Python: diccionarios

SISTEMAS DE GESTIÓN EMPRESARIAL – 148FA (DAM)

Diccionarios

- **Diccionario**: colección no ordenada de **valores** a los que se accede a través de una **clave**.
- En listas y tuplas se accede a la información mediante los índices numéricos. En el diccionario se accede a la información a través de las claves asociadas.
- Las claves son únicas: no puede haber claves repetidas. Los valores sí se pueden repetir.
- No hay forma de acceder a una clave a través de su valor.
- Las claves pueden ser: cadenas, enteros, tuplas… → variables de tipo inmutable
- Los valores pueden ser: listas, cadenas, otros diccionarios…

Diccionarios

mi_diccionario = {‘a’: 1, ‘b’: 2, ‘c’: 3}

Otra manera de crear un diccionario es crearlo vacío e ir añadiendo valores a las claves:

```
clases = {}
clases["lunes"] = [‘SGE’, ‘SOM’]
clases["martes"] = [‘SI’]
clases["miércoles"] = [‘SGE’, ‘MME’]
clases["jueves"] = []
clases["viernes"] = [‘RD’]

print(clases[“miércoles”])
```

Diccionarios

Recorrer un diccionario

Utilizando una variable y for:

```
for dia in clases:
    print (dia, ":",clases[dia])
```

Obtener los valores como tuplas donde el primer elemento es la clave y el segundo el valor

```
for dia, codigos in clases.items():
    print (dia, ":", codigos)
```

Diccionarios

- Acceder a las claves: **keys()**
- Acceder a los valores del diccionario: **values()**

```
mi_diccionario={‘a’: 1, ‘b’: 2, ‘c’: 3}

print(mi_diccionario.values())
>>> dict_values([1, 2, 3, 4])


print(mi_diccionario.keys())
>>> dict_keys([‘a’, ‘b’, ‘c’])
```

Diccionarios

- Añadir claves y valores al diccionario:

```
d = {‘clave1':’valor1’}
print(d)
# {'key': 'value’}

d[‘clave_nueva'] = ‘valor_nuevo’
print(d)

# {‘clave_nueva': ‘valor_nuevo', ‘clave1': ‘valor1'}
```

Diccionarios - Usos

- **Contar el número de apariciones** de cada palabra en un texto
- Contar número de apariciones de cada letra
- **Mantener datos** de los alumnos inscritos. Clave: DNI y valor la lista de notas asignadas a cada alumnos
- Pueden considerarse **bases de datos** simples
- **Traducciones**: la palabra clave sería la palabra en el idioma original y el valor tendría la palabra en el idioma en el que traducirla

Métodos

dict ()

Recibe como parámetro una representación de un diccionario y si es posible, devuelve un diccionario de datos.

'

```
mi_diccionario =  dict(nombre='Pepe', apellido='Perez', edad=28)
```

```
mi_diccionario → {‘nombre’ : 'Pepe', 'apellido’ : 'Perez', 'edad' : 28}
```

Métodos

zip()

Recibe como parámetro dos elementos iterables (cadena o lista). Ambos parámetros deben tener el mismo número de elementos. Se devolverá un diccionario relacionando el elemento i-ésimo de cada uno de los iterables.

```
mi_diccionario = dict(zip('abcd',[1,2,3,4]))
```

```
mi_diccionario → {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
```

Métodos

items()

Devuelve una lista de tuplas, cada tupla se compone de dos elementos: el primero será la clave y el segundo, su valor.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
items = mi_diccionario.items()
```

```
items → [(‘a’,1),(‘b’,2),(‘c’,3),(‘d’,4)]
```

Métodos

keys()

Retorna una lista de elementos, los cuales serán las claves de nuestro diccionario.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
claves= mi_diccionario.keys()
```

```
claves → [‘a’,’b’,’c’,’d’]
```

Métodos

values()

Devuelve una los valores que hay guardados en el diccionario.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}

valores= mi_diccionario.values()


valores→ [1,2,3,4]
```

Métodos

clear()

Elimina todos los ítems del diccionario y lo deja vacío.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
mi_diccionario.clean()
```

```
mi_diccionario → { }
```

Métodos

copy()

Hace una copia del diccionario original. Se puede tratar de manera independiente.

```
dic = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}

dic1 = dic.copy()
```

```
dic1 → {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
```

Métodos

fromkeys()

Recibe como parámetros un iterable y un valor, devolviendo un diccionario que contiene como claves los elementos del iterable con el mismo valor ingresado. Si el valor no es ingresado, devolverá none para todas las claves.

```
mi_diccionario = dict.fromkeys(['a','b','c','d'],1)
```

```
mi_diccionario →  {‘a’ : 1, ’b’ : 1, ‘c’ : 1 , ‘d’ : 1}
```

Métodos

get()

Recibe como parámetro una clave, devuelve el valor de la clave. Si no lo encuentra, devuelve un objeto none.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}

valor = mi_diccionario.get(‘b’)
```

```
valor → 2
```

Métodos

pop()

Recibe como parámetro una clave, elimina esta y devuelve su valor. Si no lo encuentra, devuelve error.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}

valor = mi_diccionario.pop(‘b’)


valor → 2

mi_diccionario → {‘a’ : 1, ‘c’ : 3 , ‘d’ : 4}
```

Métodos

setdefault()

Funciona de dos formas. En la primera como un método *get:*

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}

valor = mi_diccionario.setdefault(‘a’)


valor → 1
```

Métodos

setdefault()

Y por otra parte, sirve para añadir o insertar un nuevo elemento al diccionario.

```
mi_diccionario =  {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
valor = mi_diccionario.setdefault(‘e’,5)
```

```
mi_diccionario → {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4 , ‘e’ : 5}
```

Métodos

update()

Recibe como parámetro otro diccionario. Si se tienen claves iguales, actualiza el valor de la clave repetida; si no hay claves iguales, se inserta este par clave-valor al diccionario.

```
dic1 = {‘a’ : 1, ’b’ : 2, ‘c’ : 3 , ‘d’ : 4}
dic2 = {‘c’ : 6, ’b’ : 5, ‘e’ : 9 , ‘f’ : 10}
dic1.update(dic 2)
```

```
dic1 → {‘a’ : 1, ’b’ : 5, ‘c’ : 6 , ‘d’ : 4 , ‘e’ : 9 , ‘f’ : 10}
```