

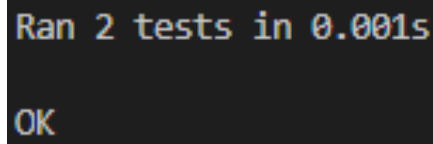
Problema 2

En el problema N°2 se implementó el TAD mazo como clase de Python con las distintas funciones, aprovechando la clase ListaDoblementeEnlazada creada previamente. A continuación se detalla la función y su correspondiente metodología:

- **__init__(self)** → Crea un mazo vacío usando una lista doblemente enlazada.
- **agregar_al_inicio(carta)** → Agrega una carta al principio del mazo.
- **agregar_al_final(carta)** → Agrega una carta al final del mazo.
- **poner_carta_arriba(self, carta)** → Pone una carta en la parte superior (inicio).
- **poner_carta_abajo(self, carta)** → Pone una carta en la parte inferior (final).
- **sacar_carta_arriba(self, mostrar=False)** → Quita la carta de arriba, la devuelve y opcionalmente la muestra.
- **__len__(self)** → Devuelve cuántas cartas hay en el mazo.

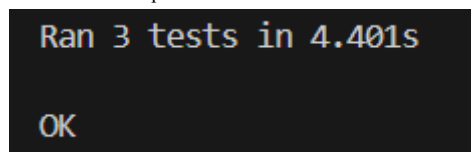
También incluimos la clase DequeEmptyError(Exception) que define una excepción personalizada para cuando el mazo está vacío.

Este código también pasó los test proporcionados, a continuación unos ejemplos luego de ejecutar el código:



```
Ran 2 tests in 0.001s
OK
```

Comprobación del test de mazo



```
Ran 3 tests in 4.401s
OK
```

Comprobación del test para el juego de guerra x

```

-----
Sacando carta: J♠
Sacando carta: J♠
Turno: 498
jugador 1:
Mazo: [Q♠, J♥]

          J♠ J♠

jugador 2:
Mazo: [10♠, 3♦, 3♠, 5♦, K♥, 5♥, 8♠, 10♥, 6♦, J♦, 4♠, 9♠, 2♠, Q♦, 4♥,
Q♥, 2♥, 10♦, 4♦, 8♠, 7♥, 9♠, 5♠, 8♦, 3♥, K♠, 7♦, Q♠, 6♠, A♦, 2♠, A♥,
3♠, 7♠, 6♠, 9♥, 4♠, A♠, 5♠, 8♥, 7♠, A♠, 9♦, K♦, 2♦, 10♠, 6♥, K♠]

-----
          **** Guerra!! ****
          ***** jugador 2 gana la partida*****

```

Resultado de la ejecución del juego guerra