HUNDIR LA FLOTA

Proceso, implementación y resultados

TABLA DE CONTENIDOS

DECLARACIÓN DE VARIABLES

CLASE BARCO:

```
class barco:

    def __init__(self):
        self.coord = {}
        self.vida = 0
```

VIDAS:

```
vidas = [20,20]
```

DECLARACIÓN DE VARIABLES

- **Una lista M = [M1,M2]**
  - Disposición de los barcos (mapa) de ambos jugadores
- **Una lista C = [C1,C2]**
  - Matriz “check” de ambos jugadores
  - Al inicio todas las casillas serán 0 y a medida que juguemos irá almacenando la información de la tirada y el resultado
- **Una lista F = [F1,F2]**
  - Se almacenan los vectores flota que incluyen los 10 objetos clase barco de cada jugador.
- **Turno**
  - Entero que toma valores entre 0 y 1, determina qué jugador está ejecutando su jugada.
- **Tirada**
  - Un contador independiente de turno, almacena cuantas jugadas se han realizado a lo largo de la partida (su valor se inicia en 1).

SET_MAPA

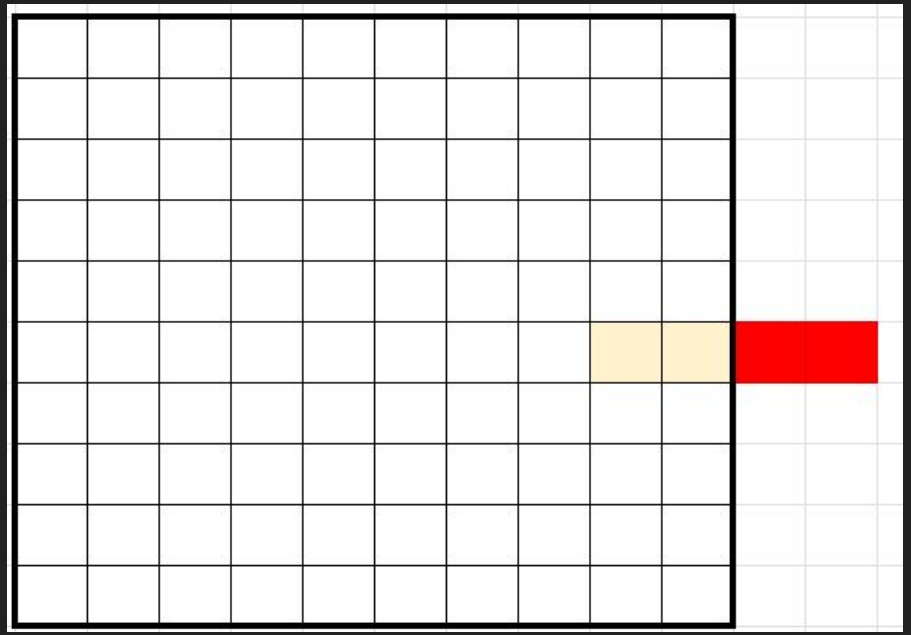
Ejecución:

```
for l in range(4,0,-1):
    # Cuanta menor sea la longitud de eslora, mayor será el número de barcos que implementaremos
    # Esto lo contabilizamos mediante el iterador j que va desde 0 a 5-l
    # En caso de l = 1, 4 barcos, l = 2, 3 barcos, etc.
    for j in range(0,5-l):
        # Inicializamos un objeto barco y lo añadimos a nuestra lista
        b = barco()
        flota1.append(b)
        # Ubicamos nuestro barco en el mapa del jugador:
        funciones.set_mapa(M1,l,flota1)
```

SET_MAPA

Casos a evitar:

longitud = 4
vertical = False



SET_MAPA

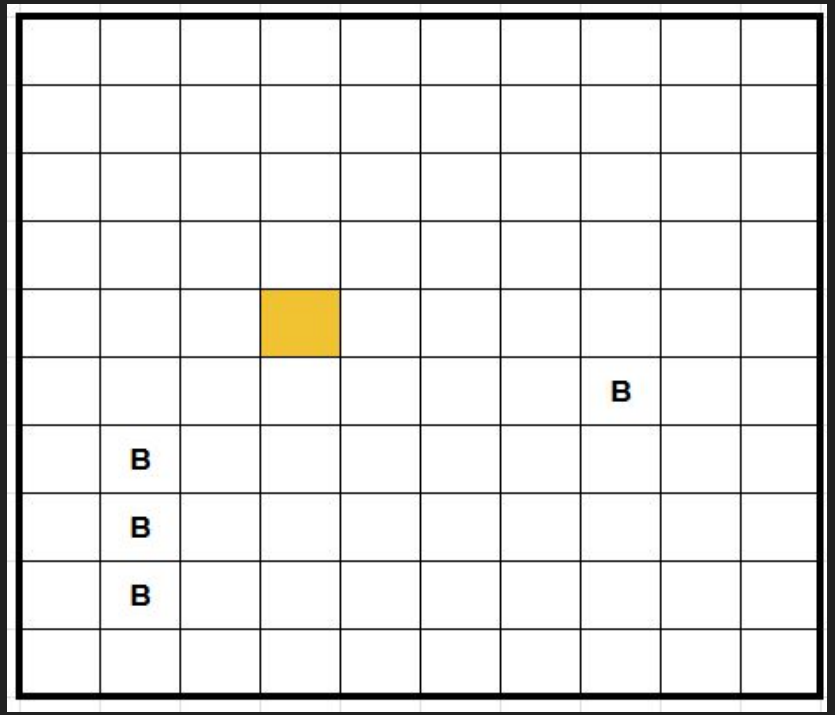
Solución:

```
if v == 1:
    while True:
        x = random.randint(0,9-l+1)
        y = random.randint(0,9)
        if valido(M,x,y,l,v):
            break
```

```
else:
    while True:
        x = random.randint(0,9)
        y = random.randint(0,9-l+1)
        if valido(M, x,y,l,v):
            break
```

SET_MAPA

longitud = 4
vertical = False
x = 4
y = 3



SET_MAPA

Función valido(...)

Implementación

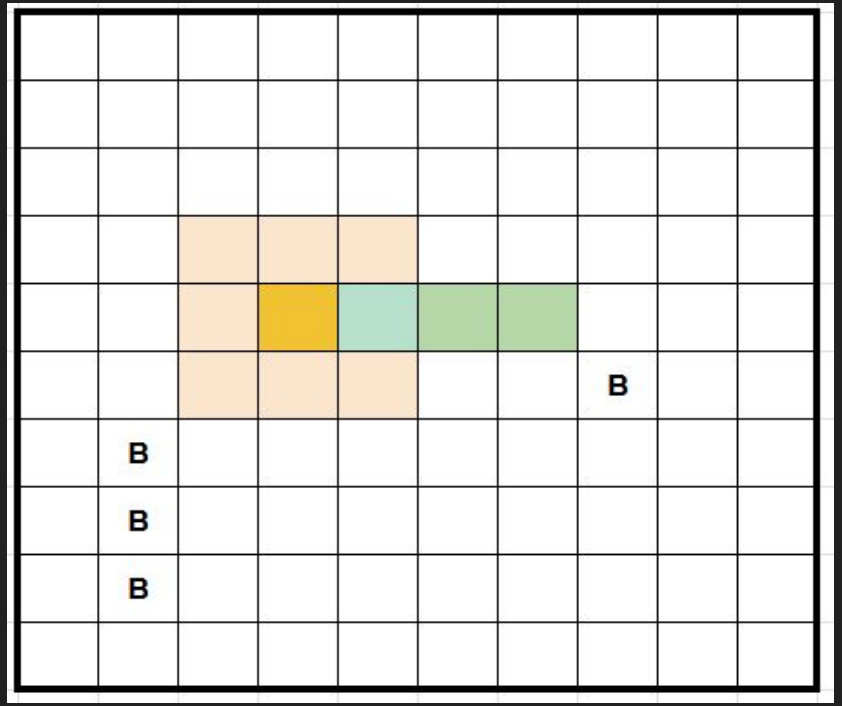
```python
def valido(M,x,y,l,v):
    # Analizaremos celda a celda.
    # Si el barco está en posición vertical, modificaremos la coordenada x
    if v == True:
        for i in range(0,l):
            if not mira(M,x+i,y):
                return False
    # Si el barco está en posición horizontal, modificaremos la coordenada y
    else:
        for i in range(0,l):
            if not mira(M,x,y+i):
                return False

    return True
```

```python
def mira(M,x,y):
    for i in range(-1,2):
        for j in range(-1,2):
            if dentro(x-j,y-i):
                if M[x-j][y-i] != 'O':
                    return False
    return True
```

SET_MAPA

Función valido(...):

Ejecución

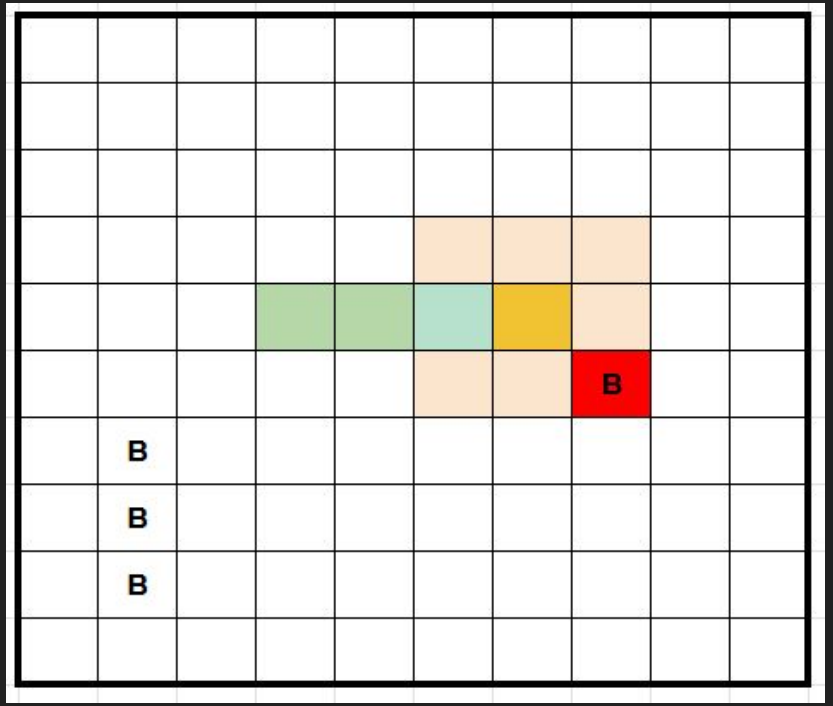


SET_MAPA

Función valido(...):

¡CONFLICTO!

retorna False



SET_MAPA

imprime_tablero(...)

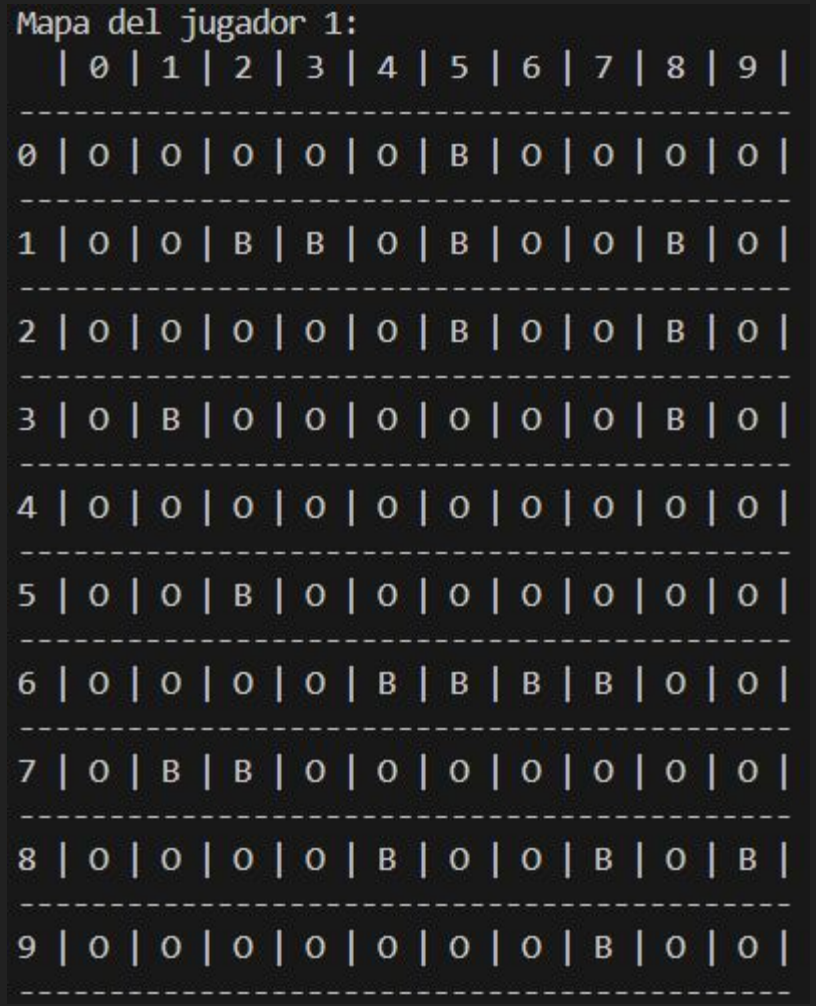
Implementación

```
def imprime_tablero(A):
    for i in range(-1,10):
        if i == -1:
            print (" ", end = " | ")
        else:
            print(i, end = " | ")
    print("")
    print("-"*43)
    for i in range(len(A)):
        for j in range(len(A[i])+1):
            if j == 0:
                print(i, end  = " | ")
            else:
                print(A[i][j-1], end = " | ")
        print("")
        print("-"*43)
```

SET_MAPA

imprime_tablero(...)

Output



FUNCIONES

Ejecución de acciones

Disparo(...)

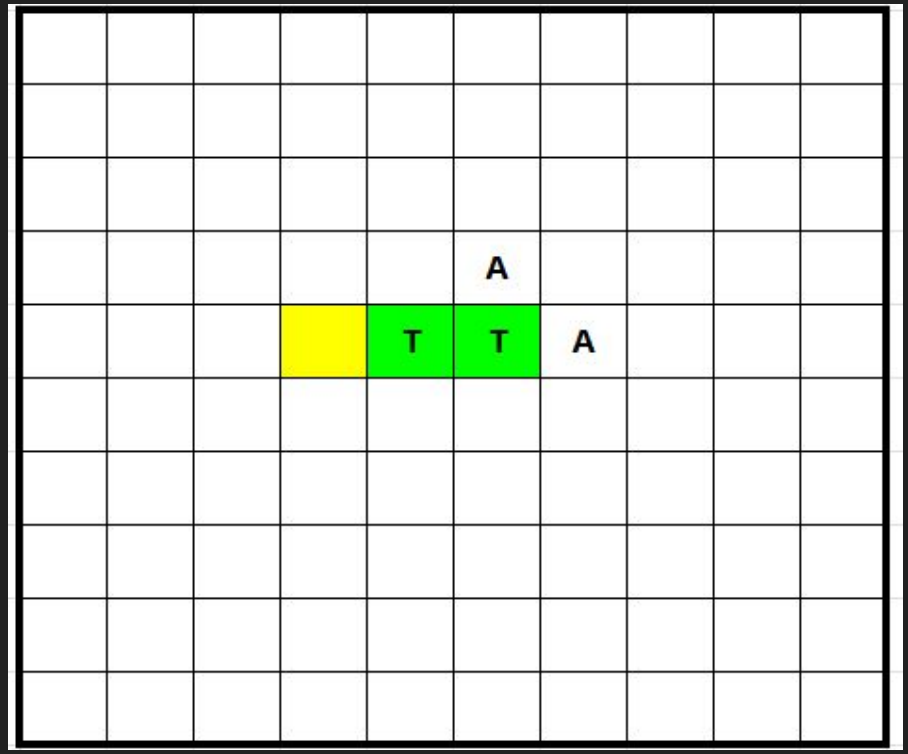
```python
def disparo(M,C,F,vidas,x,y,turno):
    t = (turno+1)%2
    acierto = True
    if dentro(x,y) and C[turno][x][y] == 0:
        if M[t][x][y] == 'O':
            C[turno][x][y] = 'A'
            print(f"Coordenada ({x},{y}): Agua. Turno del otro jugador")
            turno += 1
            acierto = False
        else:
            C[turno][x][y] = 'T'
            k = (x,y)
            aux = 0
            for i in range(len(F[t])):
                if k in F[t][i].coord.keys():
                    F[t][i].coord[k] = True
                    F[t][i].vida -= 1
                    aux = i
            if F[t][aux].vida == 0:
                print(f"Coordenada ({x},{y}): Tocado y hundido. Vuelve a tirar")

                C[turno] = pinta(C,turno, F[t][aux])
            else:
                print(f"Coordenada ({x},{y}): Tocado. Vuelve a tirar")
            vidas[t] -= 1
    else:
        if turno == 0:
            print(f"Coordenada ({x},{y}) fuera del tablero o ya visitada, prueba otra combinación")

    return turno, acierto
```

FUNCIONES

Ejecución de acciones

Disparo(...)

x = 4
y = 3



FUNCIONES

Ejecución
de acciones

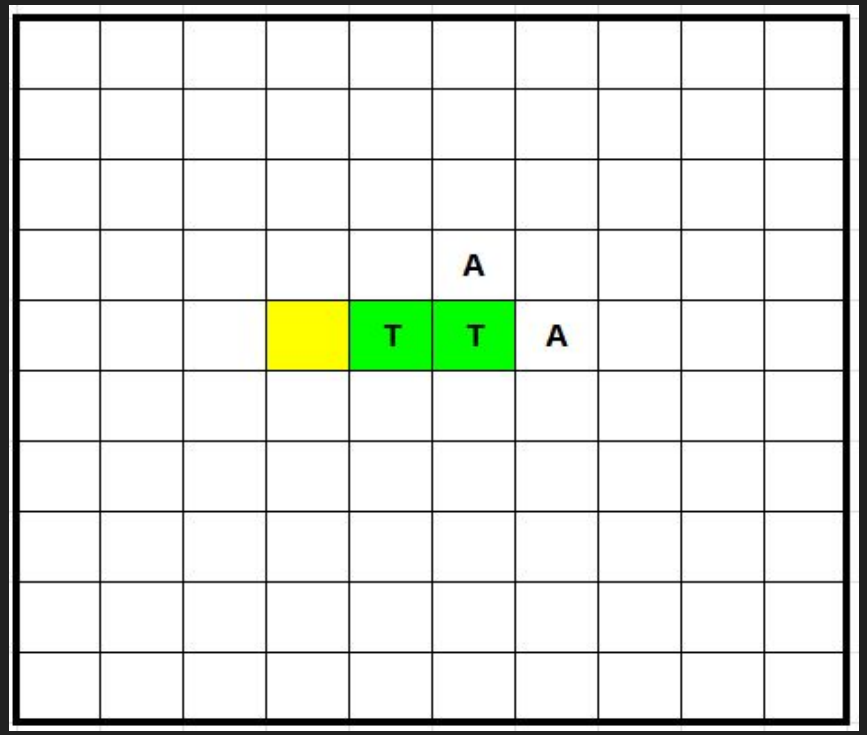
pinta(...)

```
def pinta(C, turno,b):
    for i in b.coord.keys():
        x = i[0]
        y = i[1]
        for j in range(-1,2):
            for k in range(-1,2):
                if dentro(x-j,y-k):
                    if C[turno][x-j][y-k] != 'T':
                        C[turno][x-j][y-k] = "A"
    return C[turno]
```

FUNCIONES

Ejecución
de acciones

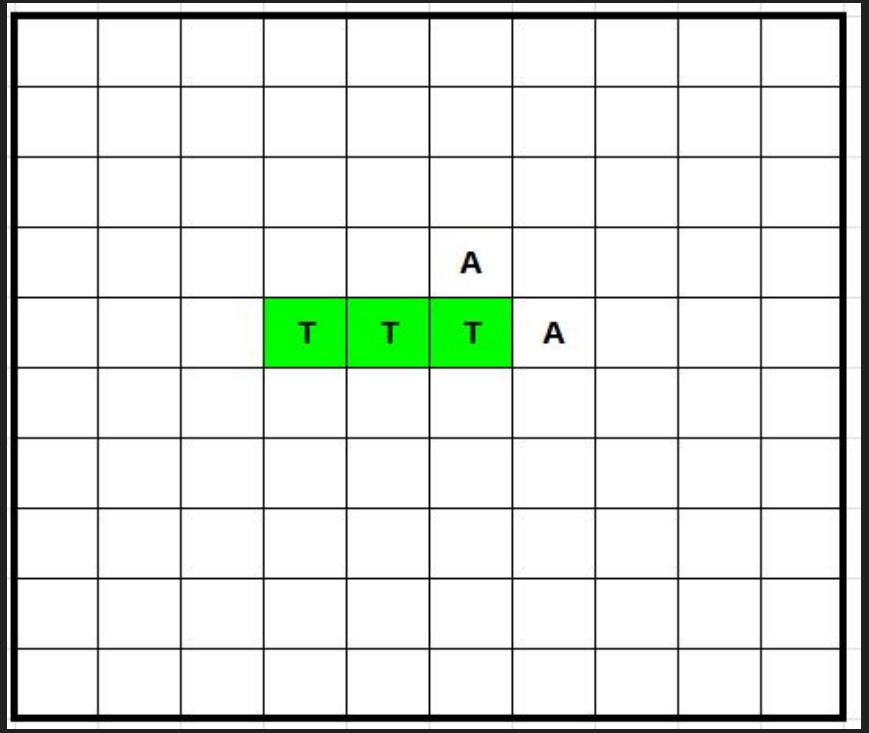
pinta(...)



FUNCIONES

Ejecución
de acciones

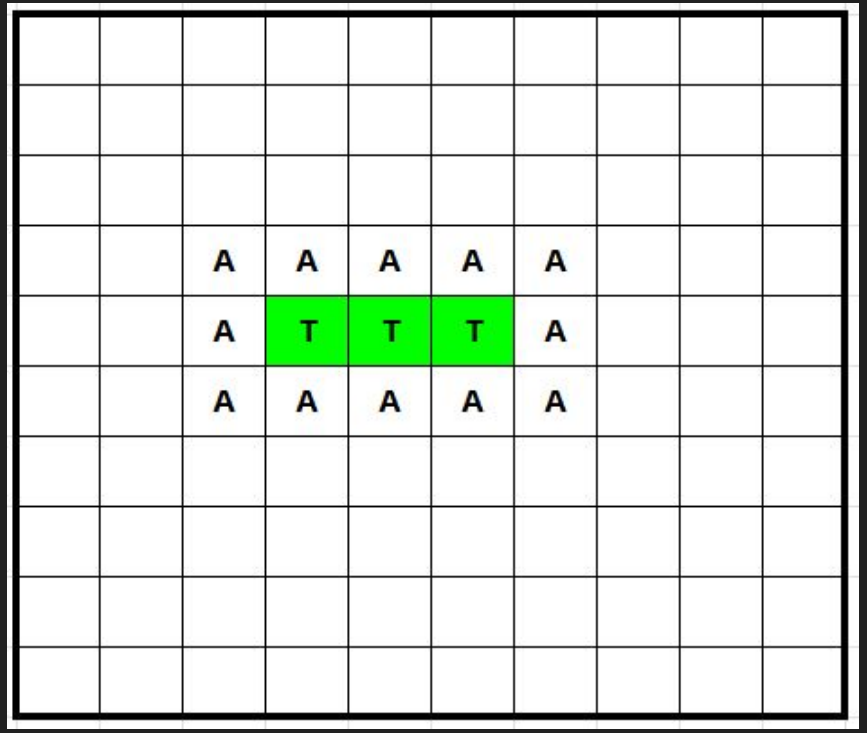
pinta(...)



FUNCIONES

Ejecución
de acciones

pinta(...)

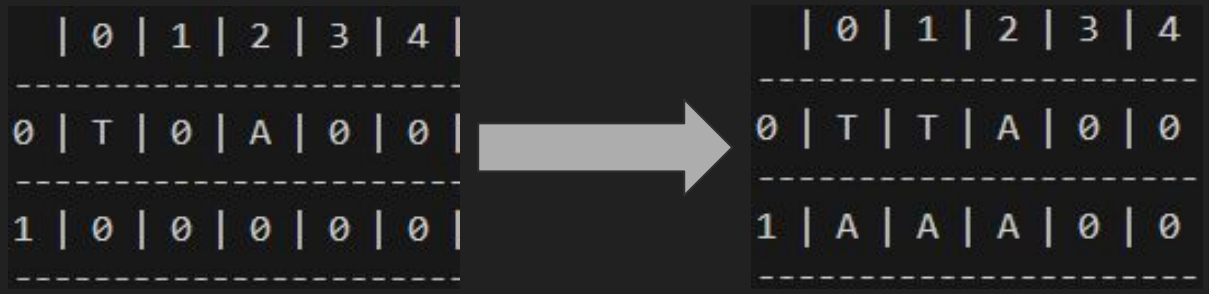


FUNCIONES

Ejecución
de acciones

pinta(...):

Output




```
Introduce la coordenada fila:0
Introduce la coordenada columna:1
Coordenada (0,1): Tocado y hundido. Vuelve a tirar
```

FUNCIONES

Niveles de dificultad

dif_0(...)

```
def dif_0(M,C,F,vidas, turno):
    while True:
        x = random.randint(0,9)
        y = random.randint(0,9)
        if C[turno][x][y] == 0:
            break
    turno, acierto = disparo(M,C,F,vidas,x,y,turno)
    return turno, acierto
```

FUNCIONES

Niveles de dificultad

dif_1(...)

sec = False

```
def dif_1_alt(M,C,F,vidas, turno, sec, futuro):
    if not sec:
        while True:
            x = random.randint(0,9)
            y = random.randint(0,9)
            if C[turno][x][y] == 0:
                break
        turno, acierto = disparo(M,C,F,vidas,x,y,turno)
        if acierto:
            tupla = (x,y)
            for n,i in enumerate(F[0]):
                if tupla in i.coord.keys():
                    if F[0][n].vida == 0:
                        sec = False
                        futuro = []
                    else:
                        futuro.append(tupla)
                        sec = True
                        for k in range(1,4):
                            if dentro(x-k,y):
                                if C[turno][x - k][y] == 0:
                                    futuro.append((x-k,y))
                                else:
                                    break
                            else:
                                break
```

FUNCIONES

Niveles de dificultad

dif_1(...)

sec = True

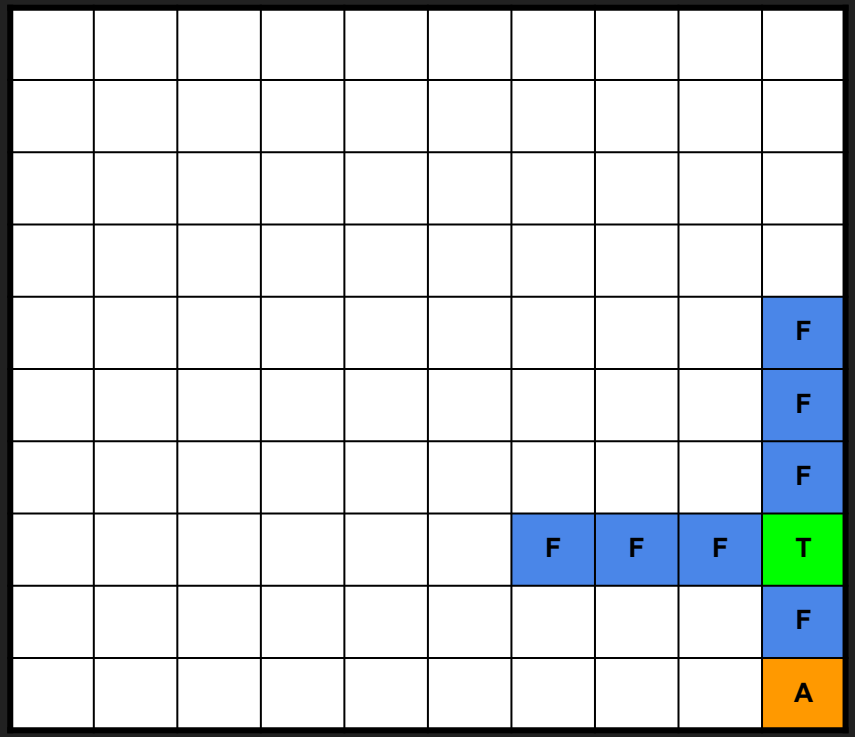
```
else:
    t = futuro[1]
    print(f"ESTOY AQUÍ {t}")
    futuro.pop(1)
    x = t[0]
    y = t[1]
    turno, acierto = disparo(M,C,F,vidas,x,y,turno)
    if acierto:
        for n,i in enumerate(F[0]):
            if t in i.coord.keys():
                if F[0][n].vida == 0:
                    sec = False
                    futuro = []
    else:
        n = len(futuro)
        if x < futuro[0][0]:
            futuro = fallo_dif_1(futuro,'N')
        elif x > futuro[0][0]:
            futuro = fallo_dif_1(futuro,'S')
        elif y < futuro[0][1]:
            futuro = fallo_dif_1(futuro,'O')
        else:
            futuro = fallo_dif_1(futuro,'E')

return turno, acierto, sec, futuro
```

FUNCIONES

Niveles de dificultad

dif_1(...)

sec = False
x = 7
y = 9

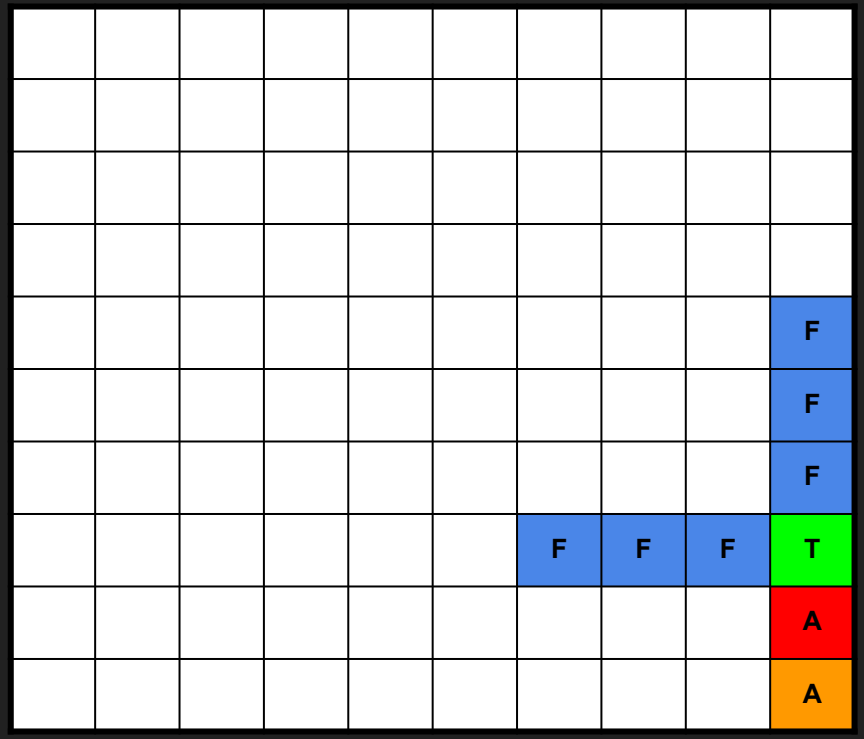


FUNCIONES

Niveles de dificultad

dif_1(...)

sec = True
Futuro[1] = (8,9)

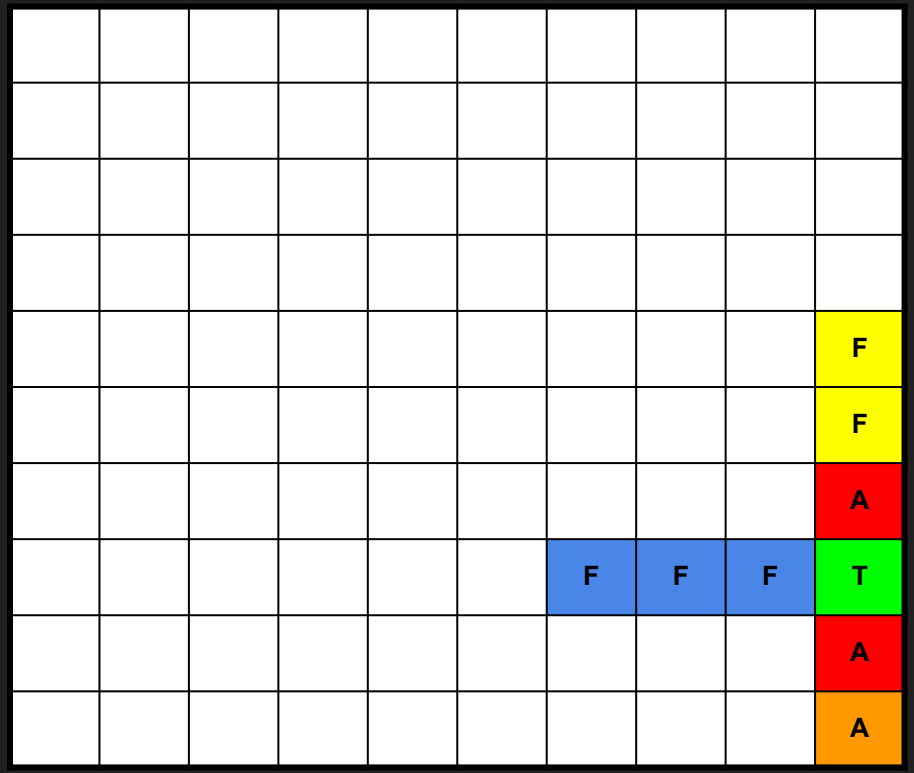


FUNCIONES

Niveles de dificultad

dif_1(...)

sec = True
Futuro[1] = (6,9)



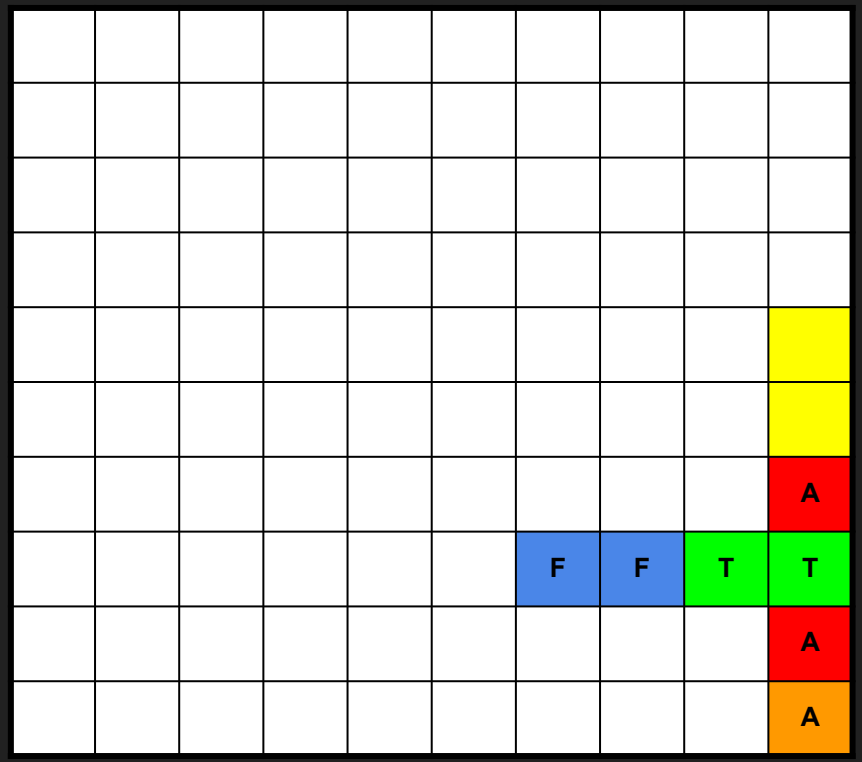
FUNCIONES

Niveles de dificultad

dif_1(...)

sec = True
Futuro[1] = (7,8)

Tocado y hundido

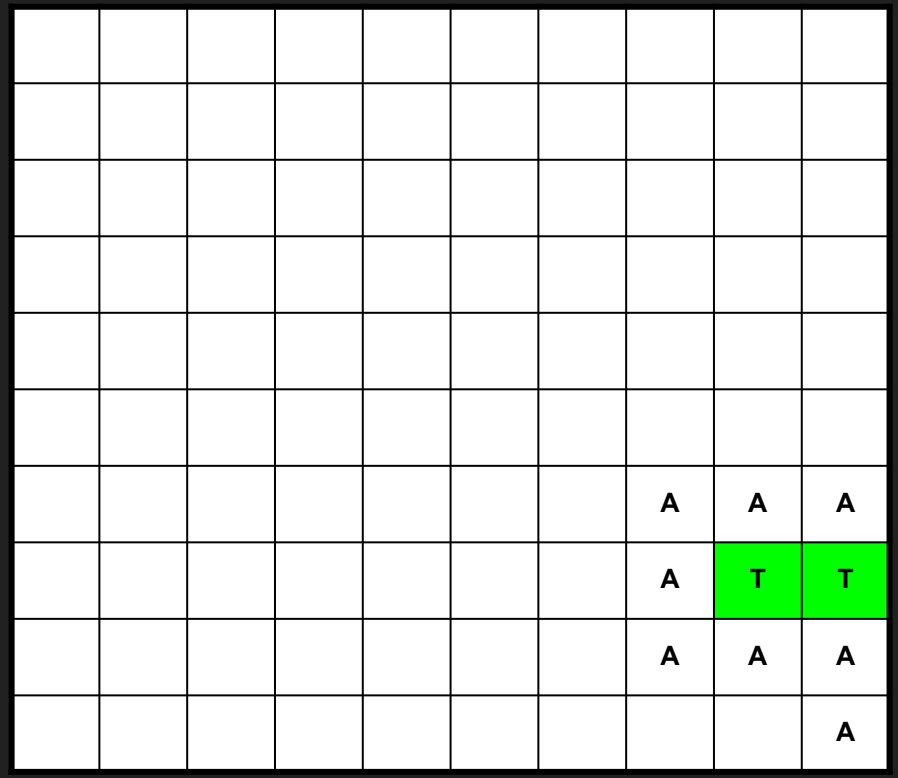


FUNCIONES

Niveles de dificultad

dif_1(...)

sec = False
futuro = []



FUNCIONES

Niveles de dificultad

dif_2(...)

```python
def dif_2(M,C,F,vidas, turno, sec, futuro):
    if sec == False:
        while True:
            x = random.randint(0,9)
            y = random.randint(0,9)
            if C[turno][x][y] == 0:
                break
        turno, acierto = disparo(M,C,F,vidas,x,y,turno)
        if acierto:
            tupla = (x,y)
            for n,i in enumerate(F[0]):
                if tupla in i.coord:
                    if F[0][n].vida == 0:
                        sec = False
                    else:
                        sec = True
                        for k in F[0][n].coord.keys():
                            if F[0][n].coord[k] == False:
                                futuro.append(k)

    else:
        t = futuro[-1]
        futuro.pop()
        x = t[0]
        y = t[1]
        turno, acierto = disparo(M,C,F,vidas,x,y,turno)
        for n,i in enumerate(F[0]):
            if t in i.coord.keys():
                if F[0][n].vida == 0:
                    sec = False
                    futuro = []
    return turno, acierto, sec
```

FUNCIONES

Niveles de dificultad

dif_3(...)

```
def dif_3(M,C,F,vidas,turno):

    for i in F[0]:
        for j in i.coord.keys():
            if i.coord[j] == False:
                x = j[0]
                y = j[1]
                turno, acierto = disparo(M,C,F,vidas,x,y,turno)
                return turno, acierto
```

ESKERRIK ASKO