# Diccionarios

March 30, 2022

**Ejecuta el siguiente bloque de código siempre antes de ejecutar el resto del notebook.**

[ ]:
```
from IPython.core.magic import  Magics, magics_class, cell_magic, line_magic

@magics_class
class Helper(Magics):

  def __init__(self, shell=None,  **kwargs):
    super().__init__(shell=shell, **kwargs)

  @cell_magic
  def debug_cell_with_pytutor(self, line, cell):
    import urllib.parse
    url_src = urllib.parse.quote(cell)
    str_begin = '<iframe width="1000" height="500" frameborder="0" src="https://
↪pythontutor.com/iframe-embed.html#code='
    str_end   = '&cumulative=false&py=3&curInstr=0"></iframe>'
    import IPython
    from google.colab import output
    display(IPython.display.HTML(str_begin+url_src+str_end))

get_ipython().register_magics(Helper)
```

## 1 Estructuras de datos 3

### 1.1 Diccionarios

El diccionario en Python es una colección ordenada de valores de datos utilizada para almacenarlos como un mapa, a diferencia de otros tipos de datos que mantienen un solo valor como elemento, mantiene un par `clave:valor`. Esta pareja `clave:valor` se proporciona en el diccionario para hacerlo más optimizado.

Crear un diccionario es tan sencillo como colocar la clave y el valor dentro de llaves (`{}`) y separarlos con una coma. Un elemento del diccionario tendrá una clave y un valor correspondiente expresados como un par `clave: valor`.

Mientras que los valores pueden ser de cualquier tipo de datos y pueden repetirse, las claves deben ser de tipo inmutable (**cadena, número o tupla**) y deben ser únicas.

**Nota**: Las claves del diccionario distinguen entre mayúsculas y minúsculas, un mismo nombre capitalizado o no (ej: `Carlos` o `carlos`) será tratado de forma distinta.

```
diccionario = {} #dict vacío

diccionario = {
    "clave":valor, #clave string
    1 : valor #clave numerica,
    (1,2) : valor #clave en tupla
}
```

### 1.1.1 Elementos del diccionario

Los elementos del diccionario están ordenados, son modificables y no admiten duplicados. Como vimos, los elementos del diccionario se presentan en pares `clave:valor`, y se puede hacer referencia a ellos utilizando el nombre de la clave, usando la notación de corchetes`[ ]`.

[ ]:
```
%%debug_cell_with_pytutor
#creemos un diccionario que simule una libreta de contactos
libreta = {
    "carlos" : 5486710,
    "hugo"   : 2483675,
    "paco"   : 4215876,
    "luis"   : 3658741
}
#accedamos al número de paco
print(libreta["paco"])
```

Al igual que las otras estructuras de datos que hemos visto, los diccionarios también pueden almacenar las demás estructuras en su interior y podemos acceder a sus datos internos con la notación corchetes `[ ]` (excepto los sets):

[ ]:
```
%%debug_cell_with_pytutor
dic_anidado = {
    "tupla" : (1,2,3,4,5),
    "lista" : [55.2,41.7,658.3],
    "dict"  : {
        1 : "hola",
        2 : "mundo"
    },
    "set": set([3,3.14,4])
}
print("tupla",dic_anidado["tupla"], dic_anidado["tupla"][1])
print("lista",dic_anidado["lista"], dic_anidado["lista"][0])
print("dict",dic_anidado["dict"], dic_anidado["dict"][1])
```

Además de la notación de corchetes, es posible acceder a los datos del diccionario con la función `get`. Al usar corchetes, si la clave no existe, el programa retornará error; si usamos get, el programa retornará `None`

[ ]:
```
libreta = {
    "carlos" : 5486710,
    "hugo"   : 2483675,
    "paco"   : 4215876,
    "luis"   : 3658741
}
```

[ ]:
```
print(libreta.get("hugo"))
```

```
2483675
```

[ ]:
```
print(libreta.get("rodolfo"))
```

```
None
```

[ ]:
```
print(libreta["rodolfo"])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-10-04e6b231bfa5> in <module>()
----> 1 print(libreta["rodolfo"])

KeyError: 'rodolfo'
```

### 1.1.2 Adición de elementos a un Diccionario

La adición de elementos se puede hacer de múltiples maneras. Se puede añadir un valor a la vez a un diccionario definiendo el valor junto con la clave, por ejemplo, `diccionario[key] = 'Value'`. También se pueden añadir valores de clave anidados a un Diccionario existente.

> **Nota**: Al añadir un valor, si la `clave:valor` ya existe, el valor se actualiza, de lo contrario se añade una nueva clave con el valor al diccionario. Recordemos que en Python, los diccionarios no admiten claves duplicadas.

[ ]:
```
%%debug_cell_with_pytutor
#creemos un diccionario que simule una libreta de contactos
libreta = {
    "carlos" : 5486710,
    "hugo"   : 2483675,
    "paco"   : 4215876,
    "luis"   : 3658741
}

libreta['rodolfo'] = 8452165
libreta["carlos"]  = 6542138
```

### 1.1.3 Eliminación de elementos del diccionario

**Usando la palabra clave `del`** La eliminación de datos se puede hacer utilizando la sentencia `del`. Con ella, se pueden borrar valores específicos de un diccionario, así como todo el diccionario. Los elementos de un diccionario anidado también pueden ser borrados usando la sentencia `del` y proporcionando la clave completa del dato a eliminar.

[ ]:
```
# diccionario de cuadrados de números
cuadrados = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
print("antes",cuadrados)
del cuadrados[3]
print("después",cuadrados)
```

```
antes {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
despues {1: 1, 2: 4, 4: 16, 5: 25}
```

[ ]:
```
# diccionario con cuadrados y cubos de números
numeros = {
    "cuadrados" : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25},
    "cubos"     : {1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
}
print('antes', numeros)
del numeros["cubos"][1]
print('después', numeros)
```

```
antes {'cuadrados': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'cubos': {1: 1, 2: 8, 3:
27, 4: 64, 5: 125}}
despues {'cuadrados': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'cubos': {2: 8, 3: 27,
4: 64, 5: 125}}
```

**Usando la función `pop`** La función `pop()` se utiliza para devolver y eliminar el valor de la clave especificada.

[ ]:
```
print('antes',libreta)
eliminado = libreta.pop('hugo')
print('después',libreta)
print('elemento eliminado:',eliminado)
```

```
antes {'carlos': 5486710, 'hugo': 2483675, 'paco': 4215876, 'luis': 3658741}
despues {'carlos': 5486710, 'paco': 4215876, 'luis': 3658741}
elemento eliminado: 2483675
```

### 1.1.4 Iterar a través de un diccionario

Podemos iterar a través de cada clave de un diccionario utilizando un ciclo `for`. En este caso, el ciclo le entrega a nuestra variable contadora (en el ejemplo es `i`) cada una de las claves del diccionario:

```
for i in diccionario:
    print(diccionario[i]) #acceso al valor a través de la i-ésima clave
```

[ ]:
```
for tel in libreta:
    print(f"{tel}, tel: {libreta[tel]}")
```

```
carlos, tel: 5486710
paco, tel: 4215876
luis, tel: 3658741
```

[ ]:
```
print(libreta)
```

```
{'carlos': 5486710, 'paco': 4215876, 'luis': 3658741}
```