CONQUER BLOCKS

PYTHON

DICCIONARIOS (PARTE 1)

REPASO

Estructuras de datos:

1. Listas → in - built
2. Arrays → importar modulo
3. Tuplas → in - built
4. Sets → in - built

QUE VEREMOS HOY

Estructuras de datos in - built:

1. Listas

2. Tuplas

3. Sets

4. Diccionarios

DICCIONARIOS

Definición: Colecciones *no ordenadas* de pares *clave-valor*

➡ **Los elementos no llevan un indice asociado si no una clave**

Podemos pensar en un diccionario real, donde buscamos una palabra y encontramos su definición

SINTAXIS BASICA DE UN DICCIONARIO

Set:

```
# sintaxis de un set
mi_set= {'fruta', 45, True}
print(mi_set)
```

✓ 0.0s

```
{'fruta', 45, True}
```

Tupla:

```
# sintaxis de una tupla
mi_tupla_1 = ("fruta", 45, True)
print(type(mi_tupla_1))
```

✓ 0.0s

```
<class 'tuple'>
```

Lista:

```
# sintaxis de una lista
mi_lista_1 = ["fruta", 45, True]
print(type(mi_lista_1))
```

✓ 0.0s

```
<class 'list'>
```

Diccionario:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario)
print(type(mi_diccionario))
```

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3}
<class 'dict'>
```

SINTAXIS BASICA DE UN DICCIONARIO

Set:

```
# sintaxis de un set
mi_set= {'fruta', 45, True}
print(mi_set)
```

✓ 0.0s

```
{'fruta', 45, True}
```

Tupla:

```
# sintaxis de una tupla
mi_tupla_1 = ("fruta", 45, True)
print(type(mi_tupla_1))
```

✓ 0.0s

```
<class 'tuple'>
```

Lista:

```
# sintaxis de una lista
mi_lista_1 = ["fruta", 45, True]
print(type(mi_lista_1))
```

✓ 0.0s

```
<class 'list'>
```

Diccionario:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario)
print(type(mi_diccionario))
```

Clave

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3}
<class 'dict'>
```

SINTAXIS BASICA DE UN DICCIONARIO

Set:

```
# sintaxis de un set
mi_set= {'fruta', 45, True}
print(mi_set)
```

✓ 0.0s

```
{'fruta', 45, True}
```

Tupla:

```
# sintaxis de una tupla
mi_tupla_1 = ("fruta", 45, True)
print(type(mi_tupla_1))
```

✓ 0.0s

```
<class 'tuple'>
```

Lista:

```
# sintaxis de una lista
mi_lista_1 = ["fruta", 45, True]
print(type(mi_lista_1))
```

✓ 0.0s

```
<class 'list'>
```

Diccionario:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario)
print(type(mi_diccionario))
```

Clave : Valor

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3}
<class 'dict'>
```

SINTAXIS BASICA DE UN DICCIONARIO

Set:

```
# sintaxis de un set
mi_set= {'fruta', 45, True}
print(mi_set)
```

✓ 0.0s

```
{'fruta', 45, True}
```

Tupla:

```
# sintaxis de una tupla
mi_tupla_1 = ("fruta", 45, True)
print(type(mi_tupla_1))
```

✓ 0.0s

```
<class 'tuple'>
```

Lista:

```
# sintaxis de una lista
mi_lista_1 = ["fruta", 45, True]
print(type(mi_lista_1))
```

✓ 0.0s

```
<class 'list'>
```

Diccionario:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario)
print(type(mi_diccionario))
```

Clave : Valor

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3}
<class 'dict'>
```

```
mi_diccionario = {}
print(mi_diccionario)
print(type(mi_diccionario))
```

✓ 0.0s

Diccionario vacío

```
{}
<class 'dict'>
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Accedemos a los valores a través de las llaves:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['manzana'])
```

✓ 0.0s

```
1
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Accedemos a los valores a través de las llaves:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['manzana'])
```

✓ 0.0s

```
1
```

Podemos modificar los valores:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario['manzana'] = 4
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 4, 'plátano': 2, 'naranja': 3}
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Accedemos a los valores a través de las llaves:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['manzana'])
```

✓ 0.0s

```
1
```

Agregar pares clave-valor:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario['pera'] = 4
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3, 'pera': 4}
```

Podemos modificar los valores:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario['manzana'] = 4
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 4, 'plátano': 2, 'naranja': 3}
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Accedemos a los valores a través de las llaves:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['manzana'])
```

✓ 0.0s

```
1
```

Agregar pares clave-valor:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario['pera'] = 4
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 1, 'plátano': 2, 'naranja': 3, 'pera': 4}
```

Podemos modificar los valores:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario['manzana'] = 4
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 4, 'plátano': 2, 'naranja': 3}
```

Eliminar pares clave-valor:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
del mi_diccionario['plátano']
print(mi_diccionario)
```

✓ 0.0s

```
{'manzana': 1, 'naranja': 3}
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Eliminar pares clave-valor:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
del mi_diccionario
print(mi_diccionario)
```

Cuidado al usar del:

0.2s

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 3
      1 mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
      2 del mi_diccionario
----> 3 print(mi_diccionario)

NameError: name 'mi_diccionario' is not defined
```

TRABAJAR CON ELEMENTOS DE UN DICCIONARIO

Eliminar pares clave-valor:

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
del mi_diccionario
print(mi_diccionario)
```

⊗ 0.2s

Cuidado al usar del:

```
NameError                                 Traceback (most recent call last)
Cell In[12], line 3
      1 mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
      2 del mi_diccionario
----> 3 print(mi_diccionario)

NameError: name 'mi_diccionario' is not defined
```

Diccionarios vacíos:

```
mi_diccionario = {}
mi_diccionario['manzana'] = 1
mi_diccionario['platano'] = 2
mi_diccionario['naranja'] = 3
print(mi_diccionario)
print(type(mi_diccionario))
```

✓ 0.0s

```
{'manzana': 1, 'platano': 2, 'naranja': 3}
<class 'dict'>
```

CASOS DE USO COMUNES:

Diccionario con informaciones diversas sobre un objeto:

```
persona = {"edad": 23, "estado civil": "casado", "DNI": "78656427A"}
edad = persona["edad"]
print(edad)
```

✓ 0.0s

```
23
```

CASOS DE USO COMUNES:

Diccionario con informaciones diversas sobre un objeto:

```
persona = {"edad": 23, "estado civil": "casado", "DNI": "78656427A"}
edad = persona["edad"]
print(edad)
```

✓ 0.0s

```
23
```

Diccionario con un tipo de información sobre varios objetos:

```
lenguaje_programacion = { "Paolo": "Python", "Lucas": "C", "Dani": "Solidity"}
lenguaje_dani = lenguaje_programacion["Dani"]
print(lenguaje_dani)
```

✓ 0.0s

```
Solidity
```

BUENAS PRACTICAS:

Diccionario con informaciones diversas sobre un objeto:

```
persona = {
    "edad": 23,
    "estado civil": "casado",
    "DNI": "78656427A",
    }
edad = persona["edad"]
print(edad)
```

✓ 0.0s

```
23
```

Diccionario con un tipo de información sobre varios objetos:

```
lenguaje_programacion = {
    "Paolo": "Python",
    "Lucas": "C",
    "Dani": "Solidity",
    }
lenguaje_dani = lenguaje_programacion["Dani"]
print(lenguaje_dani)
```

✓ 0.0s

```
Solidity
```

MÉTODOS DE DICCIONARIOS:

Keys():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario.keys())
```

✓ 0.0s

```
dict_keys(['manzana', 'plátano', 'naranja'])
```

Devuelve una lista de todas las claves en un diccionario

MÉTODOS DE DICCIONARIOS:

Values():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario.values())
```

✓ 0.0s

```
dict_values([1, 2, 3])
```

Devuelve una lista de todos los valores en un diccionario.

MÉTODOS DE DICCIONARIOS:

Items():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario.items())
```

✓ 0.0s

```
dict_items([('manzana', 1), ('plátano', 2), ('naranja', 3)])
```

Devuelve una lista de todos los pares en un diccionario.

MÉTODOS DE DICCIONARIOS:

Get():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario.get('manzana'))
print(mi_diccionario.get('pera'))
```

✓ 0.0s

```
1
None
```

Devuelve el valor de una clave en un diccionario. Si la clave no existe, devuelve un valor predeterminado

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['manzana'])
```

✓ 0.0s

```
1
```

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario['pera'])
```

⊗ 0.0s

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[28], line 2
      1 mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
----> 2 print(mi_diccionario['pera'])

KeyError: 'pera'
```

MÉTODOS DE DICCIONARIOS:

Get():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
print(mi_diccionario.get('manzana'))
print(mi_diccionario.get('pera',0))
```

✓ 0.0s

```
1
0
```

Devuelve el valor de una clave en un diccionario. Si la clave no existe, devuelve un valor predeterminado

MÉTODOS DE DICCIONARIOS:

Pop():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
valor = mi_diccionario.pop('manzana')
print(mi_diccionario)
print(valor)
```

✓ 0.0s

```
{'plátano': 2, 'naranja': 3}
1
```

Elimina y devuelve el valor de una clave en un diccionario

MÉTODOS DE DICCIONARIOS:

Clear():

```
mi_diccionario = {'manzana': 1, 'plátano': 2, 'naranja': 3}
mi_diccionario.clear()
print(mi_diccionario)
```

✓ 0.0s

```
{}
```

Elimina todos los pares clave-valor de un diccionario

DE TUPLAS A DICCIONARIOS:

Sintaxis:

```
mis_tuplas = [(key1, value1), (key2, value2), (key3, value3)]
mi_dict = dict(mis_tuplas)
```

Lista de tuplas

DE TUPLAS A DICCIONARIOS:

Sintaxis:

```
mis_tuplas = [(key1, value1), (key2, value2), (key3, value3)]
mi_dict = dict(mis_tuplas)
```

→ Lista de tuplas

Ejemplo:

```
eventos = [('2022-01-01', 100),
('2022-02-14', 50),
('2022-03-17', 75),
]
event_dict = dict(eventos)
print(event_dict)
```

✓ 0.0s

```
{'2022-01-01': 100, '2022-02-14': 50, '2022-03-17': 75}
```

Lista de tuplas con fecha de eventos y numero de asistentes convertida a un diccionario

IMPORTANTE: Si hay dos tuplas con la misma clave en la lista de tuplas, solo se conservará la última tupla. Es decir, el valor de la clave correspondiente será el valor de la última tupla en la lista.

DE TUPLAS A DICCIONARIOS:

Sintaxis:

```
mis_tuplas = [(key1, value1), (key2, value2), (key3, value3)]
mi_dict = dict(mis_tuplas)
```

→ Lista de tuplas

Ejemplo:

```
eventos = [('2022-01-01', 100),
('2022-02-14', 50),
('2022-03-17', 75),
('2022-03-17', 15),
]
event_dict = dict(eventos)
print(event_dict)
```

✓ 0.0s

```
{'2022-01-01': 100, '2022-02-14': 50, '2022-03-17': 15}
```

Lista de tuplas con fecha de eventos y numero de asistentes convertida a un diccionario

IMPORTANTE: Si hay dos tuplas con la misma clave en la lista de tuplas, solo se conservará la última tupla. Es decir, el valor de la clave correspondiente será el valor de la última tupla en la lista.

DE LISTAS A DICCIONARIOS:

Sintaxis:

```
keys = ['key1', 'key2', 'key3']
values = [value1, value2, value3]
my_dict = dict(zip(keys, values))
```

DE LISTAS A DICCIONARIOS:

Sintaxis:

```
keys = ['key1', 'key2', 'key3']
values = [value1, value2, value3]
my_dict = dict(zip(keys, values))
```

Ejemplo:

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
notas_dict = dict(zip(materias, notas))
print(notas_dict)
```

✓ 0.0s

```
{'matemáticas': 8.5, 'historia': 7.0, 'ciencias': 9.0}
```

DE LISTAS A DICCIONARIOS:

Sintaxis:

```
keys = ['key1', 'key2', 'key3']
values = [value1, value2, value3]
my_dict = dict(zip(keys, values))
```

Ejemplo:

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
notas_dict = dict(zip(materias, notas))
print(notas_dict)
```

✓ 0.0s

```
{'matemáticas': 8.5, 'historia': 7.0, 'ciencias': 9.0}
```

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
lista_tuplas = list(zip(materias, notas))
print(lista_tuplas)
```

✓ 0.0s

```
[('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0)]
```

Aquí, la función *zip()* combina las dos listas *materias* y *notas* en una lista de tuplas de elementos correspondientes, y luego el constructor de diccionarios *dict()* crea un diccionario a partir de esta lista de tuplas.

DE LISTAS A DICCIONARIOS:

Sintaxis:

```
keys = ['key1', 'key2', 'key3']
values = [value1, value2, value3]
my_dict = dict(zip(keys, values))
```

Ejemplo:

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
notas_dict = dict(zip(materias, notas))
print(notas_dict)
```

✓ 0.0s

```
{'matemáticas': 8.5, 'historia': 7.0, 'ciencias': 9.0}
```

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
lista_tuplas = list(zip(materias, notas))
print(lista_tuplas)
```

✓ 0.0s

```
[('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0)]
```

Aquí, la función *zip()* combina las dos listas *materias* y *notas* en una lista de tuplas de elementos correspondientes, y luego el constructor de diccionarios *dict()* crea un diccionario a partir de esta lista de tuplas.

IMPORTANTE: Ten en cuenta que las dos listas deben tener la misma longitud para poder crear un diccionario de esta manera. Si las dos listas tienen longitudes diferentes, la función zip() solo tomará los elementos hasta la longitud de la lista más corta.

DE LISTAS A DICCIONARIOS:

Sintaxis:

```
keys = ['key1', 'key2', 'key3']
values = [value1, value2, value3]
my_dict = dict(zip(keys, values))
```

Ejemplo:

```
materias = ['matemáticas', 'historia', 'ciencias', 'fisica']
notas = [8.5, 7.0, 9.0]
lista_tuplas = list(zip(materias, notas))
print(lista_tuplas)
```

✓ 0.0s

```
[('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0)]
```

```
materias = ['matemáticas', 'historia', 'ciencias']
notas = [8.5, 7.0, 9.0]
lista_tuplas = list(zip(materias, notas))
print(lista_tuplas)
```

✓ 0.0s

```
[('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0)]
```

Aquí, la función *zip()* combina las dos listas *materias* y *notas* en una lista de tuplas de elementos correspondientes, y luego el constructor de diccionarios *dict()* crea un diccionario a partir de esta lista de tuplas.

IMPORTANTE: Ten en cuenta que las dos listas deben tener la misma longitud para poder crear un diccionario de esta manera. Si las dos listas tienen longitudes diferentes, la función zip() solo tomará los elementos hasta la longitud de la lista más corta.

DE SETS A DICCIONARIOS:

Sintaxis:

```
my_set = {('key1', value1), ('key2', value2), ('key3', value3)}
my_dict = dict(my_set)
```

→ Set de tuplas

Ejemplo:

```
notas_set = {('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0)}
notas_dict = dict(notas_set)
print(notas_dict)
```

✓ 0.0s

```
{'ciencias': 9.0, 'matemáticas': 8.5, 'historia': 7.0}
```

Aquí, la función *zip()* combina las dos listas *materias* y *notas* en una lista de tuplas de elementos correspondientes, y luego el constructor de diccionarios *dict()* crea un diccionario a partir de esta lista de tuplas.

IMPORTANTE: Si hay dos tuplas con la misma clave en la lista de tuplas, solo se conservará la última tupla. Es decir, el valor de la clave correspondiente será el valor de una de las tuplas del set.

DE SETS A DICCIONARIOS:

Sintaxis:

```
my_set = {('key1', value1), ('key2', value2), ('key3', value3)}
my_dict = dict(my_set)
```

→ Set de tuplas

Ejemplo:

```
notas_set = {('matemáticas', 8.5), ('historia', 7.0), ('ciencias', 9.0), ('ciencias', 8.0)}
notas_dict = dict(notas_set)
print(notas_dict)
```

✓ 0.0s

```
{'ciencias': 8.0, 'matemáticas': 8.5, 'historia': 7.0}
```

Aquí, la función *zip()* combina las dos listas *materias* y *notas* en una lista de tuplas de elementos correspondientes, y luego el constructor de diccionarios *dict()* crea un diccionario a partir de esta lista de tuplas.

IMPORTANTE: Si hay dos tuplas con la misma clave en la lista de tuplas, solo se conservará la última tupla. Es decir, el valor de la clave correspondiente será el valor de una de las tuplas del set.

REPASO

1) Que es un diccionario y su sintaxis

2) Manipular pares clave-valor
   (añadir, reasignar, eliminar…)

3) Casos de uso

4) Métodos aplicables a diccionarios

5) Relación entre listas/tuplas/sets y diccionarios

CONQUER BLOCKS