

KEYWORD

El intérprete de Python tiene un conjunto de keywords que son palabras reservadas y que no pueden ser usados como variables.

| Keyword | Descripción | ejemplo |
|---------------|---|---|
| if | Se usa para realizar una declaración condicional | <pre>if 5>4: print("CONDICION True")</pre> |
| else | Usado en declaraciones condicionales | <pre>else : print("LA CONDICIÓN FUE False")</pre> |
| True | Valor del tipo bool , resultado de una operación lógica | V = 5 > 4 Como 5 es mayor que 4 por lo tanto V es True |
| False | Valor del tipo bool , resultado de una operación lógica | V = 10 < 5 Como 10 no es menor que 5 entonces V es False |
| in | Para saber si una variable se encuentra dentro de un elemento iterable | <pre>lista=[1 , 3.1 , 4] a=3.1 c=a in lista c es True , porque 3.1 está en la lista</pre> |
| for | Permite crear un bucle for y poder repetir un conjunto de iteraciones, tener en cuenta el elemento iterable | <pre>para val en cada valor de [1,2] for val in [1,2]: print(val) resultado: 1 2</pre> |
| import | Para importar módulos dentro de un archivo de Python | <pre>importando dos modulos import time,sys equivalente a lo siguiente : import time import sys</pre> |
| from | Para importar partes específicas de los módulos | <pre>Importar función sleep del módulo time from time import sleep Importar todo del módulo time from time import *</pre> |
| None | Representa un valor Nulo | valor= None |
| def | Para definir una función | <pre>def mi_function(argumento): instrucciones</pre> |
| return | Se usar para que una función devuelve uno o más valores | <pre>def mi_function(argumento): instrucciones return valor</pre> |

BUILT-IN-FUNCTIONS

El intérprete de Python tiene una serie de funciones integrados y que se encuentran disponibles para uso.

| Built-in-function | Descripción | ejemplo |
|-----------------------|---|---|
| abs() | Se utiliza para calcular el valor absoluto de un número . | <code>a=abs(-20)</code> el valor de la variable a, es de 20 |
| list(iterable) | se utiliza para crear una lista , se le tiene que pasar como argumento un elemento iterable | <code>lista1=list("hola")</code> <code>lista2=['h','o','l','a']</code> ambos son equivalentes |
| open() | Permite crear un archivo de texto o si en caso ya existe ,abrirlo. | <i>#abrir un archivo de texto</i> <i>umaker.txt</i> <i>#modo escritura</i> <code>archivo=open("umaker.txt","w")</code> |
| int() | Función para convertir un valor numérico a entero | <code>int(20.5)</code> <code>int("30")</code> |
| float() | Función para convertir un valor numérico a flotante | <code>float("30.25")</code> |
| str() | Función para convertir a dato tipo string | <code>str(10)</code> convierte el numero 10 a un strng "10" |
| len() | Devuelve la cantidad de datos de un elemento iterable | <code>len([1, 2, 3, 10])</code> el resultado es 4 ya que hay 4 elementos |

INDEXACIÓN EN ELEMENTOS ITERABLES:

La indexación permite acceder a datos de un elemento iterable mediante el uso de su índice, donde el índice indica la ubicación dentro de un elemento iterable donde se encuentra el dato de interés.

Tomaremos como ejemplo la indexación para el caso de las listas.

Tener en cuenta que una lista es un elemento iterable y que colecciona datos de cualquier tipo , además su contenido se puede modificar.

Para acceder a los elementos de una lista podemos utilizar indexación positiva o indexación negativa.

INDEXACIÓN POSITIVA:

| LISTAx | | | |
|--------|----|----|-------|
| 25 | 26 | 30 | -15.5 |
| 0 | 1 | 2 | 3 |

LISTAx [0] accede al 25 debido a que este valor se encuentra en la posición "0"

LISTAx [1] accede al 26 debido a que este valor se encuentra en la posición "1"

A continuación, se muestra un ejemplo de cómo acceder al dato de una lista mediante su índice (posición).

CODIGO DE EJEMPLO:

```
1  #creando una lista con 6 elementos
2  mi_lista=[20 , 30 ,15 ,20 ,-25 , 10.5]
3  #se presente acceder al valor que ocupa la posición 3
4  #del iterable mi_lista y mostrarlo mediante la funcion print
5  print("EL VALOR ES: ", mi_lista[3])
```

RESULTADO:

```
EL VALOR ES:  20
```

```
Process finished with exit code 0
```

SLICING EN LISTAS:

Para realizar una indexación a varios elementos de una lista podemos hacer uso de la siguiente sintaxis.

Iterable [start:stop:step]

El valor **start** : indica desde una posición de inicio

El valor de **stop**: indica el límite de la búsqueda (este valor nunca se tomara)

El valor de **step**: indica el incremento o decremento.

LISTA

| | | | | | |
|------|----|----|----|----|----|
| 10.5 | 20 | 30 | 25 | 26 | -2 |
|------|----|----|----|----|----|

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|

Se pretende acceder a los elementos de la posición 1 hasta la posición 4

LISTA

| | | | | | |
|------|----|----|----|----|----|
| 10.5 | 20 | 30 | 25 | 26 | -2 |
|------|----|----|----|----|----|

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|

start **stop**

Para acceder a dichos elementos realizamos lo siguiente:

LISTA[1 : 5 : 1] es equivalente a la lista [20,30,25,26]

El valor de start es 1 , el valor de stop es 5 , el step es 1.

Esto quiere decir que se creara una nueva lista con los elementos que van desde la posición 1 (**start**) e incremento en 1(**step**) hasta llegar a la posición menor que 5 (**stop**).

CODIGO:

```
1  #Lista llamada umaker de 6 elementos
2  umaker=[10.5,20,30,25, 26,-2]
3  #crear una sub_lista desde la posición 1 hasta la posición 4
4  sub_umaker=umaker[1:5:1]
5  print("LA SUB_LISTA ES: ", sub_umaker)
```

RESULTADO:

LA SUB_LISTA ES: [20, 30, 25, 26]

Process finished with exit code 0