Herramientas de programación en Python

3. Tipos estructurados para manejo de datos

3.1 **diccionarios**

Pedro Gomis

pedro.gomis@upc.edu

Diccionarios

- Un **diccionario** es un ***objeto*** de Python similar a una lista, excepto que sus elementos pueden accederse a través de ***índices*** que no necesariamente tienen que ser enteros.
- Lo equivalente a ***índices*** de las lsitas se llaman **claves** (keys), que pueden ser de tipo texto (***string***) o cualquier otro tipo **inmutable**.
- Así, el **diccionario** está formado por **pares** de **clave-valor**. Sintaxis con un ejemplo:

```
>>> temperatura = {'Sevilla': 25.5, 'Londres': 15.4, 'Lisboa': 17.5} # o
>>> temperatura = dict(Sevilla=25.5, Londres=15.4, Lisboa=17.5)
>>> type(temperatura)
<class 'dict'>
>>> Dic = {}           # Diccionario vacío
>>> Dic = dict()       # Diccionario vacío
```

- Son datos estructurados ***mutables***, por lo que podemos añadirle un nuevo par clave-valor:

```
>>> temperatura['Valencia'] = 26
>>> temperatura
{'Sevilla': 25.5, 'Lisboa': 17.5, 'Valencia': 26, 'Londres': 15.4}
```

- En los **diccionarios** se accede al valor los elementos a través de la **clave** y se pueden borrar elementos con **`del`**:

```
>>> temperatura['Londres']
15.4
>>> del temperatura['Sevilla']
>>> temperatura
{'Lisboa': 17.5, 'Valencia': 26, 'Londres': 15.4}
```

Claves y valores

- Cualquier objeto inmutable se puede usar como **clave** de un diccionario: **int**, **float**, **string** y (hasta) **tuplas**. Lo más usual es utilizar strings o enteros. No hay restricciones para los **valores**

```
>>> d = {0: 'a', 1: 'b', 2: 'c', 3: 'd'}
>>> d
{0: 'a', 1: 'b', 2: 'c', 3: 'd'}
>>> d2 = {'a': [1, 2, 3], 'b': [4, 5, 6], 'c': [7, 8, 9]}
>>> d2
{'a': [1, 2, 3], 'b': [4, 5, 6], 'c': [7, 8, 9]}
```

Acceso a los diccionarios

- Se accede a los diccionarios a través de sus claves

```
>>> temperatura['Lisboa']
17.5
>>> d2['a']
[1, 2, 3]
>>> d2['b'][0]
4
```

Operaciones con diccionarios

- El operador booleano **in** se puede emplear también en estas estructuras, pero se usa para averiguar si una clave forma parte del diccionario:

```
>>> 'Londres' in temperatura
True
>>> 'Lima' not in temperatura
True
```

Operaciones con diccionarios

- Se puede recorrer un diccionario con una composición ***for – in***, como otro dato estructurado, ***usando las claves como índices***:

```
>>> for ciudad in temperatura:
...     print('Temperatura en',ciudad,'es',temperatura[ciudad])
...
Temperatura en Lisboa es 17.5
Temperatura en Valencia es 26
Temperatura en Londres es 15.4

>>> for k in d2:
...     print(k, '1er elem -->',d2[k][0])
...
a 1er elem --> 1
b 1er elem --> 4
c 1er elem --> 7
```

Métodos de diccionarios

- Los objetos diccionarios tienen sus propios métodos asociados. Por ejemplo:
  se pueden extraer las ***claves*** y los ***valores*** de un diccionario con los métodos **`keys()`** y **`values()`**.
- La función **`len()`** devuelve el número de elementos (o **pares clave-valor**):

```
>>> temperatura.keys()
dict_keys(['Valencia', 'Lisboa', 'Londres'])
>>> temperatura.values()
dict_values([26, 17.5, 15.4])
>>> len(temperatura)
3
```

- Es útil convertir las claves y valores a listas

```
>>> list(temperatura.keys())
['Sevilla', 'Londres', 'Lisboa', 'Valencia']
>>> list(temperatura.values())
[25.5, 15.4, 17.5, 26]
>>> list(d2.keys())
['a', 'b', 'c']
>>> list(d2.values())
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Métodos y operaciones con diccionarios

- Recorrer un diccionario con una composición ***for – in***, equivale recorrerlo por sus ***claves***:

```
>>> for ciudad in temperatura.keys():
...     print('Temperatura en',ciudad,'es',temperatura[ciudad])
...
Temperatura en Lisboa es 17.5
Temperatura en Valencia es 26
Temperatura en Londres es 15.4
```

- También se puede recorrer un diccionario con una composición ***for – in***, recorriendo simultáneamente el par **clave-valor** de cada elemento **-útil-**:

```
>>> for ciudad, temp in temperatura.items():
...     print('Temperatura en',ciudad,'es',temp)
...
Temperatura en Lisboa es 17.5
Temperatura en Valencia es 26
Temperatura en Londres es 15.4
```

Operaciones con diccionarios

- Al recorrer un diccionario con un bucle, sus elementos no tienen necesariamente que estar ordenados.
- Podemos usar una lista luego ordenar su contenido, usando el método `d.keys()` y `lista.sort()`:
- Alternativamente, se pueden extraer las claves ordenadas en una lista con la función `sorted()`

```
>>> agenda = {"Mario": 555212223, "Yoshi": 999888777, "Luigi": 555222324}
>>> lista = list(agenda.keys())
>>> lista
['Mario', 'Yoshi', 'Luigi']
>>> lista.sort()
>>> lista
['Luigi', 'Mario', 'Yoshi']
>>>  sorted(agenda)
['Luigi', 'Mario', 'Yoshi']
>>> for nombre in lista:
        print(nombre, " -> ", agenda[nombre])

Luigi  ->  555222324
Mario  ->  111222333
Yoshi  ->  999888777
>>>
```

Métodos en diccionarios

Método	Resultado
d.clear()	Elimina todos los elementos del diccionario.
d.copy()	Crea una copia superficial del diccionario.
d.get()	Obtiene el valor de una clave buscada sin que surja error si la clave no está, Si no está la clave, devuelve None (default) o el valor que se quiere devolver
d.items()	Retorna una vista de los elementos del diccionario.
d.keys()	Retorna una vista de las claves del diccionario.
d.pop(clave)	Extrae y retorna el elemento designado por la clave, eliminándolo del diccionario.
d.popitem()	Extrae y retorna el último elemento añadido del diccionario. Usado para iterar sobre un diccionario e ir vaciándolo al mismo tiempo.
d.update(otro)	Actualiza el diccionario con los pares clave-valor de otro, sobrescribiendo las claves ya existentes.
d.values()	Retorna una vista de los valores del diccionario.

Convertir diccionarios en listas

- Usando la función `list()` en combinación con los métodos:

`list(d.items())`	→	lista de tuplas `(clave, valor)`
`list(d.keys())`	→	lista de claves
`list(d.values())`	→	lista de valores

```
>>> personajes = {"Mario": "rojo", "Wario": "amarillo", "Luigi": "verde"}
>>> list(personajes.items())
[('Mario', 'rojo'), ('Wario', 'amarillo'), ('Luigi', 'verde')]
>>> list(personajes.keys())
['Mario', 'Wario', 'Luigi']
>>> list(personajes.values())
['rojo', 'amarillo', 'verde']
>>>
```

Ejemplo 1. Frecuencias (de una lista de enteros)

```
def frecuencias(lst):
    """
    >>> d = frecuencias([1, 3, 2, 4, 2, 2, 3, 2, 4, 1, 2, 1, 2, 3, 1, 3, 1])
    >>> d == {1: 5, 2: 6, 3: 4, 4: 2}
    True
    >>> d = frecuencias([5, 3, 1, 9, 9, 9])
    >>> d == {1: 1, 3: 1, 5: 1, 9: 3}
    True
    """
    d = {}
    for item in lst:
        if item not in d:
            d[item] = 1
        else:
            d[item] += 1
    return d
```

```
>>> lista1 = [1, 2, 3, 4, 1, 2, 3, 1, 2, 1]
>>> frecuencias(lista1)
{1: 4, 2: 3, 3: 2, 4: 1}
>>> lista2= [1, 2, 3, 4, 4, 3, 2, 1, 1, 4]
>>> frecuencias(lista2)
{1: 3, 2: 2, 3: 2, 4: 3}
```

Ejemplo 2. Moda - llama la función frecuencias()

```
def moda(lista):
    """Retorna la moda de una lista de números
    Ejemplos:
    >>> moda([1, 1, 2, 2, 2, 3])
    2
    >>> moda([1, 3, 1, 2, 4, 1, 2, 4, 2, 3, 1, 4])
    1
    >>> moda([1, 3, 2, 4, 2, 2, 3, 2, 4, 1, 2, 1, 2, 3, 1, 3, 1])
    2
    """
    frec = frecuencias(lista)  #función anterior
    vmax = 0
    for clave in frec:
        if frec[clave] > vmax:
            vmax = frec[clave]
            res = clave
    return res
```

```
>>> l3 = [1, 3, 2, 4, 2, 2, 3, 2, 4, 1, 2, 1, 2, 3, 1, 3, 1]
>>> frecuencias(l3)
{1: 5, 2: 6, 3: 4, 4: 2}
>>> moda(l3)
2
```

Ejemplo 3. Frecuencias2 (de palabras en un string)

```
def frecuencias2(frase):
    """Retorna un diccionario con el número de veces que aparece cada
    palabra contenida en un string. Ejemplos:
    >>> d = frecuencias2('la silla de metal y la mesa de abajo de madera')
    >>> d == {'la': 2, 'silla': 1, 'de': 3, 'metal': 1, 'y': 1, \
              'mesa': 1, 'abajo': 1, 'madera': 1}
    True
    >>> d = frecuencias2("a a b b b c c c c d e e")
    >>> d == {'c': 4, 'a': 2, 'b': 3, 'e': 2, 'd': 1}
    True
    """
    res = {}
    for item in frase.split():
        if item in res:
            res[item] += 1
        else:
            res[item] = 1
    return res
```

```
>>> frecuencias2("esta frase es cierta y la frase anterior es falsa")
{'esta': 1, 'frase': 2, 'es': 2, 'cierta': 1, 'y': 1, 'la': 1, 'anterior': 1, 'falsa': 1}
```

Ejemplo 4. Histograma horizontal

```
def histograma(lst):
    """
    >>> histograma([1, 3, 2, 4, 2, 2, 3, 2, 4, 1, 2, 1, 2, 3, 1, 3, 1])
    1 *****
    2 ******
    3 ****
    4 **
    """
    d = frecuencias(lst)
    k = list(d.keys())
    k.sort()
    for clave in k:
        print(clave, d[clave]*'*')
```

```
>>> histograma([1, 1, 1, 5, 5, 5, 2, 2, 3, 4, 4])
1 ***
2 **
3 *
4 **
5 ***
```

Ejemplo 5. Agenda1: lista de listas a diccionario

```
def Agenda1(lst):
    """
    >>> lst = [["Luisa", "931111111"], ["José", "912222222"],\
           ["Judith", "96919391"]]
    >>> d = Agenda1(lst)
    >>> d == {"Luisa": "931111111", "José": "912222222",\
         "Judith": "96919391"}
    True
    """
    d = {}
    for item in lst:
        d[item[0]] = item[1]
    return d
```

```
def Agenda12(lst):          # Otra versión
    D={}
    D.update(lst)
    return D
```

```
def Agenda13(lst):           # version simple
    return dict(lst)
```

Ejemplo 6. Agenda2: agenda de varios números – lista de listas a diccionario

```
def Agenda2(lst):
    """
    >>> lst2 = [["Luisa", "931111111"], ["José", "912222222"], \
                ["Judith", "96919391"], ["Luisa", "65555555"],  \
                ["José", "61133333"]]
    >>> d = Agenda2(lst2)
    >>> d == {'José': ['912222222', '61133333'], 'Luisa': ['931111111', \
              '65555555'], 'Judith': ['96919391']}
    True
    """
    d = {}
    for item in lst:
        if item[0] not in d:
            d[item[0]] = [item[1]]
        else:
            d[item[0]].append(item[1])
    return d
```

Ejercicios a hacer en Python

Temperaturas de ciudades. Un diccionario guarda las temperaturas mínimas de los 3 primeros meses del año de distintas ciudades, de la forma:

```
dic = {'Londres': [3.4, 6.3, 10.5], 'Oslo': [-3.8, -5.0, 4.2], 'Rennes': [2.5, 3.1, 12.3]}
```

a) Diseña una función `mediaTemp(dic)`, en que dado un diccionario como el del ejemplo nos devuelva un diccionario con el valor medio de las temperaturas de cada ciudad (redondear a 2 decimales). Ejemplo:

```
>>> mediaTemp(dic)
{'Londres': 6.73, 'Oslo': -1.53, 'Rennes': 5.97}
```

a) Diseña una función `minimasTemp(dic)`, en que dado un diccionario como el del ejemplo nos devuelva un diccionario con el valor mínimo de las temperaturas de cada ciudad. Ejemplo:

```
>>> minimasTemp(dic)
{'Londres': 3.4, 'Oslo': -5.0, 'Rennes': 2.5}
```

a) Diseña una función `minTemp(dic)`, en que dado un diccionario como el del ejemplo nos devuelva una lista con la ciudad con la temperatura más baja y su valor. Ejemplo:

```
>>> minTemp(dic)
['Oslo', -5.0]
```

Ejercicios a hacer en Python

Diccionario de vinos. Se dispone de un diccionario de vinos donde las claves son nombres vinos y los valores para cada vino son una lista con la información de su denominación de origen, la añada y su precio.

Diseña una función en que, dado un **diccionario** de vinos **d** como el descrito y una lista con un **rango** de precios `[menorPrecio, mayorPrecio]`, nos devuelva una lista de los nombres (ordenados alfabéticamente) de los vinos que estén en este rango de precios.

```
d = {'Muga': ['Rioja',2012,8.5],'Petit caus': ['Penedes',2014, 6.75], \
     'Viña vial': ['Rioja',2016,9.5], 'Lavia': ['Jumilla',2011,12.5], \
   'Bach': ['Penedes',2015, 4.5], 'Pazo B': ['Rias Baixas', 2016, 13.0]}

def lista_vinos(d,rango):
    """ Lista de vinos en un rango de precios
    >>> d = {'Muga': ['Rioja',2012,8.5],'Petit caus': ['Penedes',2014, 6.75], \
         'Viña vial': ['Rioja',2016,9.5], 'Lavia': ['Jumilla',2011,12.5], \
         'Bach': ['Penedes',2015, 4.5], 'Pazo B': ['Rias Baixas', 2016, 13.0]}
    >>> lista_vinos(d,[10, 15])
    ['Lavia', 'Pazo B']
    """
```