ARREGLOS EN PYTHON

Los arreglos son una estructura de datos, muy útiles en programación para la manipulación de información ingresada por el usuario, de un archivo de texto y también la interacción con bases de datos e información de gran volumen, está a información se lleva en Python a listas y allí se puede manipular.

Python maneja una estructura para almacenar varios valores y lo llama colecciones. En Python hay 4 tipos de colecciones:

Listas, tuplas, set y diccionarios (List, Tupla, Set y Dictionary), todos con diferentes cualidades y uso.

Objetivos

- Proporcionar herramienta para manipular conjuntos de datos, utilizando elementos de Python.
- Diferenciar los tipos de arreglos en Python
- Asimilar la forma de almacenamiento y acceso a cada elemento de los conjuntos de datos. Además, conocer algunas de las funciones para su manipulación.

LISTAS

Las listas son estructuras de datos, se utilizan para almacenar varios elementos en una sola variable.

Las listas son uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos. Las listas se crean entre corchetes [ ]:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
print(type(thislist))
```

Los elementos de lista están ordenados, se pueden cambiar y permiten valores duplicados.

- **La lista se puede modificar**

La lista se puede cambiar, lo que significa que podemos cambiar, agregar y eliminar elementos de una lista después de que se haya creado.

- **Permitir duplicados**

Dado que las listas están indexadas, las listas pueden tener elementos con el mismo valor:

```
frutas = ["apple", "banana", "cherry", "apple", "cherry"]
print(frutas)
```

Ejercicio 1
- Crear una lista con el nombre de 6 países
- Crear una lista con el nombre de 5 ciudades

- **Longitud de la lista**

Para determinar cuántos elementos tiene una lista, use la función **len**():

```
frutas = ["apple", "banana", "cherry"]
print(len(frutas))
```

Imprimir la longitud de las dos listas creadas

- **Tipos de datos**
  las listas peuden tener varios tipos de datos, incluso elementos de diferente tipo.

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]

list4 = ["abc", 34, True, 40, "male"]
```

- Crear las 4 listas anteriores e imprimir el tipo:
  ```
  frutas = ["apple", "banana", "cherry"]
  print(type(frutas))
  ```

También es posible utilizar el constructor **list**() al crear una nueva lista.

```
frutas = list( ("apple", "banana", "cherry") ) # note doble
parentesis
print(frutas)
```

ACCESO ALOS ELEMENTOS DE UNA LISTA

```
frutas = ["apple", "banana", "cherry"]

print(frutas [1])
```

Se accede por el índice de la posición iniciando **en cero.**

```
frutas[0]:  hace referenci a “apple”

frutas[1]:  hace referenci a “banana”

frutas[2]:  hace referenci a “cherry”
```

```
Tmabien e pueden utilizar índices negativos iniciando en -1 en la ultima
posposición, -2 la penúltima etc..
```

```
frutas[-1]  hace referenci a “cherry”

frutas[-2]  hace referenci a “banana”
```

Ejercicio 2

- Imprima los dos primeros elementos de las dos listas del ejercicio 1
- Imprima los tres últimos elementos de las dos listas del ejercicio 1
- Imprima el 4 elemento de cada lista

Rango de índices

Puede especificar un intervalo de índices especificando dónde empezar y dónde finalizar el intervalo.

Al especificar un intervalo, el valor devuelto será una nueva lista con los elementos especificados. El siguiente ejemplo imprime tercer, cuarto y quinto elemento:

```
frutas = "apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(frutas [2:5])
```

A continuación, imprime desde el comienzo los primeros 4 elementos:

```
frutas = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(frutas [:4])
```

- **Cambiar los ítems de una lista**

```
frutas = ["apple", "banana", "cherry"]

print(frutas)
frutas [1] = "blackcurrant"
print(frutas)
```

- **Cambiar un rango de ítems**

```
frutas = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
print(frutas)
frutas [1:3] = ["blackcurrant", "watermelon"] # cambio
print(frutas)
```

Cambiar el segundo valor por 2 nuevos ítems

```
frutas = ["apple", "banana", "cherry"]
frutas [1:2] = ["blackcurrant", "watermelon"]
print(frutas)
```

- **Adicionar ítems a la lista**

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
thislist.append("orange")
print(thislist)
```

- **Insertar ítems a la lista**

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
thislist.insert(1, "orange")
print(thislist)
```

- **Eliminar** ítems de la lista, **remove**().
  Si existen repetidos elimina primera ocurrencia.

```
frutas = ["apple", "banana", "cherry"]
print(frutas)
frutas.remove("banana")
print(frutas)
```

- **Ordenar** listas **sort**().

```
frutas = ["orange", "mango", "kiwi", "pineapple", "banana"]
print(frutas)
frutas.sort()
print(frutas)
```

Ordenar descendente

```
frutas = ["orange", "mango", "kiwi", "pineapple", "banana"]
print(frutas)
frutas.sort(reverse = True)
print(frutas)
```

- **Copiar una Lista**

```
frutas = ["apple", "banana", "cherry"]
mylist = frutas.copy()  # nueva lista copiada de la original
print(mylist)
```

Otra forma de copiar

```
frutas = ["apple", "banana", "cherry"]
mylist = list(frutas)
print(mylist)
```

- **Unión de dos listas**

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

Otra forma de unir 2 Listas

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

Ejercicio 3
Las dos listas creadas en el ejercicio 1

- Ordene cada lista
- Unir las dos listas
- Inserte une elemento en cada lista, en la segunda posición
- Adicione un elemento en cada lista
- Elimine el primer elemento de cada lista

ALGUNOS METODOS PARA MANIPULAR LISTAS

append() Agrega un elemento al final de la lista
clear() Elimina todos los elementos de la lista
copy() Devuelve una copia de la lista
count() Devuelve el número de elementos con el valor especificado
extend() Añade los elementos de una lista (o cualquier iterable), al final de la lista actual
index() Devuelve el índice del primer elemento con el valor especificado
insert() Agrega un elemento en la posición especificada
pop() Elimina el elemento en la posición especificada
remove() Elimina el elemento con el v especificado

Fecha Creación	Enero 23 2024
Responsable	Plinio Neira Vargas
Revisado por	Sonia Escobar
Fecha Revisión	Enero 25 2024