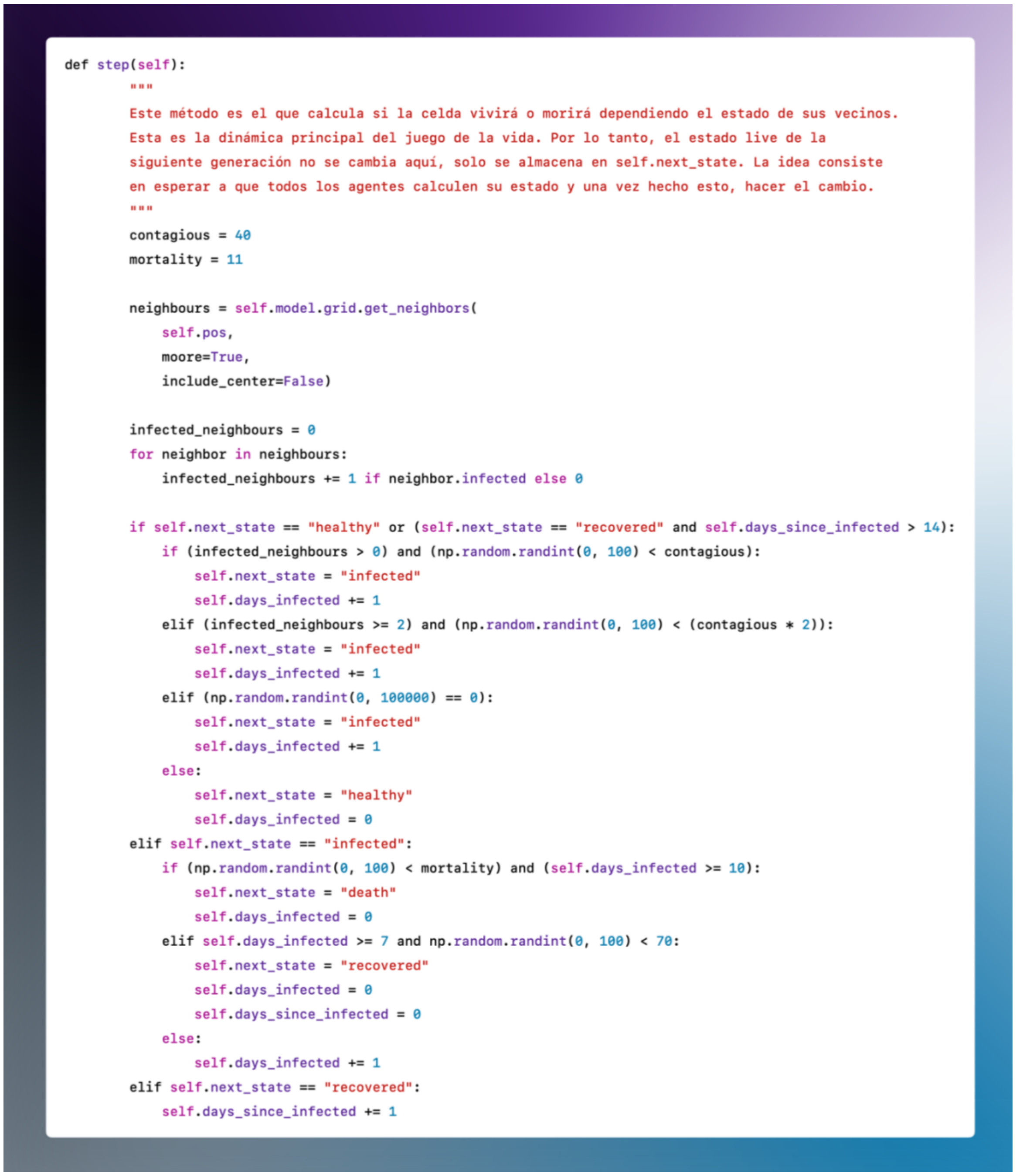
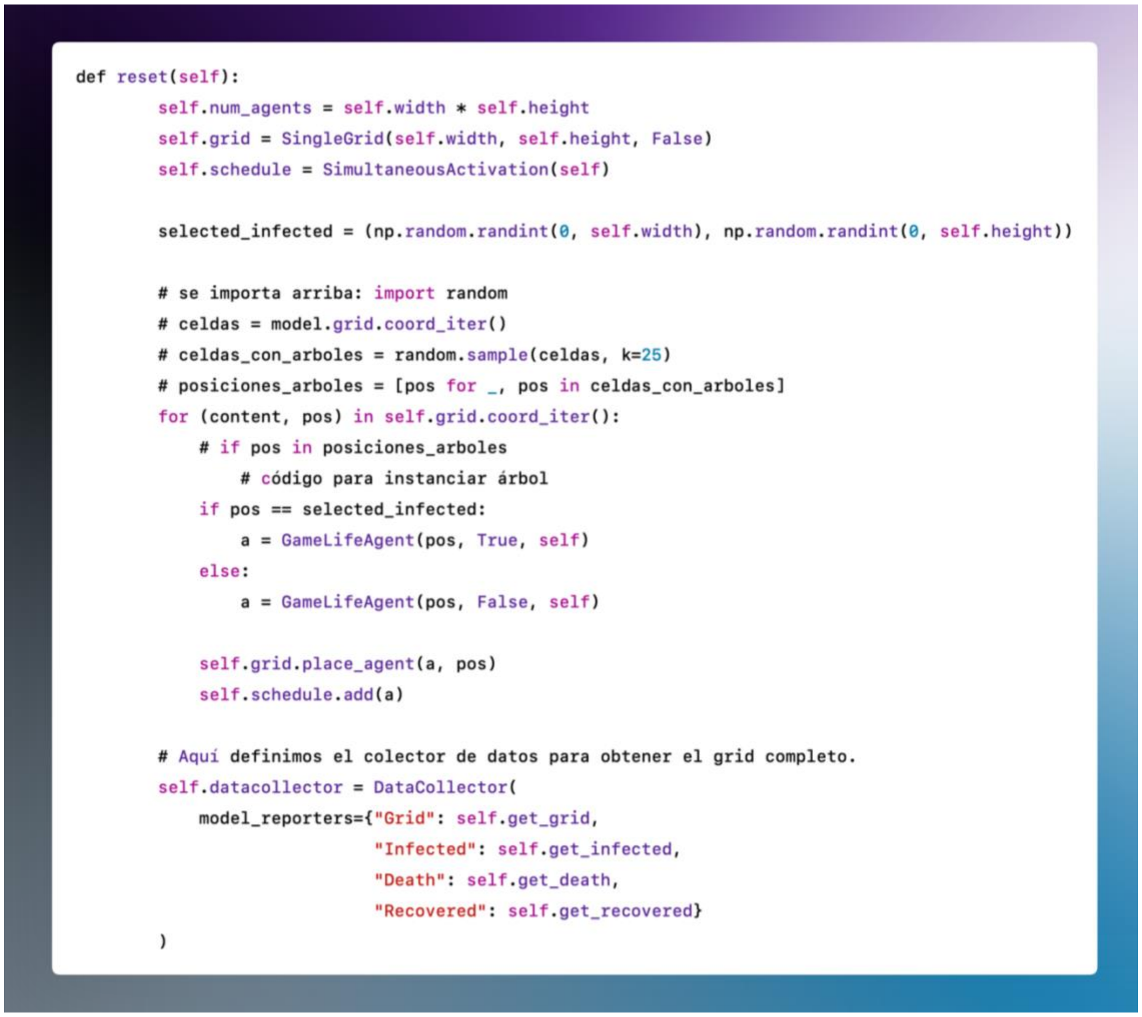
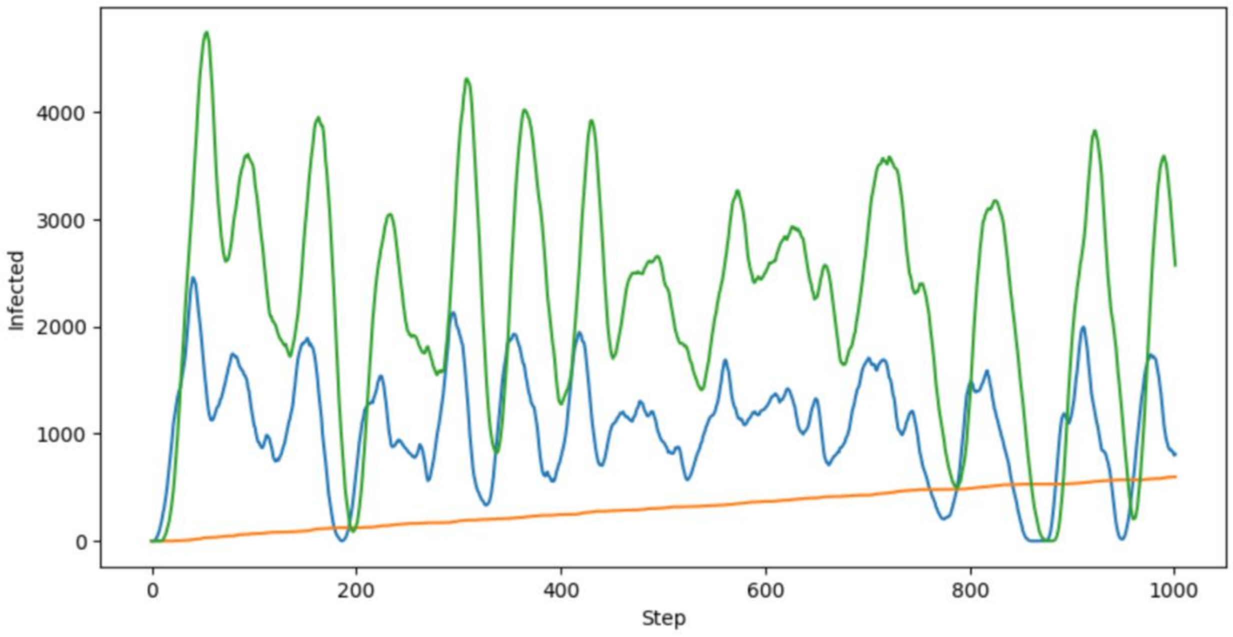
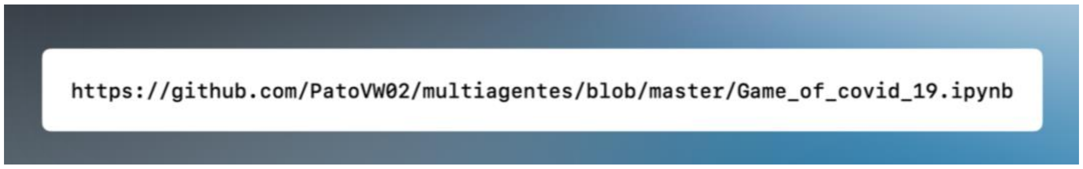


```python
def step(self):
        """
        Este método es el que calcula si la celda vivirá o morirá dependiendo el estado de sus vecinos.
        Esta es la dinámica principal del juego de la vida. Por lo tanto, el estado live de la
        siguiente generación no se cambia aquí, solo se almacena en self.next_state. La idea consiste
        en esperar a que todos los agentes calculen su estado y una vez hecho esto, hacer el cambio.
        """
        contagious = 40
        mortality = 11

        neighbours = self.model.grid.get_neighbors(
            self.pos,
            moore=True,
            include_center=False)

        infected_neighbours = 0
        for neighbor in neighbours:
            infected_neighbours += 1 if neighbor.infected else 0

        if self.next_state == "healthy" or (self.next_state == "recovered" and self.days_since_infected > 14):
            if (infected_neighbours > 0) and (np.random.randint(0, 100) < contagious):
                self.next_state = "infected"
                self.days_infected += 1
            elif (infected_neighbours >= 2) and (np.random.randint(0, 100) < (contagious * 2)):
                self.next_state = "infected"
                self.days_infected += 1
            elif (np.random.randint(0, 100000) == 0):
                self.next_state = "infected"
                self.days_infected += 1
            else:
                self.next_state = "healthy"
                self.days_infected = 0
        elif self.next_state == "infected":
            if (np.random.randint(0, 100) < mortality) and (self.days_infected >= 10):
                self.next_state = "death"
                self.days_infected = 0
            elif self.days_infected >= 7 and np.random.randint(0, 100) < 70:
                self.next_state = "recovered"
                self.days_infected = 0
                self.days_since_infected = 0
            else:
                self.days_infected += 1
        elif self.next_state == "recovered":
            self.days_since_infected += 1
```



```
def reset(self):
        self.num_agents = self.width * self.height
        self.grid = SingleGrid(self.width, self.height, False)
        self.schedule = SimultaneousActivation(self)

        selected_infected = (np.random.randint(0, self.width), np.random.randint(0, self.height))

        # se importa arriba: import random
        # celdas = model.grid.coord_iter()
        # celdas_con_arboles = random.sample(celdas, k=25)
        # posiciones_arboles = [pos for _, pos in celdas_con_arboles]
        for (content, pos) in self.grid.coord_iter():
            # if pos in posiciones_arboles
                # código para instanciar árbol
            if pos == selected_infected:
                a = GameLifeAgent(pos, True, self)
            else:
                a = GameLifeAgent(pos, False, self)

            self.grid.place_agent(a, pos)
            self.schedule.add(a)

        # Aquí definimos el colector de datos para obtener el grid completo.
        self.datacollector = DataCollector(
            model_reporters={"Grid": self.get_grid,
                             "Infected": self.get_infected,
                             "Death": self.get_death,
                             "Recovered": self.get_recovered}
        )
```





https://github.com/PatoVW02/multiagentes/blob/master/Game_of_covid_19.ipynb