Diccionarios en Python – Parte 2

Materia: Redes Neuronales.

Profesora: Camacho Vázquez Vanessa Alejandra.

items

In [25]:
```
print(my_dic.items())
```

```
dict_items([('Colores', 'Negro'), ('Animales', 'Gato'), ('Calzado', ['Botas', 'Botines', 'Deportivos', 'Sandalias'])])
```

In [ ]:
```
print(list(my_dic.items()))
```

```
[('Colores', 'Negro'), ('Animales', 'Gato'), ('Calzado', ['Botas', 'Botines', 'Deportivos', 'Sanadalias'])]
```

In [ ]:
```
list(my_dic.items())[0][1]
```

Out[ ]:
```
'Negro'
```

También existen otros métodos útiles. ¡Averigua para que sirve cada uno!

- **d.keys()**
- **d.values()**
- **d.pop(<key>[, <default>])**
- **d.popitem()**
- **d.update(<obj>)**

Ejercicio

A partir de las siguientes líneas de código, entienda lo que hace cada uno de los métodos y cree sus propio ejemplo de mayor complejidad.

In [26]:
```
print(my_dic)
```

```
{'Colores': 'Negro', 'Animales': 'Gato', 'Calzado': ['Botas', 'Botines', 'Deportivos', 'Sandalias']}
```

In [27]:
```
print(my_dic.items())
print(my_dic.keys())
print(my_dic.values())
print(my_dic.pop('Colores'))
print(my_dic)
print(my_dic.popitem())
print(my_dic)
print(my_dic.update(persona))
```

```
dict_items([('Colores', 'Negro'), ('Animales', 'Gato'), ('Calzado', ['Botas', 'Botines', 'Deportivos', 'Sandalias'])])
dict_keys(['Colores', 'Animales', 'Calzado'])
dict_values(['Negro', 'Gato', ['Botas', 'Botines', 'Deportivos', 'Sandalias']])
Negro
{'Animales': 'Gato', 'Calzado': ['Botas', 'Botines', 'Deportivos', 'Sandalias']}
('Calzado', ['Botas', 'Botines', 'Deportivos', 'Sandalias'])
{'Animales': 'Gato'}
None
```

char2int

Como último ejemplo construimos un diccionario para el alfabeto, de tal manera que dado un caracter retorne un código asociado.

In [ ]:
```
alfabeto = 'abcdefghijklmnñopqrstuvwxyz'
alfaL = []
for j in alfabeto:
    alfaL.append(j)


char2int = {}
num = list(range(len(alfabeto)))

for car,val in zip(alfaL, num):
    char2int[car] = num[val]

print(char2int)
print('\n')
print('char2int[\'c\']=',char2int['c'])
```

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'ñ': 14, 'o': 15, 'p': 16, 'q': 17, 'r': 18, 's': 19, 't': 20, 'u': 21, 'v': 22, 'w': 23, 'x': 24, 'y': 25, 'z': 26}


char2int['c']= 2
```

Métodos de los diccionarios

Método	Descripción
clear()	Elimina todos los elementos del diccionario.
copy()	Devuelve una copia poco profunda del diccionario.
get(clave[,valor])	Devuelve el valor de la clave. Si no existe, devuelve el valor valor si se indica y si no, None.
items()	Devuelve una vista de los pares clave: valor del diccionario.
keys()	Devuelve una vista de las claves del diccionario.
pop(clave[,valor])	Devuelve el valor del elemento cuya clave es clave y elimina el elemento del diccionario. Si la clave no se encuentra, devuelve valor si se proporciona. Si la clave no se encuentra y no se indica valor, lanza la excepción KeyError.
popitem()	Devuelve un par (clave, valor) aleatorio del diccionario. Si el diccionario está vacío, lanza la excepción KeyError.
setdefault(clave[,valor])	Si la clave está en el diccionario, devuelve su valor. Si no lo está, inserta la clave con el valor valor y lo devuelve (si no se especifica valor, por defecto es None).
update(iterable)	Actualiza el diccionario con los pares clave: valor del iterable.
values()	Devuelve una vista de los valores del diccionario.

Tips para manipulación de diccionarios

Mezclar dos diccionarios

[32]:
```
dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}

dict3 = {**dict1, **dict2}

print(dict3)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

Hacer la unión de dos diccionarios

Disponible desde Python 3.9

In [34]:
```
dict1 = {'a':1, 'b':2}
dict2 = {'c':3, 'b':4}

dict3 = dict1 | dict2
print(dict3)
```

```
{'a': 1, 'b': 4, 'c': 3}
```

Chequear si una clave existe en un diccionario

In [35]:
```
dict1 = {'a': 1, 'b': 2}

print('a' in dict1)
print('c' in dict1)
```

```
True
False
```

Remover un item de un diccionario

Revise las siguientes tres salidas y desarrolle su propio ejemplo.

In [36]:
```
dict1 = {'a': 1, 'b': 2}

print(dict1.pop('a'))
print(dict1.pop('c', 'Missing Key'))
print(dict1.pop('d'))
```

```
Missing Key
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In [36], line 5
      3 print(dict1.pop('a'))
      4 print(dict1.pop('c', 'Missing Key'))
----> 5 print(dict1.pop('d'))

KeyError: 'd'
```

Diccionarios por comprensión

n [ ]:
```
dict1 = {i:i**2 for i in range(1,11)}
print(dict1)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Eliminar items vacios (None) en un diccionario

n [ ]:
```
dict1 = {'a': 'manzana', 'b': 'banano', 'c':None}

dict1 = {key:value for (key, value) in dict1.items() if value is not None}
print(dict1)
```

```
{'a': 'manzana', 'b': 'banano'}
```

Uso de `get` para acceder a un ítem en un diccionario

Observe que con el uso de *get* no se genera un error cuando se intenta accesar una clave que no existe. En ese caso se recibe el valor `None`. Por el contrario se accesa directamente por la clave, se obtiene un error si la clave no existe en el diccionario.

In [ ]:
```
dict1 = {'a':1, 'b':2}

print(dict1.get('c'))
print(dict1['c'])

# Returns: None
# Returns: KeyError
```

```
None
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-68-035bdafe920c> in <module>
      2
      3 print(dict1.get('c'))
----> 4 print(dict1['c'])
      5
      6 # Returns: None

KeyError: 'c'
```

Filtrar un diccionario

Se pueden obtener los ítems en un diccionario que cumplen con un criterio. En ejemplo se filtran las personas que miden 170 cm o más.

In [ ]:
```
alturas = {'John': 175, 'Luis': 150, 'Carlos': 155, 'María': 170}

alto = {key:value for (key, value) in alturas.items() if value >= 170}

print(tall)
```

```
{'John': 175, 'María': 170}
```

Iterando en un diccionario un diccionario

Vamos iterar a lo largo de un diccionario, primero por las claves y luego por los valores.

In [ ]:
```
# creamos el diccionario

salarios = {'León': 175, 'Luis': [150, 100], 'Carlos': 155, 'Angel': None}
print(salarios)
```

```
{'León': 175, 'Luis': [150, 100], 'Carlos': 155, 'Angel': None}
```

In [ ]:
```
# recorremos por claves

for salario in salarios:
    print(salario)
```

```
León
Luis
Carlos
Angel
```

In [ ]:
```
# de nuevo por claves
for persona in salarios.keys():
    print(persona)
```

```
León
Luis
Carlos
Angel
```

In [ ]:
```
# recorremos por valores
for salario in salarios.values():
    print(salario)
```

```
175
[150, 100]
```