

# NIVEL 2

---

DICCIONARIOS: OPERACIONES BÁSICAS



# DICCIONARIO

---

- ✓ Un diccionario es una «estructura de datos». Es decir: un tipo de dato compuesto que nos permite manejar correspondencias entre claves y valores
- ✓ Y ante todo: es un tipo de dato. Esto quiere decir que podemos tener variables y parámetros de tipo diccionario y también funciones que retornen diccionarios



# EJEMPLOS



- Un diccionario español-inglés:

Pone en correspondencia  
*palabras en español* (claves)  
con *palabras en inglés*  
(valores)

- Un directorio de contactos telefónicos:

Pone en correspondencia  
*nombres* (claves) con  
*números de teléfono*  
(valores)

- Una agenda:

Pone en correspondencia  
*fechas* (claves) con *citas*  
(valores)

- El directorio de unos grandes almacenes (o tiendas por departamento):

Pone en correspondencia  
*secciones* como «moda caballero», «electrónica», «música», etc. (claves) con  
*pisos del edificio* (valores)

- Google:

Pone en correspondencia  
*palabras aparecidas en páginas web* (claves) con la  
*URL* de dichas páginas  
(valores)

# CREACIÓN DE UN DICCIONARIO

- ✓ Antes de usar un diccionario es obligatorio **crearlo**
- ✓ El diccionario vacío se denota con un par de llaves, así:



Aquí creamos un nuevo diccionario que estamos guardando en la variable llamada directorio

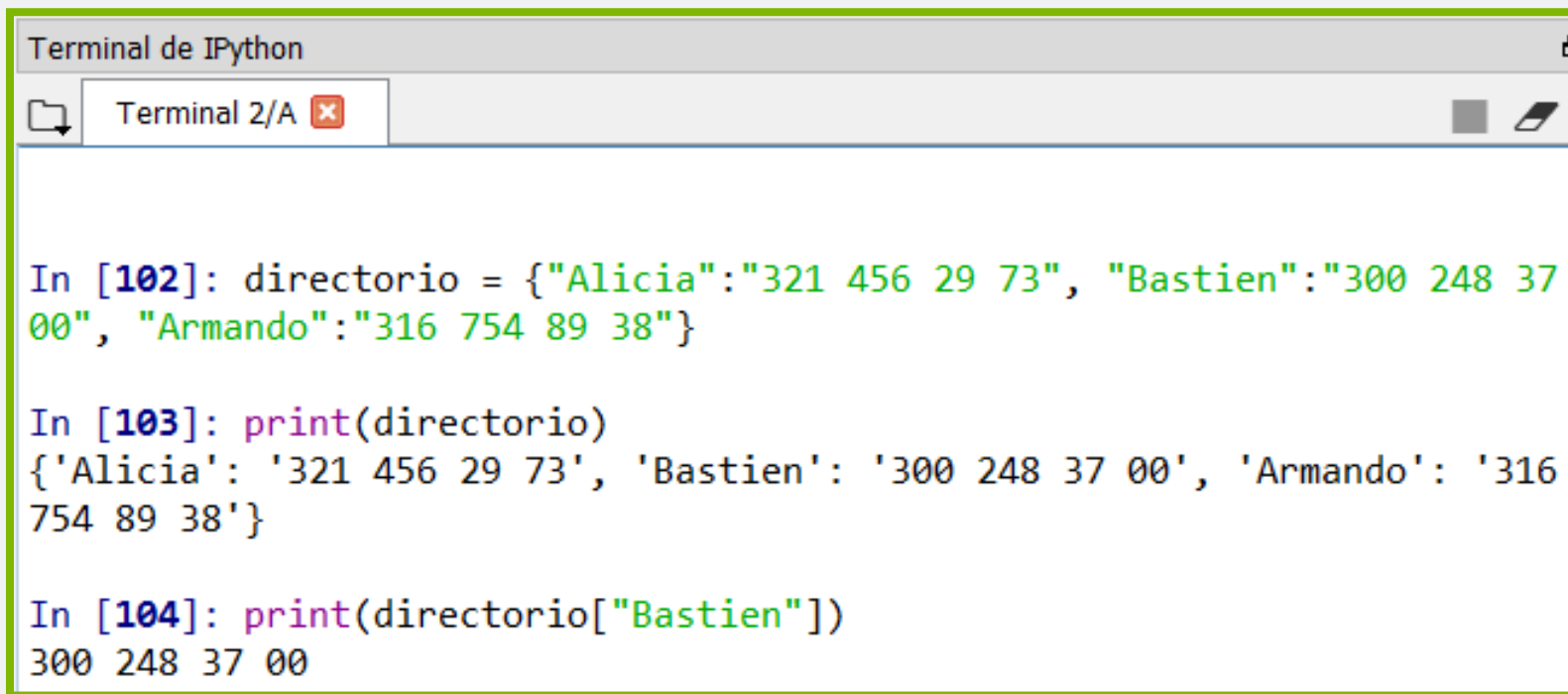
```
Terminal 4/A ✕  
  
In [5]: directorio={}  
  
In [6]: print(directorio)  
{}  
  
In [7]: type(directorio)  
Out[7]: dict
```

Si lo imprimimos, vemos que está vacío

El tipo de la variable directorio es dict

# TAMBIEN ES POSIBLE CREAR UN DICCIONARIO DE UNA VEZ CON DATOS

Basta con poner la lista de parejas **clave:valor**, así:



```
Terminal de IPython
Terminal 2/A

In [102]: directorio = {"Alicia":"321 456 29 73", "Bastien":"300 248 37 00", "Armando":"316 754 89 38"}

In [103]: print(directorio)
{'Alicia': '321 456 29 73', 'Bastien': '300 248 37 00', 'Armando': '316 754 89 38'}

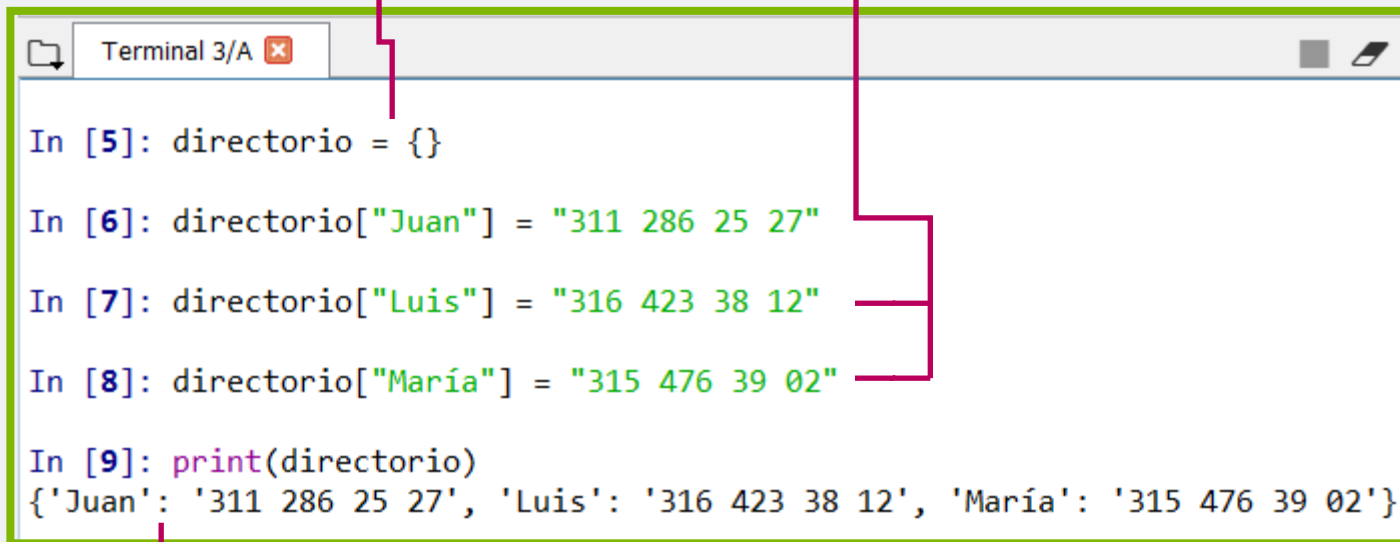
In [104]: print(directorio["Bastien"])
300 248 37 00
```

# ADICIÓN DE DATOS A UN DICCIONARIO

Como vimos en los ejemplos anteriores, lo que añadimos a un diccionario son parejas **clave-valor**, así:

Primero se crea el diccionario

Después se añaden las parejas **clave-valor** (nombre, número de teléfono)



```
Terminal 3/A x
```

```
In [5]: directorio = {}  
In [6]: directorio["Juan"] = "311 286 25 27"  
In [7]: directorio["Luis"] = "316 423 38 12"  
In [8]: directorio["María"] = "315 476 39 02"  
In [9]: print(directorio)  
{'Juan': '311 286 25 27', 'Luis': '316 423 38 12', 'María': '315 476 39 02'}
```

Si lo imprimimos, vemos los datos almacenados

# OTRO EJEMPLO

```
Terminal 2/A x
```

```
In [7]: agenda_día = {}  
In [8]: agenda_día[6] = "Empieza el día"  
In [9]: agenda_día[8] = "Desayuno con Clara"  
In [10]: agenda_día[9.5] = "Clase de IP"  
In [11]: agenda_día[11] = "Clase de cálculo"  
In [12]: agenda_día[13] = "Almuerzo"  
In [13]: agenda_día[15] = "Natación"  
In [14]: agenda_día[16] = "Repaso de química"  
In [15]: agenda_día[19.5] = "Visita a los abuelos"  
In [16]: print(agenda_día)  
{6: 'Empieza el día', 8: 'Desayuno con Clara', 9.5: 'Clase de  
IP', 11: 'Clase de cálculo', 13: 'Almuerzo', 15: 'Natación', 16:  
'Repaso de química', 19.5: 'Visita a los abuelos'}
```

Primero se crea el  
diccionario

Después se añaden  
las parejas **clave-  
valor** (hora, actividad)



OJO! Tanto claves  
como valores pueden  
ser de cualquier tipo:  
entero, flotante,  
string... y hasta otro  
diccionario!

# OTRO EJEMPLO

```
Terminal de IPython
Terminal 2/A x
In [31]: cuentas_mercado = {}
In [32]: cuentas_mercado["Leche"] = 2500
In [33]: cuentas_mercado["Pan"] = 1500
In [34]: cuentas_mercado["Huevos"] = 6000
In [35]: cuentas_mercado["Naranjas"] = 11500
In [36]: cuentas_mercado["Queso"] = 7300
In [37]: print(cuentas_mercado)
{'Leche': 2500, 'Pan': 1500, 'Huevos': 6000, 'Naranjas': 11500, 'Queso': 7300}
```

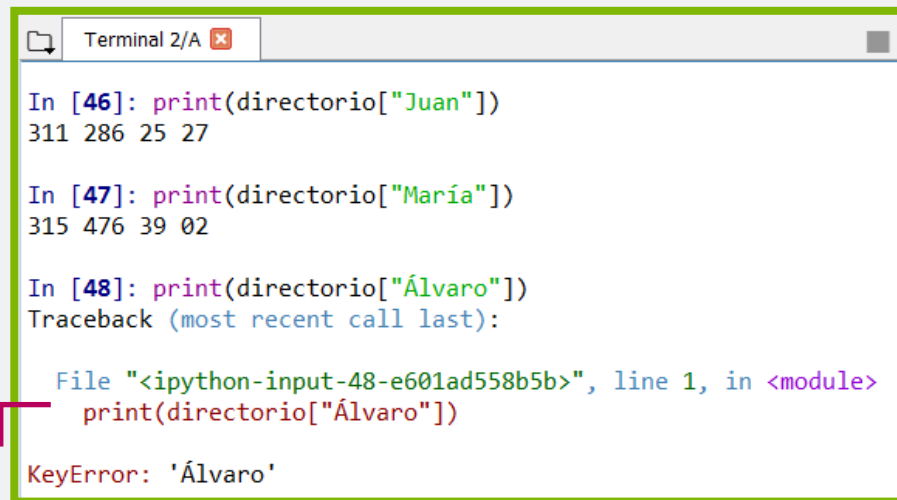
Tanto claves como valores pueden ser de cualquier tipo: entero, flotante, string ... y hasta otro diccionario!



# CONSULTA DE DATOS EN UN DICCIONARIO

Si queremos consultar un **valor**, solo tenemos que indexar por la **clave** con los `[ ]`:

Si accedemos a un elemento inexistente se produce un error del tipo **KeyError** (error de clave) y nos especifica que la clave errónea es 'Álvaro'



```
Terminal 2/A

In [46]: print(directorio["Juan"])
311 286 25 27

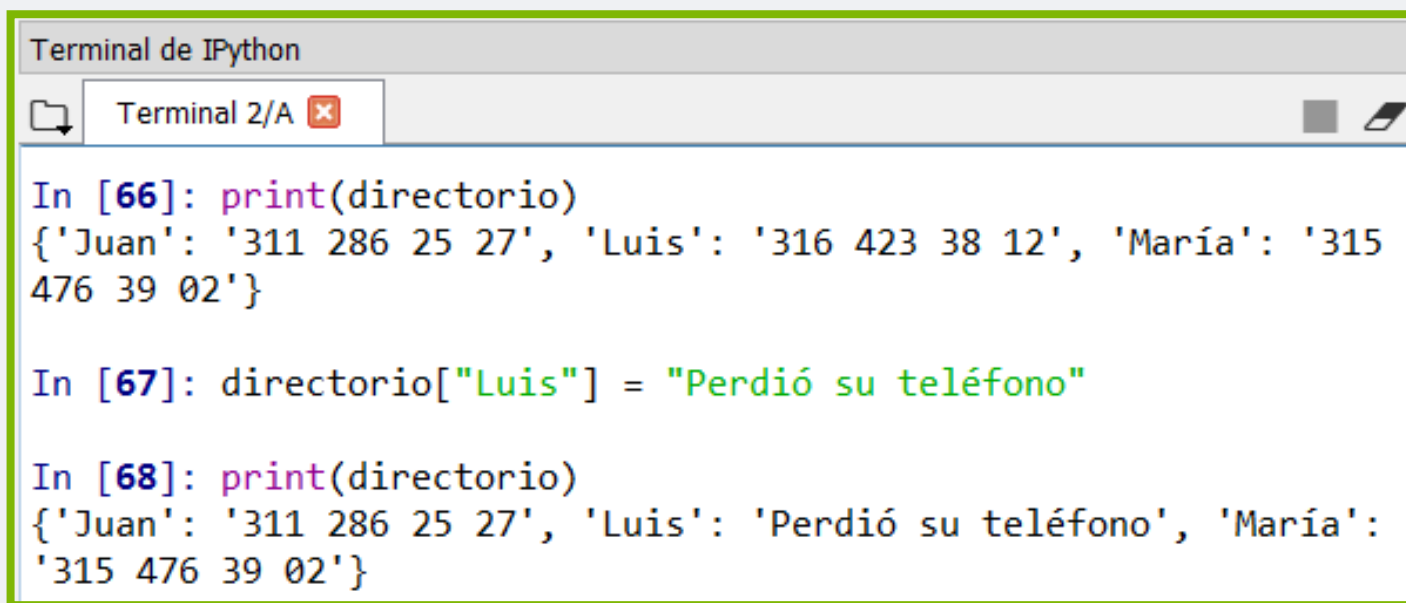
In [47]: print(directorio["María"])
315 476 39 02

In [48]: print(directorio["Álvaro"])
Traceback (most recent call last):

  File "<ipython-input-48-e601ad558b5b>", line 1, in <module>
    print(directorio["Álvaro"])
KeyError: 'Álvaro'
```

# MODIFICACIÓN DE DATOS EN UN DICCIONARIO

Se puede cambiar el **valor** de una **clave** por un **nuevo valor**:

A screenshot of an IPython terminal window. The title bar says "Terminal de IPython". Below it, there's a tab labeled "Terminal 2/A". The terminal shows three lines of code and their output. The first line prints a dictionary with three entries: 'Juan', 'Luis', and 'María'. The second line updates the value for 'Luis'. The third line prints the dictionary again, showing the update.

```
Terminal de IPython
Terminal 2/A

In [66]: print(directorio)
{'Juan': '311 286 25 27', 'Luis': '316 423 38 12', 'María': '315 476 39 02'}

In [67]: directorio["Luis"] = "Perdió su teléfono"

In [68]: print(directorio)
{'Juan': '311 286 25 27', 'Luis': 'Perdió su teléfono', 'María': '315 476 39 02'}
```

# LOS OPERADORES IN Y NOT IN

Terminal de IPython



Terminal 2/A

```
In [57]: print("Juan" in directorio)
True

In [58]: print("Constanza" in directorio)
False

In [59]: print("Pan" in cuentas_mercado)
True

In [60]: print("Jamón" in cuentas_mercado)
False

In [61]: print(6 in agenda_día)
True

In [62]: print(7.5 in agenda_día)
False
```

✓ El operador **in** nos permite preguntar si una **clave** determinada aparece en un diccionario. Devuelve **True** o **False**.

✓ Recuerda que el operador **not in** es el opuesto



# LA FUNCIÓN LEN

Retorna la cantidad de parejas **clave-valor** que hay en un diccionario:

```
Terminal 2/A x
```

```
In [94]: print(cuentas_mercado)
{'Leche': 2500, 'Pan': 2500, 'Huevos': 6000, 'Naranjas': 11500,
'Queso': 7300}

In [95]: len(cuentas_mercado)
Out[95]: 5

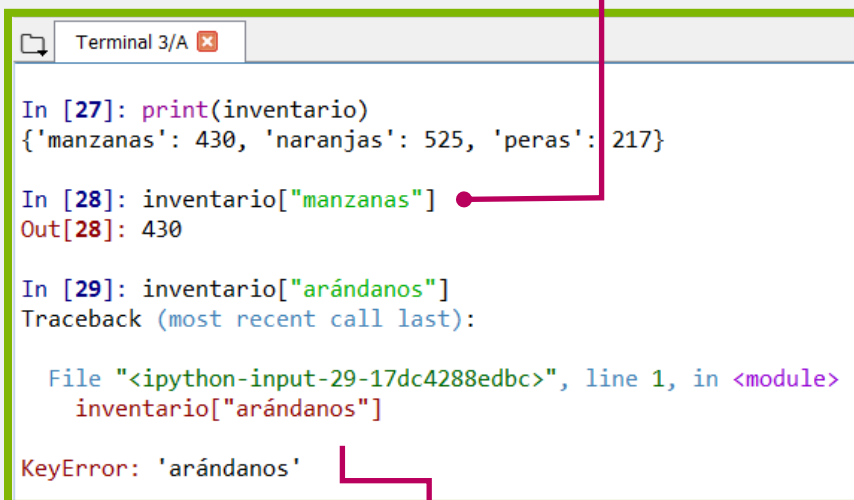
In [96]: print(agenda_día)
{6: 'Empieza el día', 8: 'Desayuno con Clara', 9.5: 'Clase de
IP', 11: 'Clase de cálculo', 13: 'Almuerzo', 15: 'Natación', 16:
'Repaso de química', 19.5: 'Visita a los abuelos', 'Fin del día':
'A dormir'}
```

```
In [97]: len(agenda_día)
Out[97]: 9
```

# EL MÉTODO GET

- Es una «función» que poseen los diccionarios y que es muy útil para consultar datos en un diccionario si no estamos seguros de que estos existan.
- Su sintaxis es:  
`diccionario.get(clave, valor si no existe)`

Si accedemos a una clave que existe en el diccionario, obtenemos su valor



```
Terminal 3/A x

In [27]: print(inventario)
{'manzanas': 430, 'naranjas': 525, 'peras': 217}

In [28]: inventario["manzanas"]
Out[28]: 430

In [29]: inventario["arándanos"]
Traceback (most recent call last):

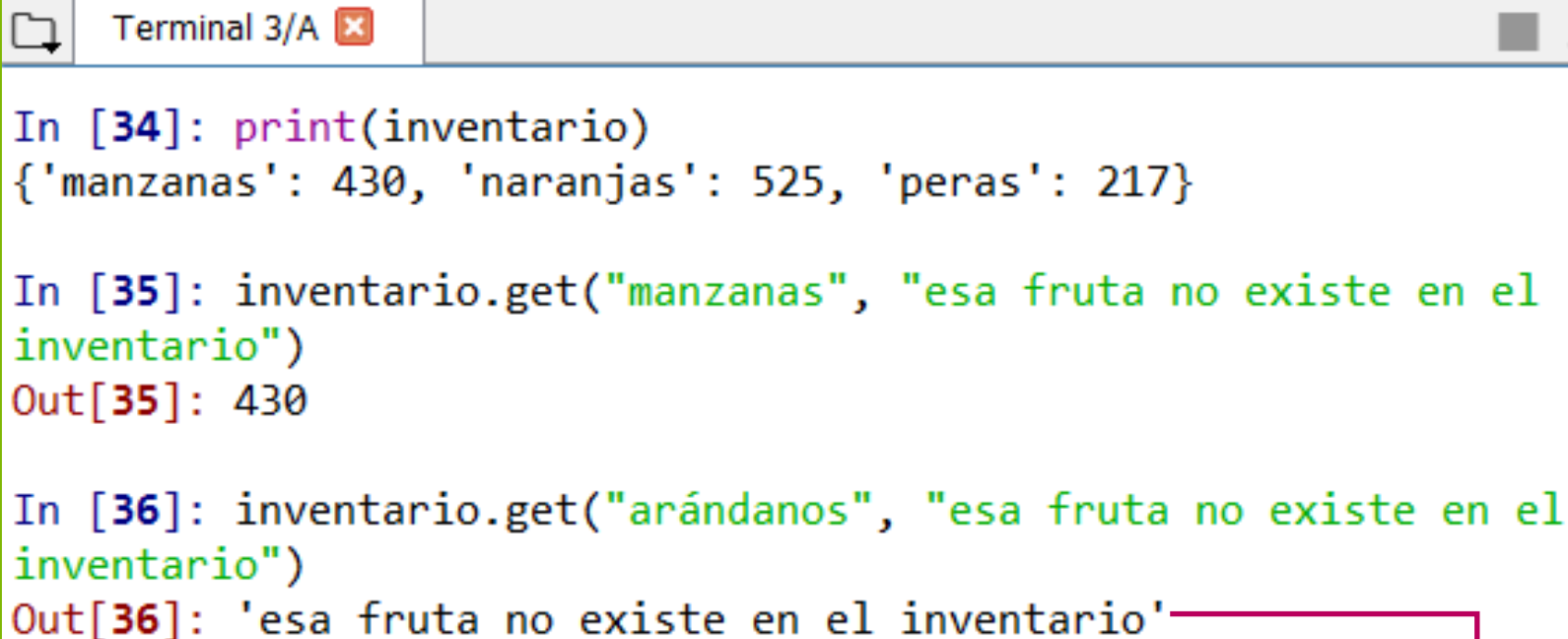
File "<ipython-input-29-17dc4288edbc>", line 1, in <module>
    inventario["arándanos"]

KeyError: 'arándanos'
```

Si accedemos a una clave que no existe en el diccionario, se produce un error

# EJEMPLO DE GET

Si accedemos con get a una clave que existe en el diccionario, obtenemos su valor



```
Terminal 3/A x
```

```
In [34]: print(inventario)
{'manzanas': 430, 'naranjas': 525, 'peras': 217}

In [35]: inventario.get("manzanas", "esa fruta no existe en el
inventario")
Out[35]: 430

In [36]: inventario.get("arándanos", "esa fruta no existe en el
inventario")
Out[36]: 'esa fruta no existe en el inventario'
```

Si accedemos con get a una clave que no existe en el diccionario, NO se produce un error y obtenemos el valor por defecto