

Diccionarios

- Definiciones
- Búsqueda en diccionarios
- Operaciones



Diccionarios

Ejemplo con dinosaurios

Queremos gestionar información sobre dinosaurios

```
names = ["aardonyx", "abelisaurus", "zephyrosaurus", "zuniceratops"]
```

```
diet = ["herbivorous", "carnivorous", "herbivorous", "herbivorous"]
```

```
species = ["celestae", "comahuensis", "schaffi", "christopheri"]
```

- Tenemos una lista para cada elemento
- Todas tienen que tener el mismo largo
- La información está en cada índice, emparejada por índices.

¿Cómo obtenemos/actualizamos datos?

```
def get_dinosaur_index(dinosaur, names):  
    try:  
        i = names.index(dinosaur)  
    except ValueError:  
        i = -1  
    return i
```

¿Cómo obtenemos/actualizamos datos?

```
def get_diet(dinosaur, names, diets):  
    i = get_dinosaur_index(dinosaur, names)  
    if i >= 0:  
        return diets[i]  
    return ""
```

¿Cómo obtenemos/actualizamos datos?

```
def get_diet(dinosaur, names, diets):  
    i = get_dinosaur_index(dinosaur, names)  
    if i >= 0:  
        return diets[i]  
    return ""
```

```
def set_diet(dinosaur, diet, names, diets):  
    i = get_dinosaur_index(dinosaur, names)  
    if i >= 0:  
        diets[i] = diet
```

¿Cómo obtenemos/actualizamos datos?

```
def add_dinosaur(dinosaur, diet, specie, names, diets, species):  
    i = get_dinosaur_index(dinosaur, names)  
    if i < 0:  
        for i, name in enumerate(names):  
            if dinosaur < name:  
                break  
    else:  
        i = len(names)  
    names.insert(i, dinosaur)  
    diets.insert(i, diet)  
    species.insert(i, species)
```

¿Cómo obtenemos/actualizamos datos?

```
def remove_dinosaur(dinosaur, names, diets, species):  
    i = get_dinosaur_index(dinosaur, names)  
    if i >= 0:  
        names.pop(i)  
        diets.pop(i)  
        species.pop(i)
```


¿Cómo obtenemos/actualizamos datos?

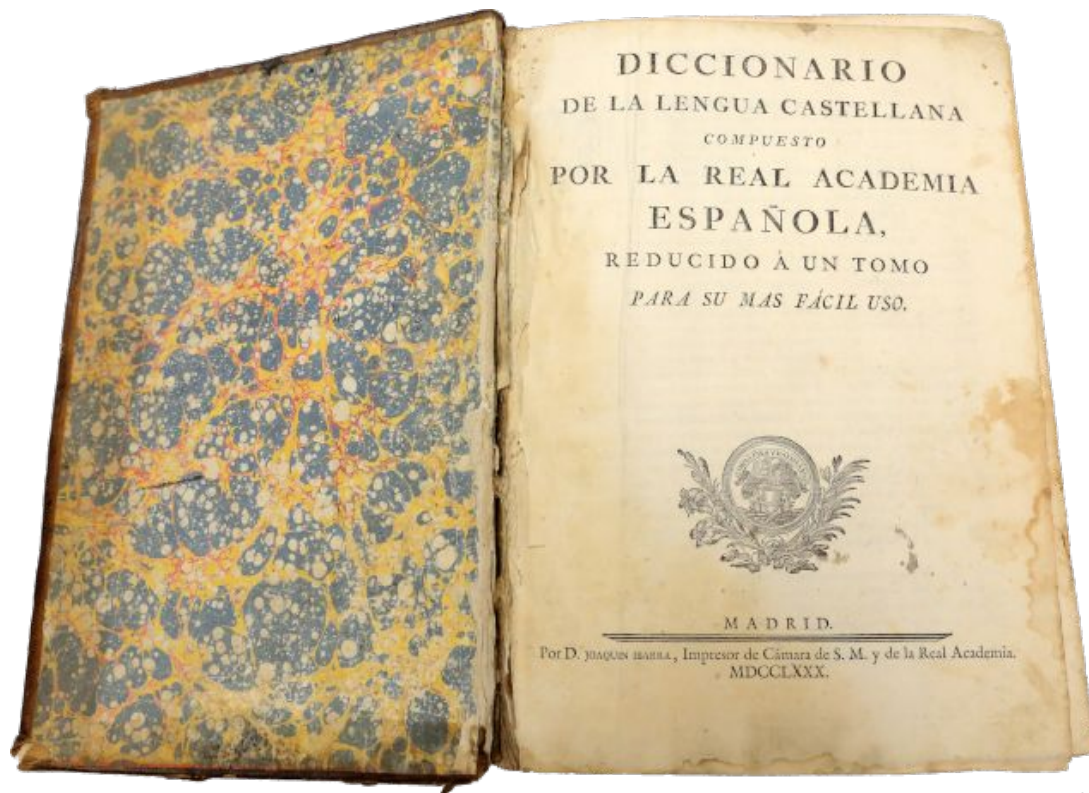
```
def remove_dinosaur(dinosaur, names, diets, species):  
    i = get_dinosaur_index(dinosaur, names)  
    if i >= 0:  
        names.pop(i)  
        diets.pop(i)  
        species.pop(i)
```

- *Desprolijo* si tenemos mucha información.
- Hay que mantener *ordenadas* muchas listas.
- Sólo se pueden indexar con índices enteros
- Hay que recordar *modificar todas* las listas.
- Hay que *pasarle todas las listas* a todas las funciones.

Mejor pensemos en otra estructura de datos

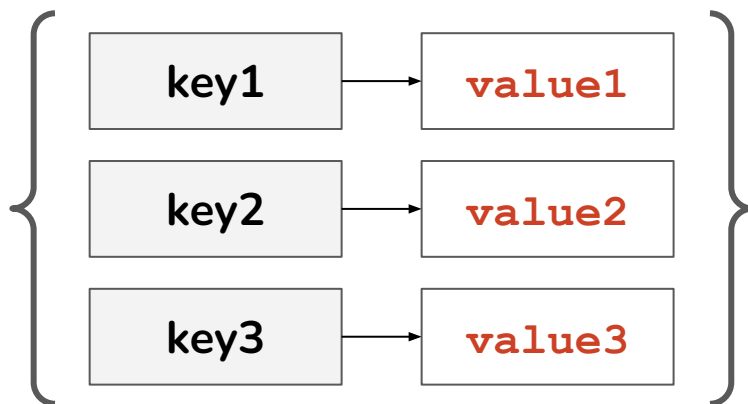
- Quisiéramos buscar la información directamente con el nombre del dinosaurio, no un índice numérico.
- Además sería deseable tener todos los datos juntos, no en listas separadas.

- Es un elemento con **entradas** asociadas a **uno o más valores**.
- En papel está ordenado, pero lo que nos importa son las **entradas**, no el orden.
- Buscamos por **claves**.



Diccionarios en Python

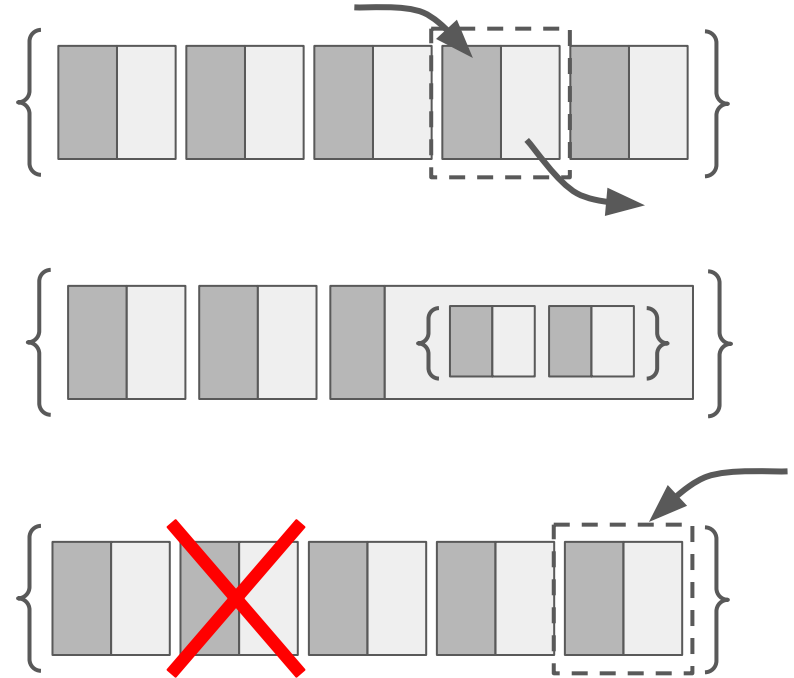
- Colección de elementos, pares **key-value**
 - **key:** es la clave para acceder al elemento
 - **value:** es el valor o contenido del elemento



- Key
 - Tienen que ser **únicas**
 - Tienen que ser **inmutables** (**int**, **float**, **str**, **tuple**, **bool**)
 - *Hashable*
 - ¿Qué puede pasar si usamos **float**?
- Value
 - Cualquier tipo es válido.
 - Puede haber duplicados.
 - Pueden ser números, cadenas, listas, tuplas, otros diccionarios.
- **No** hay **orden** en los pares key / value

Diccionarios en Python

- **Indexados:** se accede a cada elemento a través de la **clave** (key)
- **Anidados:** cada elemento puede contener otro diccionario en el campo *value*
- **Dinámicos:** se pueden añadir o eliminar elementos. **Son mutables**



Secuencias (tuplas, listas)

index	valor
0	"aardonyx"
1	"abelisaurus"
1531	"zuniceratops"

Necesitamos conocer la ubicación exacta de cada elemento: **¡el orden importa!**

Diccionario / Mapa / Tabla

clave		valor
"abelisaurus"	→	"carnivorous"
"aardonyx"	→	"herbivorous"
"zuniceratops"	→	"herbivorous"

**El orden no
importa**


```
{<key1>:<value1>, <key1>:<value1>, ... }
```

Pueden crearse diccionarios de varias formas distintas:

```
d1 = {"Nombre": "Sara", "Edad": 27, "DNI": 30038821}
```

```
d2 = dict([('Nombre', 'Sara'), ('Edad', 27), ('DNI', 30038821)])
```

```
...
```

Diccionarios en Python: ejemplos

```
dinosaurs = {}
```

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

Diccionarios en Python: ejemplos

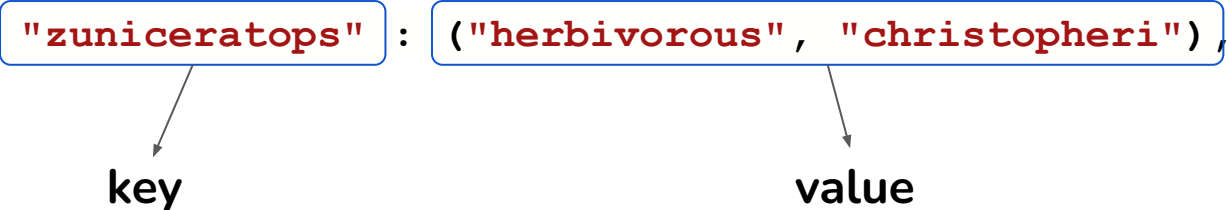
dinosaurs = {} → Diccionario vacío

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

Diccionarios en Python: ejemplos

```
dinosaurs = {}
```

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```



key

value

Búsqueda en Diccionarios

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}  
  
dinosaurs["aardonyx"]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")
```


Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

key

value

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")  
dinosaurs["velociraptor"]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")  
dinosaurs["velociraptor"] → KeyError
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("  
dinosaurs["velociraptor"] -
```

```
def get_diet(dinosaur, names, diets):  
    i = names.index(dinosaur)  
    return diets[i]
```

```
def get_diet(dinosaur, dinosaurs):  
    return dinosaurs[dinosaur][0]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")  
dinosaurs["velociraptor"] → KeyError
```

```
def get_diet(dinosaur, dinosaurs):  
    return dinosaurs[dinosaur][0]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : ("herbivorous", "celestae"),  
    "zephyrosaurus" : ("herbivorous", "schaffi"),  
    "abelisaurus" : ("carnivorous", "comahuensis"),  
    "zuniceratops" : ("herbivorous", "christopheri"),  
}
```

```
dinosaurs["aardonyx"] → ("herbivorous", "celestae")
```

```
dinosaurs["velociraptor"] → KeyError
```

```
def get_diet(dinosaur, dinosaurs):  
    return dinosaurs[dinosaur][0]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : {"diet": "herbivorous", "species": "celestae"},  
    "zephyrosaurus" : {"diet": "herbivorous", "species": "schaffi"},  
    "abelisaurus" : {"diet": "carnivorous", "species": "comahuensis"},  
    "zuniceratops" : {"diet": "herbivorous", "species": "christopherei"},  
}
```

```
dinosaurs["aardonyx"] → {"diet": "herbivorous", "species": "celestae"}  
dinosaurs["aardonyx"]["diet"] → "herbivorous"
```

```
def get_diet(dinosaur, dinosaurs):  
    return dinosaurs[dinosaur]["diet"]
```

Búsqueda en diccionarios

```
dinosaurs = {  
    "aardonyx" : {"diet": "herbivorous", "species": "celestae"},  
    "zephyrosaurus" : {"diet": "herbivorous", "species": "schaffi"},  
    "abelisaurus" : {"diet": "carnivorous", "species": "comahuensis"},  
    "zuniceratops" : {"diet": "herbivorous", "species": "christopherei"},  
}
```

```
dinosaurs["aardonyx"] → {"diet": "herbivorous", "species": "celestae"}  
dinosaurs["aardonyx"]["diet"] → "herbivorous"
```

```
def get_diet(dinosaur, dinosaurs):  
    return dinosaurs[dinosaur]["diet"]
```


Operaciones

Operaciones – Añadir

```
dinosaurs = {  
    "aardonyx" : {"diet": "herbivorous", "species": "celestae"},  
    "zephyrosaurus" : {"diet": "herbivorous", "species": "schaffi"},  
    "abelisaurus" : {"diet": "carnivorous", "species": "comahuensis"},  
    "zuniceratops" : {"diet": "herbivorous", "species": "christopheri"},  
}  
  
dinosaurs['velociraptor'] = {  
    "diet": "carnivorous",  
    "species": "mongoliensis"  
}
```

Se agrega un nuevo
elemento al diccionario

Operaciones – Ver si existe la clave

```
dinosaurs = {  
    "aardonyx" : {"diet": "herbivorous", "species": "celestae"},  
    "zephyrosaurus" : {"diet": "herbivorous", "species": "schaffi"},  
    "abelisaurus" : {"diet": "carnivorous", "species": "comahuensis"},  
    "zuniceratops" : {"diet": "herbivorous", "species": "christopheri"},  
}
```

"Deinonychus" in dinosaurs → False

"zuniceratops" in dinosaurs → True

Permite verificar si una **clave** está incluida en el diccionario

Operaciones – Borrar clave

```
dinosaurs = {  
    "aardonyx" : {"diet": "herbivorous", "species": "celestae"},  
    "zephyrosaurus" : {"diet": "herbivorous", "species": "schaffi"},  
    "abelisaurus" : {"diet": "carnivorous", "species": "comahuensis"},  
    "zuniceratops" : {"diet": "herbivorous", "species": "christopheri"},  
}
```

```
del dinosaurs["zuniceratops"]  
del dinosaurs
```

Elimina un elemento
del diccionario o el
diccionario completo

Operaciones – claves y valores

dict → **.keys()** → **<iterable_keys>**

dict → **.values()** → **<iterable_values>**

dict → **.items()** → **<iterable_keys_values>**

dict →
key → **.get()** → **<value_or_default>**
default →

dict →
key → **.pop()** → **<value_or_default>**
default →

Operaciones – claves y valores

```
agenda = {'Douglas': 44175352, 'Jorge': 53214411,  
          'Tony': 46471257, 'Latania': 61235562}
```

```
for itm in agenda.items():  
    print(itm)
```

✓ 0.5s

('Douglas', 44175352)

('Jorge', 53214411)

('Tony', 46471257)

('Latania', 61235562)

```
for nombre, tel in agenda.items():  
    print(nombre, tel)
```

✓ 0.2s

Douglas 44175352

Jorge 53214411

Tony 46471257

Latania 61235562

Operaciones – claves y valores

```
print(agenda)
for nombre in agenda.keys():
    print(nombre)
```

✓ 0.7s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
```

Douglas

Jorge

Tony

Latania

```
print(agenda)
for tel in agenda.values():
    print(tel)
```

✓ 0.6s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
```

44175352

53214411

46471257

61235562

Operaciones – claves y valores

```
print(agenda)
print(agenda.get('Tony'))
```

✓ 0.6s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
46471257
```

```
print(agenda)
print(agenda.get('Raul', 'No existe'))
```

✓ 0.6s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
No existe
```

```
print(agenda)
print(agenda.get('Raul'))
```

✓ 0.6s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
None
```


Operaciones – claves y valores

```
print(agenda)
print(agenda.pop('Tony'))
print(agenda)
```

✓ 0.7s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Tony': 46471257, 'Latania': 61235562}
46471257
{'Douglas': 44175352, 'Jorge': 53214411, 'Latania': 61235562}
```

```
print(agenda)
print(agenda.pop('Raul', 'No existe'))
print(agenda)
```

✓ 0.6s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Latania': 61235562}
No existe
{'Douglas': 44175352, 'Jorge': 53214411, 'Latania': 61235562}
```

Operaciones – claves y valores

```
print(agenda)
print(agenda.pop('Raul'))
print(agenda)
```

⊗ 0.5s

```
{'Douglas': 44175352, 'Jorge': 53214411, 'Latania': 61235562}
```

KeyError

Traceback (most recent call last)

Input In [140], in <cell line: 2>()

```
1 print(agenda)
----> 2 print(agenda.pop('Raul'))
      3 print(agenda)
```

KeyError: 'Raul'

Diccionarios de compresión

```
{<new_key>:<new_value> for <item> in <iterable>}
```

```
d1 = {10: 'diez', 20: 'veinte', 30: 'treinta', 40: 'cuarenta'}  
d2 = {k:v.upper() for (k,v) in d1.items()}  
d3 = {k:v+'!!' for (k,v) in d1.items()}  
print(d1)  
print(d2)  
print(d3)
```

✓ 0.5s

```
{10: 'diez', 20: 'veinte', 30: 'treinta', 40: 'cuarenta'}  
{10: 'DIEZ', 20: 'VEINTE', 30: 'TREINTA', 40: 'CUARENTA'}  
{10: 'diez!!', 20: 'veinte!!', 30: 'treinta!!', 40: 'cuarenta!!'}
```

Diccionarios y funciones

```
def sum(a,b):  
    return a+b  
  
def rest(a,b):  
    return a-b  
  
def mult(a,b):  
    return a*b  
  
def div(a,b):  
    return a/b  
  
def square(a):  
    return a**2
```

```
op = {"+": sum, "-": rest, "*": mult,  
      "/": div, "**2": square}  
  
def main():  
    x = 4  
    y = 5  
    print( op["+"](x,y) )  
    print( op["-"](x,y) )  
    print( op["*"](x,y) )  
    print( op["/"](x,y) )  
    print( op["**2"](x) )  
  
if __name__ == "__main__":  
    main()
```

[Python tutor](#)

Número variable de argumentos

Número variable de argumentos – *args

El asterisco “*” delante del último argumento indica se permiten múltiples parámetros

```
def usar_args_variables(x,*arg_values):  
    print("primer argumento", x)  
    print(arg_values)  
    for values in arg_values:  
        print("argumentos de *varios:", values)  
  
usar_args_variables('primero', 'segundo',  
                    'tercero', 'cuarto')
```

[Python tutor](#)

Número variable de argumentos – ****kargs**

```
def usar_args_variables(x, **args_dict):  
    print("primer argumento", x)  
    print(args_dict)  
    for key, value in args_dict.items():  
        print(f"{key} = {value}")
```

```
usar_args_variables('primero', a2='segundo',  
                    a3='tercero', a4='cuarto')
```

El doble asterisco “**” delante del último argumento indica se permiten múltiples parámetros con un nombre explícito. Con el nombre y valor de cada uno se arma un diccionario dentro de la función

[Python tutor](#)