلف كمية المياه على حسب حالة الجو :-
* لو الجو ممطر (Rainy) ← محتاج 1 Liter
* لو الجو عادى (Normal) ← محتاج 2 Liters
* لو الجو مشمس (Sunny) ← محتاج 3 Liters
- لو النبات في أى مرحلة من مراحل النمو إتعرض لآفة (Bug) يتم إزالة الحشرة وإستكمال النمو

Declare our code:

Plant

Goal: Simulate an environment for reinforcement learning.
Steps:
Initialize the environment with specified dynamics.
Provide methods to interact with the environment (e.g., step for taking actions and receiving observations and rewards).

```
import random
from gym import Env
from gym.spaces import Discrete, Tuple, Box
import gym
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from PIL import Image
from IPython.display import display
from PIL import Image

sns.set()



class Plant(Env):
    def __init__(self,plant):
```

```python
def __init__(self,plant):

    # Define action space: Plant, Kill Bug, Water 1L, Water 2L, Water 3L, Harvest
    self.action_space = Discrete(6)

    # Define observation space: soil condition (empty/planted), Plant growin, presence of bugs (0/1), weather condotion(0/2)
    self.observation_space = Tuple((
        Discrete(2),  # Soil condition: 0 for empty, 1 for planted
        Discrete(5),  # Plant growin: 0 for small, 1 for gester, 2 for bigger , .....
        Discrete(2),  # Presence of bugs: 0 for no bugs, 1 for bugs
        Discrete(2),  # weather condotion : 0 for sunny, 1 for normal, 2 for rainy
    ))



    # Initialize state variables
    # plant age 0 for tomato , 1 for corn , 2 for bean
    if plant == "tomato" :
        self.age= 2
    elif plant == "corn" :
        self.age = 3
    elif plant == "bean" :
        self.age = 4
    else :
        print("There is only three plants: (Tomato, Corn, Bean)!!")
    self.soil_condition = 0  # 0 for empty, 1 for planted
    self.plant_stage = 0     # 0 for small plant, 1 for gester, 2 for fully grown
    self.bugs_presence = 0   # 0 for no bugs, 1 for bugs
    self.Harvest = 0  # Set start Harvest
    self.weather_condotion =0
```

reset

Goal:Reset the environment to its initial state.
Steps:
Reset the internal state of the environment.
Return the initial observation.

```
def reset(self):

    # initial observation
    self.soil_condition = 0
    self.plant_stage = 0
    self.bugs_presence = random.randint(0, 1)
    self.weather_condotion = random.randint(0, 2)
    self.Harvest = 0
    observation = (self.soil_condition, self.plant_stage,  self.bugs_presence,self.weather_condotion)

    return observation
```

step

Goal: Take an action in the environment and observe the outcome.
Steps:
Receive an action from the agent.
Apply the action to the environment and observe the next state, reward, and termination signal.
Return the next state, reward, and termination signal.

```
def step(self, action):
    # Update the environment based on the action

    reward = 0  # Default reward
    # Plant action
    if action == 0:
        if self.soil_condition == 0:  # Check if soil is empty before planting
            self.soil_condition = 1
            reward += 20  # Reward for planting
        else:
             reward -= 0  # Penalty for planting in non-empty soil

    # Kill bug action
    elif action == 1:
        if self.bugs_presence == 1:
            self.bugs_presence = 0  # Remove bugs if present
            reward += 10  # Reward for killing bugs
        else:
            reward = 0  # Penalty for attempting to kill non-existent bugs

    # Water 1L action
    elif action == 2:
        if self.soil_condition == 1 and self.plant_stage < self.age:
            if self.bugs_presence == 0 :
                if self.weather_condotion== 2 :
                    self.plant_stage += 1
                    self.bugs_presence = random.randint(0, 1)
                    self.weather_condotion = random.randint(0, 2)
                    reward += 10  # Reward for appropriate watering the plant
                else :
                    self.bugs_presence = random.randint(0, 1)
                    self.weather_condotion = random.randint(0, 2)
                    reward -= 3  # Penalty for Inappropriate watering the plant
```

```
        else :
            self.plant_stage += 1
            reward = -1  # Penalty for watering Plant infested with bugs

    else:
        reward = -1  # Penalty for watering unplanted soil or fully grown plant

# Water 2L action
elif action == 3:
    if self.soil_condition == 1 and self.plant_stage < self.age:
        if self.bugs_presence == 0 :
            if self.weather_condotion== 1 :
                self.plant_stage += 1
                self.bugs_presence = random.randint(0, 1)
                self.weather_condotion = random.randint(0, 2)
                reward += 10  # Reward for appropriate watering the plant
            elif self.weather_condotion== 2 :
                self.plant_stage += 1
                self.bugs_presence = random.randint(0, 1)
                self.weather_condotion = random.randint(0, 2)
                reward -= 1 # Penalty for Inappropriate watering the plant
            else :
                self.bugs_presence = random.randint(0, 1)
                self.weather_condotion = random.randint(0, 2)
                reward -= 3  # Penalty for Inappropriate watering the plant

        else :
            self.plant_stage += 1
            reward = -1  # Penalty for watering Plant infested with bugs

    else:
        reward = -1  # Penalty for watering unplanted soil or fully grown plant
```

```
            reward = -1  # Penalty for watering unplanted soil or fully grown plant

    # Water 3L action
    elif action == 4:
        if self.soil_condition == 1 and self.plant_stage < self.age:
            if self.bugs_presence == 0 :
                if self.weather_condotion== 0 :
                    self.plant_stage += 1
                    self.bugs_presence = random.randint(0, 1)
                    self.weather_condotion = random.randint(0, 2)
                    reward += 10  # Reward for appropriate watering the plant
                else :
                    self.plant_stage += 1
                    self.bugs_presence = random.randint(0, 1)
                    self.weather_condotion = random.randint(0, 2)
                    reward -= 3  # Penalty for Inappropriate watering the plant
            else :
                self.plant_stage += 1
                reward = -1  # Penalty for watering Plant infested with bugs

        else:
            reward = -1  # Penalty for watering unplanted soil or fully grown plant

    # Harvest action
    elif action == 5:
        if self.soil_condition == 1 and self.plant_stage == self.age:
            if self.bugs_presence == 0 :
                reward += 100  # Reward for harvesting a fully grown plant
                self.soil_condition = 0  # Clear soil after harvesting
                self.Harvest = 1 # plant growth stage reaches the end
                self.plant_stage = 0
            else :
                reward -= 10  # Penalty for harvesting a fully grown Plant infested with bugs
                self.soil_condition = 0  # Clear soil after harvesting
```

```
            self.soil_condition = 0  # Clear soil after harvesting
            self.plant_stage = 0
    else:
        self.soil_condition = 0
        self.plant_stage = 0
        reward -= 5  # Penalty for attempting to harvest before plant is fully grown


# Episode done if Harvest done
done = self.Harvest

observation = (self.soil_condition, self.plant_stage,  self.bugs_presence,self.weather_condotion)

# Return observation, reward, done, info
return observation, reward, done, {}
```

render

Goal: Visualize the current state of the environment.
Steps:
Display the current state of the environment in a graphical or textual format.
Optionally, provide controls for user interaction or observation.

```
# Visualize the environment
def render(self):

    width, height = 200, 200  # Specify the desired width and height

    if self.age == 2:
      if self.soil_condition == 1:
        if self.bugs_presence == 0:
          if self.plant_stage == 0:
           if self.weather_condotion == 0:
            image_path = "/content/drive/MyDrive/farmm/tomato/so_small_tomato.png"
            image = Image.open(image_path)
            resized_image = image.resize((width, height))
            print("Sunny & No bugs")
            display(resized_image)
           elif self.weather_condotion == 1:
            image_path30 = "/content/drive/MyDrive/farmm/tomato/so_small_tomato.png"
            image30 = Image.open(image_path30)
            resized_image30 = image30.resize((width, height))
            print("Normal & No bugs")
            display(resized_image30)
           else:
            image_path31 = "/content/drive/MyDrive/farmm/tomato/so_small_tomato.png"
            image31 = Image.open(image_path31)
            resized_image31 = image31.resize((width, height))
            print("Rainy & No bugs")
            display(resized_image31)
          elif self.plant_stage == 1:
            if self.weather_condotion == 0:
              image_path1 = "/content/drive/MyDrive/farmm/tomato/gesture_tomato.png"
              image1 = Image.open(image_path1)
              resized_image1 = image1.resize((width, height))
              print("Sunny & No bugs")
```

```
      else:
        image_path39 = "/content/drive/MyDrive/farmm/tomato/gesture_bad_tomato.png"
        image39 = Image.open(image_path39)
        resized_image39 = image39.resize((width, height))
        print("Rainy & Bugs!!")
        display(resized_image39)
    elif self.plant_stage == 2:
      if self.weather_condotion == 0:
        image_path6 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image6 = Image.open(image_path6)
        resized_image6 = image6.resize((width, height))
        print("Sunny & Bugs!!")
        display(resized_image6)
      elif self.weather_condotion == 1:
        image_path40 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image40 = Image.open(image_path40)
        resized_image40 = image40.resize((width, height))
        print("Normal & Bugs!!")
        display(resized_image40)
      else:
        image_path41 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image41 = Image.open(image_path41)
        resized_image41 = image41.resize((width, height))
        print("Rainy & Bugs!!")
        display(resized_image41)
else:
  image_path3 = "/content/drive/MyDrive/farmm/soil.png"
  image3 = Image.open(image_path3)
  resized_image3 = image3.resize((width, height))  # Resize the image
  display(resized_image3)
```

```
else:
  if self.plant_stage == 0:
    if self.weather_condotion == 0:
      image_path4 = "/content/drive/MyDrive/farmm/tomato/so_small_bad_tomato.png"
      image4 = Image.open(image_path4)
      resized_image4 = image4.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image4)
    elif self.weather_condotion == 1:
      image_path36 = "/content/drive/MyDrive/farmm/tomato/so_small_bad_tomato.png"
      image36 = Image.open(image_path36)
      resized_image36 = image36.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image36)
    else:
      image_path37 = "/content/drive/MyDrive/farmm/tomato/so_small_bad_tomato.png"
      image37 = Image.open(image_path37)
      resized_image37 = image37.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image37)
  elif self.plant_stage == 1:
    if self.weather_condotion == 0:
      image_path5 = "/content/drive/MyDrive/farmm/tomato/gesture_bad_tomato.png"
      image5 = Image.open(image_path5)
      resized_image5 = image5.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image5)
    elif self.weather_condotion == 1:
      image_path38 = "/content/drive/MyDrive/farmm/tomato/gesture_bad_tomato.png"
      image38 = Image.open(image_path38)
      resized_image38 = image38.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image38)
```

```
      else:
        image_path39 = "/content/drive/MyDrive/farmm/tomato/gesture_bad_tomato.png"
        image39 = Image.open(image_path39)
        resized_image39 = image39.resize((width, height))
        print("Rainy & Bugs!!")
        display(resized_image39)
    elif self.plant_stage == 2:
      if self.weather_condotion == 0:
        image_path6 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image6 = Image.open(image_path6)
        resized_image6 = image6.resize((width, height))
        print("Sunny & Bugs!!")
        display(resized_image6)
      elif self.weather_condotion == 1:
        image_path40 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image40 = Image.open(image_path40)
        resized_image40 = image40.resize((width, height))
        print("Normal & Bugs!!")
        display(resized_image40)
      else:
        image_path41 = "/content/drive/MyDrive/farmm/tomato/full_grown_bad_tomato.png"
        image41 = Image.open(image_path41)
        resized_image41 = image41.resize((width, height))
        print("Rainy & Bugs!!")
        display(resized_image41)
else:
  image_path3 = "/content/drive/MyDrive/farmm/soil.png"
  image3 = Image.open(image_path3)
  resized_image3 = image3.resize((width, height))  # Resize the image
  display(resized_image3)
```

```
elif self.age == 3:
  if self.soil_condition == 1:
    if self.bugs_presence == 0:
      if self.plant_stage == 0:
        if self.weather_condotion == 0:
          image_path7 = "/content/drive/MyDrive/farmm/corn/so_small_corn.png"
          image7 = Image.open(image_path7)
          resized_image7 = image7.resize((width, height))
          print("Sunny & No bugs")
          display(resized_image7)
        elif self.weather_condotion == 1:
          image_path42 = "/content/drive/MyDrive/farmm/corn/so_small_corn.png"
          image42 = Image.open(image_path42)
          resized_image42 = image42.resize((width, height))
          print("Normal & No bugs")
          display(resized_image42)
        else:
          image_path43 = "/content/drive/MyDrive/farmm/corn/so_small_corn.png"
          image43 = Image.open(image_path43)
          resized_image43 = image43.resize((width, height))
          print("Rainy & No bugs")
          display(resized_image43)
      elif self.plant_stage == 1:
        if self.weather_condotion == 0:
          image_path8 = "/content/drive/MyDrive/farmm/corn/small_corn.png"
          image8 = Image.open(image_path8)
          resized_image8 = image8.resize((width, height))
          print("Sunny & No bugs")
          display(resized_image8)
        elif self.weather_condotion == 1:
          image_path44 = "/content/drive/MyDrive/farmm/corn/small_corn.png"
          image44 = Image.open(image_path44)
          resized_image44 = image44.resize((width, height))
          print("Normal & No bugs")
```

```
        display(resized_image44)
      else:
        image_path45 = "/content/drive/MyDrive/farmm/corn/small_corn.png"
        image45 = Image.open(image_path45)
        resized_image45 = image45.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image45)
    elif self.plant_stage == 2:
      if self.weather_condotion == 0:
        image_path9 = "/content/drive/MyDrive/farmm/corn/gesture_corn.png"
        image9 = Image.open(image_path9)
        resized_image9 = image9.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image9)
      elif self.weather_condotion == 1:
        image_path46 = "/content/drive/MyDrive/farmm/corn/gesture_corn.png"
        image46 = Image.open(image_path46)
        resized_image46 = image46.resize((width, height))
        print("Normal & No bugs")
        display(resized_image46)
      else:
        image_path47 = "/content/drive/MyDrive/farmm/corn/gesture_corn.png"
        image47 = Image.open(image_path47)
        resized_image47 = image47.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image47)
    elif self.plant_stage == 3:
      if self.weather_condotion == 0:
        image_path10 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
        image10 = Image.open(image_path10)
        resized_image10 = image10.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image10)
      elif self.weather_condotion == 1:
        image_path48 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
        image48 = Image.open(image_path48)
        resized_image48 = image48.resize((width, height))
        print("Normal & No bugs")
        display(resized_image48)
      else:
        image_path49 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
        image49 = Image.open(image_path49)
        resized_image49 = image49.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image49)
    else:
      if self.plant_stage == 0:
        if self.weather_condotion == 0:
          image_path11 = "/content/drive/MyDrive/farmm/corn/so_small_bad_corn.png"
          image11 = Image.open(image_path11)
          resized_image11 = image11.resize((width, height))
          print("Sunny & Bugs!!")
          display(resized_image11)
        elif self.weather_condotion == 1:
          image_path50 = "/content/drive/MyDrive/farmm/corn/so_small_bad_corn.png"
          image50 = Image.open(image_path50)
          resized_image50 = image50.resize((width, height))
          print("Normal & Bugs!!")
          display(resized_image50)
        else:
          image_path51 = "/content/drive/MyDrive/farmm/corn/so_small_bad_corn.png"
          image51 = Image.open(image_path51)
          resized_image51 = image51.resize((width, height))
          print("Rainy & Bugs!!")
          display(resized_image51)
      elif self.plant_stage == 1:
        if self.weather_condotion == 0:
```

```
elif self.plant_stage == 1:
  if self.weather_condotion == 0:
    image_path12 = "/content/drive/MyDrive/farmm/corn/small_bad_corn.png"
    image12 = Image.open(image_path12)
    resized_image12 = image12.resize((width, height))
    print("Sunny & Bugs!!")
    display(resized_image12)
  elif self.weather_condotion == 1:
    image_path52 = "/content/drive/MyDrive/farmm/corn/small_bad_corn.png"
    image52 = Image.open(image_path52)
    resized_image52 = image52.resize((width, height))
    print("Normal & Bugs!!")
    display(resized_image52)
  else:
    image_path53 = "/content/drive/MyDrive/farmm/corn/small_bad_corn.png"
    image53 = Image.open(image_path53)
    resized_image53 = image53.resize((width, height))
    print("Rainy & Bugs!!")
    display(resized_image53)
elif self.plant_stage == 2:
  if self.weather_condotion == 0:
    image_path13 = "/content/drive/MyDrive/farmm/corn/gesture_bad_corn.png"
    image13 = Image.open(image_path13)
    resized_image13 = image13.resize((width, height))
    print("Sunny & Bugs!!")
    display(resized_image13)
  elif self.weather_condotion == 1:
    image_path54 = "/content/drive/MyDrive/farmm/corn/gesture_bad_corn.png"
    image54 = Image.open(image_path54)
    resized_image54 = image54.resize((width, height))
    print("Normal & Bugs!!")
    display(resized_image54)
```

```
    else:
      image_path55 = "/content/drive/MyDrive/farmm/corn/gesture_bad_corn.png"
      image55 = Image.open(image_path55)
      resized_image55 = image55.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image55)
  elif self.plant_stage == 3:
    if self.weather_condotion == 0:
      image_path23 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
      image23 = Image.open(image_path23)
      resized_image23 = image23.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image23)
    elif self.weather_condotion == 1:
      image_path56 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
      image56 = Image.open(image_path56)
      resized_image56 = image56.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image56)
    else:
      image_path57 = "/content/drive/MyDrive/farmm/corn/fully_corn.png"
      image57 = Image.open(image_path57)
      resized_image57 = image57.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image57)
else:
  image_path14 = "/content/drive/MyDrive/farmm/soil.png"
  image14 = Image.open(image_path14)
  resized_image14 = image14.resize((width, height))  # Resize the image
  display(resized_image14)

elif self.age == 4:
  if self.soil_condition == 1:
```

```
if self.soil_condition == 1:
  if self.bugs_presence == 0:
    if self.plant_stage == 0:
      if self.weather_condotion == 0:
        image_path15 = "/content/drive/MyDrive/farmm/bean/so_small_bean.png"
        image15 = Image.open(image_path15)
        resized_image15 = image15.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image15)
      elif self.weather_condotion == 1:
        image_path58 = "/content/drive/MyDrive/farmm/bean/so_small_bean.png"
        image58 = Image.open(image_path58)
        resized_image58 = image58.resize((width, height))
        print("Normal & No bugs")
        display(resized_image58)
      else:
        image_path59 = "/content/drive/MyDrive/farmm/bean/so_small_bean.png"
        image59 = Image.open(image_path59)
        resized_image59 = image59.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image59)
    elif self.plant_stage == 1:
      if self.weather_condotion == 0:
        image_path16 = "/content/drive/MyDrive/farmm/bean/small_bean.png"
        image16 = Image.open(image_path16)
        resized_image16 = image16.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image16)
      elif self.weather_condotion == 1:
        image_path60 = "/content/drive/MyDrive/farmm/bean/small_bean.png"
        image60 = Image.open(image_path60)
        resized_image60 = image60.resize((width, height))
        print("Normal & No bugs")
        display(resized_image60)
```

```
        display(resized_image60)
      else:
        image_path61 = "/content/drive/MyDrive/farmm/bean/small_bean.png"
        image61 = Image.open(image_path61)
        resized_image61 = image61.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image61)
    elif self.plant_stage == 2:
      if self.weather_condotion == 0:
        image_path17 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
        image17 = Image.open(image_path17)
        resized_image17 = image17.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image17)
      elif self.weather_condotion == 1:
        image_path62 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
        image62 = Image.open(image_path62)
        resized_image62 = image62.resize((width, height))
        print("Normal & No bugs")
        display(resized_image62)
      else:
        image_path63 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
        image63 = Image.open(image_path63)
        resized_image63 = image63.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image63)
    elif self.plant_stage == 3:
      if self.weather_condotion == 0:
        if self.weather_condotion == 0:
          image_path18 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
          image18 = Image.open(image_path18)
          resized_image18 = image18.resize((width, height))
          print("Sunny & No bugs")
          display(resized_image18)
```

```
        display(resized_image18)
      elif self.weather_condotion == 1:
        image_path64 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
        image64 = Image.open(image_path64)
        resized_image64 = image64.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image64)
      else:
        image_path65 = "/content/drive/MyDrive/farmm/bean/gesture_bean.png"
        image65 = Image.open(image_path65)
        resized_image65 = image65.resize((width, height))
        print("Normal & No bugs")
        display(resized_image65)
    elif self.plant_stage == 4:
      if self.weather_condotion == 0:
        image_path24 = "/content/drive/MyDrive/farmm/bean/fully_bean.png"
        image24 = Image.open(image_path24)
        resized_image24 = image24.resize((width, height))
        print("Sunny & No bugs")
        display(resized_image24)
      elif self.weather_condotion == 1:
        image_path66 = "/content/drive/MyDrive/farmm/bean/fully_bean.png"
        image66 = Image.open(image_path66)
        resized_image66 = image66.resize((width, height))
        print("Normal & No bugs")
        display(resized_image66)
      else:
        image_path67 = "/content/drive/MyDrive/farmm/bean/fully_bean.png"
        image67 = Image.open(image_path67)
        resized_image67 = image67.resize((width, height))
        print("Rainy & No bugs")
        display(resized_image67)
    else:
```

```
      else:
        if self.plant_stage == 0:
          if self.weather_condotion == 0:
            image_path19 = "/content/drive/MyDrive/farmm/bean/so_small_bad_bean.png"
            image19 = Image.open(image_path19)
            resized_image19 = image19.resize((width, height))
            print("Sunny & Bugs!!")
            display(resized_image19)
          elif self.weather_condotion == 1:
            image_path68 = "/content/drive/MyDrive/farmm/bean/so_small_bad_bean.png"
            image68 = Image.open(image_path68)
            resized_image68 = image68.resize((width, height))
            print("Normal & Bugs!!")
            display(resized_image68)
          else:
            image_path69 = "/content/drive/MyDrive/farmm/bean/so_small_bad_bean.png"
            image69 = Image.open(image_path69)
            resized_image69 = image69.resize((width, height))
            print("Rainy & Bugs!!")
            display(resized_image69)
        elif self.plant_stage == 1:
          if self.weather_condotion == 0:
            image_path20 = "/content/drive/MyDrive/farmm/bean/small_bad_bean.png"
            image20 = Image.open(image_path20)
            resized_image20 = image20.resize((width, height))
            print("Sunny & Bugs!!")
            display(resized_image20)
          elif self.weather_condotion == 1:
            image_path70 = "/content/drive/MyDrive/farmm/bean/small_bad_bean.png"
            image70 = Image.open(image_path70)
            resized_image70 = image70.resize((width, height))
            print("Normal & Bugs!!")
            display(resized_image70)
```

```
    else:
      image_path71 = "/content/drive/MyDrive/farmm/bean/small_bad_bean.png"
      image71 = Image.open(image_path71)
      resized_image71 = image71.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image71)
  elif self.plant_stage == 2:
    if self.weather_condotion == 0:
      image_path21 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image21 = Image.open(image_path21)
      resized_image21 = image21.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image21)
    elif self.weather_condotion == 1:
      image_path72 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image72 = Image.open(image_path72)
      resized_image72 = image72.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image72)
    else:
      image_path73 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image73 = Image.open(image_path73)
      resized_image73 = image73.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image73)
  elif self.plant_stage == 3:
    if self.weather_condotion == 0:
      image_path25 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image25 = Image.open(image_path25)
      resized_image25 = image25.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image25)
    elif self.weather_condotion == 1:
      image_path74 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
```

```
      image_path74 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image74 = Image.open(image_path74)
      resized_image74 = image74.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image74)
    else:
      image_path75 = "/content/drive/MyDrive/farmm/bean/gesture_bad_bean.png"
      image75 = Image.open(image_path75)
      resized_image75 = image75.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image75)
  elif self.plant_stage == 4:
    if self.weather_condotion == 0:
      image_path26 = "/content/drive/MyDrive/farmm/bean/fully_bad_bean.png"
      image26 = Image.open(image_path26)
      resized_image26 = image26.resize((width, height))
      print("Sunny & Bugs!!")
      display(resized_image26)
    elif self.weather_condotion == 1:
      image_path76 = "/content/drive/MyDrive/farmm/bean/fully_bad_bean.png"
      image76 = Image.open(image_path76)
      resized_image76 = image76.resize((width, height))
      print("Normal & Bugs!!")
      display(resized_image76)
    else:
      image_path77 = "/content/drive/MyDrive/farmm/bean/fully_bad_bean.png"
      image77 = Image.open(image_path77)
      resized_image77 = image77.resize((width, height))
      print("Rainy & Bugs!!")
      display(resized_image77)
```

```
    else:
        image_path22 = "/content/drive/MyDrive/farmm/soil.png"
        image22 = Image.open(image_path22)
        resized_image22 = image22.resize((width, height))  # Resize the image
        display(resized_image22)




else:
        image_path3 = "/content/drive/MyDrive/farmm/soil.png"
        image3 = Image.open(image_path3)
        resized_image3 = image3.resize((width, height))  # Resize the image
        display(resized_image3)
print()  # New line
```

test env

Goal: Get an action input from the user.

Steps:

a. **Prompt the user to enter an action.**

b. **Store the user input in action_input.**

This is an part just for interactive simulation experience and test the environment yourself.

```
print("Enter plant type (tomato, corn, or bean): Plant age at harvest: Tomatoes: 2, Corn: 3, Beans: 4.")
plant_type = input()
env = Plant(plant_type.lower())



obs = env.reset()
print("obs:", obs)
for _ in range(50):
    env.render()

    # Get user input for action
    action_input = input("Enter action (0: plant, 1: kill bug, 2: Water 1L, 3:Water 2L, 4:Water 3L, 5: harvest): ")
    action = int(action_input)

    obs, reward, done, _ = env.step(action)
    print("action: ", action)
    print("obs: ", obs)
    print("reward: ", reward)
    if done:
        print("Episode done!")
        break
```

ActorCriticNetwork

Goal: Design neural networks for policy and value functions.
Steps:
Define the architecture for both the actor (policy) and critic (value) neural networks.
Provide methods for forward pass (e.g., forward) to compute action probabilities and value estimates.

```python
import torch
from torch import nn
from torch import optim
from torch.distributions.categorical import Categorical

# Policy and value model
class ActorCriticNetwork(nn.Module):
  def __init__(self, obs_space_size, action_space_size):
    super().__init__()

    self.shared_layers = nn.Sequential(
        nn.Linear(obs_space_size, 64),
        nn.ReLU(),
        nn.Linear(64, 64),
        nn.ReLU())

    self.policy_layers = nn.Sequential(
        nn.Linear(64, 64),
        nn.ReLU(),
        nn.Linear(64, action_space_size))

    self.value_layers = nn.Sequential(
        nn.Linear(64, 64),
        nn.ReLU(),
        nn.Linear(64, 1))
```

```python
def value(self, obs):
    z = self.shared_layers(obs)
    value = self.value_layers(z)
    return value
```

```python
def policy(self, obs):
    z = self.shared_layers(obs)
    policy_logits = self.policy_layers(z)
    return policy_logits
```

```python
def forward(self, obs):
    z = self.shared_layers(obs)
    policy_logits = self.policy_layers(z)
    value = self.value_layers(z)
    return policy_logits, value
```

PPOTrainer

Goal: Initialize the PPO trainer with the actor-critic model and training parameters.
Steps:
Store the provided parameters and actor-critic model.
Set up optimizers for policy and value functions.

```
class PPOTrainer():
  def __init__(self,
               actor_critic,
               ppo_clip_val=0.2,
               target_kl_div=0.01,
               max_policy_train_iters=80,
               value_train_iters=80,
               policy_lr=3e-4,
               value_lr=1e-2):
    self.ac = actor_critic
    self.ppo_clip_val = ppo_clip_val
    self.target_kl_div = target_kl_div
    self.max_policy_train_iters = max_policy_train_iters
    self.value_train_iters = value_train_iters

    policy_params = list(self.ac.shared_layers.parameters()) + \
        list(self.ac.policy_layers.parameters())
    self.policy_optim = optim.Adam(policy_params, lr=policy_lr)

    value_params = list(self.ac.shared_layers.parameters()) + \
        list(self.ac.value_layers.parameters())
    self.value_optim = optim.Adam(value_params, lr=value_lr)
```

```python
def train_policy(self, obs, acts, old_log_probs, gaes):
  for _ in range(self.max_policy_train_iters):
    self.policy_optim.zero_grad()

    new_logits = self.ac.policy(obs)
    new_logits = Categorical(logits=new_logits)
    new_log_probs = new_logits.log_prob(acts)

    policy_ratio = torch.exp(new_log_probs - old_log_probs)
    clipped_ratio = policy_ratio.clamp(
        1 - self.ppo_clip_val, 1 + self.ppo_clip_val)

    clipped_loss = clipped_ratio * gaes
    full_loss = policy_ratio * gaes
    policy_loss = -torch.min(full_loss, clipped_loss).mean()

    policy_loss.backward()
    self.policy_optim.step()

    kl_div = (old_log_probs - new_log_probs).mean()
    if kl_div >= self.target_kl_div:
      break
```

```python
def train_value(self, obs, returns):
  for _ in range(self.value_train_iters):
    self.value_optim.zero_grad()

    values = self.ac.value(obs)
    value_loss = (returns - values) ** 2
    value_loss = value_loss.mean()

    value_loss.backward()
    self.value_optim.step()
```

discount_rewards

Goal: Apply discounting to rewards in a rollout for training.
Steps:
Receive a list of rewards from a rollout.
Apply discounting to each reward using a specified discount factor.

```python
def discount_rewards(rewards, gamma=0.99):
    """
    Return discounted rewards based on the given rewards and gamma param.
    """
    new_rewards = [float(rewards[-1])]
    for i in reversed(range(len(rewards)-1)):
        new_rewards.append(float(rewards[i]) + gamma * new_rewards[-1])
    return np.array(new_rewards[::-1])
```

calculate_gaes

Goal: Calculate Generalized Advantage Estimation (GAE) advantages.
Steps:
Receive a list of rewards, values, and done flags from a rollout.
Compute advantages using the Generalized Advantage Estimation (GAE) formula.

```python
def calculate_gaes(rewards, values, gamma=0.99, decay=0.97):
    """
    Return the General Advantage Estimates from the given rewards and values.
    """
    next_values = np.concatenate([values[1:], [0]])
    deltas = [rew + gamma * next_val - val for rew, val, next_val in zip(rewards, values, next_values)]

    gaes = [deltas[-1]]
    for i in reversed(range(len(deltas)-1)):
        gaes.append(deltas[i] + decay * gamma * gaes[-1])

    return np.array(gaes[::-1])
```

rollout

Goal: Generate rollouts (trajectories) in the environment for data collection.
Steps:
Reset the environment to its initial state.
Interact with the environment using the current policy to collect observations, actions, rewards, and next states.

```python
def rollout(model, env, max_steps=50):
    """
    Returns training data in the shape (n_steps, observation_shape)
    and the cumulative reward.
    """
    # Create data storage
    train_data = [[], [], [], [], []] # obs, act, reward, values, act_log_probs
    obs = env.reset()

    ep_reward = 0
    for _ in range(max_steps):
        logits, val = model(torch.tensor([obs], dtype=torch.float32))
        act_distribution = Categorical(logits=logits)
        act = act_distribution.sample()
        act_log_prob = act_distribution.log_prob(act).item()


        act, val = act.item(), val.item()
        #print(act)


        next_obs, reward, done, _ = env.step(act)


        for i, item in enumerate((obs, act, reward, val, act_log_prob)):
          train_data[i].append(item)

        obs = next_obs
        ep_reward += reward
        if done:
            #print("done")
            break
```

```
            break

    train_data = [np.asarray(x) for x in train_data]

    # Do train data filtering
    train_data[3] = calculate_gaes(train_data[2], train_data[3])

    return train_data, ep_reward
```

```
#env = Plant("corn")
env = Plant("tomato")
model = ActorCriticNetwork(len(env.observation_space), env.action_space.n)
train_data, reward = rollout(model, env)
reward
```

Main Training Loop

Goal: Train the PPO model over a number of episodes.
Steps:
Perform rollouts to collect training data.
Shuffle and prepare the training data.
Train the policy and value networks using the prepared data.
Record and print episode rewards.

```
# Define training params
n_episodes = 2000

ppo = PPOTrainer(
    model,
    policy_lr = 1e-1,
    value_lr = 1e-1,
    target_kl_div = 0.02,
    max_policy_train_iters = 100,
    value_train_iters = 100)
```

```python
ep_rewards = []
for episode_idx in range(n_episodes):
  # Perform rollout
  train_data, reward = rollout(model, env)
  ep_rewards.append(reward)
  print('Episode {} | Reward {:.1f}'.format(
         episode_idx + 1, reward))

  # Shuffle
  permute_idxs = np.random.permutation(len(train_data[0]))

  # Policy data
  obs = torch.tensor(train_data[0][permute_idxs],
                     dtype=torch.float32)

  acts = torch.tensor(train_data[1][permute_idxs],
                      dtype=torch.int32)
  #print(acts)
  gaes = torch.tensor(train_data[3][permute_idxs],
                      dtype=torch.float32)
  act_log_probs = torch.tensor(train_data[4][permute_idxs],
                               dtype=torch.float32)

  # Value data
  returns = discount_rewards(train_data[2])[permute_idxs]
  returns = torch.tensor(returns, dtype=torch.float32)

  # Train model
  ppo.train_policy(obs, acts, act_log_probs, gaes)
  ppo.train_value(obs, returns)
```

```python
data_labels = ['Observations', 'Actions', 'Rewards', 'Value function results', 'Action Log Probabilities']

for label, data in zip(data_labels, train_data):
    print(f"{label}:")
    if isinstance(data, np.ndarray):
        for row in data:
            print(row)
    else:
        print(data)
    print()
```

Plotting Learning Curve

Goal: Visualize the training progress.
Steps:
Compute the running average of rewards.
Plot the running average against the episode numbers.
Display the plot with appropriate titles and labels.

```
x = np.arange(1, len(ep_rewards)+1)

# Plot the learning curve with the collected rewards
running_avg = np.zeros(len(ep_rewards))
for i in range(len(running_avg)):
    running_avg[i] = np.mean(ep_rewards[max(0, i-100):(i+1)])
plt.plot(x, running_avg)

plt.title(f'Learning Curve With The Rewards over  {len(ep_rewards)} Episode')
plt.xlabel('Episode')
plt.ylabel(' Reward ')
plt.grid(True)
plt.tight_layout()
plt.show()
```

click on links:

Drive link(for the images)

Colab link(for the code)

Warning !

Please upload images from google drive to colab before runing.

Thank You