Estructuras de datos en Python

Características

- Son formas de organizar los datos
- Permiten manipulación, extracción, búsqueda e inserción
- Son más eficientes
- Son mutables si sus elementos pueden ser modificados después de creados

Tipos de estructuras de datos en Python

- Listas
- Tuplas
- Sets
- Diccionarios

```
lista = [el_1, el_2, el_3 ... el_n]
tupla = (el_1, el_2, el_3 ... el_n)
diccionario = {llave: valor}
set = {el_unico_1, el_unico_2 ... el_unico_2}
```

📌 Las listas y las tuplas pueden contener elementos duplicados, mientras que los sets sólo contienen elementos únicos.

	Lista	Tupla	Diccionario	Set
Ordenado	Sí	Sí	No	No
Mutable	Sí	No	Sí	Sí
Permite duplicados	Sí	Sí	No	No

Listas

- Permiten almacenar elementos de forma ordenada
- Sus elementos se separan por comas
- Son ordenadas, se mantienen en el orden en el que son declarados

```
languages = ["python", "java", "golang"];
print(languages) # python, java, golang

languages[0] # Imprime el valor indicado entre corchetes -> p

languages[-1] # Imprimirá el último elemento

languages[-3] # Imprimirá el primer elemento

languages[0:2] # Imprimirá python, java

len(languages) # Imprime la longitud de la lista -> 3
```

📌 El conteo interno de las listas empieza por 0. Por tanto, si la longitud de la lista *languages* es de 3, para acceder al último elemento habría que acceder de forma: languages[2] → Imprimiría “golang”.

```
# Una lista puede contener un mix de tipos de variables
mix = [1, 2.0, True, "Python", 1]
print(mix) # 1, 2.0, True, Python, 1
```

```
# Las listas pueden ser anidadas
languages = ["python", "java", "golang"]
programming = [languages, "patience", "effort"]

print(programming) # ["python", "java", "golang"], languages,
programming[0][0] # -> Imprimirá python

languages[0] = "dart" # Reemplaza python por dart
```

```
# Añadir elementos a la última posición de la lista con "appe
languages = ["dart", "java", "golang"]
languages.append("python")

# Extend une dos listas
other_lang = ["c", "c++"]
languages.extend(other_lang) # Imprimirá ["dart", "java", "go
```

Tuplas

- Almacenan elementos

- Similares a las listas, pero las tuplas son INMUTABLES
- Se definen entre paréntesis y separan sus elementos con comas

```
languages = ("dart", "java", "golang")

other_languages = "c", "c++" # Puedes declarar una tupla sin

languages[0] # Imprimirá dart

languages[-1] # Imprime la última posición de la tupla

languages[0] = "java" # No se puede reasignar un valor en una
```

Diccionarios

- Almacenan información
- A diferencia de listas y tuplas, almacenan en pares de datos de llave/valor
- Cada valor corresponde a una llave
- En otros lenguajes de programación se conocen como JSON o Hash maps
- Se definen entre llaves
- Cada elemento es un par de datos, separados por dos puntos
- Se separa cada par de elementos con una coma
- No puedes acceder a sus elementos mediante índices, sino mediante sus llaves
- No puede haber dos llaves iguales
- Las llaves normalmente son de tipo texto
- Los valores no tienen restricción sobre qué tipo de dato almacenan
- Pueden tener elementos anidados

```
language = {
    "name": "python",
    "author": "Guido"
}

language["name"] # Imprimirá python

# AÑADIENDO ELEMENTOS NUEVOS

language["release_year"] = 1991 # Añadiendo un nuevo par de i

language["features"] = ["easy", "great"]

# IMPRIMIR SU CONTENIDO MEDIANTE FUNCIONES BUILT-IN

language.items() # Devuelve una lista de tuplas, combinando c
# dict_items([('name', 'python), ('author', 'Guido')...])

language.keys() # Devuelve sólo las llaves

language.values() # Devuelve sólo los valores
```

Sets

- Permite guardar elementos únicos
- Se definen entre llaves
- Sus elementos se separan entre comas
- No son estructuras ordenadas
- No se puede acceder a ellos mediante un índice
- Pueden almacenar elementos de diferentes tipos
- Son mutables → podemos alterar el contenido, pero no las variables que contiene

```
set1 = {1, 2, 3}
```

```
set2 = {5, 6, 7}

# AÑADIR ELEMENTOS EN EL SET

set2.add(4) # Añade un 4 al conjunto

set2.update(8, 9) # Añade un 8 y un 9 al conjunto

# SABER LA LONGITUD DEL SET

len(set2) # Para conocer la longitud del set

# ELIMINACIÓN DE ELEMENTOS EN EL SET

set2.dicard(4) # Para eliminar un elemento de un set

set2.remove(8) # Para eliminar un elemento de un set

set2.clear() # Eliminar TODOS los elementos del set
```

@Desiré Marrón