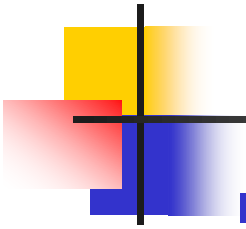
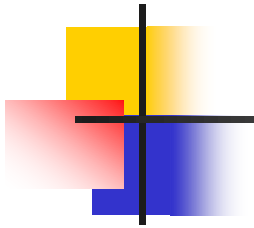
Estructuras de datos nativas de Python

Diccionarios



- Estructuras de datos nativas de Python
  - Secuencias
    - Listas
    - Tuplas
    - Strings
  - Diccionarios ←

DICCIONARIOS

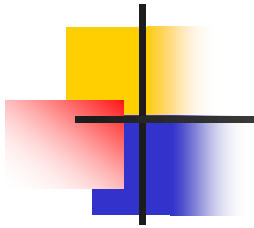


- Grupo de elementos no ordenados que asocian valores con índices a los cuales se les llama llaves o claves

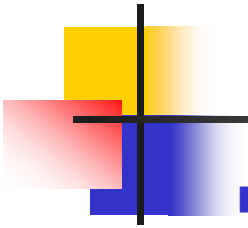
- Cada elemento tiene la estructura

  llave:valor

- Objetivo de la estructura: acceder directamente el valor asociado a la llave



- **Llave**: es el índice de los elementos en los diccionarios, pueden ser de cualquier tipo de datos inmutable. La llave es definida por el programador
  - Nota: Los índices en los diccionarios no son como las secuencias que se indexan exclusivamente con un número entero (0, 1, 2, etc.)
- **Valor**: datos asociados a las llaves, pueden ser de cualquier tipo de datos



- Los elementos del diccionario (par de datos **llave:valor**) están encerrados entre llaves { } y separados por una “,”
- Ejemplos

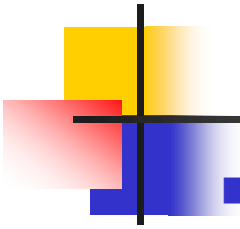
{201055555:“Martha”, 201344444:“William”}

{“arroz”:100, “frijoles”:50, “café”:50}

{5:“cinco”, 2:“dos”, 0:“cero”, 6:“seis”}

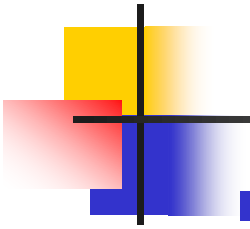
Diccionario vacío: { } o dict()

Operaciones con diccionarios

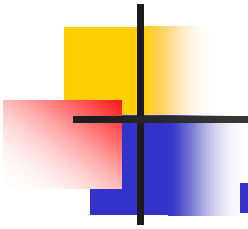


- Crear diccionarios
- Insertar un elemento (par **llave:valor**)
- Obtener el valor asociado a una llave
- Membresía de la llave
- Eliminar un elemento asociado a una llave
- Cambiar el valor asociado a una llave

Crear diccionarios

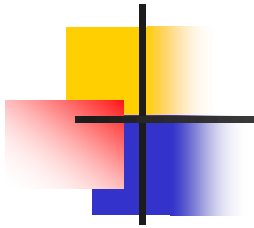


- Podemos crear el diccionario vacío {} y luego agregarle elementos
- Un elemento es agregado cuando damos un par de datos **llave:valor** y la llave no existe.
- Si la llave existe entonces la parte de valor se sustituye



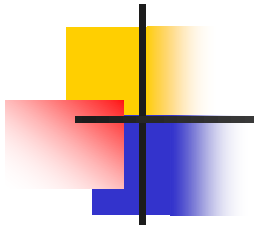
- Ejemplo: traductor de números a letras
  - En este caso los índices son de tipo int

```
>>> dic_nl = {}
>>> dic_nl[15] = "quince"
>>> dic_nl[2] = "dos"
>>> dic_nl
{2: 'dos', 15: 'quince'}
```



- Otras formas de crear un diccionario
  - Directamente en instrucción de asignación

```
>>> dic_nl = {1:"uno", 15: "quince", 3:"tres"}
>>> dic_nl
{3: 'tres', 15: 'quince', 1: 'uno'}
```

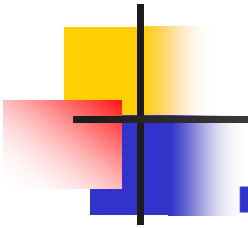


- Accediendo un elemento: damos la llave como el índice y obtenemos su valor asociado:

nombre_diccionario[llave]

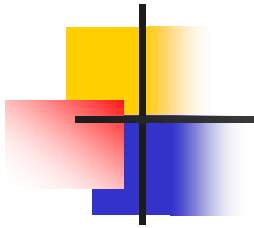
```
>>> dic_ie = {"one":"uno","two":"dos","three":"tres"}

>>> dic_ie["two"]
'dos'
```

La llave debe existir, de lo contrario se produce un error.



- Operador de membresía en diccionarios: busca una llave
  - Si la llave existe retorna verdadero(True) y sino retorna falso (False)

```
>>> "one" in dic_ie
True
>>> "six" in dic_ie
False
```

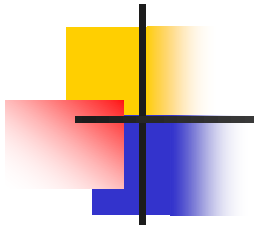


- Función ***len***: número de elementos (o pares) en el diccionario

```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> len(inventario)
3
```

- Funciones ***min*** y ***max*** se aplican a las llaves de los elementos

```
>>> min(inventario)
'arroz'
>>> max(inventario)
'café'
```



- Función ***del***: borra un elemento del diccionario

```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> del inventario["arroz"]
>>> inventario
{'azucar': 400, 'café': 300}
```

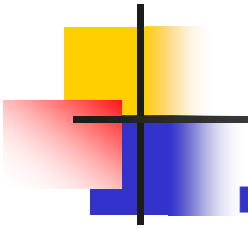
- Función ***pop***: borra un elemento del diccionario y retorna su valor asociado

```
>>> inventario.pop("azucar")
400
```

- Actualizando un elemento de un diccionario: se actualiza el valor asociado a una llave que debe existir

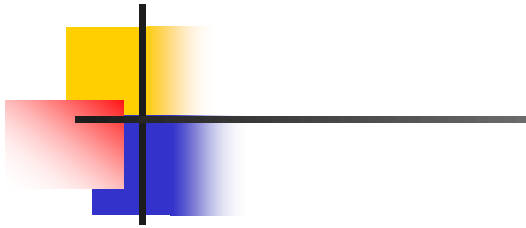
```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> inventario["arroz"]
500
>>> inventario["arroz"] = inventario["arroz"]+200
>>> inventario["arroz"]
700
>>> inventario["aceite"]
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    print (inventario["aceite"])
KeyError: 'aceite'
```

Alias y copias



- Cuando se asigna una variable de tipo diccionario a otra variable, ambas se refieren al mismo objeto, por tanto un cambio en una también modifica a la otra (alias)

- Método ***copy***: crea una copia de un diccionario de tal forma que sea un objeto diferente al original, por tanto los cambios en uno no modifican al otro



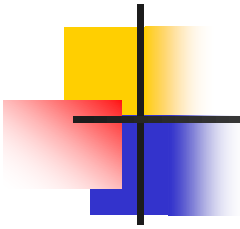
```
>>> opuestos = {"arriba":"abajo","verdadero":"falso"}
>>> copiaOpuestos = opuestos.copy()
>>> opuestos["arriba"]
'abajo'
>>> copiaOpuestos["arriba"] = "nada"
>>> opuestos["arriba"]
'abajo'
>>> copiaOpuestos["arriba"]
'nada'
```

Algunos métodos de diccionarios

- diccionario.keys()

Retorna un objeto iterable que contiene solo las llaves

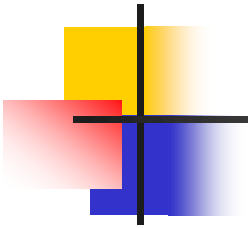
```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> inventario.keys()
dict_keys(['azucar', 'arroz', 'café'])

>>> list(inventario.keys())    # convierte objeto en lista
['azucar', 'arroz', 'café']
```



diccionario.values()

Retorna un objeto iterable que contiene solo los valores asociados a las llaves

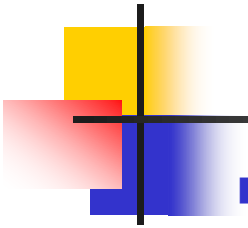
```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> inventario.values()
dict_values([400, 500, 300])

>>> list(inventario.values())
[400, 500, 300]
```



- # diccionario.items()

Retorna un objeto iterable con tuplas: cada tupla corresponde a un elemento del diccionario con la llave y el valor asociado

```
>>> inventario = {"arroz":500, "café":300, "azucar":400}
>>> inventario.items()
dict_items([('azucar', 400), ('arroz', 500), ('café', 300)])

>>> list(inventario.items())
[('azucar', 400), ('arroz', 500), ('café', 300)]
```



- Una forma de recorrer un diccionario obteniendo la llave y el valor a la vez

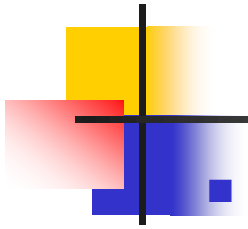
```
>>> for llave, valor in inventario.items():
        print(llave, valor)
```

```
azucar 400
arroz 500
café 300
```

FUNCIÓN *sorted*

- Crea una nueva lista ordenada a partir de una secuencia, conjunto o diccionario
  - ✓ Aplicándola a diccionarios se basa en las llaves

```
>>> inventario = {"azucar": 200, "arroz":500, "frijoles":150, "cafe":100}

>>> inventario_ord = sorted(inventario)

>>> inventario_ord
['arroz', 'azucar', 'cafe', 'frijoles']
```



- Desplegar los elementos de un diccionario en forma ordenada según la llave

```
>>> for producto in sorted(inventario):
        print("Hay", inventario[producto], "de", producto)
```

```
Hay 500 de arroz
Hay 200 de azucar
Hay 100 de cafe
Hay 150 de frijoles
```

Práctica

- Considere el diccionario **estudiantes** con la siguiente estructura
  - Llave: número de carnet (entero positivo)
  - Valor: lista con datos del estudiante: nombre (string), código de la carrera (string) y cantidad de créditos ganados (entero positivo)

Haga un programa principal que tenga un menú para realizar estas funciones sobre ese diccionario.

- Actualizar diccionario: recibe los datos solicitados al usuario desde un programa principal y el diccionario. Si la llave no existe se crea el elemento, si la llave existe se cambian todos sus datos. **Los diccionarios son pasados a las funciones por referencia, por tanto al modificarlos en una función se están modificando los argumentos**.
- Desplegar los datos de un estudiante en particular. Recibe el carnet del estudiante y el diccionario. Sino existe el carnet enviar el mensaje de error respectivo.
- Desplegar los datos de los estudiantes de una carrera en particular. Recibe la carrera y el diccionario.
- Desplegar la cantidad de estudiantes en el diccionario.
- Desplegar todos los estudiantes: llaves y valores, cada elemento en una línea separada. Ordenar por número de carnet. Recibe el diccionario.

Agregue una opción para terminar el programa.

Sobre aprender haciendo:

"Dime y lo olvido,
enséñame y lo recuerdo,
involúcrame y lo aprendo."

Benjamin Franklin