Diccionarios en Python

De la misma forma que con listas, es posible definir un diccionario directamente con los miembros que va a contener, o bien inicializar el diccionario vacío y luego agregar los valores de a uno o de a muchos.

Para definirlo junto con los miembros que va a contener, 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 :.

```
punto = {'x': 2, 'y': 1, 'z': 4}
```

Para declararlo vacío y luego ingresar los valores, se lo declara como un par de llaves sin nada en medio, y luego se asignan valores directamente a los índices.

```
materias = {}
materias["lunes"] = [6103, 7540]
materias["martes"] = [6201]
materias["miércoles"] = [6103, 7540]
materias["jueves"] = []
materias["viernes"] = [6201]
```

Para acceder al valor asociado a una determinada clave, se lo hace de la misma forma que con las listas, pero utilizando la clave elegida en lugar del índice.

```
print materias["lunes"]
```

Sin embargo, esto falla si se provee una clave que no está en el diccionario. Es posible, por otro lado, utilizar la función get, que devuelve el valor None si la clave no está en el diccionario, o un valor por omisión que se establece opcionalmente.

```
>>> print materias["domingo"]
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
KeyError: 'domingo'
>>> print materias.get("domingo")
None
>>> print materias.get("domingo", []) []
```

Existen diversas formas de recorrer un diccionario. Es posible recorrer sus claves y usar esas claves para acceder a los valores.

```
for dia in materias:
   print dia, ":", materias[dia]
```

Es posible, también, obtener los valores como tuplas donde el primer elemento es la clave y el segundo el valor.

```
for dia, codigos in materias.items():
   print dia, ":", codigos
```

Para verificar si una clave se encuentra en el diccionario, es posible utilizar la función has_key o la palabra reservada in.

```
d = {'x': 12, 'y': 7}
if d.has_key('x'):
   print d['x']   # Imprime 12
if d.has_key('z'):
   print d['z']   # No se ejecuta
if 'y' in d:
   print d['y']   # Imprime 7
```

Más allá de la creación y el acceso, hay muchas otras operaciones que se pueden realizar sobre los diccionarios, para poder manipular la información según sean nuestras necesidades, algunos de estos métodos pueden verse en la referencia al final de la unidad.

NOTA
El algoritmo que usa Python internamente para buscar un elemento en un diccionario es muy distinto que el que utiliza para buscar en listas.

Para buscar en las listas, se utiliza un algoritmo de comparación que tarda cada vez más a medida que la lista se hace más larga. En cambio, para buscar en diccionarios se utiliza un algoritmo llamado hash, que se basa en realizar un cálculo numérico sobre la clave del elemento, y tiene una propiedad muy interesante: sin importar cuántos elementos tenga el diccionario, el tiempo de búsqueda es siempre aproximadamente igual.

Este algoritmo de hash es también la razón por la cual las claves de los diccionarios deben ser inmutables, ya que la operación hecha sobre las claves debe dar siempre el mismo resultado, y si se utilizara una variable mutable esto no sería posible.

No es posible obtener slices de un diccionario usando [:], ya que al no tener un orden determinado para los elementos, no sería posible tomarlos en orden.