Ejercicio 1

```
# librerias necesarias
import random
```

Ejercicio 1

Escoger el comportamiento de nuestro enemigo, este podrá atacar, moverse, defenderse o curarse/usars item

```
def set_action_probs(m, a, d, h):

    return {
                "Move": m,
                "Attack": a,
                "Defend": d,
                "Heal": h
            }
```

Ejercicio 2 y 3

Implemente una simulación de Monte Carlo que genere posibles resultados del comportamiento del enemigo durante una serie de turnos.

Analice la distribución de los posibles resultados y determine la probabilidad de éxito o fracaso del jugador.

```
def simulate_enemy_behavior(num_turns, action_probabilities):

    # Ejercicio 2
    successes = 0
    failures = 0

    for i in range(num_turns):

        # Escoger una accion random por probabilidades
```

```
            enemy_action =
            random.choices(list(action_probabilities.keys()),
            action_probabilities.values())[0]

            # Analizar el turno
            if enemy_action == "Attack" or enemy_action == "Defend":
                failures += 1
            else:
                successes += 1

    # Ejercicio 3
    success_probability = successes / num_turns
    failure_probability = failures / num_turns

    return success_probability, failure_probability

# generar probabilidades
action_probabilities = set_action_probs(0.1, 0.5, 0.1, 0.3)

def simulation(action_probabilities):
    # Paremtros para la simulacion
    num_iterations = 10000
    num_turns_per_iteration = 15

    avg_success_prob = 0
    avg_failure_prob = 0

    for _ in range(num_iterations):
        success_prob, failure_prob =
        simulate_enemy_behavior(num_turns_per_iteration,
        action_probabilities)
        avg_success_prob += success_prob
        avg_failure_prob += failure_prob

    print(f"Situaciones de éxito: {avg_success_prob:.2f}")
    print(f"Situaciones de fracaso: {avg_failure_prob:.2f}")

    avg_success_prob /= num_iterations
    avg_failure_prob /= num_iterations
    print(f"\nProbabilidad de éxito: {avg_success_prob:.2f}")
    print(f"Probabilidad de fracaso: {avg_failure_prob:.2f}\n")
```

```
simulation(action_probabilities)
```

```
Situaciones de éxito: 3998.93
Situaciones de fracaso: 6001.07

Probabilidad de éxito: 0.40
Probabilidad de fracaso: 0.60
```

Ejercicio 4

Ajuste las reglas y los parámetros para explorar diferentes estrategias y resultados de los jugadores.

```
# Explorar diferentes probabilidades

# generar probabilidades
action_probabilities = set_action_probs(0.3, 0.3, 0.1, 0.3)
simulation(action_probabilities)

# generar probabilidades
action_probabilities = set_action_probs(0.1, 0.7, 0.1, 0.1)
simulation(action_probabilities)

# generar probabilidades
action_probabilities = set_action_probs(0.4, 0.1, 0.1, 0.4)
simulation(action_probabilities)
```

```
Situaciones de éxito: 5990.60
Situaciones de fracaso: 4009.40

Probabilidad de éxito: 0.60
Probabilidad de fracaso: 0.40
Situaciones de éxito: 1982.27
Situaciones de fracaso: 8017.73

Probabilidad de éxito: 0.20
Probabilidad de fracaso: 0.80
Situaciones de éxito: 8011.47
Situaciones de fracaso: 1988.53
```

```
Probabilidad de éxito: 0.80
Probabilidad de fracaso: 0.20
```