DIGITALHOUSE

Estructuras de Datos en Python

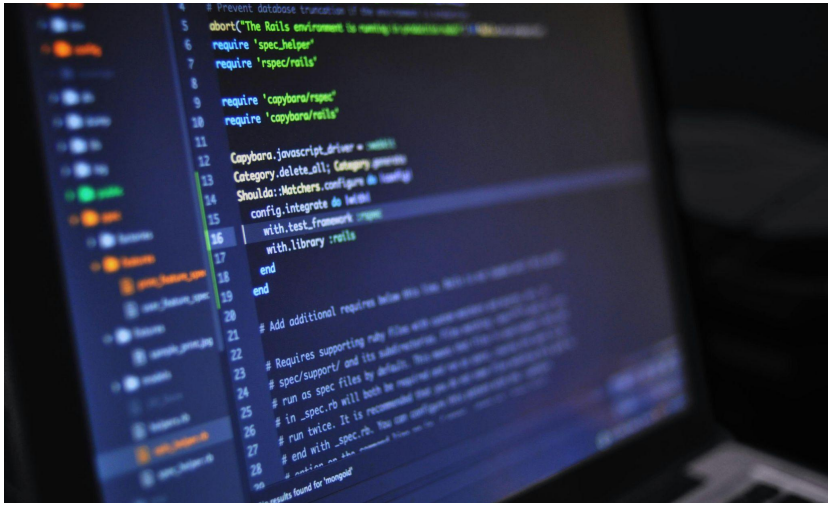
01

¿Qué son las Estructuras de Datos?

Estructuras de Datos

En Python, una estructura de datos es una forma de organizar y almacenar datos de manera que puedan ser utilizados eficientemente. Estas estructuras proporcionan métodos para insertar, acceder, modificar y eliminar datos de manera ordenada y eficiente.

Las estructuras de datos en Python pueden ser simples, como listas y tuplas, o más complejas, como diccionarios y conjuntos. Cada tipo de estructura de datos tiene sus propias características y métodos que facilitan diversas operaciones.



02

Características de las estructuras de datos

Listas

Una lista en Python es una estructura de datos que permite almacenar una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo (números, strings, booleanos, incluso otras listas) y pueden ser modificados una vez que la lista ha sido creada.

```
# Creación de Listas:

mi_lista = [1, 2, 3, 4, 5]

# También es posible crear listas vacías
# y luego agregar elementos a ellas:

lista_vacia = []
lista_vacia.append(10)
lista_vacia.append(20)
lista_vacia.append(30)
```

Tuplas

Una tupla en Python es una estructura de datos similar a una lista, pero con la principal diferencia de que es inmutable, es decir, una vez creada, no se puede modificar. Las tuplas se utilizan para almacenar colecciones ordenadas de elementos de diferentes tipos, al igual que las listas.

```
# Creación de Tuplas:

mi_tupla = (1, 2, 3, 4, 5)

# También se pueden crear tuplas sin paréntesis,
# simplemente separando los elementos por comas.
# Sin embargo, se recomienda el uso de paréntesis
# para mayor claridad y consistencia:

otra_tupla = 10, 20, 30
```

Diccionarios

Es una estructura de datos que permite almacenar pares de datos llamados "clave-valor". Cada elemento en un diccionario consta de una clave única y su correspondiente valor. Las claves suelen ser de tipo inmutable (como strings o números), mientras que los valores pueden ser de cualquier tipo (números, strings, listas, diccionarios, etc.).

```
# Creación de Diccionarios:

mi_diccionario = {"nombre": "Juan", "edad": 30, "ciudad": "Madrid"}

# Para acceder a un valor en un diccionario,
# se utiliza la clave correspondiente:

print(mi_diccionario["nombre"])  # Imprime "Juan"
print(mi_diccionario["edad"])    # Imprime 30
```

Sets

Es una colección desordenada de elementos únicos. En otras palabras, un conjunto no puede contener elementos duplicados y no tiene un orden definido. Los conjuntos son útiles cuando necesitas almacenar una colección de elementos sin preocuparte por el orden o la duplicación, y cuando necesitas realizar operaciones matemáticas de conjuntos como unión, intersección, diferencia, etc.

```
# Creación de Sets:

mi_conjunto = {1, 2, 3, 4, 5}

# También es posible crear conjuntos a partir de listas
# o cualquier otra secuencia utilizando la función set().
# Por ejemplo:

lista = [1, 2, 3, 3, 4, 4, 5]
conjunto_desde_lista = set(lista)  # Elimina duplicados
```

Strings

Es una secuencia de caracteres encerrados entre comillas simples ' o dobles ". Los strings son inmutables, lo que significa que una vez que se crean, no se pueden modificar. Los strings son una parte fundamental de Python y se utilizan para almacenar y manipular texto de cualquier tipo, como nombres, direcciones, mensajes, etc.

```python
# Strings con comillas simples
string_comillas_simples = 'Hola, mundo!'
print(string_comillas_simples)

# Strings con comillas dobles
string_comillas_dobles = "¡Hola, mundo!"
print(string_comillas_dobles)

# Strings con comillas triples (pueden ser simples o dobles)
string_comillas_triples = '''Este es un string
con múltiples líneas.'''
print(string_comillas_triples)
```