Diccionarios y Estructuras de Datos

Al igual que una lista, un diccionario es una colección de muchos valores. Pero a diferencia de los índices enteros que utilizan las listas, los índices de los diccionarios pueden ser otros tipos de datos, no solo enteros. A los índices de los diccionarios se los llama keys o claves.
En Python, los diccionarios se escriben con llaves {}

```
>>> miGato = {'tamaño': 'gordo', 'color': 'gris', 'carácter': 'ruidoso'}
```

Este código le asigna un diccionario a la variable miGato. Las claves de este diccionario son 'tamaño', 'color', y 'carácter'. Los valores de estas claves son 'gordo', 'gris', y 'ruidoso', respectivamente. Podemos acceder a estos valores utilizando sus claves:

```
>>> miGato['tamaño']
>>> 'Mi gato tiene pelaje de color ' + miGato['color']
```

Los Diccionarios todavía pueden utilizar números enteros como clave, pero no necesitan empezar en 0 y pueden ser cualquier número.

```
>>> spam = {12345: 'Combinación del candado', 42: 'La respuesta'}
```

Diferencias entre Diccionarios y Listas

A diferencia de las listas, los elementos de un diccionario no tienen un orden. El primer elemento de la lista spam sería spam[0]. Pero no hay un "primer" elemento en un diccionario. Y si necesitáramos saber si 2 diccionarios son iguales, el orden de sus elementos o par clave-valor no importaría.

```
>>> spam = ['gatos', 'perros', 'alces']
>>> cerdo = ['perros', 'alces', 'gatos']
>>> spam == cerdo
>>> huevos = {'nombre': 'Sofía', 'especie': 'gato', 'edad': '8'}
>>> jamón = {'especie': 'gato', 'edad': '8', 'nombre': 'Sofía'}
>>> huevos == jamón
```

Como los diccionarios no tienen orden no podremos utilizar rangos o Slices con estos.
Si tratamos de acceder a una clave que no existe en un diccionario recibiremos un error del tipo KeyError, parecido al IndexError que sucede con listas cuando nos salimos de rango.

```
>>> spam = {'nombre': 'Sofía', 'edad': 7}
>>> spam['color']
```

Ejemplo trabajando con Diccionarios

Los métodos keys(), values() e ítems()

Existen 3 métodos de diccionario que devuelven valores parecidos a una lista de: claves, valores o ambos. Estos valores parecen listas, pero no pueden ser modificados porque pertenecen a otros tipos de datos llamados dict_keys, dict_values y dict_items y que principalmente se utilizan en ciclos de repetición.

```
>>> spam = {'color': 'rojo', 'edad': 42}
>>> for v in spam.values():
        print(v)
>>> for k in spam.keys():
        print(k)
>>> for i in spam.items():
        print(i)
```

Si queremos listas de estos valores se puede utilizar la función list()

```
>>> spam = {'color': 'rojo', 'edad': 42}
>>> spam.keys()
>>> list(spam.keys())
```

La función list(spam.keys) toma los valores de la dict_key y los devuelve en forma de lista. Podemos utilizar el truco de asignación múltiple en un ciclo for para asignar la clave y el valor a distintas variables.

```
>>> spam = {'color': 'rojo', 'edad': 42}
>>> for k, v in spam.items():
        print('Clave: ' + k + ' Valor: ' + str(v))
```

Operadores in y not in con Diccionarios

Al igual que con listas, podes utilizar los operadores in y not in para verificar si una clave o valor existe dentro de un diccionario en particular.

```
>>> spam = {'nombre': 'Sofía', 'edad': 7}
>>> 'nombre' in spam.keys()
>>> 'Sofía' in spam.values()
>>> 'color' in spam.keys()
>>> 'color' not in spam.keys()
>>> 'color' in spam
```

El método get()

Para no tener que prevenir los errores de no encontrar una clave dentro de un diccionario existe el método get(). Este método toma 2 argumentos: la clave del valor que estamos buscando, y el valor que devolverá en caso de no encontrarlo.

```
>>> elementosPicnic = {'manzanas': 5, 'vasos': 2}
>>> 'Yo voy a traer ' + str(elementosPicnic.get('vasos', 0)) + ' vasos.'
>>> 'Yo voy a traer ' + str(elementosPicnic.get('huevos', 0)) + ' huevos.'
>>> elementosPicnic = {'manzanas': 5, 'vasos': 2}
>>> 'Yo voy a traer' + str(elementosPicnic['huevos']) + ' huevos.'
```

El método setdefault()

El método setdefault() acepta 2 argumentos, el primero para buscar una clave en el diccionario y el segundo para asignarle un valor si esa clave no existe. De esta manera, podemos en una sola línea de código, crear una clave solo si no existe y asignarle un valor.

```
>>> spam = {'nombre': 'Pooka', 'edad': 5}
>>> spam.setdefault('color', 'negro')
>>> spam
>>> spam.setdefault('color', 'blanco')
>>> spam
```

Pretty Printing

Existe un módulo o librería llamada Pretty Printing que nos puede ayudar a mostrar nuestros diccionarios en pantalla. Pretty Printing nos da acceso a las funciones pprint() y pformat() para lograr esto.

```
import pprint
mensaje = 'Era un día frio de abril, cuando dieron las 12'
contar = {}
for caracter in mensaje:
   contar.setdefault(caracter, 0)
   contar[caracter] = contar[caracter] + 1
pprint.pprint(count)
```

SI en lugar de imprirlo directamente por pantalla queremos utilizar la string para algo más, se puede utilizar pformat()

```
print(pprint.pformat(contar))
```