

# Diccionarios

Jonatan Gómez Perdomo, Ph.D.

[jgomezpe@unal.edu.co](mailto:jgomezpe@unal.edu.co)

Arles Rodríguez, Ph.D.

[aerodriguezp@unal.edu.co](mailto:aerodriguezp@unal.edu.co)

Camilo Cubides, Ph.D.(c)

[eccubidesg@unal.edu.co](mailto:eccubidesg@unal.edu.co)

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Universidad Nacional de Colombia



# Agenda

- 1 **Introducción**
- 2 Operadores
- 3 Diccionarios y estructuras de control
- 4 Métodos



# Pareja clave-valor I

En algunas ocasiones, determinar que un objeto es igual a otro (por su **valor**), resulta una tarea muy dispendiosa, ya sea por la complejidad de los objetos o por su tamaño. Por ejemplo, determinar si dos ciudadanos son el mismo sería difícil usando las características físicas (con dispositivos biométricos es cada vez más fácil), que si les asignáramos un número, etiqueta o similar (llamado **clave**) con el que los pudiéramos reconocer. Por ejemplo, las cédulas y tarjetas de identidad ya no son sólo el número que sirve para identificarnos, sino que también llevan información biométrica de nosotros en ellas.

Una pareja clave-valor relaciona un par de objetos, la **clave** con que se identifica o localiza a un objeto, y el **valor** que es el objeto referenciado.



# Pareja clave-valor II

Las parejas clave-valor en Python se escriben en la forma `clave:valor` separando el objeto que va a servir de clave (a la izquierda) con un dos puntos del objeto valor, que será identificado por la clave. En Python se tienen estas restricciones para las parejas clave-valor:

- Una clave, referencia a un sólo valor, pero un valor puede ser referenciado por diferentes claves.
- Cadena, enteros, reales y tuplas que no tenga listas como elementos, pueden ser usados como claves.
- Los valores pueden ser cualquier tipo de dato.



# Pareja clave-valor III

Los siguientes son ejemplos de parejas clave-valor en Python:

- 22: 'SSH'
- 80: 'HTTP'
- 'edad': 18
- 'color': 'rojo'
- 'sopa': 'Plato compuesto de un caldo y uno o '  
          'más ingredientes sólidos cocidos en él'
- 12100100: "Juan María de los Santos"
- 'primos': [2, 3, 5, 7, 11, 13, 17, 19]
- (3, 5, 7): 105



# Definición

Un **diccionario** es una colección de parejas clave-valor donde los valores pueden ser recuperados principalmente por su clave. Cada pareja clave-valor en un diccionario es considerada un **ítem**. En Python, un diccionario se escribe separando los ítems por comas (,) y entre { }.

- {22: 'SSH', 23: 'Telnet', 80: 'HTTP', 3306: 'MySQL'}
- {"edad": 40, "genero": 'Masculino',  
"nombre": 'Juan Salvador', "apellido": 'Gaviota'}
- {"producto": 'Leche', "presentación": [900,1100,1300]}
- {'compañía': 'La Gallina Saraviada',  
"precio": {"grande": 23000, "mediana": 18000,  
"pequeña": 12000}}



# Variables

Un diccionario se puede asignar a una variable.

`x = {}` : Le asigna el diccionario vacío a la variable `x`.

`puertos = {22:'SSH', 23:'Telnet', 80:'HTTP', 3306:'MySQL'}`



# Agenda

- 1 Introducción
- 2 Operadores**
- 3 Diccionarios y estructuras de control
- 4 Métodos





# Unir diccionarios (update)

El método update agrega ítems de un diccionario en otro. Para el programa

```
dict_ports1 = {22: "SSH", 80: "Http"}  
dict_ports2 = {53 : "DNS", 443 : "https"}  
dict_ports1.update(dict_ports2)  
print(dict_ports1)
```

La salida obtenida es (puede variar el orden)

```
{22: 'SSH', 80: 'Http', 53: 'DNS', 443: 'https'}
```



# Comparar

Se usa los operadores convencionales (`==`, `!=`) para comparar diccionarios. Se mira si los diccionarios tienen los mismos ítems. Para el programa

```
a = {123:'Rojas',87:'Rosas'} == {87:'Rosas',123:"Rojas"}  
print(a)  
print({'Rosas':123} != {'rosas':123})  
b = {123:'Rosas',87:'rojas'} == {'Rosas':123,87:'rojas'}  
print(b)
```

La salida obtenida es

```
True  
True  
False
```



# Comparar (is)

Se puede usar el operador `is` para determinar si dos diccionarios son el mismo (referencian al mismo objeto). Para el programa

```
a = {123:'Rojas', 87:'rosas'}  
b = {87:'rosas', 123:'Rojas'}  
c = a  
print(a == b)  
print(a is b)  
print(a == c)  
print(a is c)
```

La salida obtenida es

```
True  
False  
True  
True
```



# Accediendo [ ]

Accede al valor de un ítem con la clave dada. Si no existe un ítem con la clave dada en el diccionario se genera un error. Para el programa

```
puertos = {22:'SSH',23:'Telnet',80:'HTTP',3306:'MySQL'}
protocolo = puertos[22]
print(protocolo)
puertos[443]
```

La salida obtenida es

```
SSH
```

```
-----
KeyError      Traceback (most recent call last)
<ipython-input-5-b7ffcdde03a52> in <module>
----> 1 puertos[443]
KeyError: 443
```



# Agregando/Modificando [ ]

Agrega (si no existe) o modifica (si existe) el valor de un ítem con la clave dada. Para el programa

```
puertos = {80: "HTTP", 23: "SMTP", 443: "HTTPS"}  
print(puertos)  
puertos[23] = 'Telnet'  
print(puertos)  
puertos[110] = 'POP'  
print(puertos)
```

La salida obtenida es (puede variar el orden)

```
{80: 'HTTP', 23: 'SMTP', 443: 'HTTPS'}  
{80: 'HTTP', 23: 'Telnet', 443: 'HTTPS'}  
{80: 'HTTP', 23: 'Telnet', 443: 'HTTPS', 110: 'POP'}
```



# Eliminando elementos (del)

La palabra clave del permite eliminar el ítem con la clave dada. Para el programa

```
puertos = {22:'SSH',23:'Telnet',80:'HTTP',3306:'MySQL'}  
print(puertos)  
del puertos[23]  
print(puertos)
```

La salida obtenida es (el orden puede cambiar)

```
{22: 'SSH', 23: 'Telnet', 80: 'HTTP', 3306: 'MySQL'}  
{22: 'SSH', 80: 'HTTP', 3306: 'MySQL'}
```



# Agenda

- 1 Introducción
- 2 Operadores
- 3 Diccionarios y estructuras de control**
- 4 Métodos



# Consultando un diccionario

Es posible determinar si en un diccionario existe un ítem tiene asociada una clave. Para el programa

```
puertos = {80: "HTTP", 23: "SMTP", 443: "HTTPS"}  
if 80 in puertos:  
    print('yes')
```

La salida obtenida es:

```
yes
```





# Iterando un diccionario I

Es posible iterar un diccionario usando el ciclo for para obtener las claves.  
Para el programa

```
dict_ports = {22: "SSH", 23: "telnet", 80: "Http"}  
for key in dict_ports:  
    print(key)
```

La salida obtenida es (el orden puede variar):

```
22  
23  
80
```



# Iterando un diccionario II

Es posible usar el ciclo `for` y el método `items` para obtener los ítems de un diccionario. Para el programa

```
dict_ports = {22: "SSH", 23: "telnet", 80: "Http"}  
for k,v in dict_ports.items():  
    print(k,'->',v)
```

La salida obtenida es (el orden puede variar)

```
22 -> SSH  
23 -> telnet  
80 -> Http
```



# Agenda

- 1 Introducción
- 2 Operadores
- 3 Diccionarios y estructuras de control
- 4 Métodos**



# Longitud (len)

la función `len` determina el número de ítems en un diccionario. Para el programa

```
puertos = {80: "HTTP", 23 : "SMTP", 443 : "HTTPS"}  
print(len(puertos))
```

La salida obtenida es

3



# Obteniendo valores (get)

Se puede obtener el valor de un ítem a partir de la llave con el método `get`. Adicionalmente se puede decir que retornar si no se encuentra un ítem con dicha clave. Para el programa

```
dict1 = {'a': 1, 'b': 2, 'c': 3}  
print(dict1.get('a'))  
print(dict1.get('d', 'clave no encontrada.'))
```

La salida obtenida es:

```
1  
clave no encontrada.
```



# Máximo y mínimo (max, min)

El método max/min obtiene la máxima/mínima clave de los ítems en el diccionario. Para el programa

```
puertos = {80: "HTTP", 23: "SMTP", 443: "HTTPS"}  
print(max(puertos))  
print(min(puertos))
```

La salida obtenida es:

```
443  
23
```



# Listas de claves y valores (keys, values)

El método `keys` obtiene una lista con todas las claves de los ítems de un diccionario, mientras el método `values` obtiene todos los valores. Para el programa

```
dict1 = {'a': 1, 'b': 2, 'c': 3}  
print(list(dict1.keys()))  
print(list(dict1.values()))
```

La salida obtenida es (el orden puede cambiar):

```
['a', 'b', 'c']  
[1, 2, 3]
```



# Convertir a diccionarios

El método `dict` permite convertir listas de listas y listas de tuplas a diccionarios. Para el programa

```
puertos = [[80, "http"], [20, "ftp"], [23, "telnet"]]
d_port = dict(puertos)
print(d_port)
puertos = [(20, "ftp"), (80, "http"), (23, "telnet")]
d_port = dict(puertos)
print(d_port)
```

La salida obtenida es (el orden puede variar):

```
{80: 'http', 20: 'ftp', 23: 'telnet'}
{20: 'ftp', 80: 'http', 23: 'telnet'}
```





# Eliminar entradas (clear)

El método `clear` se usa para eliminar todo ítem de un diccionario. Para el programa

```
dict_ports = {22: "SSH", 23: "telnet", 80: "Http"}  
print(dict_ports)  
dict_ports.clear()  
print(dict_ports)
```

La salida obtenida es (el orden puede variar):

```
{22: 'SSH', 23: 'telnet', 80: 'Http'}  
{}
```



# Sugerencia

Se sugiere consultar un manual de Python o de sus librerías para determinar si ya existe un método para lo que se quiera realizar con un diccionario.

