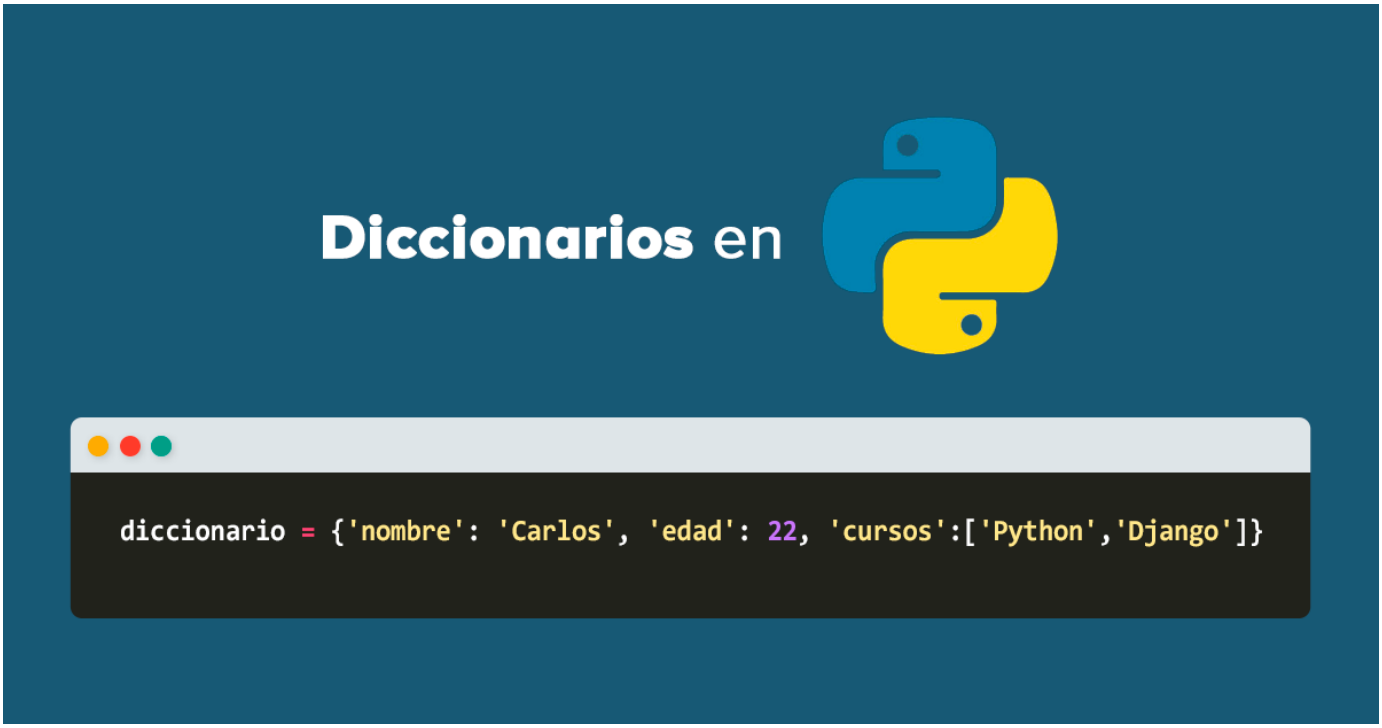
Diccionarios en Python

Por Carlos Eduardo Plasencia Prado



En este tutorial aprenderemos a utilizar diccionarios de datos en Python y algunos de sus métodos más importantes.

Python es un lenguaje de programación interpretado de alto nivel y orientado a objetos, con el cual podemos crear todo tipo de aplicaciones. Entre sus diversos tipos de estructuras de datos, se encuentra "Diccionarios de Datos". En este tutorial aprenderemos a utilizar esta estructura revisando sus méetodos más utilizados.

¿Qué es un Diccionario de datos?

Un Diccionario es una estructura de datos y un tipo de dato en Python con características especiales que nos permite almacenar cualquier tipo de valor como enteros, cadenas, listas e incluso otras funciones. Estos diccionarios nos permiten además identificar cada elemento por una clave (Key).

1. Para definir un diccionario, se encierra el listado de valores entre llaves. Las parejas de clave y valor se separan con comas, y la clave y el valor se separan con dos puntos.

```
diccionario = {'nombre' : 'Carlos', 'edad' : 22, 'cursos': ['Python','D
```

2. Podemos acceder al elemento de un Diccionario mediante la clave de este elemento, como veremos a continuación:

```
print diccionario['nombre'] #Carlos
print diccionario['edad']#22
print diccionario['cursos'] #['Python','Django','JavaScript']
```

3. También es posible insertar una lista dentro de un diccionario. Para acceder a cada uno de los cursos usamos los índices:

```
print diccionario['cursos'][0]#Python
print diccionario['cursos'][1]#Django
print diccionario['cursos'][2]#JavaScript
```

4. Para recorrer todo el Diccionario, podemos hacer uso de la estructura for:

```
for key in diccionario:
  print key, ":", diccionario[key]
```

Métodos de los Diccionarios

- dict ()

  Recibe como parámetro una representación de un diccionario y si es factible, devuelve un diccionario de datos.

```
dic =  dict(nombre='nestor', apellido='Plasencia', edad=22)

dic → {‘nombre’ : 'nestor', ‘apellido’ : 'Plasencia', ‘edad’ : 22}
```

- zip()

  Recibe como parámetro dos elementos iterables, ya sea una cadena, una lista o una tupla. Ambos parámetros deben tener el mismo número de elementos. Se devolverá un diccionario relacionando el elemento i-esimo de cada uno de los iterables.

```
dic = dict(zip('abcd',[1,2,3,4]))

dic →   {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
```

- items()

  Devuelve una lista de tuplas, cada tupla se compone de dos elementos: el primero será la clave y el segundo, su valor.

```
dic =   {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
items = dic.items()

items → [(‘a’,1),(‘b’,2),(‘c’,3),(‘d’,4)]
```

- keys()

  Retorna una lista de elementos, los cuales serán las claves de nuestro diccionario.

```
dic =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
keys= dic.keys()

keys→ [‘a’,’b’,’c’,’d’]
```

- values()

  Retorna una lista de elementos, que serán los valores de nuestro diccionario.

```
dic =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
values= dic.values()

values→ [1,2,3,4]
```

- clear()

  Elimina todos los ítems del diccionario dejándolo vacío.

```
dic 1 =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
dic1.clean()

dic1 → { }
```

- copy()

  Retorna una copia del diccionario original.

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
dic1 = dic.copy()

dic1 → {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
```

- fromkeys()

  Recibe como parámetros un iterable y un valor, devolviendo un diccionario que contiene como claves los elementos del iterable con el mismo valor ingresado. Si el valor no es ingresado, devolverá none para todas las claves.

```
dic = dict.fromkeys(['a','b','c','d'],1)

dic →  {‘a’ : 1, ’b’ : 1, ‘c’ : 1 , ‘d’ : 1}
```

- get()

  Recibe como parámetro una clave, devuelve el valor de la clave. Si no lo encuentra, devuelve un objeto none.

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
valor = dic.get(‘b’)

valor → 2
```

- pop()

  Recibe como parámetro una clave, elimina esta y devuelve su valor. Si no lo encuentra, devuelve error.

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
valor = dic.pop(‘b’)

valor → 2
dic → {‘a’ : 1, ‘c’ : 3 , ‘d’ : 4}
```

- setdefault()

  Funciona de dos formas. En la primera como get

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
valor = dic.setdefault(‘a’)

valor → 1
```

  Y en la segunda forma, nos sirve para agregar un nuevo elemento a nuestro diccionario.

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
valor = dic.setdefault(‘e’,5)
```

```
dic → {'a' : 1, 'b' : 2, 'c' : 3 , 'd' : 4 , 'e' : 5}
```

- update()

  Recibe como parámetro otro diccionario. Si se tienen claves iguales, actualiza el valor de la clave repetida; si no hay claves iguales, este par clave-valor es agregado al diccionario.

```
dic 1 = {'a' : 1, 'b' : 2, 'c' : 3 , 'd' : 4}
dic 2 = {'c' : 6, 'b' : 5, 'e' : 9 , 'f' : 10}
dic1.update(dic 2)

dic 1 → {'a' : 1, 'b' : 5, 'c' : 6 , 'd' : 4 , 'e' : 9 , 'f' : 10}
```

Estos son algunos de los métodos más útiles y más utilizados en los Diccionarios. Python es un gran lenguaje de programación que nos permite programar de una manera realmente sencilla. Si deseas conocer mucho más y aprender a profundidad esta tecnología, ingresa al Curso de Python que tenemos en Devcode. ¡Te esperamos!

Carlos Eduardo Plasencia Prado

Backend Developer | Python / Django junior - Javascript / Node.js

@plasenciacar