ARREGLOS UNIDIMENSIONALES

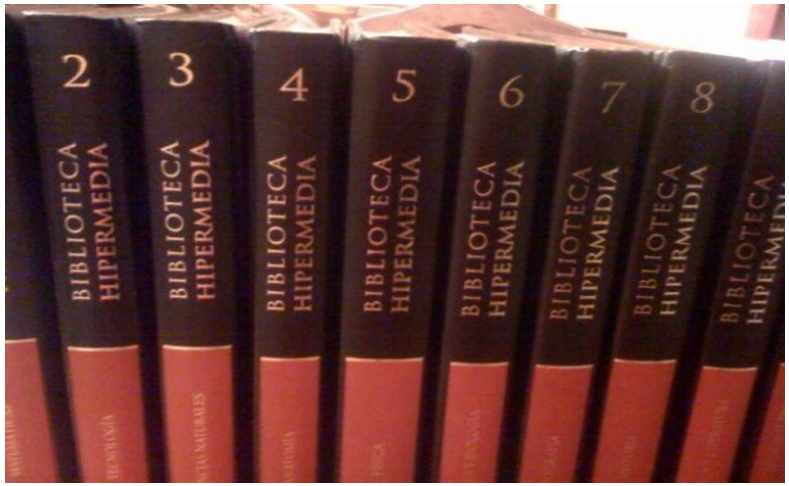
CONTENIDO

Definiendo los arreglos unidimensionales.

¿Qué es un arreglo?

*Un arreglo es una **lista (conjunto) de datos** con un **número fijo de componentes**, todos del **mismo tipo,** que están referenciados **bajo un mismo nombre**.

*Cada componente del arreglo se puede **acceder mediante índices** (0, 1, 2, 3, ...) encerradas entre corchetes [ ].



¿Para qué sirven los arreglos?

Los arreglos permiten manejar de forma **sencilla y directa** conjuntos de datos del mismo tipo, **de los cuales conocemos su cantidad** y con los cuales se realizarán operaciones similares.

Ejemplo1: Escriba un programa en Java que solicite los nombres de cada estudiante del curso y los muestre todos al final.

¿Para qué sirven los arreglos?

Ejemplo1: Escriba un programa en Python que solicite los nombres de cada estudiante del curso y los muestre todos al final.

Sin arreglos tendríamos un programa cuyo código sería algo como esto:

```
nombre1 =input(“Digite su nombre”)
nombre2=input(“Digite su nombre”)
. . .
nombre50=input (“Digite su nombre”)
. . .
```

¿Para qué sirven los arreglos?

Ejemplo1: Escriba un programa en Python que solicite los nombres de cada estudiante del curso y los muestre todos al final.

Sin arreglos tendríamos un programa cuyo código sería algo como esto:

```
nombre1 =input(“Digite su nombre”)
nombre2=input(“Digite su nombre”)
. . .
nombre50=input (“Digite su nombre”)
. . .
```

¡NO ES EFICIENTE!

¿Para qué sirven los arreglos?

Ejemplo2: Escriba un programa en Python que solicite los nombres de cada estudiante de **cualquier** curso y los muestre todos al final.

Sin arreglos tendríamos que declarar un número arbitrariamente grande de variables de tipo String (¿100?, ¿200?) de tal forma que nunca hubiera más estudiantes que variables del programa.

```
nombre1 =input(“Digite su nombre”)
nombre2=input(“Digite su nombre”)
. . .
nombre50=input (“Digite su nombre”)
. . .
. . .
```

¿Para qué sirven los arreglos?

Ejemplo2: Escriba un programa en Python que solicite los nombres de cada estudiante de **cualquier** curso y los muestre todos al final.

Sin arreglos tendríamos que declarar un número arbitrariamente grande de variables de tipo String (¿100?, ¿200?) de tal forma que nunca hubiera más estudiantes que variables del programa.

```
nombre1 =input(“Digite su nombre”)
nombre2=input(“Digite su nombre”)
. . .
nombre50=input (“Digite su nombre”)
. . .
. . .
```

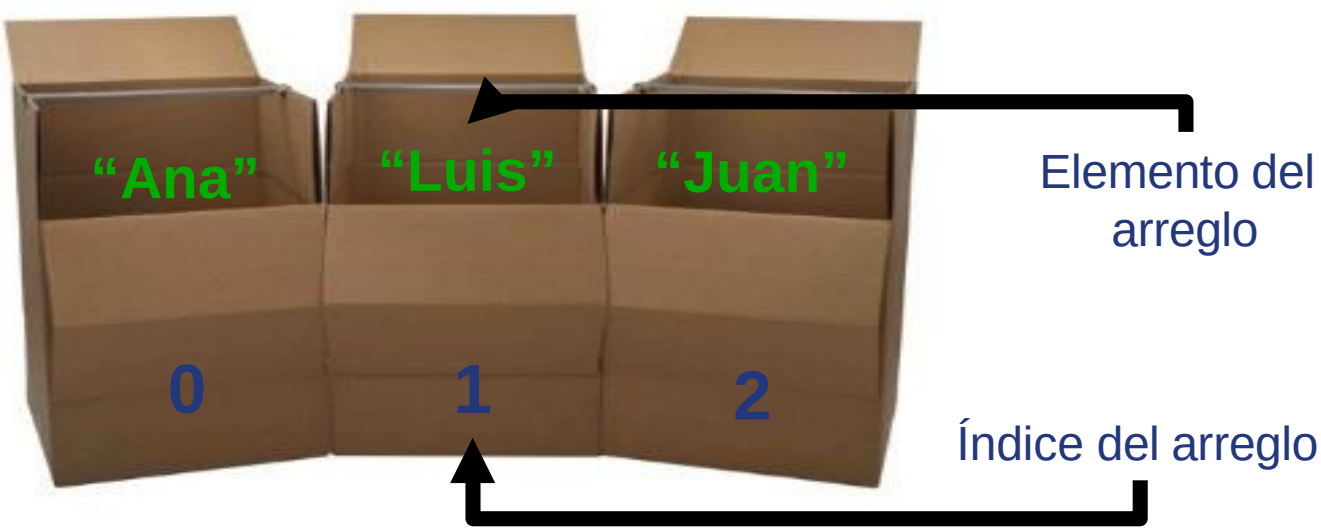
¡NO ES PRÁCTICO!

Arreglos: Espacios ordenados

Un arreglo se puede ver como un **conjunto de espacios finitos** donde se almacenan elementos (todos del mismo tipo).

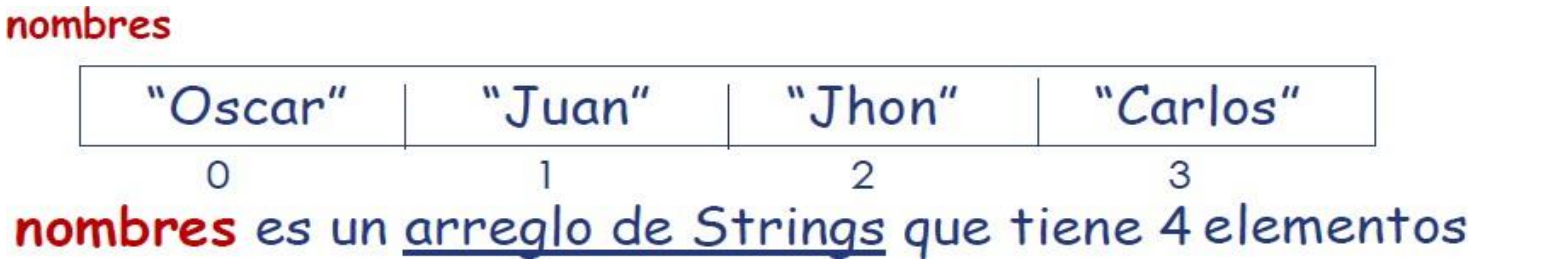
Un arreglo también puede verse como cajas ordenadas en fila y numeradas, donde en cada caja se almacena un solo elemento u objeto.

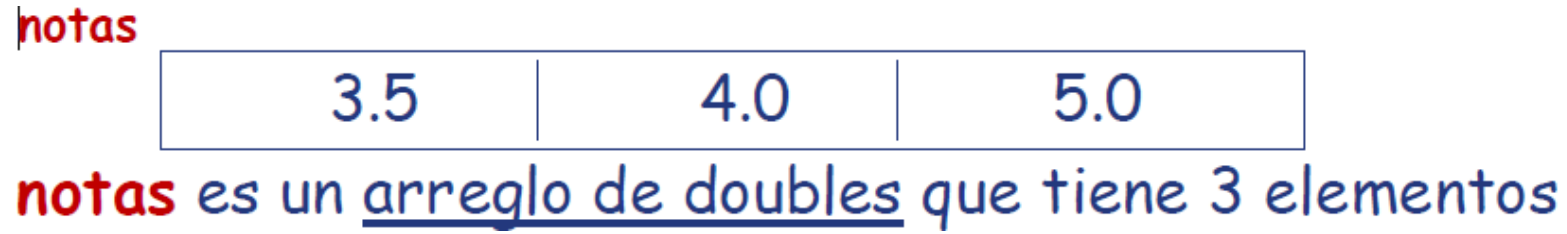
cajasDeNombres



Para recordar:

* Un arreglo se usa para almacenar elementos del mismo tipo.
* Cada elemento se guarda en un espacio independiente.
* Cada espacio se referencia con un índice (0,1,2,3,...,n).





Los Arreglos y Pyhton

Declarando y trabajando con arreglos en Python.

Declarando Arreglos

La declaración de un arreglo se hace de la siguiente forma:

arreglo = [ ]

Se Crea un arreglo Vacío

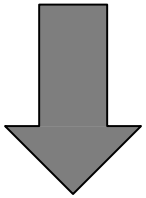
Ejemplos:

Notas = [ ]
edades = [ ]
nombres = [ ]

Inserción de datos

Para insertar datos en un arreglo se usa la función ***append()***

```
arreglo = []
arreglo.append(10)
arreglo.append(20)
arreglo.append(30)
```



0	1	2
10	20	30

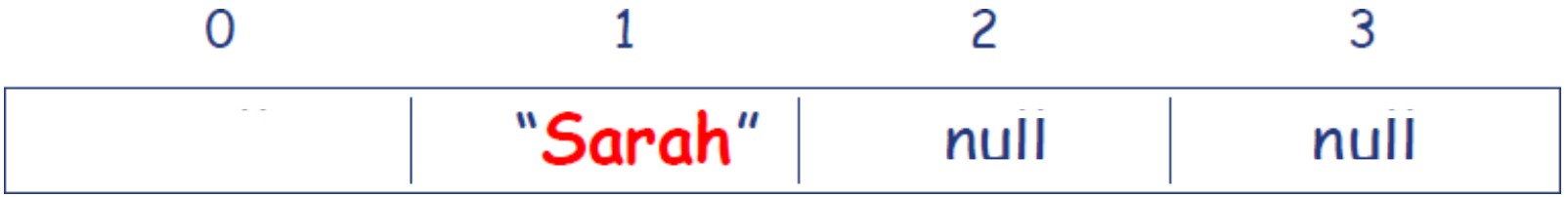
Inserción de datos

Otra forma de definir un arreglo, en este caso de cuatro elementos

```
nombres = [None]*4
```

```
nombres[1] = “Sarah”
```

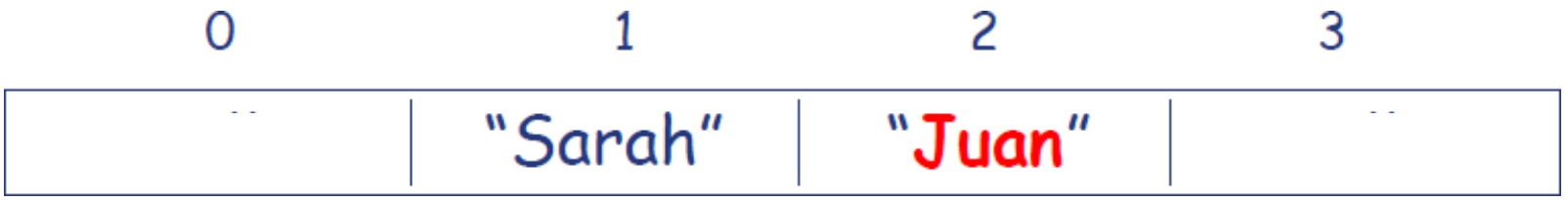
Guarda el texto “Sarah” En la posición 1 del arreglo.



Inserción de datos

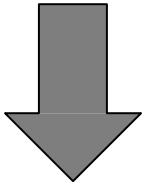
```
nombres = [None]*4

nombres[1] = “Sarah”

nombres[2] = “Juan”
```

Guarda el texto “Juan” En la posición 2 del arreglo



Inserción de datos

```
arreglo = []
arreglo.append(10)
arreglo.append(20)
arreglo.append(30)
```

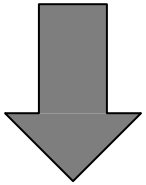


0	1	2
10	20	30

Cada **elemento** del arreglo se almacena en una posición identificada mediante un ***índice*** que inicia en cero.

Inserción de datos

```
arreglo = []
arreglo.append(10)
arreglo.append(20)
arreglo.append(30)
```



0	1	2
10	20	30

La función ***append*** coloca los elementos en el arreglo de acuerdo al orden de llegada.

Inserción de datos

Aunque *Python* permite insertar datos de diferentes ***tipos*** en un arreglo, una buena práctica de programación es que todos los elementos de un arreglo sean del ***mismo tipo*** .

```
notas = []
notas.append(4.5)
notas.append(5.0)
notas.append(3.0)
```

```
nombres = []
nombres.append(“Carlos”)
nombres.append(“Juan”)
nombres.append(“Marta”)
```

Inserción de datos

Otra forma de insertar datos en un arreglo es usar la función *insert ()*.

insert (índice, valor)

Se debe especificar el ***índice*** donde se desea insertar el nuevo ***valor***.

Inserción de datos

Otra forma de insertar datos en un arreglo es usar la función *insert ().*

insert (índice, valor)

Ejemplo:

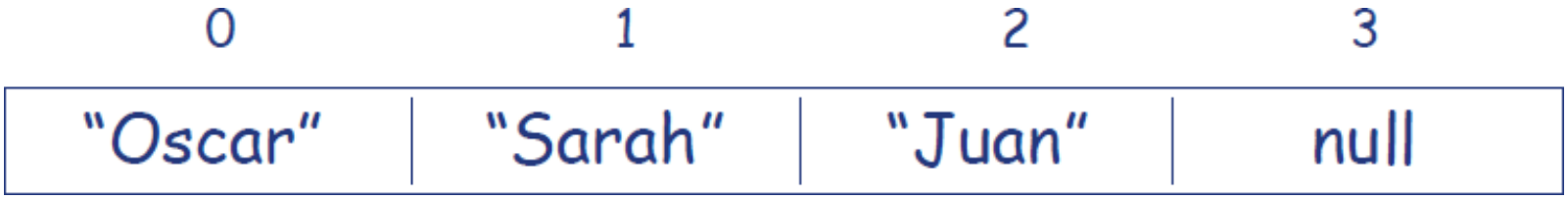
```
arreglo = []
arreglo.insert(0, 500)
arreglo.insert(1, 700)
arreglo.insert(2, 1000)
```

0	1	2
500	700	1000

Leer los datos de un arreglo

Para recuperar datos de un arreglo se debe indicar la posición(índice) del arreglo que se quiere conocer:

nombreArreglo[posición]



nombres[2] indica que el valor en la posición 2 es "Juan".

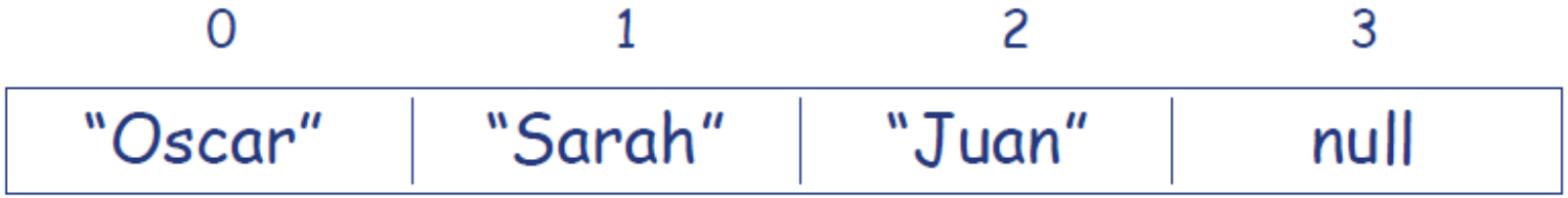
Leer los datos de un arreglo

```
print ("El valor en la posición 2 es " + nombres[2]);
```

0	1	2	3
"Oscar"	"Sarah"	"Juan"	null

```
El valor de la posición dos  es    Juan
```

Extracción de datos




```
x  =  nombres[2]
print (x)
```

En este caso se guarda el valor en una variable.

Imprimir un arreglo

```
arreglo = []
arreglo.append(10)
arreglo.append(20)
arreglo.append(30)
print ("Datos del arreglo")
print (arreglo)
```

```
>>>
Datos del arreglo
[10, 20, 30]
```

Recorrer un arreglo

Para recorrer un arreglo se utiliza el ciclo *for*

0	1	2	3	4	5	6	7	8	9
Juan	Pedro	Maria	Luis	Jose	Miguel	David	Lili	Luz	Ana

```
for i in range (0,10):
    print (nombres [i] )
```

```
Juan
Pedro
María
Luis
....
```

Trabajando con arreglos

Ejemplos prácticos de cómo se crean programas usando arreglos.

Trabajando con arreglos

- Presente el conjunto de instrucciones Python para **crear** un arreglo de enteros
- Adicione las instrucciones que necesite para **solicitar** al usuario cada uno de los 100 números
- después, **muestre todos** los números en un solo mensaje
- Luego, **muestre** sólo los **números** almacenados en **posiciones pares**
- y, **muestre** los **números impares** contenidos en el arreglo

Trabajando con arreglos

- Presente el conjunto de instrucciones Python para **crear** un arreglo de enteros

```
numeros = []
```

Trabajando con arreglos

Adicione las instrucciones que necesite para **solicitar** al usuario cada uno de los 100 números

```
numeros [0] = int (input (“Digite un número”))

numeros [1] = int (input (“Digite un número”))

numeros [2] = int (input (“Digite un número”))

numeros [3] = int (input (“Digite un número”))

…

numeros [99] = int (input (“Digite un número”))
```

Trabajando con arreglos

Adicione las instrucciones que necesite para **solicitar** al usuario cada uno de los 100 números

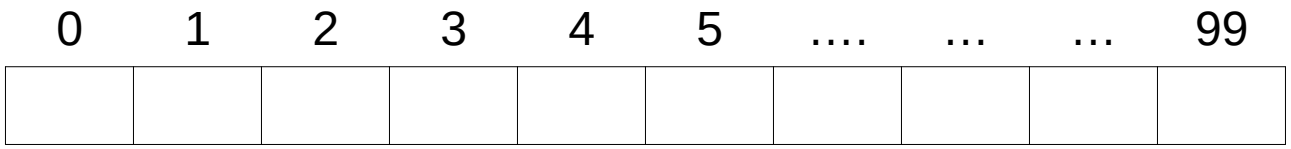
Una mejor Solución…

```
numeros = []
for i in range (0,100):
    números [i] = int (input (“Digite un número”))
```

Trabajando con arreglos

Una mejor Solución…

```
numeros = []
for i in range (0,100):
    numeros [i] = int (input (“Digite un numero”))
```



Trabajando con arreglos

- Después, **muestre todos** los números en un solo mensaje

?

Trabajando con arreglos

- Después, **muestre todos** los números en un solo mensaje

```
mensaje = “ ”
for i in range (0,100):
      mensaje = mensaje + numeros[i]  + “  ”


print (mensaje)
```

Trabajando con arreglos

- Luego, **muestre** sólo los **números** almacenados en **posiciones pares**

?

Trabajando con arreglos

- Muestre sólo los **números** almacenados en **posiciones pares**

```
for i in range (0,100):
      if (i%2 == 0)
            print (numeros[i])
```

Trabajando con arreglos

- muestre los números impares **contenidos** en el arreglo

?

Trabajando con arreglos

- muestre los números impares **contenidos** en el arreglo

```
for i in range (0,100):
     if ((numeros[i] % 2)!= 0)
        print (numeros[i])
```

Trabajando con arreglos

- Generar números aleatorios y almacenarlos en un arreglo.

```
import random
arreglo =[]

for i in range (0,10):
    arreglo.append (random.randint(0,100))


print ("Números generados aleatoriamente")
print (arreglo)
```

Trabajando con arreglos

- Generar números aleatorios enteros y almacenarlos en un arreglo.

```
import random
arreglo =[]

for i in range (0,10):
    arreglo.append (random.randint(0,100))


print ("Números generados aleatoriamente")
print (arreglo)
```

random. randint (inicio, fin) = Genera números aleatorios, en el rango inicio – fin. Donde estos dos valores son enteros

Trabajando con arreglos

- Generar números aleatorios y almacenarlos en un arreglo.

```
import random
arreglo =[]

for i in range (0,10):
    arreglo.append (random.randint(0,100))


print ("Números generados aleatoriamente")
print (arreglo)
```

random. randint (inicio, fin) = Genera números aleatorios, en el rango inicio – fin. Donde estos dos valores son enteros

Ejercicios a realizar con arreglos

- En una competencia de natación se desea implementar una aplicación en Python para almacenar el tiempo por cada competidor y además determinar, con base en todos los tiempos de los competidores, cuál es el ganador. El usuario debe especificar cuántos tiempos de competidores desea ingresar.

- Desarrolle un programa en Python que dado un arreglo de enteros, determine cuales valores son múltiplos de 7 y la posición de éstos en el arreglo. Muestre en pantalla los resultados.

Ejercicios a realizar con arreglos

- Escriba un programa en Python que lea una lista de n enteros, calcule el promedio de los datos ingresados, el mayor valor y el menor valor de ellos.
- Escriba un programa en Python que dada una lista de enteros y un valor ***x*** por parte del usuario, diga cuántas veces ***x*** aparece en la lista. El programa de mostrar cuántas veces aparece el valor ***x*** y en cuáles posiciones.