Ejercicios Extra I

1 Invierte una lista

Escribe una función `invertir_lista` que reciba una lista y devuelva una nueva lista con los elementos en orden inverso.

```
print(invertir_lista([1, 2, 3, 4, 5]))
```

[5, 4, 3, 2, 1]

Pista Recuerda todas las posibilidades que ofrece `range()`.

Solución

```
def invertir_lista(lista):
    lista_invertida = []
    for i in range(len(lista) - 1, -1, -1):
        lista_invertida.append(lista[i])
    return lista_invertida
```

Solución alternativa

```
def invertir_lista(lista):
    lista_invertida = [0] * len(lista)
    for i in range(len(lista)):
        lista_invertida[i] = lista[len(lista) - 1 - i]
    return lista_invertida
```

2 Encuentra el mayor de cada columna

Escribe una función `mayores_de_columna(matriz)` que reciba una matriz y devuelva una lista con el mayor de cada columna.

```
print(mayores_de_columna([
    [1, 2, 9],
    [4, 8, 6],
    [7, 5, 3]
]))
```

[7, 8, 9]

Solución

```
def mayores_de_columna(matriz):
    mayores = []
    for j in range(len(matriz[0])):
```

```
        mayor = matriz[0][j]
        for i in range(1, len(matriz)):
            if matriz[i][j] > mayor:
                mayor = matriz[i][j]
        mayores.append(mayor)
    return mayores
```

3 Encuentra los dos mayores de una lista

Escribe una función `dos_mayores(lista)` que reciba una lista y devuelva una lista de dos elementos donde el primero será el mayor y el segundo el segundo mayor de la lista. El `mayor` y el `segundo mayor` pueden ser iguales si el mayor se repite.

```
print(dos_mayores([5,6,3,8,4,9,2,3,1]))
print(dos_mayores([5,6,3,9,4,9,2,3,1]))
```

[9, 8]
[9, 9]

Solución

```
def dos_mayores(lista):
    mayor = lista[0]
    segundo_mayor = lista[0]
    for i in range(len(lista)):
        if lista[i] > mayor:
            segundo_mayor = mayor
            mayor = lista[i]
        elif lista[i] > segundo_mayor:
            segundo_mayor = lista[i]
    return [mayor, segundo_mayor]
print(dos_mayores([5, 6, 3, 8, 4, 9, 2, 3, 1]))
print(dos_mayores([5, 6, 3, 9, 4, 9, 2, 3, 1]))
```

4 Matriz transpuesta

Escribe una función `transponer(matriz)` que reciba una matriz y devuelva una nueva matriz que sea la transpuesta de la original.

```
print(transponer([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]))
```

[[1, 4, 7], [2, 5, 8], [3, 6, 9]]

Pista

```python
def transponer(matriz):
    nueva = [0] * len(matriz[0])
    for i in range(len(matriz[0])):
        nueva[i] = [0] * len(matriz)

    for i in range(len(matriz)):
        for j in range(len(matriz[0])):
            nueva[j][i] = matriz[i][j]
    return nueva
```

5 Mover pieza

Partiendo del ejercicio del tablero de ajedrez, escribe una función `mover_pieza(tablero, origen, destino)` que reciba un tablero y dos posiciones (origen y destino) y mueva la pieza de la posición origen a la posición destino.

i Nota

- De momento puedes obviar que las columnas deberían nombrarse con letras
- Si quieres hacer el ejercicio más fácil, puedes:
  - Hacer que `Posicion` tenga `fila:int` y `columna: str`.
    - Deberás implementar el mecanismo de traducción de letras a números y viceversa.
  - Implementar checks de control:
    - Los peones blancas sólo pueden moverse a filas menores y los negros a filas mayores.
    - Los alfiles sólo pueden moverse en diagonal.
    - etc.

```python
import dataclasses

@dataclasses.dataclass
class Posicion:
    fila: int
    columna: int

tablero = [
    ['t', 'c', 'a', 'd', 'r', 'a', 'c', 't'],
    ['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'],
    ['T', 'C', 'A', 'D', 'R', 'A', 'C', 'T']
]

print(mover_pieza(tablero, Posicion(1, 0), Posicion(3, 0)))
```

[[‘t’, ‘c’, ‘a’, ‘d’, ‘r’, ‘a’, ‘c’, ‘t’],
[‘p’, ‘p’, ‘p’, ‘p’, ‘p’, ‘p’, ‘p’, ‘p’],
[‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’],
[‘p’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’],
[‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’],
[‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’, ‘.’],
[‘P’, ‘P’, ‘P’, ‘P’, ‘P’, ‘P’, ‘P’, ‘P’],
[‘T’, ‘C’, ‘A’, ‘D’, ‘R’, ‘A’, ‘C’, ‘T’]]

Solución

```
def copiar_tablero(tablero):
    nuevo_tablero = [""] * len(tablero)
    for i in range(len(tablero)):
        nuevo_tablero[i] = [""] * len(tablero[0])
        for j in range(len(tablero[0])):
            nuevo_tablero[i][j] = tablero[i][j]
    return nuevo_tablero

def mover_pieza(tablero, origen, destino):
     # Recuerda siempre hacer una copia de los datos de entrada si vas a
modificarlos
    nuevo_tablero = copiar_tablero(tablero)
    if tablero[origen.fila][origen.columna] != '.':
         nuevo_tablero[destino.fila][destino.columna] = tablero[origen.fila]
[origen.columna]
        nuevo_tablero[origen.fila][origen.columna] = '.'
    return nuevo_tablero
```