</ Ayudantía 8:
Listas 2
/>

Rolando Rojas

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Temas a tratar

{01}
Listas anidadas

{02}
Obtención de valores en listas anidadas

{03}
Recorrer listas anidadas (for)

{04}
Recorrer listas anidadas (while)

{05}
Ejemplos / Ejercicios

{06}
Preguntas / Tarea

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Listas anidadas

En Python, una lista anidada es una lista que contiene otras listas como elementos. Esto significa que dentro de una lista, los elementos pueden ser otras listas en lugar de simplemente valores individuales. La anidación proporciona una forma de organizar y estructurar datos de manera más compleja y jerárquica.

```
lista_anidada = [1, [2, 3], 4, [5, 6, [7, 8]]]
                 0    1   2       3
```

En este ejemplo, “lista_anidada” contiene elementos que son tanto valores individuales como otras listas.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Obtención de valores en listas anidadas

Cuando se accede a elementos en listas anidadas, cada conjunto de corchetes representa un nivel de profundidad en la estructura anidada. Por ejemplo, si tienes una lista anidada llamada lista_anidada, acceder a un elemento específico implica indicar el índice correspondiente en cada nivel de anidación.

```
>>> lista_anidada = [1, [2, 3], 4, [5, 6, [7, 8]]]
>>> lista_anidada[1][1]
3
>>> lista_anidada[3][0]
5
>>> lista_anidada[3][2][1]
8
```

Obtiene el segundo elemento (indice 1) dentro del segundo elemento (indice 1), es decir: [2,3] → 3

Obtiene el primer elemento (indice 0) dentro del cuarto elemento (indice 3), es decir: [5,6,[7,8]] → 5

Obtiene el segundo elemento (indice 1) de la lista (indice 2) dentro del cuarto elemento (indice 3), es decir: [5,6,[7,8]] → [7,8] → 8

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Recorrer en listas anidadas (for)

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

*Para recorrer todos los elementos de esta lista anidada, puedes utilizar dos bucles for anidados. El primer bucle recorrerá las **listas externas** y el segundo bucle recorrerá los **elementos dentro de cada lista**:*

```
for lista_exterior in lista_anidada:
    for elemento in lista_exterior:
        # Realizar acciones con cada elemento
        print(elemento)
```

Salida:

```
1
2
3
4
5
6
7
8
9
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Recorrer en listas anidadas (while)

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

Este código utiliza dos ciclos while para recorrer las listas anidadas. El primer while se encarga de iterar a través de las listas exteriores (lista_anidada[i]), y el segundo while itera a través de los elementos dentro de cada lista (lista_anidada[i][j]).

```
# Índices de las listas
i = 0
while i < len(lista_anidada):
    # Índices de los elementos dentro de la lista actual
    j = 0
    while j < len(lista_anidada[i]):
        # Acciones con cada elemento
        print(lista_anidada[i][j])
        j += 1

    i += 1
```

Salida:

```
1
2
3
4
5
6
7
8
9
```

</ Ejercicio: SansaCine

Contexto: En el cine “SansaCine" se necesita desarrollar un programa que gestione los asientos de la sala. La sala de cine tiene un total de 5 filas y 8 columnas, lo que representa un total de 40 asientos.

El objetivo del programa es permitir a los usuarios ver la disponibilidad de los asientos en tiempo real y reservar los asientos disponibles para una función determinada. El cine quiere mostrar visualmente la sala de cine en forma de matriz, donde cada asiento vacío se representará con la letra 'O' y los asientos ocupados se representarán con la letra 'X'.

Requerimientos del programa:

- Mostrar la disposición actual de los asientos en la sala de cine en forma de matriz.
- Permitir al usuario seleccionar un asiento ingresando el número de fila y columna correspondiente al asiento deseado.
- Verificar si el asiento seleccionado está disponible. Si está ocupado, mostrar un mensaje de error y permitir al usuario seleccionar otro asiento.
- Si el asiento está disponible, marcarlo como ocupado y actualizar la matriz de asientos.
- Mostrar la matriz actualizada después de la reserva del asiento.

Nota: Se recomienda utilizar números de fila y columna basados en índices, es decir, comenzando desde 0.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Solución: SansaCine

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0

main.py

```python
from random import choice
# Crear la matriz de asientos vacía
sala_cine = []
for _ in range(5):
    fila = [0] * 8
    sala_cine.append(fila)
# Función para mostrar la sala de cine
def mostrar_sala_cine():
    print("----- Sala de Cine -----")
    for fila in sala_cine:
        for asiento in fila:
            if asiento == 0:
                print('O', end=' ')
            else:
                print('X', end=' ')
        print()
    print("-------------------------")
# Mostrar la sala de cine inicial
mostrar_sala_cine()
# realizaremos el ciclo indefinidamente hasta que el usuario ingrese X en el número de fila.
flag = True
while flag:
# Solicitar al usuario seleccionar una función
    fila = int(input("Ingrese el número de fila del asiento: "))
    columna = int(input("Ingrese el número de columna del asiento: "))
# Verificar si el asiento está disponible
    if sala_cine[fila][columna] == 0:
        sala_cine[fila][columna] = 1
        print("¡Asiento reservado con éxito!")
        # Mostrar la sala de cine actualizada
        mostrar_sala_cine()
    else:
        print("Lo siento, el asiento seleccionado está ocupado.")
```

Tarea

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</ />

Ayudantía 8: Listas 2

¿Dudas?

rolando.rojass@usm.cl

Rolando Rojas

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1