TEL101 - Iniciación a la Programación

Clase 9: Diccionarios

Prof. Nicolás Torres
nicolas.torresr@usm.cl
Ingeniería Civil Telemática
Departamento de Electrónica
Universidad Técnica Federico Santa María

Diccionarios

Los diccionarios son colecciones **indexadas**, desordenadas y mutables de llaves (*keys*) y valores (*values*) asociados en pares. La mejor forma de pensar en un diccionario es como un conjunto de pares `llave:valor`, con el requisito de que las llaves no se pueden repetir dentro del diccionario.

Creación

Los diccionarios se declaran entre paréntesis curvos `{ }`, y sus elementos, que son pares `llave:valor`, se separan por comas.

In [1]:
```
capitales = {'Chile': 'Santiago', 'Peru': 'Lima'}
```

In [2]:
```
telefonos = {'jack': 4098, 'jill': 4139, 'joan': 4401}
```

Indexación

Los diccionarios están **indexados por llaves**, que pueden ser de **cualquier tipo inmutable**.

Un valor dentro de un diccionario es accesible por medio del operador `[ ]`, usando como índice su llave. En consecuencia, `d[k]`, entrega el valor asociado a la llave `k` del diccionario `d`.

In [3]:
```
d = {1: 'uno', 2: 'dos', 3: 'tres', 4: 'cuatro'}
```

In [4]:
```
print(d[1])
```

```
uno
```

In [5]:
```
print(d[0])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 print(d[0])

KeyError: 0
```

Mutabilidad

Los diccionarios son un tipo de dato **mutable**. Por lo tanto, sus elementos se pueden modificar. En consecuencia, `d[k] = v`, reemplaza el valor de la llave `k` en el diccionario `d` por el valor `v`. Si la llave `k` no existe en el diccionario `d`, entonces se agrega el par `k:v`.

In [6]:
```
d = {1: 'uno', 2: 'dos', 3: 'tres', 4: 'cuatro'}
```

In [7]:
```
# La llave 1 existe, se reemplaza su valor por 'one'
d[1] = 'one'
```

In [8]:
```
print(d)
```

```
{1: 'one', 2: 'dos', 3: 'tres', 4: 'cuatro'}
```

In [9]:
```
# La llave 5 no existe, se crea con valor 'cinco' asociado.
d[5] = 'cinco'
```

In [10]:
```
print(d)
```

```
{1: 'one', 2: 'dos', 3: 'tres', 4: 'cuatro', 5: 'cinco'}
```

Eliminar un elemento

La instrucción `del` permite eliminar un elemento en un diccionario a través de la llave.

In [11]:
```
d = {1: 'uno', 2: 'dos', 3: 'tres', 4: 'cuatro'}
```

In [12]:
```
del d[1]
```

In [13]:
```
print(d)
```

```
{2: 'dos', 3: 'tres', 4: 'cuatro'}
```

Operaciones

Las operaciones elementales de concatenación y repetición no son compatibles con los diccionarios. Sin embargo, sí es posible evaluar membresía sobre las llaves de un diccionario.

In [14]:
```
d = {1: 'uno', 2: 'dos', 3: 'tres', 4: 'cuatro'}
```

In [15]:
```
1 in d
```

Out[15]:
```
True
```

In [16]:
```
'uno' in d
```

Out[16]:
```
False
```

Funciones

Las funciones `len()`, `min()`, `max()` y `sum()`, se pueden aplicar sobre diccionarios. Sin embargo, siempre actúan sobre las llaves.

In [17]:
```
len({1: 2, 3: 4, 5: 6})
```

Out[17]: 3

In [18]:
```
min({1: 2, 3: 4, 5: 6})
```

Out[18]: 1

In [19]:
```
max({1: 2, 3: 4, 5: 6})
```

Out[19]: 5

In [20]:
```
sum({1: 2, 3: 4, 5: 6})
```

Out[20]: 9

Iteraciones

No es necesario iterar sobre un diccionario para buscar un valor si se conoce la llave. Sin embargo, es posible recorrer un diccionario a través de un ciclo `for`.

In [21]:
```
telefonos = {'jack': 4098, 'jill': 4139, 'joan': 4401}

for nombre in telefonos:
    print("El telefono de",nombre,"es",telefonos[nombre])
```

```
El telefono de jack es 4098
El telefono de jill es 4139
El telefono de joan es 4401
```

Uso de los diccionarios en Python

- Contabilizar el número de ocurrencias de todos los elementos:

  `{'Manzanas': 2, 'Naranjas': 1, 'Fideos': 4}`
- Categorizar cada elemento en una categoría:

  `{'Frutas': ['Manzanas', 'Naranjas'], 'Despensa': ['Fideos']}`

Patrón de Contabilizar

Los diccionarios pueden ser utilizados para llevar un registro del número de ocurrencias de diferentes elementos y actualizar sus valores fácilmente a modo de inventario. Dada una secuencia de elementos, es posible definir un patrón para contabilizar el número de veces que se repite cada elemento en los datos.

```
d = {}
for elemento in datos:
    if elemento not in d:
        d[elemento] = 0
    d[elemento] += 1
```

In [22]:
```
lista = ['luis', 'ana', 'pedro', 'ana', 'juan', 'ana', 'ana', 'pedro']

d = {}
for nombre in lista:
    if nombre not in d:
        d[nombre] = 0
    d[nombre] += 1
print (d)
```

```
{'luis': 1, 'ana': 4, 'pedro': 2, 'juan': 1}
```

Patrón de Categorizar

Los diccionarios pueden ser utilizados para categorizar elementos en diferentes clases. Dada una secuencia de elementos y sus categorías, es posible definir un patrón para agrupar elementos por cada categoría presente en los datos.

```
d = {}
for elemento,categoría in datos:
    if categoría not in d:
        d[categoría] = []
    d[categoría].append(elemento)
```

In [23]:
```
lista = [('Roma', 'Italia'), ('Paris', 'Francia'), ('Venecia','Italia'), ('Lyon','Francia'),
         ('Frankfurt', 'Alemania'), ('Florencia', 'Italia')]

d = {}
for ciudad,país in lista:
    if país not in d:
        d[país] = []
    d[país].append(ciudad)
print (d)
```

```
{'Italia': ['Roma', 'Venecia', 'Florencia'], 'Francia': ['Paris', 'Lyon'], 'Alemania': ['Frankfu
rt']}
```