fidélitas
Universidad

Programación Básica
Clase 09

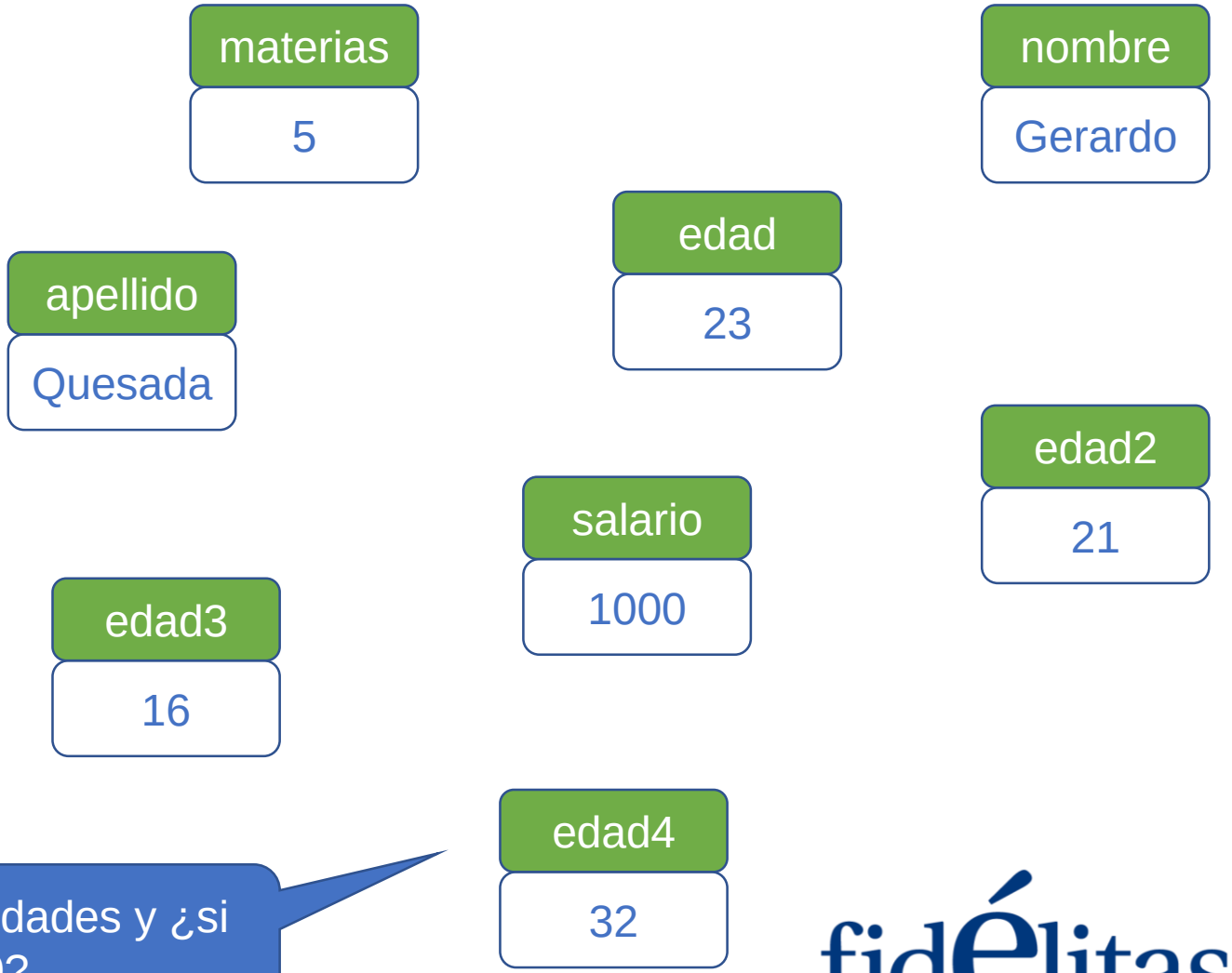
Arreglos unidimensionales

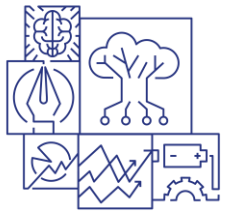
Agenda

Arreglos unidimensionales
- Concepto y utilización
- Creación
- Escritura y lectura
- Ejemplos

Arreglos unidimensionales

Hemos tenido la oportunidad de almacenar información en nuestras variables, una alternativa suficiente para los programas que desarrollamos, pero encontraremos requerimientos en donde estas variables simples no serán tan funcionales.

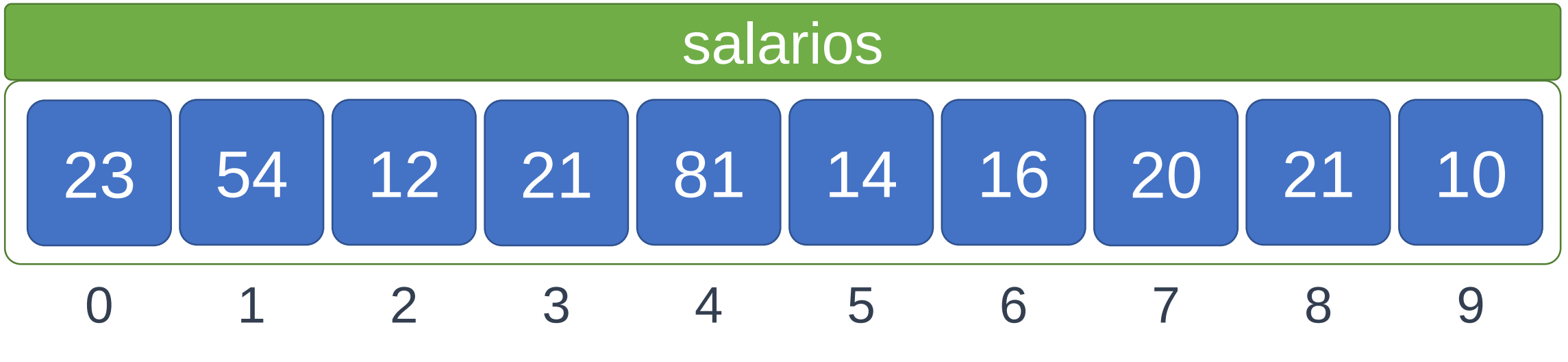


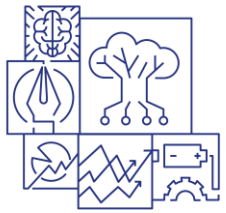


fidélitas Universidad

Arreglos unidimensionales

Las variables nos permiten almacenar múltiples valores pero solo uno a la vez, por lo que sería mejor una estructura que nos permita almacenar múltiples valores de manera simultánea.





fidélitas Universidad

Definición de un arreglo

En Python podemos crear un arreglo con sus valores ya predefinidos y hacer referencia a cada uno de ellos por medio de su posición.

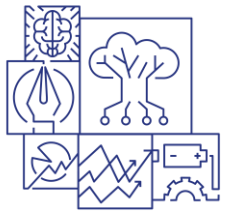
Arreglo con valores predefinidos

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades[1])
print(edades[1] - 22)
print(edades[edades[2]-3])
```

Muestra el valor en la segunda posición

Aplica una operación sobre el valor almacenado

Utiliza un valor de otra posición, con una operación para identificar la posición que se desea mostrar



fidélitas Universidad

Muestra y asigna valores en un arreglo

También podemos asignar un nuevo valor a una de estas posiciones, con lo perderemos el valor almacenado anteriormente.

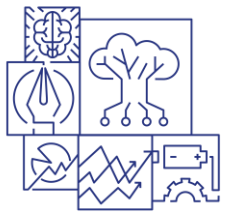
```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades[1])
edades[1] = 100
print(edades[1])
```

Muestra el valor de la posición 1

Modifica el valor de la posición 1

Muestra la misma posición pero con un valor diferente

¿Qué sucede si asignamos un valor en una posición que no existe?



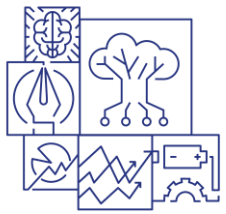
fidélitas Universidad

Agregar un nuevo valor a un arreglo

En pocas ocasiones definiremos los arreglos y no se modificarán, será necesario agregar nuevos valores de la siguiente manera.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades[1])
edades.append(100)
print(edades[10])
```

Agrega un nuevo valor que se adjuntará al final del arreglo



fidélitas Universidad

Eliminar una posición de un arreglo

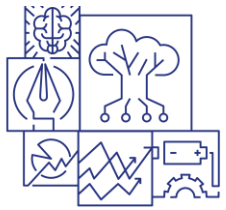
En algunos casos podría ser necesario eliminar un valor de un arreglo, pero no solamente será remplazar el valor almacenado, sino eliminar inclusive la posición.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades)
edades.pop()
print(edades)
```

Si no recibe ningún valor como parámetro, eliminará el último del arreglo

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades)
edades.pop(2)
print(edades)
```

Si se indica un valor se eliminará esa posición



fidélitas Universidad

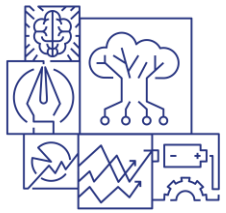
Eliminar una posición de un arreglo

En algunos casos podría ser necesario eliminar un valor de un arreglo, pero no solamente será remplazar el valor almacenado, sino eliminar inclusive la posición.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
print(edades)
edades.remove(81)
print(edades)
```

También existe la posibilidad de eliminar una posición haciendo referencia al valor directamente.

¿Qué sucede si eliminamos el valor 21 que aparece 2 veces?



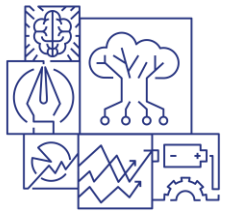
fidélitas Universidad

Ciclos y arreglos

Los ciclos y los arreglos tienen una relación muy cercana, ya que en muchos casos el uso de arreglos requiere un procesamiento masivo de datos y los ciclos nos ayudan a resolver estos temas.

En el caso de que quisiéramos recorrer los valores de un arreglo haciendo referencia a su posición podemos hacerlo de la siguiente manera.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
for x in range(10):
    print(edades[x])
```

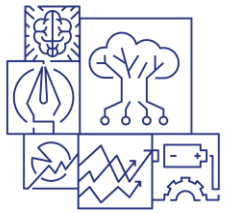


fidélitas Universidad

Ciclos y arreglos

Si no conocemos la cantidad de elementos del arreglo podríamos utilizar la función **len()** para conocerlo.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
for x in range(len(edades)):
    print(edades[x])
```



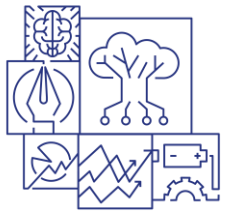
fidélitas Universidad

Ciclos y arreglos

También podríamos tomar en cada iteración del ciclo directamente el elemento que se encuentra en el arreglo, colocándolo como la fuente de los valores de la siguiente manera.

```
edades = [23, 54, 12, 21, 81, 14, 16, 20, 21, 10]
for x in edades:
    print(x)
```

Cualquiera de los dos métodos es válido, pero este nos permite tomar directamente los valores del arreglo en la definición del ciclo

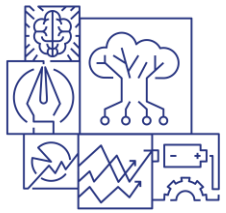


fidélitas Universidad

Ciclos y arreglos

Si quisiéramos agregar datos de manera vacía a un arreglo lo podríamos hacer de la siguiente manera.

```
edades = []
for x in range(10):
    valor = int(input("Ingrese una edad: "))
    edades.append(valor)
for x in edades:
    print(x)
```



fidélitas Universidad

Ejercicio # 01

20 minutos

Desarrolle un programa que le solicite al usuario un número y cree un arreglo con la cantidad de posiciones de acuerdo al número que ingresó el usuario.

Cada una de las posiciones debe almacenar el valor 5.

Ejercicio # 02

Modifique el programa anterior para que en cada posición coloque números consecutivos iniciando en el número 10.

20 minutos

Ejercicio # 03

Modifique el programa anterior para que además de ingresar coloque números consecutivos iniciando en el número 10, y al final los imprima en orden inverso.

20 minutos

Ejercicio # 04

Desarrolle un programa que le solicite al usuario 10 edades, luego debe mostrar al usuario la edad promedio y cuales de las edades ingresadas son mayores al ese promedio.

40 minutos

Con los arreglos se abre una gran cantidad de posibilidades de gestionar en memoria mayor cantidad de información y de una manera más eficiente.

fidélitas
Universidad