Colección de datos lineales (listas)

CI-0202 Principios de informática

Sivana Hamer - sivana.hamer@ucr.ac.cr
Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica

UNIVERSIDAD DE COSTA RICA

Estructuras de datos

Son **estructuras** que **guardan datos**. Algunos son:

- Listas
- Diccionarios
- Tuplas
- Conjuntos
- Pilas

Nota

Las listas son de una dimensión, es decir **unidimensionales**.

Nota

Se pueden **combinar** estructuras de datos.

Listas (1)

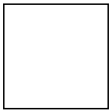
Es una estructura de datos que guarda elementos **ordenados**. El orden se refiere a que los elementos están en una **secuencia**.

Ejemplo en Python

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

Listas (2)

Las listas pueden estar vacias.



Ejemplo en Python

```
>> lista = []
>> print(lista)
[]
>> print(type(lista))
<class 'list'>
```

Listas (3)

En Python, las listas pueden tener distintos tipos de elementos.

Ejemplo en Python

```
lista = [1, 2, 3.2, "Hola"]
```

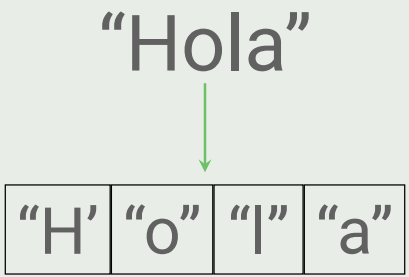
Nota

No es común guardar distintos tipos de elementos.

Listas (4)

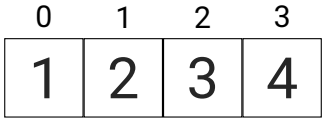
Fun fact!

Los strings se guardan como listas en memoría. Cada letra es un byte.



Accesos a elementos (1)

Se puede acceder solamente **un elemento** en la lista. Se accede al i-ésimo elemento de un arreglo x con `x[i]`.

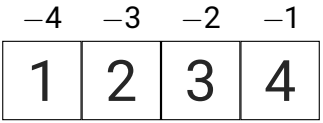


Ejemplo en Python

```
>> lista=[1, 2, 3, 4]
>> print(lista[0])
1
>> print(lista[1])
2
>> print(lista[2])
3
>> print(lista[3])
4
```

Accesos a elementos (2)

En Python, se puede acceder los elementos utilizando indices negativos.



Ejemplo en Python

```
>> lista=[1, 2, 3, 4]
>> print(lista[-1])
4
>> print(lista[-2])
3
>> print(lista[-3])
2
>> print(lista[-4])
1
```

Accesos a elementos (3)

Importante

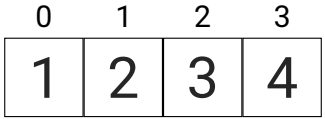
Si se accede a un elemento fuera del rango de la lista, da una excepción.

Importante

Los indices tienen que ser números enteros.

Sublistas

En Python, se puede acceder los elementos utilizando indices negativos.

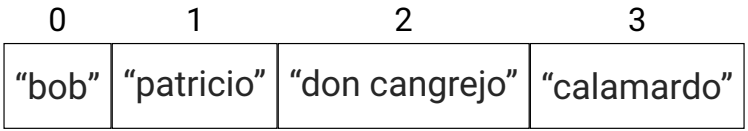


Ejemplo en Python

```
lista=[1,2,3,4]
print(lista[1:]) # Devuelve los elementos del segundo en adelante
print(lista[:3]) # Devuelve los elementos hasta el cuarto (sin incluirlo)
print(lista[1:3]) # Devuelve los elementos del segundo hasta el cuarto (sin incluirlo)
print(lista[:]) # Devuelve una copia del arreglo
```

Recorrer (1)

Recorrer una lista implica **acceder en una secuencia los elementos de una lista**.



Ejemplo en Python

```
personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]

for indice in range(0, 4):
    #Nota en range esta hardcodeado el limite
    print(personajes_bob_esponja[indice])
```

Recorrer (2)

También se puede recorrer los **elementos**.

“bob”	“patricio”	“don cangrejo”	“calamardo”

Ejemplo en Python

```
personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]

for personaje in personajes_bob_esponja:
    print(personaje)
```

Recorrer (3)

Se puede obtener los **elementos** y los **indices** con **enumerate**.

Ejemplo en Python

```
personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]

for indice, personaje in enumerate(personajes_bob_esponja):
    print(f"El personaje {indice + 1} es {personaje}.")
```

Length

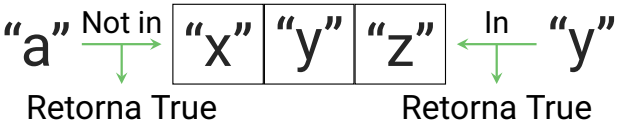
Se puede calcular el **largo** de una lista con **len** (de length).

Ejemplo en Python

```
>>> lista = [1, 2, 5]
>>> print(len(lista))
3
```

In y not in

Se puede ver si los **elementos existen en la lista** con **in** (si existe en la lista) and **not in** (si no existe en la lista).

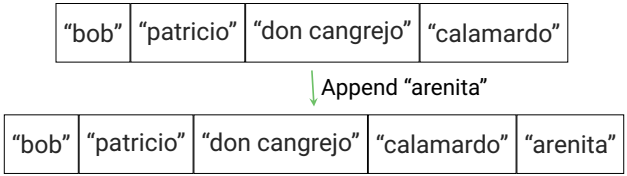


Ejemplo en Python

```
>> lista = ["x", "y", "z"]
>> print("y" in lista)
True
>> print("a" in lista)
False
>> print("a" not in lista)
True
```

Append

Agrega el elemento al final de la lista.

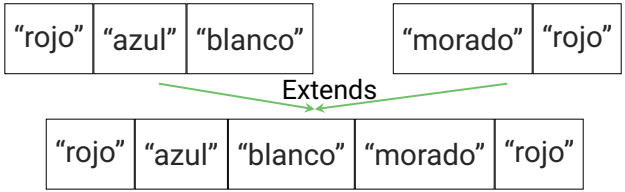


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> print(personajes_bob_esponja)
['bob', 'patricio', 'don cangrejo', 'calamardo']
>> personajes_bob_esponja.append("arenita")
>> print(personajes_bob_esponja)
['bob', 'patricio', 'don cangrejo', 'calamardo', 'arenita']
```

Extend

Extiende la primera lista con los elementos de la segunda al final.

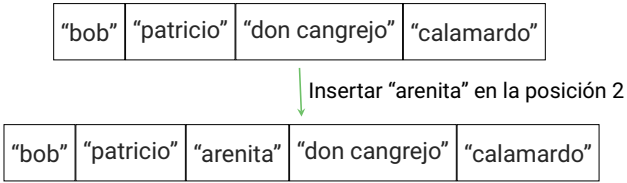


Ejemplo en Python

```
>> colores1 = ["rojo", "azul", "blanco"]
>> colores2 = ["morado", "rojo"]
>> colores1.extend(colores2)
>> print(colores1)
['rojo', 'azul', 'blanco', 'morado', 'rojo']
```

Insert

Agrega el elemento en la **posición i**.

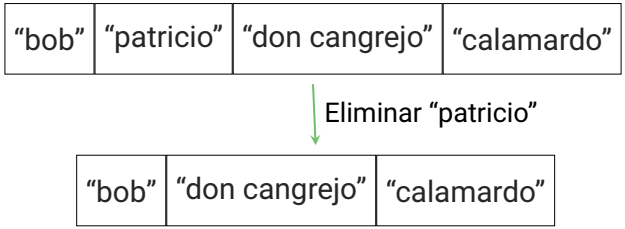


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> personajes_bob_esponja.insert(2, "arenita")
>> print(personajes_bob_esponja)
['bob', 'patricio', 'arenita', 'don cangrejo', 'calamardo']
```

Remove

Elimina el elemento de la lista.

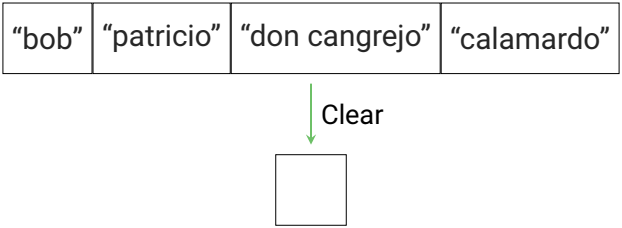


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> personajes_bob_esponja.remove("patricio")
>> print(personajes_bob_esponja)
['bob', 'don cangrejo', 'calamardo']
```

Clear

Limpia la lista.

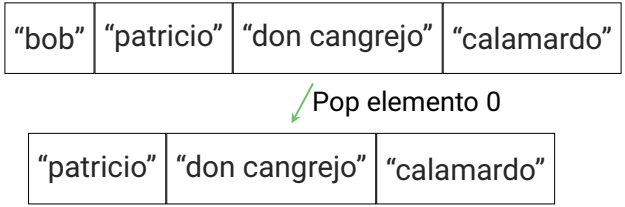


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> personajes_bob_esponja.clear()
>> print(personajes_bob_esponja)
[]
```

Pop

Remueve y retorna el elemento en la **posición i** del arreglo.

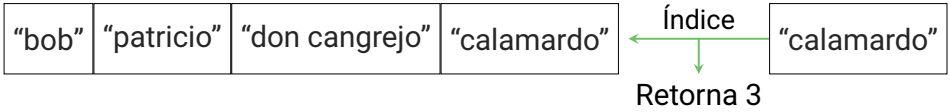


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> personaje = personajes_bob_esponja.pop(0)
>> print(personajes_bob_esponja)
['patricio', 'don cangrejo', 'calamardo']
>> print(personaje)
bob
```

Index

Retorna el índice de la **primera instancia** del elemento.

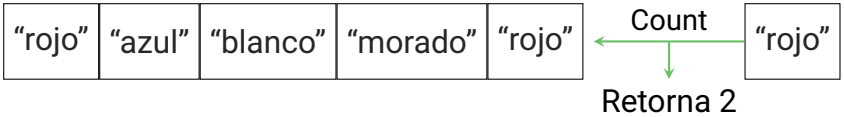


Ejemplo en Python

```
>> personajes_bob_esponja = ["bob", "patricio", "don cangrejo", "calamardo"]
>> indice_personaje = personajes_bob_esponja.index("calamardo")
>> print(indice_personaje)
3
```

Count

Retorna la cantidad de veces que se repite un elemento.



Ejemplo en Python

```
>> colors = ['rojo', 'azul', 'blanco', 'morado', 'rojo']
>> cuenta = colors.count('rojo')
>> print(cuenta)
2
```

Referencias I

L. Villalobos, “Colección de datos lineales (arreglos),” Material del curso CI-0202 Universidad de Costa Rica de Leonardo Villalobos, 2019.

C. Swaroop, *A Byte of Python*. Independent, 2020.

J. Elkner, A. B. Downey, and C. Meyers, “How to think like a computer scientist: Learning with python,” 2012.