COMANDOS
listas=[]
- mutables

count = cuantas veces hay ese objeto

```
lista_nueva = [1, 2, 3, 4, 5]

lista_nueva.append(3)

print(lista_nueva)

print(lista_nueva.count(3))
```

```
[1, 2, 3, 4, 5, 3]
2
```

```
print(lista_nueva.index(4))
```

```
[1, 2, 3, 4, 5, 3]
2
3
```

index devuelve la posicion del 1er elemento

```
lista_nueva.remove(3)

print(lista_nueva)
```

```
[1, 2, 3, 4, 5, 3]
2
3
[1, 2, 4, 5, 3]
```

remove quita la 1ra posición de ese elemento
TUPLA=()
- ordenada
- heterogenea

CONJUNTOS={}
- no ordenados
- mutables
- noo se repite

```
# Conjunto

print(set([5, 2, 5, 1, 1.5]))
print(set((5, 2, 5, 1, 1.5)))
print(set("52511.5"))
```

```
{1, 2, 5, 1.5}
{1, 2, 5, 1.5}
{'.', '5', '1', '2'}
```

```
conjunto_2 = set([5, 3, 5, 6])
conjunto_3 = set([4, 2])
print(conjunto, conjunto_2, conjunto_3)
print(conjunto.intersection(conjunto_2))
```

```
{2, 3, 4} {3, 5, 6} {2, 4}
{3}
```

intersection es el punto donde las 2 coinciden
o utilizar &

```
print(conjunto & conjunto_2)
```

```
print(conjunto_2.issubset(conjunto))
print(conjunto_3.issubset(conjunto))
```

issubset si los elementos de un conjuntop estan en otro conjunto

DICCIONARIO= {clave:valor}

```
diccionario = {1: "Uno", 2: "Dos"}
diccionario[3] = "Tres"
print(diccionario)

dict_lista_tuplas = dict([(1, "Uno"), (2, "Dos"), (3, "Tres")])
print(dict_lista_tuplas)

dict_lista_string = dict(Uno = 1, Dos = 2, Tres = 3)
print(dict_lista_string)
```

```
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{'Uno': 1, 'Dos': 2, 'Tres': 3}
```

```python
diccionario = {1: "Uno", 2: "Dos"}
diccionario[3] = "Tres"
print(diccionario)

dict_lista_tuplas = dict([(1, "Uno"), (2, "Dos"), (3, "Tres")])
print(dict_lista_tuplas)

dict_lista_string = dict(Uno = 1, Dos = 2, Tres = 3)
print(dict_lista_string)

dict_tipos = {1: "integer", 2.2: "float", "texto": "string", (1, 2): "tupla"}
print(dict_tipos)

dict_repeticion = {1: "Primero", 1: "Último"}
print(dict_repeticion)

print(diccionario, diccionario.keys(), diccionario.values(), diccionario.items())
claves = diccionario.values()
print(claves)
diccionario[1] = "One"
print(claves)
```

```
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{'Uno': 1, 'Dos': 2, 'Tres': 3}
{1: 'integer', 2.2: 'float', 'texto': 'string', (1, 2): 'tupla'}
{1: 'Último'}
{1: 'Uno', 2: 'Dos', 3: 'Tres'} dict_keys([1, 2, 3]) dict_values(['Uno', 'Dos', 'Tres']) dict_items([(1, 'Uno'), (2, 'Dos'), (3, 'Tres')])
dict_values(['Uno', 'Dos', 'Tres'])
dict_values(['One', 'Dos', 'Tres'])
```

```python
print(claves)
diccionario.pop(2)
print(diccionario)
```

```
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{1: 'Uno', 2: 'Dos', 3: 'Tres'}
{'Uno': 1, 'Dos': 2, 'Tres': 3}
{1: 'integer', 2.2: 'float', 'texto': 'string', (1, 2): 'tupla'}
{1: 'Último'}
{1: 'Uno', 2: 'Dos', 3: 'Tres'} dict_keys([1, 2, 3]) dict_values(['Uno', 'Dos', 'Tres']) dict_items([(1, 'Uno'), (2, 'Dos'), (3, 'Tres')])
dict_values(['Uno', 'Dos', 'Tres'])
dict_values(['One', 'Dos', 'Tres'])
{1: 'One', 3: 'Tres'}
```

len saber el total de posiciones

```
s = 'nombre,edad,carrera,matricula'
partes = s.split(',')
```

.split considera los espacios

startswith

Regresa verdadero si una cadena empieza con una subcadena

```
print('hola mundo'.startswith('hola'))
```

True

endswith

Regresa verdadero si una cadena termina con cierta subcadena

```
print('hola mundo'.endswith('mundo'))
```

True

join

Concatena cadenas en una lista de cadenas, usando la cadena actual como separador

```
print(','.join(['hola', 'mundo', 'mundial']))
```

hola,mundo,mundial

Si necesitas modificar caracteres de una cadena, una forma simple y eficiente de hacerlo es primero convertir la cadena a lista, mediante la función list y luego de las modificaciones regresar a cadena con la función join usando la cadena vacía como separador

```
s = 'hola mundo'
ls = list(s)
print(ls)
ls[0] = 'H'
s = ''.join(ls)
print(s)
```

Insert
Al inicio
Cuidado: insert permite posiciones inválidas (mayores a la longitud o menores a 0)

```
l = [2, 3, 4]
l.insert(0, 1)
print(l)
```

R= [1, 2, 3, 4]

Al final
Usar método append de lista

```
l = [1, 2, 3]
l.append(4)
print(l)
```

R=[1, 2, 3, 4]

En cualquier posición
Con el método insert

```
l = [1, 2, 4]
l.insert(2, 3)
print(l)
```

[1, 2, 3, 4]

Remplazar elementos
Se puede con asignación directa
Es una operación in-place

```
l = [0, 2, 3]
l[0] = 1
print(l)
```

[1, 2, 3]
Borrar elementos
Operación in-place
Se utiliza la función general del que funciona para varias estructuras de datos de Python

```
l = [0, 1, 2, 3]
del(l[0])
```

```
print(l)
[1, 2, 3]
```

Obtener sublistas
Es una operación no mutable
Usando rebanadas (como se vio en el tema anterior)
Las rebanadas regresan nueva memoria (por eso son no mutables)

```
l = [1, 2, 3, 4]
print(l[:-1]) # todos menos último
print(l[1:]) # todos menos primero
print(l[1:-1]) # sin primero y último
[1, 2, 3]
[2, 3, 4]
[2, 3]
```

Orden ascendente

```
l = [44, 11, 7, 22]
l2 = sorted(l)
l.sort()
print(l)
print(l2)
[7, 11, 22, 44]
[7, 11, 22, 44]
```

Orden descendente
Se logra con el parámetro nombrado (keyword) reverse

```
l = [44, 11, 7, 22]
l2 = sorted(l, reverse=True)
l.sort(reverse=True)
print(l)
print(l2)
[44, 22, 11, 7]
[44, 22, 11, 7]
```

PILAS

Push: agrega un elemento al tope

Pop: saca y regresa el elemento al tope

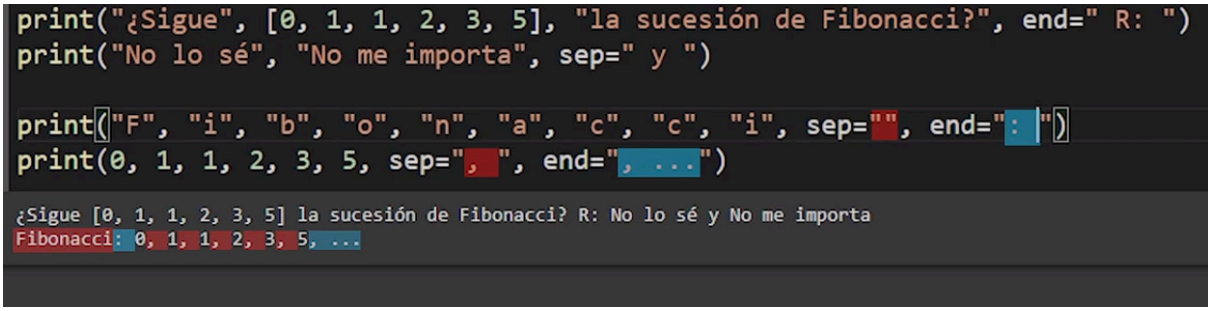
Peek: sólo regresa el valor del elemento del tope sin sacarlo

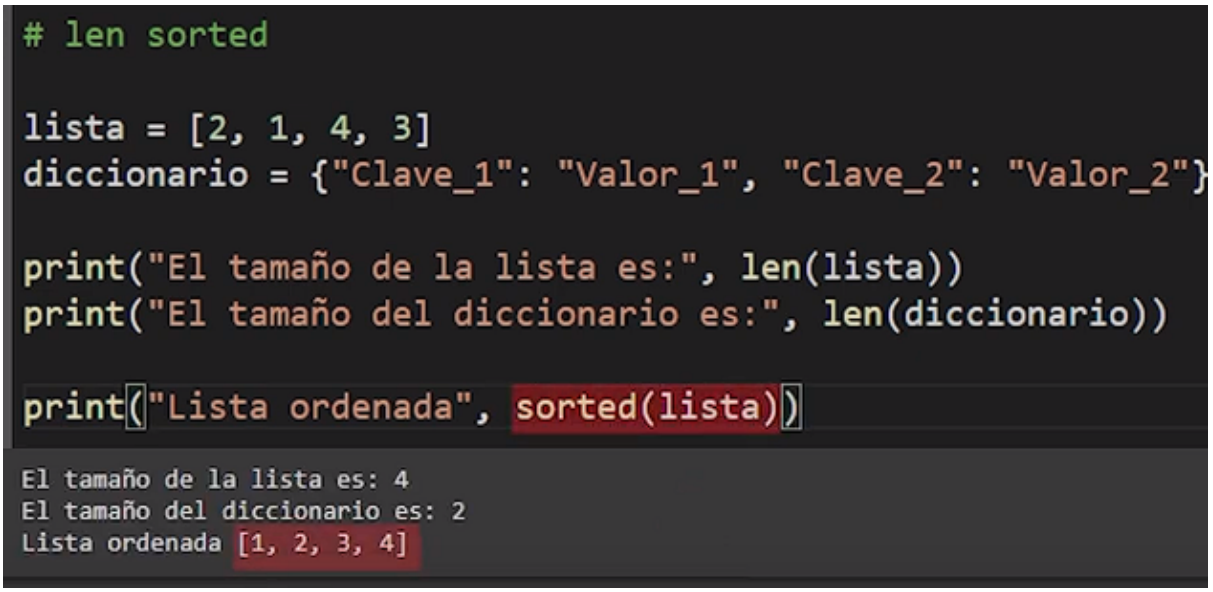
```
pila = []
pila.append(1) # equivalente de push
pila.append(2)
pila.append(3)

tope = pila[-1] # equivalente de peek
```

print(tope)

tope = pila.pop()
print(tope)
print(pila)

COLAS
append: agrega un elemento al final (igual que en una lista)
shift: saca el elemento del frente y lo regresa
peek: sólo regresa el valor del elemento del frente sin sacarlo





```python
# len sorted

lista = [2, 1, 4, 3]
diccionario = {"Clave_1": "Valor_1", "Clave_2": "Valor_2"}

print("El tamaño de la lista es:", len(lista))
print("El tamaño del diccionario es:", len(diccionario))

print("Lista ordenada", sorted(lista))
print("Lista ordenada inversa:", sorted(lista, reverse=True))
```

```
El tamaño de la lista es: 4
El tamaño del diccionario es: 2
Lista ordenada [1, 2, 3, 4]
Lista ordenada inversa: [4, 3, 2, 1]
```

```python
class Personaje:

    def __init__(self, nombre, fuerza, inteligencia, defensa, vida):
        self.nombre = nombre
        self.fuerza = fuerza
        self.inteligencia = inteligencia
        self.defensa = defensa
        self.vida = vida

mi_personaje = Personaje("BitBoss", 10, 1, 5, 100)
print("El nombre del jugador es", mi_personaje.nombre)
print("La fuerza del jugador es", mi_personaje.fuerza)
```

```
El nombre del jugador es BitBoss
La fuerza del jugador es 10
```