DICCIONARIOS

Estructura de datos que almacena elementos en pares clave : valor (key : value).

Características

- La **clave** es **única e inmutable:** booleano, strings int, float,tuplas
- El **valor** puede ser de **cualquier tipo**: números, cadenas, listas, tuplas y otros diccionarios
- Son **mutables**, se puede agregar, modificar y eliminar elementos

Se pueden **crear**:

- Usando **{}** y clave **:** valor separados por dos puntos
  frutas = {"manzana": 0.99, "plátano": 0.75, "naranja": 0.60}
  diccionario4 = { 'lunes': [1, "peras"], 'martes':[2, "manzanas"], 'miércoles':[3, "brocoli"] }
- **dict()**
  - diccionario1 = dict(jueves = “thursday", viernes = Friday”)
  - diccionario6 = dict([('sábado','saturday'), ('domingo','sunday')]) → {'sábado': 'saturday', 'domingo': 'sunday'}

  Se puede crear un diccionario de una **tupla o lista de tuplas** dict(tupla)
- **.fromkeys()** para crear un diccionario con claves y valores **predeterminados**
  - claves = ['jueves', 'viernes', 'sabado'] valor_predeterminado = []
    diccionario7 = dict.fromkeys(claves, valor_predeterminado) → {'jueves': [], 'viernes': [], 'sabado': []

Propiedades:

- **len(): cantidad** de elementos del diccionario, **clave:valor** es un elemento. Se puede conocer **solo las keys o values**: len(diccionario.keys())
- **.keys():** devuelve una **lista con las claves del diccionario**
- **.values():** devuelve una **lista con los valores del diccionario**
- **.items():** devuelve una **lista de tuplas clave-valor**

- **in, not in:** puede ser in diccionario y lo **busca** sobre las **claves** → 'clave' in diccionario. Para buscar en los **valores** in diccionario.values()

Métodos:

- **.copy(): copia** el diccionario en **uno nuevo**

- **.clear(): elimina** todos los elementos

- **.update(): actualiza** el diccionario **con nuevas claves-valor, con otro diccionario o una clave o valor.** Sobreescribe. diccionario1.update(diccnuevo) O diccionario1.update('clave' : 'valor') O diccionario.update({'clavepredeterminada' : 'nuevovalor'})

- **.get(): devuelve el valor de una clave** → diccionario.get(clave) es igual que **diccionario[clave]**. Si **no existe** devuelve **None** y se puede añadir después del valor una , 'una frase' → diccionario.get(clave, 'una frase'). **No sobreescribe.**

- **.setdefault():** igual que get, pero **si no existe sobreescribe** el diccionario insertando una con **valor None** o uno '**predeterminado'**. diccionario.setdefault (clave, 'una frase')

- **sorted(): ordena alfabéticamente** los elementos por las **claves:** sorted(diccionario1) → la claves de diccionario1 ordenadas alfabéticamente. También se pueden **ordenar los valores** solo si son del **mismo type:** sorted(diccionario1.values()) o sorted(diccionario1.values(), reverse=True)

- **.pop(): elimina y devuelve** el valor de una **clave especificada:** diccionario.pop('clave')→devuelve el valor. **Sobreescribe** el diccionario, si la clave no existe genera un error.

- **.popitem(): elimina y devuelve** el valor de la **última clave** añadida**.** diccionario.popitem('clave')→devuelve el valor. **Sobreescribe** el diccionario.

Notas

1. No se puede acceder por índice. Para **acceder a clave** →diccionario['clave']

2. Para **acceder al valor** o valores de una clave: diccionario[‘clave’][posicióndelvalor/es] → valor/es
3. Cuando es un **diccionario con diccionarios** dentro y queremos **acceder a un valor** concreto. → Diccionario1[clave1][indicedentroclave1]

Para **saber si hay una clave** en el diccionario: diccionario[‘clave’]

Para **modificar el valor** de una clave, o cambiar de string a lista: diccionario[‘clave] = ‘nuevovalor’ , o []

Para **añadir valores** a una clave que es una lista podemos usar el método .append() de las listas: diccionario[clave].append(‘nuevovalor’).

Cuando hay una lista se pude usar las mismas funciones que en listas.

Ej1: Diccionario[valorqueeslista].append(9) añade el valor 9 dentro de la key lista Ana

Ej: Diccionario[valorqueeslista][nºindice] accede al dato que queremos dentro de la lista que son valores.

Input:

Variable = input(‘qué mascota quieres eliminar’) ENTER, EN LA BARRA ESCRIBIR mascota2

Diccionario_animales[opcion] DEVUELVE el valor de mascota2 que es perro

SETS

Definidos por **`{}` o set()**

Variable = {x, y, z,}

Podemos **convertir una lista o tupla en un set**

Variable1 = set(tupla1) Variable2 = set(lista1)

- **Únicos** (no pueden estar repetidos)
- **Mutables** (se pueden agregar y eliminar)
- **No** tienen **orden, no** podemos acceder a los elementos por su **posición**
- **No** pueden contener **listas** o **diccionarios**

len(): conocer el **número de elementos**

in, not in: verificar si un **elemento está presente o no** en set.

.**add**(X): añade **un elemento** X al set

.update(): **añade varios** elementos de una **variable o lista**

.copy(): la copia **se crea en una nueva variable**

.pop(): elimina y devuelve un elemento **al azar**

.remove(): elimina solo un elemento especificado. Si **no existe** dará **error**.

.discard(): elimina solo un elemento especificado. Si **no existe** NO **error**.

.**clear()**: para **eliminar todos los elementos**

set1**.union(**set2**):** para unir **2 sets sin repetir** elementos. Crear nueva variable

set1**.intersection(**set2**):** devuelve los **elementos en común** de los dos sets. Crear nueva variable

set1**.difference(**set2**):** devuelve los **elementos** del set1 que **no están** en set2. Crear nueva variable

set1**.symmetric_difference(**set2**):** devuelve los **elementos de ambos set** que aparecen en uno pero no en el otro. Crear nueva variable

set1**.isdisjoint(**set2**):** devuelve **True** si los sets tienen elementos completamente **diferentes** y **False** si tienen elementos **en común**

set1**.issubset(**set2**):** devuelve **True** si **todos** los elementos **del set1** están **en el set2** y **False** si no es así

set1**.issuperset(**set2**):** devuelve **True** si **todos** los elementos **del set2** están **en el set1** y **False** si no es así