

Estructuras de Datos

Listas, Pilas y Colas en Python

Por José Panaza y Julio Zaldaña



IEEE
**COMPUTER
SOCIETY**

Universidad de San Carlos de Guatemala
Student Branch Chapter



Clase 4

1. Listas
2. Tuplas
3. Diccionarios
4. Pilas
5. Colas



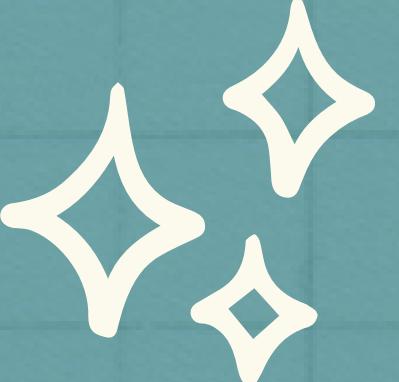
Listas



Clase list en python

Este tipo de clases nativas de python son de las más utilizadas en el día a día de un programador. Esto se debe a su naturaleza y dinamismo al momento de utilizarlas.

Proximamente se abordará qué es el tipo list, cuáles son sus operaciones más comunes para sacarle lo máximo que se pueda a este tipo de dato



¿Qué es una lista?

Se hace la referencia de que son como un tipo de contenedor, el cual se utiliza para guardar elementos ya sea del mismo, o diferente tipo de datos. Normalmente se hacen listas de un mismo tipo, pero se puede hacer una lista de distintos elementos.

Las lista en python son muy versátiles, a comparación de otros lenguajes, las listas aquí se declaran y recorren de maneras más sencillas.



Declarar una lista

Para esto es necesario igualar una variable a un par de corchetes, para indicarle al lenguaje que la variable será una lista.

Se puede declarar solo de numeros, de varios elementos o una lista dentro de otra para darle origen a una matriz

```
>>> numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> elementos = [3, 'a', 8, 7.2, 'hola']
```

```
>>> lista = [1, ['a', 'e', 'i', 'o', 'u'], 8.9, 'hola']
```

Declarar listas vacías



Se puede declarar una lista vacía con la función `list()`, ésta por si solo es una lista vacía. Sin embargo, si tiene un string adentro, divide el string en varias posiciones de la lista.

```
>>> vocales = list('aeiou')
>>> vocales
['a', 'e', 'i', 'o', 'u']
```

Y para declarar listas vacías se pueden utilizar estas dos opciones

```
>>> lista_1 = [] # Opción 1
>>> lista_2 = list() # Opción 2
```



Acceder a una lista

Las listas poseen posiciones de 0 hasta n numeros, hay que tener en cuenta que siempre inicián en 0. Y se puede acceder a ellas de manera sencilla

```
>>> lista = ['a', 'b', 'd', 'i', 'j']
>>> lista[0] # Primer elemento de la lista. Índice 0
'a'
```

Cuando se trabaja con listas anidadas, o comúnmente conocidas como matrices, se puede acceder a los datos, mediante el uso de doble corchete

```
>>> lista = ['a', ['d', 'b'], 'z']
>>> lista[1][1] # lista[1] hace referencia a la lista anidada
'b'
```

Recorrer una lista

En este momento entra la importancia de los ciclos, ya que para recorrer una lista es necesario conocer la estructura de los ciclos. Se pueden utilizar los ciclos `for` y `while`, pero el más utilizado por su simplicidad es el ciclo `for`.

```
>>> colores = ['azul', 'blanco', 'negro']
>>> for color in colores:
    print(color)

azul
blanco
negro
```

Funciones que debemos saber para manejar listas

Añadir elementos

Se puede añadir con:

- `append():` agrega un elemento
- `extend():` agrega varios elementos a la lista
- `insert(n,'dato'):` inserta un elemento en la lista

Modificar elementos

Se puede modificar de manera sencilla con:

- `la_lista[n] = dato:` se coloca la lista, entre corchetes el item de la lista y se iguala a un valor cualquiera.

Eliminar elementos

Se puede eliminar elementos con 4 funciones, las cuales hacen lo mismo:

- `del la_lista[n]:` del de delete y la lista en su item n
- `la_lista.remove('dato0'):` la lista y en su función remove se coloca el dato a remover.
- `la_lista.pop():` elimina el ultimo elemento
- `la_lista.clear():` elimina toda la lista

Funciones que debemos saber para manejar listas

Longitud de una lista: `len(la_lista)`

Ordenar una lista de integers: `lista.sort()`

obtener el índice de un dato: `la_lista.index('dato1')`

obtener lista en orden inverso: `la_lista.reverse()`

Diccionarios



¿Qué son y cómo crearlos?

Son una estructura de datos que permite almacenar el contenido en forma de llave y valor.

Son tipos de datos que se utilizan en una programación más avanzada tipo flask y django. Su estructura es así

```
d1 = {  
    "Nombre": "Sara",  
    "Edad": 27,  
    "Documento": 1003882  
}  
print(d1)  
#{'Nombre': 'Sara', 'Edad': 27, 'Documento': 1003882}
```

Tuplas



¿Qué son?

Son colecciones de datos cuyo orden no se puede alterar. Posee elementos ordenados en una secuencia específica, donde el orden tiene importancia.

Estas se declaran entre paréntesis y se puede acceder a ellas como una lista normal, o con el signo - se puede acceder desde el último elemento.

Su principal uso es para unificar o agrupar valor que por defecto deben ir juntos, por ejemplo un nombre y un apellido. Se utiliza una tupla para facilitar el trabajo

```
>>> hoy = (2011, 4, 19)
>>> ayer = (2011, 4, 18)
>>> navidad = (2011, 12, 25)
>>> anno_nuevo = (2012, 1, 1)
>>> hoy < ayer
False
>>> hoy < navidad < anno_nuevo
True
```



Ejemplos

Ejemplo 1

Declarar una lista de numeros y recorrerla con un ciclo for y while

Ejemplo 2

Declarar una lista vacía y que el usuario la llene con inputs

Ejemplo 3

usar las funciones de una lista para poder mostrar palabra por palabra en un texto

Pilas



¿Qué es una pila?

Una Pila (stack) es un TDA lineal, que representa a un conjunto de elementos que se colocan uno después del otro.

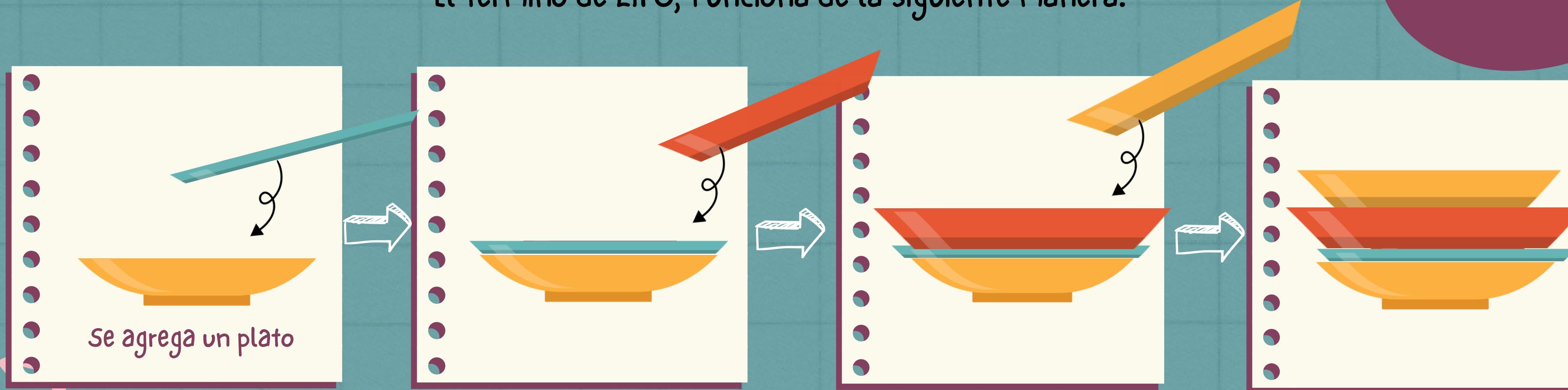
Utiliza el concepto de:
LIFO
(Last IN, First OUT)

(El último en entrar,
es el primero en salir)



¿Cómo funciona?

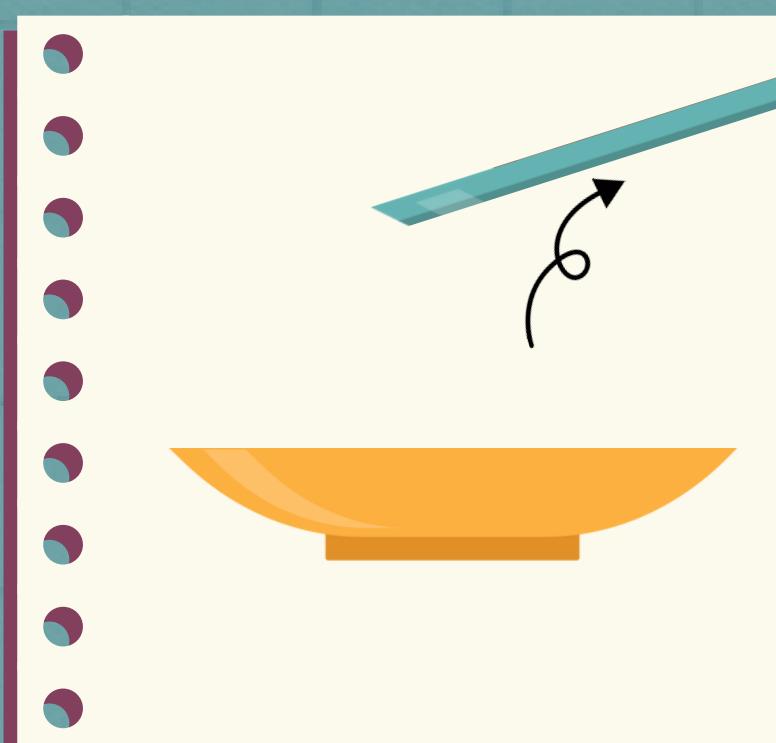
El término de LIFO, funciona de la siguiente manera:



Si queremos agregar un plato a nuestra pila de platos, se coloca uno por encima de otro.

¿Cómo funciona?

El término de LIFO, funciona de la siguiente manera:



Si queremos quitar un plato de nuestra pila de platos, se comenzará quitando desde el último plato, de nuestra pila.

*No sería correcto quitar un plato que esté en medio o al principio

Resumen Pila

La pila agrega y elimina elementos únicamente por un extremo.



Añadir Elemento

Podemos utilizar el método `.append()`

```
pila = [3, 4, 5]
pila.append(6)
pila.append(7)
print(pila)
```

```
[3, 4, 5, 6, 7]
```



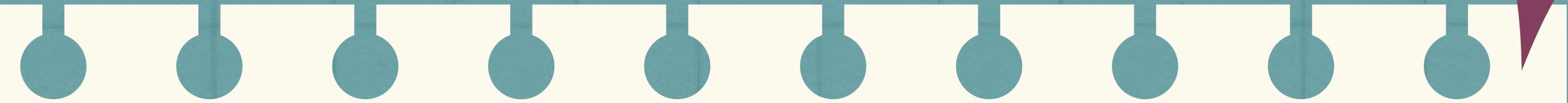
Sacar Elemento

Podemos utilizar el método `.pop()`

```
print(pila.pop())
print(pila)
```

```
7
[3, 4, 5, 6]
```

Colas



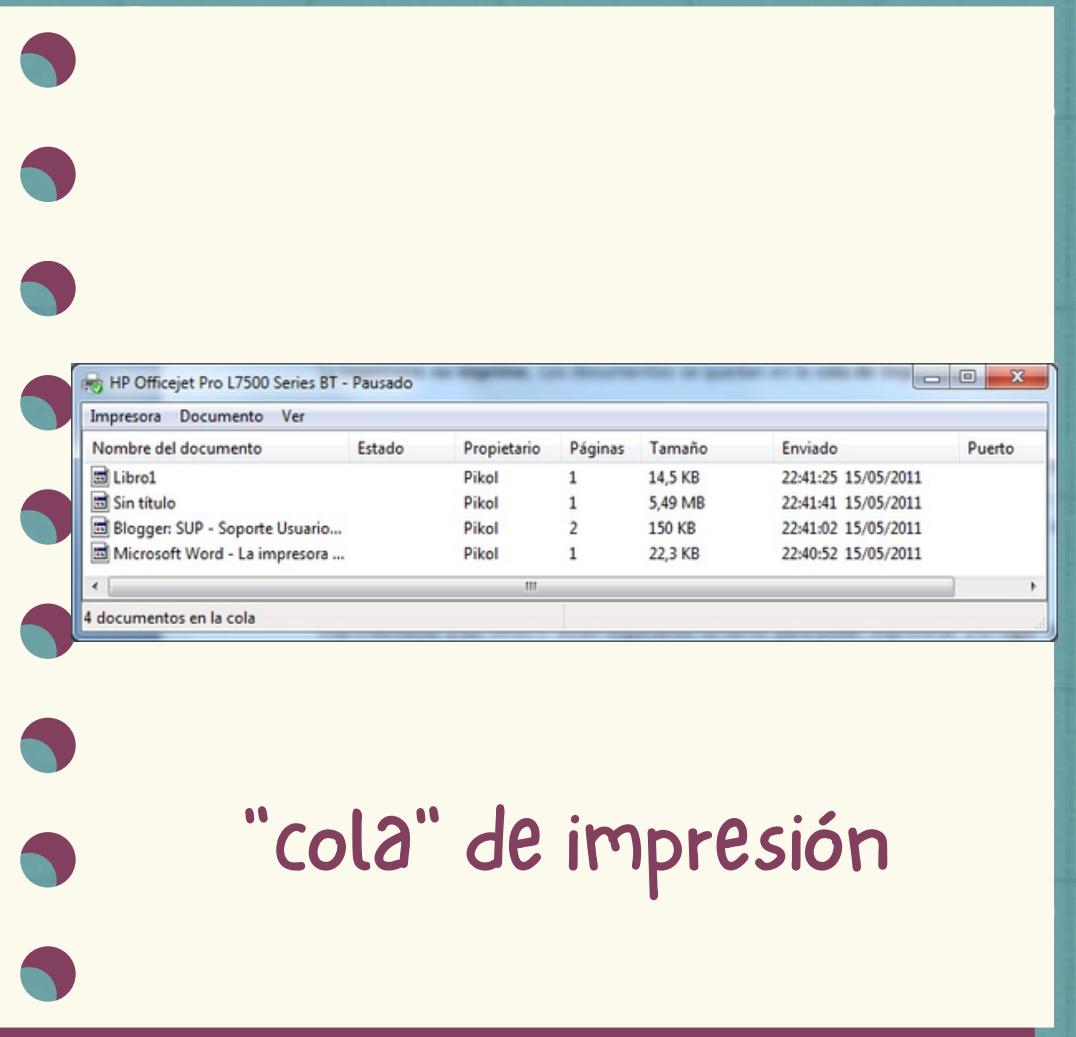
¿Qué es una cola?

Una cola (queue) es un TDA, que representa a un grupo ordenado de elementos o una secuencia de elementos.

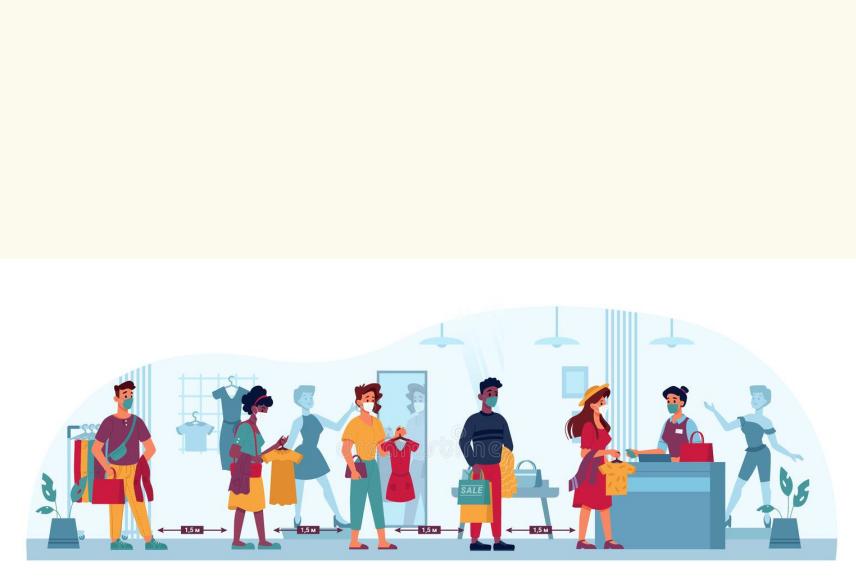
Utiliza el concepto de:

FIFO
(First IN, First OUT)

(El primero en entrar,
es el primero en salir)



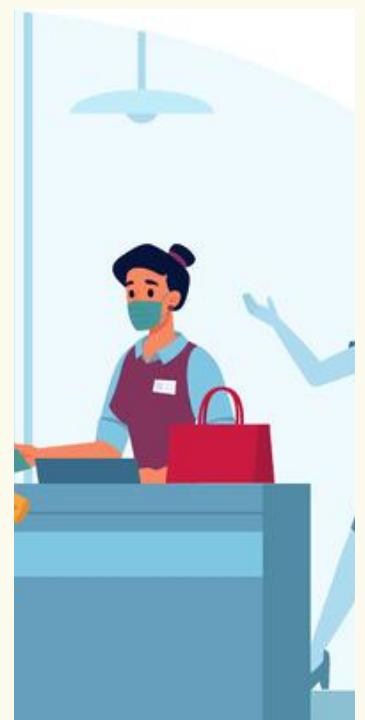
"cola" de impresión



"cola" de un banco,
supermercado, tienda

¿Cómo funciona?

El término FIFO, funciona de la siguiente manera.



Las personas hacen "cola" para poder comprar sus prendas de ropa



Jeniffer es la primera en llegar



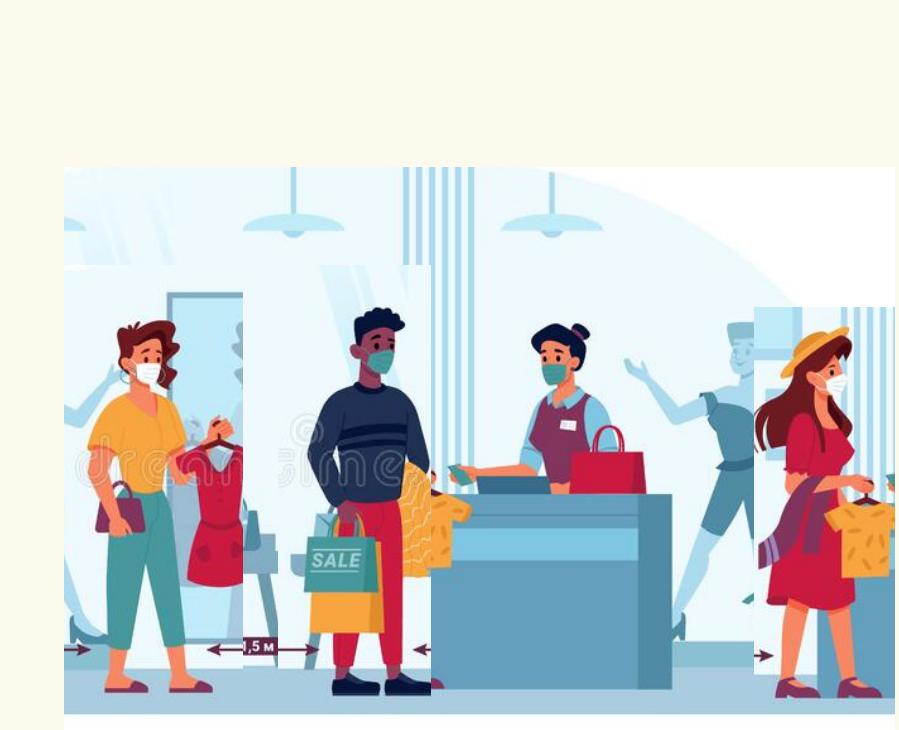
Diego es el segundo en meterse a la cola

¿Cómo funciona?

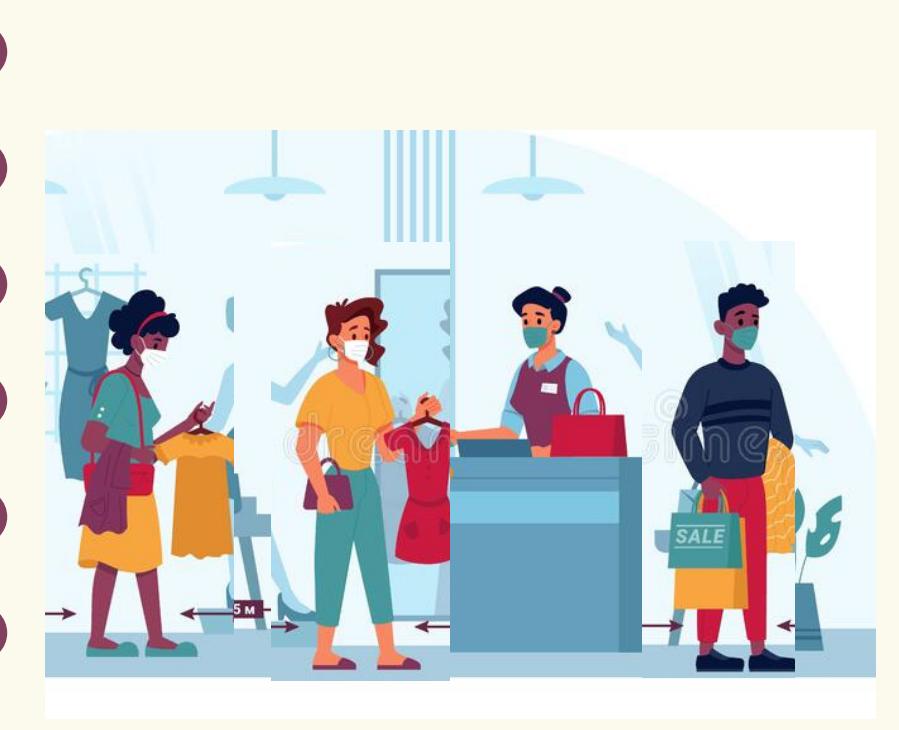
El término FIFO, funciona de la siguiente manera.



Cuando jennifer termina de pagar, sale de la cola.
(ES LA PRIMERA EN SALIR)

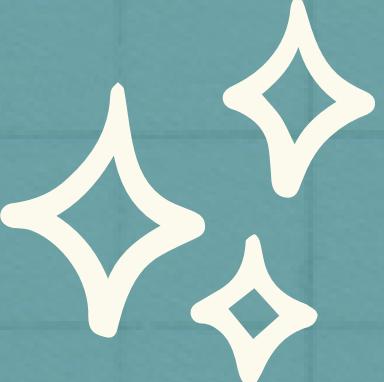


No cambia el hecho de que
jennifer fue la primera en
entrar, y por consiguiente,
la primera en salir



Diego sale de la cola, después de haber
entrado por detrás de la primera
posición.

Y así se repite sucesivamente....



Resumen Colas

Elementos entran por el final y salen por el principio.

- Se puede usar librería collections

```
cola = deque()  
cola.append('Maria')  
print(cola.popleft())
```

```
from collections import deque
```

- Se puede usar .append() para insertar un elemento por el final. Y podría salir usando .pop(0), para simular que sale desde la primera posición.





¡Muchas
Gracias!

Próxima Clase: Uso de excepciones y Manejo de Archivos -> 15 de septiembre