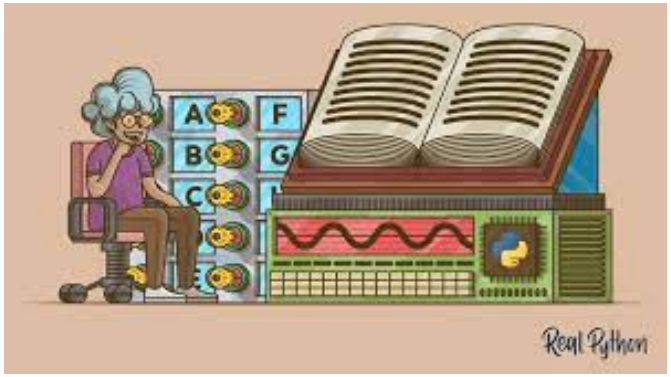
Lenguaje de Programación Python

Diccionarios

Dr. Mario Marcelo Berón
Argentina Programa
Universidad Nacional de San Luis

Características

- Colección mutable de valores.
- Son colecciones de clave-valor. La clave tiene que ser inmutable y los valores pueden ser mutables.

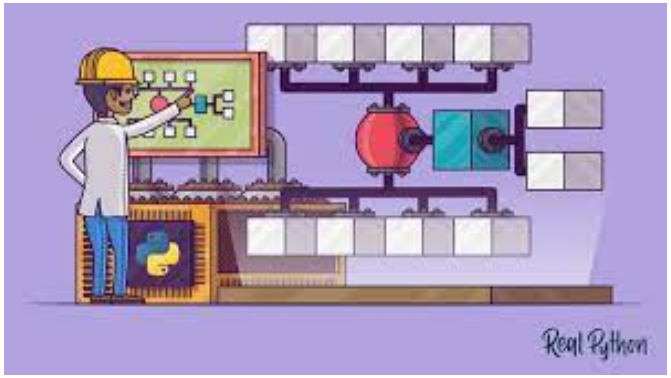


- Los diccionarios se denotan con llaves $\{\}$.
- El acceso a los valores que se encuentran almacenados en el diccionario se hace a través de la clave.

Características

- Los diccionarios son desordenados.
- No importa el orden de los pares clave-valor para determinar si dos diccionarios son iguales.

- Dado que los diccionarios no están ordenados no se pueden sacar slice de los mismos.
- Si se intenta acceder a un diccionario con una clave que no existe se producirá un error *keyError*.



Operaciones

dict

$$dict()$$

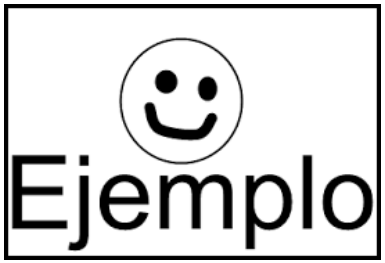
Tarea

Crea un diccionario.

Tarea

- Sin argumentos retorna un diccionario vacío.
- Con argumentos puede suceder:
  - Si es un diccionario crea una copia superficial del argumento.
  - Si es una secuencia de pares el primer objeto se usa como clave y el segundo se usa como valor.
- Los diccionarios también se pueden crear usando $\{\}$. Si no se coloca nada entre llaves el diccionario que se crea es vacío sino se deben colocar pares clave:valor separados por comas.

Operaciones



dict

```
d1=dict({'id': 1948, "Nombre": "Lavadora", "Tamaño": 3
d2=dict(id=1948, Nombre="Lavadora", Tamaño=3)
d3=dict([('id', 1948), ("Nombre", "Lavadora"), \
          ("Tamaño", 3)])
d5={'id': 1948, "Nombre": "Lavadora", "Tamaño": 3}
```

Operaciones

Acceso a los Elementos del Diccionario

$$d[expr]$$

Tarea

Retorna como resultado el valor asociado a la clave indicada por expresión.

Operaciones

Observaciones

- La expresión expr tiene que evaluar a una clave válida del diccionario caso contrario producirá un error.
- $d[expr] = valor$ modifica el valor asociado a expr si el mismo existe en d caso contrario incorpora un nuevo par clave,valor al diccionario.
- Cuando se utiliza en combinación con *del* borra un elemento del diccionario.

Ejemplo

```
d={1:"Hola", 2:"Buen día", 3:"Hasta Luego"}
d[1]  #'Hola'
```

Operaciones

clear()

$d.clear()$

Tarea

Elimina todos los ítems del diccionario d.

Ejemplo

```
d={1:"Soledad",2:"Daniel",3:"José"}
d.clear()
d
{}
```

Operaciones

d.copy()

$$d.copy()$$

Tarea

Retorna una copia superficial del diccionario d.

Ejemplo

```
d={1:"Soledad",2:["Daniel","Gabriel"],3:"José"}
e=d.copy()
e
{1:"Soledad",2:["Daniel","Gabriel"],3:"José"}
```

Operaciones

d.fromkeys(s,v)

$$d.fromkeys(s, v)$$

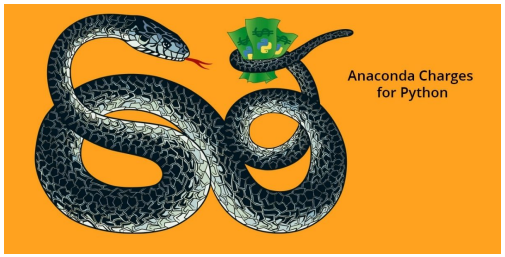
Tarea

Retrona un diccionario cuyas claves son los ítems de la secuencia s y cuyos valores son None o v si v se especifica.

Ejemplo

```
{}.fromkeys((3,51),2)
{3:2,51:2}
{}.fromkeys((3,51))
{3:None,51:None}
```

Métodos: keys, values, items



- *keys*:retorna las claves de un diccionario.
- *values*: retorna los valores de un diciconario.
- *items*: retorna los ítems de un diccionario

Importante

Los valores retornados por esos métodos no son listas no se pueden modificar ni agregar elementos pero pueden ser usados en las iteraciones.

Métodos: keys, values, items

Para iterar por las claves:

```
for v in d.keys():
  print(v)
```

Para iterar por los valores:

```
for v in d.values():
  print(v)
```

Para iterar por ítems:

```
for i in spam.items():
  print(i)
```

Para recuperar clave y valor por separado:

```
for k, v in spam.items():
  print(k,v)
```

Verificación de si una Clave o Valor existe en un Diccionario

Para este propósito se pueden usar los operadores *in* y *not in* descritos cuando se abordó la temática de listas y tuplas.

Ejemplo

```
#Verificar si una clave esta en el diccionario
clave in diccionario.keys()

#verificar si un valor esta en el diciconario
valor in diccionario.values()
```

Diccionarios: get

El método *get* permite recuperar un valor de relacionado con una clave de un diccionario. Este método recibe dos parámetros la clave del valor que se desea recuperar y un valor que será retornado en caso de que la clave no exista en el diccionario.

Ejemplo

```
d={1:"Juan",2:"Pedro",3:"Carlos",4:"Juan"}
#r tendrá "Juan"
r=d.get(1,"No se encuentra la clave")

#r tendrá "No se encuentra la clave"
r=d.get(43,"No se encuentra la clave")
```

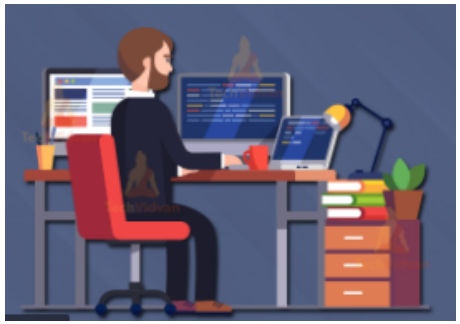
Diccionarios: setDefault

El método *setDefault* permite incorporar una clave al diccionario con un valor por defecto. Es decir si la clave no se encuentra en el diccionario la misma se incorpora con un valor por defecto definido por el programador. El método recibe dos parámetros el primero la clave que se va a buscar en el diccionario y el segundo el valor por defecto que se asociará con la clave en el caso de que la misma no exista.

Ejemplo

```
d={1:"Juan",2:"Pedro",3:"Carlos",4:"Juan"}
d.setDefault(5,"Humberto")
d
{1:"Juan",2:"Pedro",3:"Carlos",5:"Humberto",4:"Juan"}
```

Diccionarios: Impresión



El módulo *pprint* provee dos funciones *pprint()* y *pformat()* que permiten imprimir de forma más "bonita"los valores de un diccionario.
El método *pprint* es más adecuado cuando el diccionario contiene listas o diccionarios o ambas estructuras anidadas.

Diccionarios: Impresión

Ejemplo

```
>>> d
{1: 'Negocio', 2: 'Institución', 3: 'Organización'}
>>> import pprint
>>> pprint.pprint(d)
{1: 'Negocio', 2: 'Institución', 3: 'Organización'}
>>>
```

Conclusión

- Listas y diccionarios son valores que pueden contener múltiples valores incluyendo listas y diccionarios.
- Los diccionarios son útiles porque mapean un ítem (la clave) en otro ítem (el valor).
- Los valores del diccionario se acceden utilizando el nombre del diccionario y entre corchetes la clave.