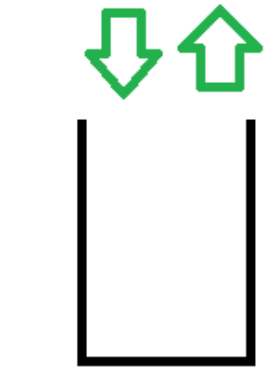
..(sigue) Colecciones

Pilas y Colas

PILA
(LAST IN, FIRST OUT)



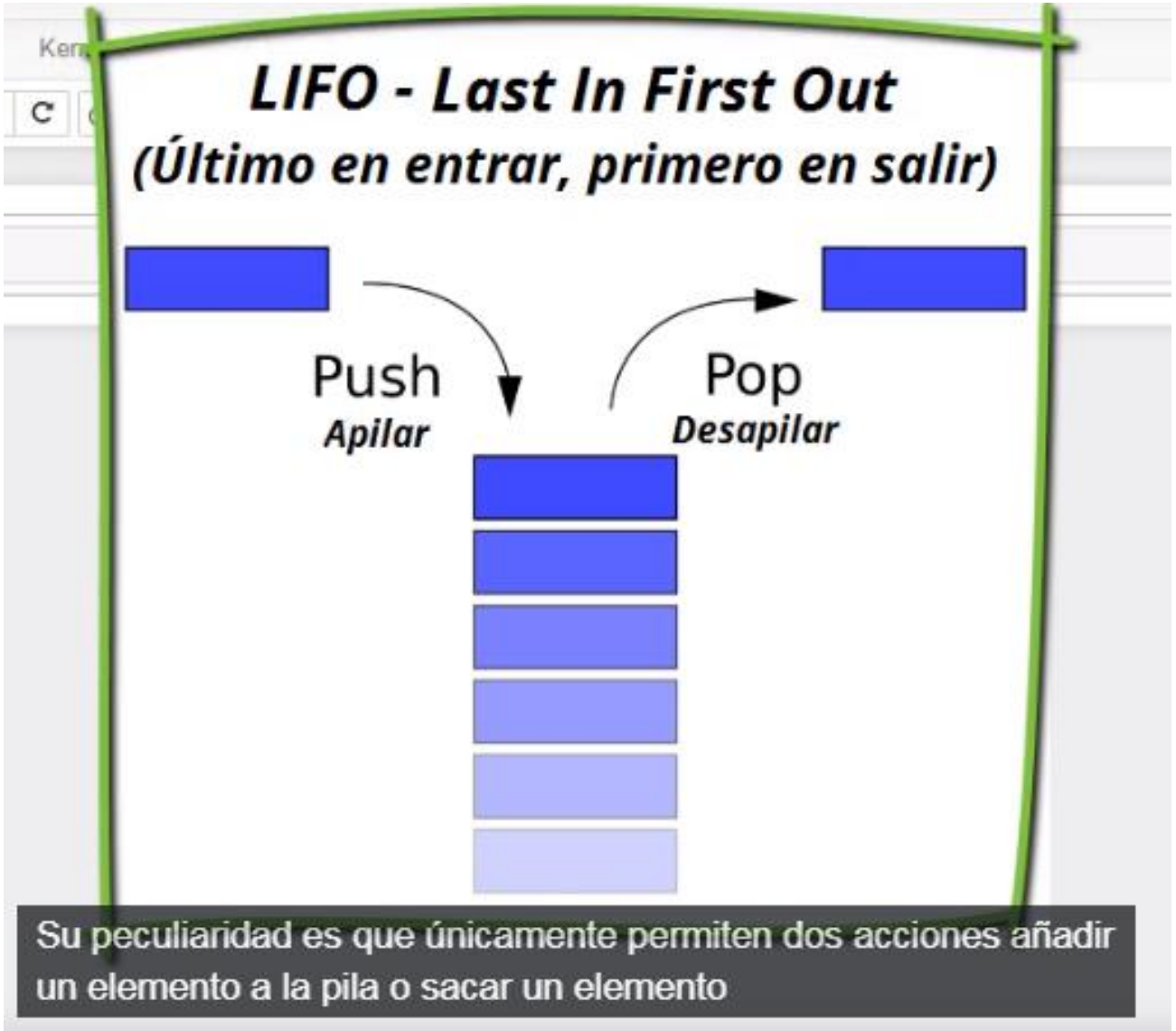
- Tuplas
- Conjuntos
- Diccionarios

- Pilas
- Colas

simulables con listas

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Pilas


LIFO - Last In First Out
(Último en entrar, primero en salir)
Push
Apilar
Pop
Desapilar
Su peculiaridad es que únicamente permiten dos acciones añadir un elemento a la pila o sacar un elemento

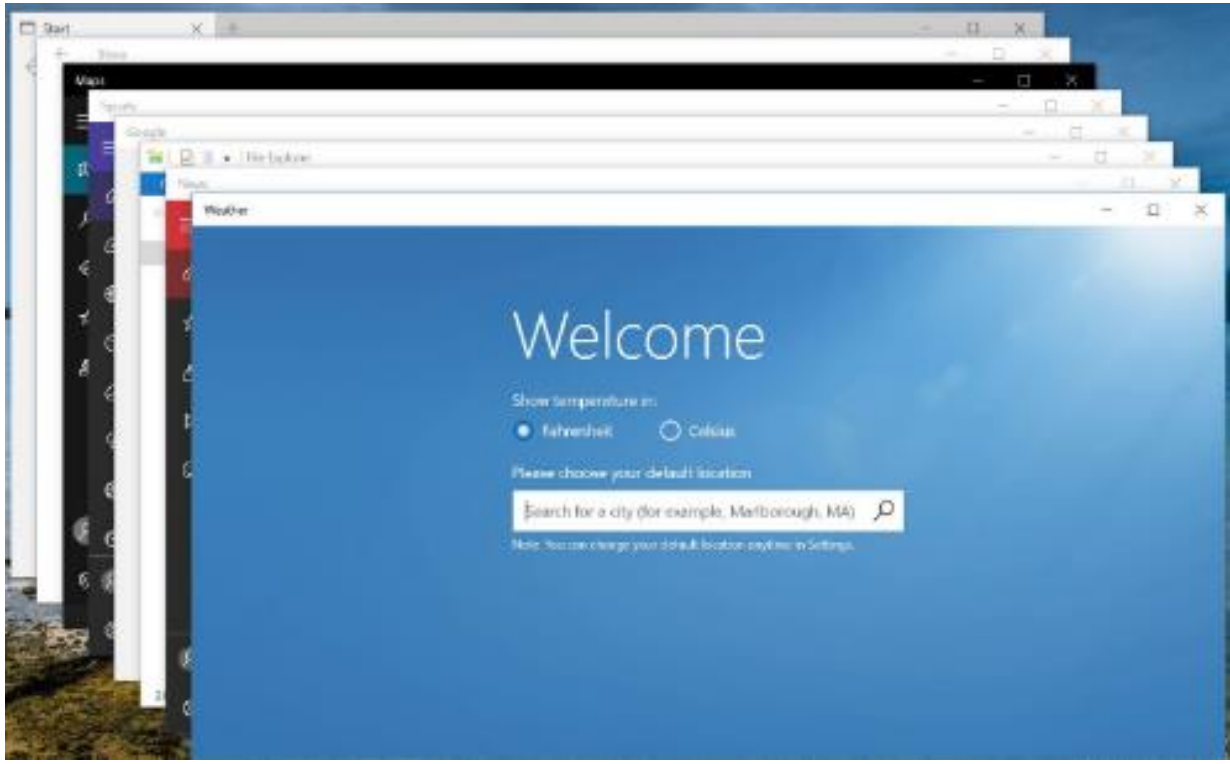
```
fruits = ['apple', 'banana', 'cherry']

x = fruits.pop() ; print(x)
x = fruits.pop() ; print(x)
x = fruits.pop() ; print(x)
```

cherry
banana
apple

```
fruits = ['apple', 'banana', 'cherry']

x = fruits.pop(0) ; print("sale : ", x, " queda: ", fruits)
x = fruits.pop(1) ; print("sale : ", x, " queda: ", fruits)
x = fruits.pop() ; print("sale : ", x, " queda: ", fruits)
```

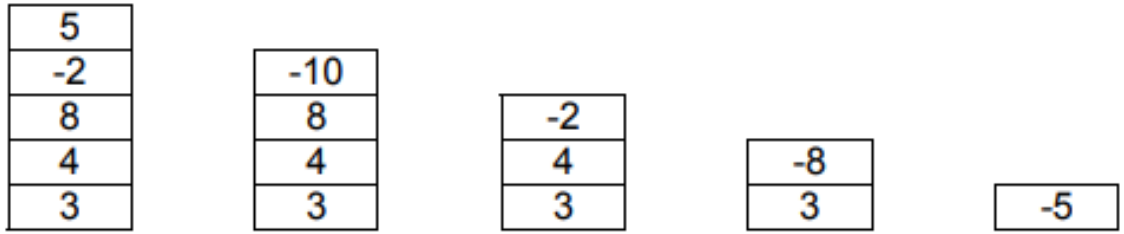
sale : apple queda: ['banana', 'cherry']
sale : cherry queda: ['banana']
sale : banana queda: []

```
In [1]: pila = [3,4,5]

In [2]: pila.append(6)

In [3]: pila.append(7)

In [4]: pila
Out[4]: [3, 4, 5, 6, 7]

In [5]: pila.pop()
Out[5]: 7

In [6]: pila
Out[6]: [3, 4, 5, 6]
```



Las pilas se utilizan en muchas aplicaciones que utilizamos con frecuencia. Por ejemplo, la gestión de ventanas en Windows (cuando cerramos una ventana siempre recuperamos la que teníamos detrás). Otro ejemplo es la evaluación general de cualquier expresión matemática para evitar tener que calcular el número de variables temporales que hacen falta. Por ejemplo:

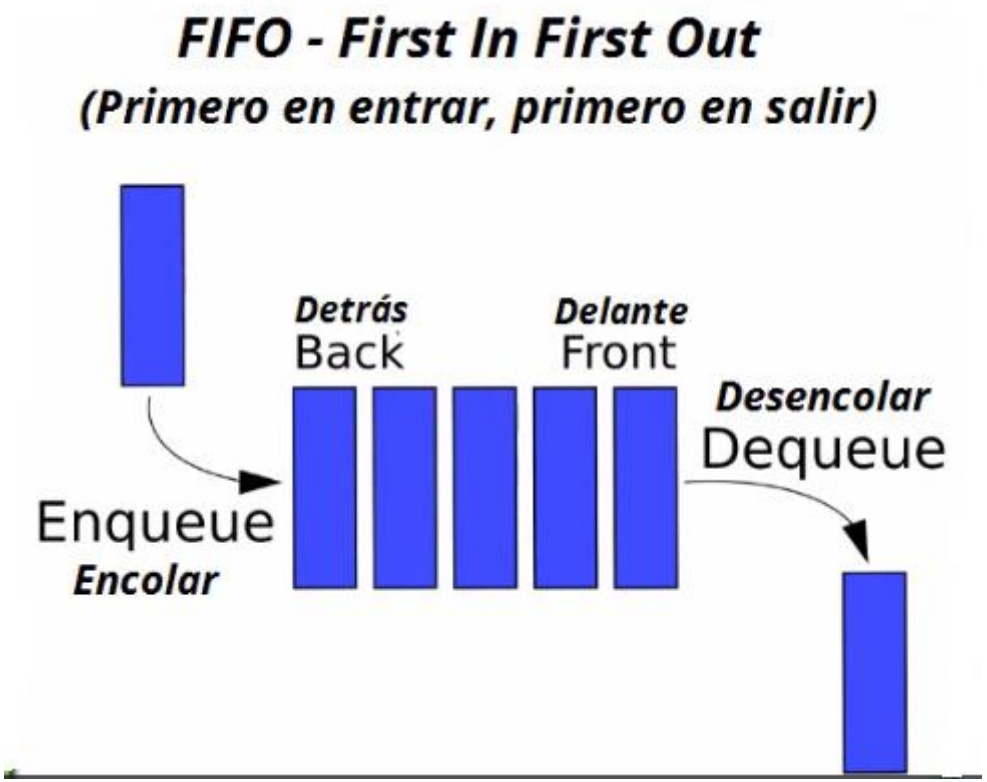
3 + 4 * (8 – 2 * 5)



```
In [9]: pila
Out[9]: [3, 4, 5]

In [10]: pila.pop()
         pila.pop()
         pila.pop()
Out[10]: 3

In [11]: pila
Out[11]: []

In [12]: pila.pop()
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-12-3900970cfbef> in <module>()
----> 1 pila.pop()

IndexError: pop from empty list
```

con pop no sucede un error.

Colas



```
In [13]: from collections import deque

In [14]: cola = deque()

In [15]: cola
Out[15]: deque([])

In [16]: cola = deque(['Hector','Juan','Miguel'])

In [17]: cola
Out[17]: deque(['Hector', 'Juan', 'Miguel'])
```

```
In [18]: cola.append('Maria')

In [19]: cola.append('Arnaldo')

In [20]: cola
Out[20]: deque(['Hector', 'Juan', 'Miguel', 'Maria', 'Arnaldo'])

In [21]: cola.popleft()
Out[21]: 'Hector'

In [22]: cola
Out[22]: deque(['Juan', 'Miguel', 'Maria', 'Arnaldo'])

In [23]: p = cola.popleft()

In [24]: p
Out[24]: 'Juan'

In [25]: cola
Out[25]: deque(['Miguel', 'Maria', 'Arnaldo'])
```

Práctica P01 Colas

Necesitamos un programa para gestionar la cola de un establecimiento, mediante un Menú con las siguientes opciones :

0.Salir 1.Coger Ticket 2.Despachar 3.Listar

- El número de ticket al iniciar el programa empieza en cero y va aumentando en uno cada vez que se emite un ticket.
- Cada vez que llega una persona pulsa "1. Coger Ticket" , el programa suma 1 al ticket y imprime ("Su número de ticket es : ", ticket), y lo guarda en la cola.
- Cada vez que un empleado queda disponible pulsa "2. Despachar" y el programa obtiene el elemento de la cola y muestra el mensaje "Por favor pase al mostrador el ticket <n>", donde <n> es el elemento obtenido de la cola.
- En algunas ocasiones pulsaremos 3. para ver que tickets están todavía en cola.
- A continuación se muestra el algoritmo.
- Determina si el algoritmo funciona y realiza el programa en Python.

```
# Algoritmo gestiona cola

#-------- definiciones
variables
   entero ticket
   array cola


#-------- proceso
inicio

   ticket = 0
   opcion = -1

   mientras ( opcion != 0 )
      escribir ("0.Salir    1.Coger Ticket    2.Despachar a   3.Listar")
      opcion = leer("opcion")

      si ( opcion != 0 )
        si ( opcion == 1 )
           ticket = ticket +1
           escribir ("Su ticket es : ", ticket)
           poner_en_cola (cola, ticket)
        sino
           si ( opcion == 2 )
              sigue = leer_cola (cola)
              escribir ("Pase al mostrador el número : ", sigue)
           sino
              si ( opcion == 3 )
                 imprimir_cola (cola)
              fin_si
           fin_si
        fin_si
      fin_si
   fin_mientras

   escribir("Adios hasta otra")

fin
```

Al programar adapta la **definición de cola**, y las funciones

poner_en_cola,

leer_cola,

imprimir_cola a la existentes en Python.

Solución Ejercicio colas

```
from collections import deque
cola = deque()
opcion = -1
ticket = 0


while opcion != 0 :
    print ("0.Salir    1.Coger Ticket    2.Despachar a   3.Listar" )
    print ("-------------------------------------------------" )
    opcion = int (input ("opcion :") )

    if opcion != 0:
        if opcion == 1 :
            ticket +=1
            print (">>> Su número de ticket  es :", ticket)
            cola.append(ticket)
        elif  opcion == 2 :
              if len(cola) > 0 :
                 p = cola.popleft()
                 print (">>>Por favor pase al mostrador el ticket : ", p )
        elif opcion == 3 :
             print (cola)
```