



Sesión 7 

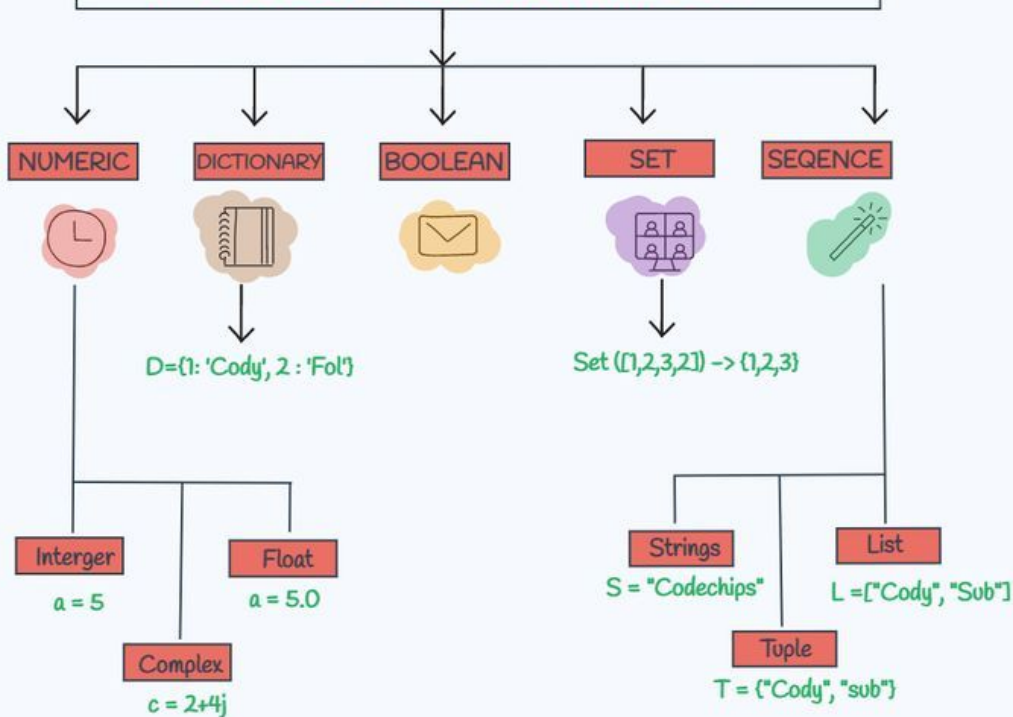
¡Bienvenidos!

Colecciones en Python





PYTHON DATA TYPES



Una de las razones del éxito de Python, es su amplia gama de estructuras de datos incorporadas, que permiten a los programadores manejar y organizar eficientemente la información en sus programas.

Entre estas estructuras, las listas, tuplas y diccionarios son fundamentales.

Listas



List in Python

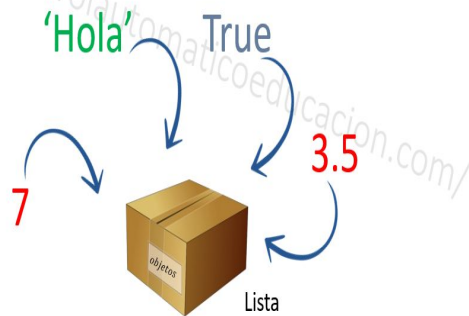
List = [10, 'Favtutor', 10, [5, 10, 15]]

↓ ↓ ↓ ↓

List[0] List[1] List[2] List[3]

- ✓ *Ordered:* Items have defined order which cannot be changed
- ✓ *Mutable:* Items can be modified anytime
- ✓ *Allow duplicates:* Items with the same value is allowed

objetos = [7, 'Hola', True, 3.5]



Ejemplo 1: Creación de listas

```
# Creating a list of numbers
numbers = [1, 2, 3, 4, 5]
# Creating a list of strings
fruits = ['apple', 'banana', 'orange']
# Creating a list of mixed data types
mixed_list = [1, 'hello', True, 3.14]
```

Ejemplo 2: Accediendo a elementos de la lista

```
# Creating a list of strings
fruits = ['apple', 'banana', 'orange']
# Accessing the first element of the list
first_element = fruits[0]
print(first_element)
# Output: 'apple'
# Accessing the last element of the list
last_element = fruits[-1]
print(last_element)
# Output: 'orange'
```

Ejemplo 3: Añadiendo o eliminando elementos

```
fruits = ['apple', 'banana', 'orange']
# Adding an element to the end of the list
fruits.append('grape')
print(fruits)
# Output: ['apple', 'banana', 'orange', 'grape']
# Removing an element by value
fruits.remove('banana')
print(fruits)
# Output: ['apple', 'orange', 'grape']
```

Ejemplo 4: Recorrer listas en bucle

```
fruits = ["apple", "banana",  
"cherry"]  
for x in fruits:  
    print(x)  
# Output:apple  
# banana  
# cherry
```

Ejemplo 5: Comprensión de listas

```
# Using list comprehension to  
create a list of squares  
squares = [x**2 for x in range(1,  
6)] print(squares)  
# Output: [1, 4, 9, 16, 25]
```

PYTHON METHODS CHEAT SHEET



   .append() →    







   .clear()

   .count() → 2

   .copy() →   

   .index() → 2

   .insert(1, ) →    

    .pop(3) →   

   .remove() →  

   .reverse() →   



Techie Programmer

©2024



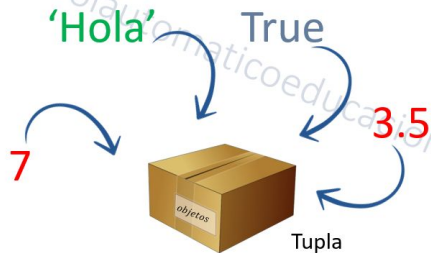
Tuplas



```
objetos = (7, 'Hola', True, 3.5)
```

7	'Hola'	True	3.5
0	1	2	3

```
objetos = (7, 'Hola', True, 3.5)
```



Python
Tupla

Definición de tupla

Una **tupla** es un conjunto **ordenado** e **inmutable** de **elementos** del mismo o diferente tipo.

Las **tuplas** se representan escribiendo los **elementos** entre paréntesis y separados por comas. Una **tupla** puede estar vacía.

Ejemplo 1: Creación de tuplas

```
colores = ("Rojo", "Negro", "Verde",  
"Azul", "Naranja")
```

```
numeros = (23, 45, 12, -4)
```

```
colores numeros = (colores, numeros, 123,  
"Violeta", "Marrón", 458)
```

Ejemplo 2: Accediendo a elementos de la tupla

Las tuplas en Python admiten métodos genéricos como *str()*, *type()* o *len()*. Por ejemplo, si ejecutamos un ***len()*** en la tupla «colores» definida anteriormente:

```
colores = ("Rojo", "Negro", "Verde",  
"Azul", "Naranja")  
print(f"El número de elementos de la  
tupla colores es: {len(colores)}")
```

Ejemplo 3: Añadiendo o eliminando elementos

```
colores = ("Rojo", "Negro", "Verde", "Azul", "Naranja")  
numeros = (23, 45, 12, -4)  
colores_numeros = (colores, numeros, 123, "Violeta", "Marrón",  
458)  
print(colores_numeros[1])
```

```
1 numeros = (23, 45, 12, -4, 12, 4, 12, 87)  
2 print(numeros.count(12))
```

```
numeros = (23, 45, 12, -4, 12, 4, 12, 87)  
print(numeros.index(88))
```



Diccionarios

Es una colección de elementos, donde cada uno tiene una llave `key` y un valor `value`. Los diccionarios se pueden crear con paréntesis `{}` separando con una coma cada par `key: value`.

```
d1 = {  
    "Nombre": "Sara",  
    "Edad": 27,  
    "Documento": 1003882  
}  
print(d1)  
#{'Nombre': 'Sara', 'Edad': 27, 'Documento': 1003882}
```


Otra forma equivalente de crear un diccionario en Python es usando `dict()` e introduciendo los pares `key: value` entre paréntesis.

```
d2 = dict([
    ('Nombre', 'Sara'),
    ('Edad', 27),
    ('Documento', 1003882),
])
print(d2)
#{'Nombre': 'Sara', 'Edad': '27', 'Documento':
'1003882'}
```

¡Nos vemos pronto!

