7_Más sobre diccionarios

June 18, 2025

Creado por:

Isabel Maniega

1 Diccionarios

Los diccionarios se usan en: * Machine learning * Base de datos: NoSQL como es MongoDB * en JSON, etc

1.1 Parte 1

[1]:
```
# clave-valor
# "key": "value"
# { "key": "value"}
# { "key": "value", "key2": "value2", "key3": "value3", ...}
```

[2]:
```
diccionario = {"A": 10, "B": 2, "C": 35}
diccionario
```

[2]:
```
{'A': 10, 'B': 2, 'C': 35}
```

[3]:
```
diccionario["A"]
```

[3]:
```
10
```

[4]:
```
diccionario["B"]
```

[4]:
```
2
```

[5]:
```
diccionario["C"]
```

[5]:
```
35
```

[6]:
```
len(diccionario)
```

[6]:
```
3
```

1.2 Parte 2

[7]:
```
diccionario1 = {"clave1": 1, "clave2": 2, "clave3": 3}
diccionario1
```

[7]:
```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

[8]:
```
diccionario1.keys()
```

[8]:
```
dict_keys(['clave1', 'clave2', 'clave3'])
```

[9]:
```
diccionario1.values()
```

[9]:
```
dict_values([1, 2, 3])
```

[10]:
```
type(diccionario1.values())
```

[10]:
```
dict_values
```

[11]:
```
# Obtener los valores de Keys/claves
for key in diccionario1.keys():
    print(key)
```

```
clave1
clave2
clave3
```

[12]:
```
# Obtener los valores de Keys/claves en forma de listado
listado_keys = [key for key in diccionario1.keys()]
listado_keys
```

[12]:
```
['clave1', 'clave2', 'clave3']
```

[13]:
```
# Obtener los valores de values/valores
for value in diccionario1.values():
    print(value)
```

```
1
2
3
```

[14]:
```
# Obtener listado
listado_values = [value for value in diccionario1.values()]
listado_values
```

[14]:
```
[1, 2, 3]
```

[15]:
```
diccionario1.items()
```

[15]:
```
dict_items([('clave1', 1), ('clave2', 2), ('clave3', 3)])
```

[16]:
```
# obtención de clave/valor mediante for:
for key, value in diccionario1.items():
    print("clave: ", key)
    print("valor: ", value)
```

```
clave:  clave1
valor:  1
clave:  clave2
valor:  2
clave:  clave3
valor:  3
```

[17]:
```
diccionario1["clave1"]
```

[17]: 1

[18]:
```
# Modificación de valores en el diccionario
diccionario1["clave1"] = 5
diccionario1
```

[18]: {'clave1': 5, 'clave2': 2, 'clave3': 3}

[19]:
```
len(diccionario1)
```

[19]: 3

[20]:
```
# Eliminar un campo del diccionario
del diccionario1["clave3"]
diccionario1
```

[20]: {'clave1': 5, 'clave2': 2}

[21]:
```
len(diccionario1)
```

[21]: 2

[22]:
```
# otra forma de eliminar...
diccionario1.pop("clave2")
diccionario1
```

[22]: {'clave1': 5}

[23]:
```
len(diccionario1)
```

[23]: 1

[24]:
```
# Borrar todos los elementos del diccionario .clear()
diccionario1.clear()
```

[25]:
```
diccionario1
```

[25]:
```
{}
```

[26]:
```
len(diccionario1)
```

[26]:
```
0
```

1.3 Parte 3

[27]:
```
dic = {"clave1": 10, "clave2": 20, "clave3": 30}
dic
```

[27]:
```
{'clave1': 10, 'clave2': 20, 'clave3': 30}
```

[28]:
```
from collections import Counter
```

[29]:
```
Counter(dic)
```

[29]:
```
Counter({'clave3': 30, 'clave2': 20, 'clave1': 10})
```

[30]:
```
Counter(dic).most_common()
```

[30]:
```
[('clave3', 30), ('clave2', 20), ('clave1', 10)]
```

[31]:
```
Counter(dic).most_common()[0]
```

[31]:
```
('clave3', 30)
```

[32]:
```
Counter(dic).most_common()[1]
```

[32]:
```
('clave2', 20)
```

[33]:
```
Counter(dic).most_common()[2]
```

[33]:
```
('clave1', 10)
```

[34]:
```
Counter(dic).most_common()[-1]
```

[34]:
```
('clave1', 10)
```

[35]:
```
Counter(dic).most_common()[-2]
```

[35]:
```
('clave2', 20)
```

[36]:
```
Counter(dic).most_common()[-3]
```

[36]:
```
('clave3', 30)
```

[37]:
```
# Si queremos seleccionar unos elementos del dicionario,
# en este caso los primeros valores
Counter(dic).most_common()[:2]
```

[37]:
```
[('clave3', 30), ('clave2', 20)]
```

[38]:
```
Counter(dic).most_common()[1:]
```

[38]:
```
[('clave2', 20), ('clave1', 10)]
```

1.4 Parte 4

[39]:
```
diccionario2 = {"clave1": 10, "clave2": 20, "clave3": 30}
diccionario2
```

[39]:
```
{'clave1': 10, 'clave2': 20, 'clave3': 30}
```

[40]:
```
diccionario2.keys()
```

[40]:
```
dict_keys(['clave1', 'clave2', 'clave3'])
```

[41]:
```
listado_keys = []
for key in diccionario2.keys():
    listado_keys.append(key)
listado_keys
```

[41]:
```
['clave1', 'clave2', 'clave3']
```

[42]:
```
listado_values = []
for value in diccionario2.values():
    listado_values.append(value)
listado_values
```

[42]:
```
[10, 20, 30]
```

[43]:
```
# pip install pandas
import pandas as pd
```

[44]:
```
df_diccionario = pd.DataFrame(listado_keys, columns=["claves"])
df_diccionario
```

[44]:

	claves
0	clave1
1	clave2
2	clave3

[45]:
```
df_diccionario["Valores"] = listado_values
df_diccionario
```

[45]:

	claves	Valores
0	clave1	10
1	clave2	20
2	clave3	30

1.5 Parte 5

[46]:
```
# Como ordenar un dataframe
df_diccionario.sort_values(by='Valores')
# ascendente
```

[46]:

	claves	Valores
0	clave1	10
1	clave2	20
2	clave3	30

[47]:
```
# si no se especifica ascending (caso anterior) por defecto ascending = True
df_diccionario.sort_values(by='Valores', ascending=True)
```

[47]:

	claves	Valores
0	clave1	10
1	clave2	20
2	clave3	30

[48]:
```
df_diccionario.sort_values(by='Valores', ascending=False)
# descendente
```

[48]:

	claves	Valores
2	clave3	30
1	clave2	20
0	clave1	10

1.6 -5.1- Strings

1.6.1 Importancia de los strings en AI

Strings como introducción al Procesamiento de Lenguaje Natural (NLP- Natural Language Processing)

Proyectos típicos de Inteligencia Artificial con NLP: * Chatbots, * analítica de textos, * análisis de sentimientos en redes sociales, * Etc.

1.7 Index en los Strings

[49]:
```
s1 = "Hi, How are you?"
s1
```

[49]: 'Hi, How are you?'

[50]:
```
s1[0]
```

[50]: 'H'

[51]:
```
s1[0], s1[1], s1[2]
```

[51]: ('H', 'i', ',')

[52]:
```
s1[-1]
```

[52]: '?'

Longitud

[53]:
```
len(s1)
```

[53]: 16

[54]:
```
for letra in s1:
    print(letra)
```

```
H
i
,

H
o
w

a
r
e

y
o
u
?
```

1.8 most_common()

[55]:
```
s1 = "Hi, How are you?"
s1
```

[55]: 'Hi, How are you?'

[56]:
```
from collections import Counter
```

[57]:
```
# Número de veces que esta en el strings
Counter(s1).most_common()
```

[57]:
```
[(' ', 3),
 ('H', 2),
 ('o', 2),
 ('i', 1),
 (',', 1),
 ('w', 1),
 ('a', 1),
 ('r', 1),
 ('e', 1),
 ('y', 1),
 ('u', 1),
 ('?', 1)]
```

1.9 count

[58]:
```
# Frecuencia aparece la palabra
s1 = "Hi, How are you?"
s1
```

[58]:
```
'Hi, How are you?'
```

[59]:
```
s1.count("Hi")
```

[59]:
```
1
```

[60]:
```
s1.count("How")
```

[60]:
```
1
```

[61]:
```
# debe ser exacta para que la busque, si está en H pues debe buscarse así
s1.count("hi")
```

[61]:
```
0
```

1.10 upper / lower

[62]:
```
s1 = "Hi, How are you?"
s1
```

[62]:
```
'Hi, How are you?'
```

[63]:
```
mayusculas = s1.upper()
mayusculas
```

[63]:
```
'HI, HOW ARE YOU?'
```

[64]:
```
minuscula = s1.lower()
minuscula
```

[64]:
```
'hi, how are you?'
```

1.11 find

[65]:
```
s1 = "Hi, How are you?"
s1
```

[65]:
```
'Hi, How are you?'
```

[66]:
```
len(s1)
```

[66]:
```
16
```

[67]:
```
# Buscar la letra "o"
s1[5], s1[13]
```

[67]:
```
('o', 'o')
```

[68]:
```
# busqueda de la posición
s1_find = s1.find("o")
s1_find
```

[68]:
```
5
```

[69]:
```
# busqueda de la posición
s1_interrogación = s1.find("?")
s1_interrogación
```

[69]:
```
15
```

[70]:
```
s1[-1]
```

[70]:
```
'?'
```

[71]:
```
# si no encuentra la letra, en ese caso pone -1
s1_notfound = s1.find("p")
s1_notfound
```

[71]:
```
-1
```

[72]:
```
# Diferencia entre mayúsculas y minúsculas
s1 = "Hi, how are you?"
s1
```

[72]:
```
'Hi, how are you?'
```

[73]:
```
s1_H = s1.find("H")
s1_H
```

[73]:
```
0
```

[74]:
```
s1_h = s1.find("h")
s1_h
```

[74]:
```
4
```

1.12 startswith, endswith

[75]:
```
s1 = "Hi, how are you?"
s1
```

[75]:
```
'Hi, how are you?'
```

[76]:
```
s1_startswith = s1.startswith("hi")
s1_startswith
```

[76]:
```
False
```

[77]:
```
s1_startswith = s1.startswith("Hi")
s1_startswith
```

[77]:
```
True
```

[78]:
```
s1_endswith = s1.endswith("you")
s1_endswith
```

[78]:
```
False
```

[79]:
```
s1_endswith = s1.endswith("you?")
s1_endswith
```

[79]:
```
True
```

1.13 Split

[80]:
```
# hacemos el split de un string (division en substrings)
# creando una lista de elementos que componen el string
s1 = "Hi, how are you?"
s1
```

[80]:
```
'Hi, how are you?'
```

[81]:
```
s1_split = s1.split()
s1_split
```

[81]:
```
['Hi,', 'how', 'are', 'you?']
```

[82]:
```
s1 = "Hi, how are you?"
s1
```

[82]: 'Hi, how are you?'

[83]:
```
s1_split = s1.split(",")
s1_split
```

[83]: ['Hi', ' how are you?']

[84]:
```
s2 = "Hi , how are you?"
s2
```

[84]: 'Hi , how are you?'

[85]:
```
s2_split = s2.split()
s2_split
```

[85]: ['Hi', ',', 'how', 'are', 'you?']

[86]:
```
s3 = "No cuentes los días, haz que los días cuenten"
s3_split = s3.split(" ", 3)
s3_split
```

[86]: ['No', 'cuentes', 'los', 'días, haz que los días cuenten']

1.14 Replace

[87]:
```
# reemplazar algo
s1 = "Hi, How are you?"
s1
```

[87]: 'Hi, How are you?'

[88]:
```
# sustituir la "H" por "h"
# el primer valor es el valor a sustituir
# el segundo valor el valor que quiero poner
s1.replace("H", "h")
```

[88]: 'hi, how are you?'

1.15 Join

[89]:
```
# une todos los elementos del string por un simbolo específico
# "-" en este caso
```

[90]:
```
s1 = "Hi, How are you?"
s1
```

[90]: 'Hi, How are you?'

[91]:
```
s1_join = "-".join(s1)
s1_join
```

[91]: 'H-i-,- -H-o-w- -a-r-e- -y-o-u-?'

[92]:
```
s1_join = "+".join(s1)
s1_join
```

[92]: 'H+i+,+ +H+o+w+ +a+r+e+ +y+o+u+?'

1.16 Sleep y time

[93]:
```
from time import sleep
```

[94]:
```
%%time
print("Hola")
sleep(2)
print("Mundo")
```

```
Hola
Mundo
CPU times: user 3.58 ms, sys: 1.04 ms, total: 4.61 ms
Wall time: 2 s
```

[95]:
```
%%time
print("Hola")
sleep(10)
print("Mundo")
```

```
Hola
Mundo
CPU times: user 5.16 ms, sys: 1.96 ms, total: 7.12 ms
Wall time: 10 s
```

[96]:
```
import time
```

[97]:
```
%%time
print("Hola")
time.sleep(2)
print("Mundo")
```

```
Hola
Mundo
CPU times: user 3.79 ms, sys: 1.05 ms, total: 4.84 ms
Wall time: 2 s
```

[98]:
```
# Ejemplo 1
```

[99]:
```
%%time
a = time.time()
x = 2
print(x)
b = time.time()

tiempo = b - a
tiempo
```

```
2
CPU times: user 219  s, sys: 0 ns, total: 219  s
Wall time: 192  s
```

[99]: 0.0001461505889892578

[100]:
```
%%time
a = time.time()
print("Hello")
time.sleep(5)
print("World")
b = time.time()

tiempo = b -a
tiempo
```

```
Hello
World
CPU times: user 3.64 ms, sys: 2.05 ms, total: 5.68 ms
Wall time: 5 s
```

[100]: 5.000880002975464

[101]:
```
# Ejemplo 2
```

[102]:
```
%%time

import time

tiempo_inicial = time.time()

contador = 0

for numero in range(1000000):
    contador += 1

tiempo_final = time.time()

tiempo_ejecucion = tiempo_final - tiempo_inicial
```

```
print("tiempo de ejecución: t_final - t.inicial = ", tiempo_ejecucion)
```

```
tiempo de ejecución: t_final - t.inicial =  0.03804612159729004
CPU times: user 38.6 ms, sys: 13  s, total: 38.6 ms
Wall time: 38.2 ms
```

[103]:
```
%%time

import time
import numpy as np

tiempo_inicial = time.time()

contador = 0

for numero in np.arange(1000000):
    contador += 1

tiempo_final = time.time()

tiempo_ejecucion = tiempo_final - tiempo_inicial
print("tiempo de ejecución: t_final - t.inicial = ", tiempo_ejecucion)
```

```
tiempo de ejecución: t_final - t.inicial =  0.048943519592285156
CPU times: user 48 ms, sys: 1.93 ms, total: 49.9 ms
Wall time: 49.1 ms
```

1.17 Operaciones Elementales y algunas cosas más

[104]:
```
5 < 7
```

[104]: True

[105]:
```
5 < 7, 5 < 5, 5 <= 5, 7 > 5, 7 >= 5, 5 > 5
```

[105]: (True, False, True, True, True, False)

[106]:
```
# Ojo con estos, porque van en condicionales if para testear
6 != 5, 6 != 6, 5 == 5, 5 == 6
```

[106]: (True, False, True, False)

[107]:
```
# division:
12/4
```

[107]: 3.0

[108]:
```
# cociente:
11//4
```

[108]: 2

[109]:
```
# resto:
11 % 4
```

[109]: 3

[110]:
```
print('División exacta: ', 12/4, "cociente:", 11//4, "resto", 11%4)
```

```
División exacta:  3.0 cociente: 2 resto 3
```

[111]:
```
resultado = 12/4 + 11//4 + 11%4
resultado
```

[111]: 8.0

[112]:
```
# Multiplicacion
2 * 4
```

[112]: 8

[113]:
```
# Elevado:
2**4
```

[113]: 16

Creado por:

Isabel Maniega