Tema 9: Estructuras de Datos Complejas

Lectura fácil

En Python, además de las estructuras de datos simples como números y cadenas de texto, también existen **estructuras de datos complejas** que nos permiten almacenar y organizar grandes cantidades de información. Estas estructuras son muy útiles para trabajar con conjuntos de datos, realizar operaciones de búsqueda, ordenar información y mucho más.

En este tema, aprenderemos sobre las **listas avanzadas**, los **diccionarios**, los **conjuntos** (*sets*) y las **tuplas**.

Listas avanzadas

Una **lista** en Python es una colección de elementos que pueden ser de diferentes tipos (números, cadenas de texto, otras listas, etc.). Ya hemos visto cómo crear listas, pero ahora profundizaremos en algunas operaciones más avanzadas.

Acceso a elementos y sublistas

Puedes acceder a un elemento de la lista usando su índice, y también puedes extraer partes de la lista, lo que llamamos **slicing**.

Ejemplo:

```
frutas = ["manzana", "banana", "cereza", "mango", "uva"]
print(frutas[1])  # Accede al segundo elemento
print(frutas[1:4])  # Extrae una sublista desde el
segundo al cuarto elemento
```

- El programa mostrará:
  banana
  ['banana', 'cereza', 'mango']

Modificar elementos de una lista

Puedes cambiar el valor de un elemento en una lista simplemente asignándole un nuevo valor.

Ejemplo:

```
frutas[2] = "naranja"
print(frutas)
```

- Esto reemplazará "cereza" por "naranja".
- El programa mostrará:
  ['manzana', 'banana', 'naranja', 'mango', 'uva']

Añadir y eliminar elementos

- **append()**: Añade un nuevo elemento al final de la lista.
- **insert()**: Inserta un nuevo elemento en una posición específica.
- **remove()**: Elimina un elemento específico de la lista.

Ejemplo:

```
frutas.append("piña")  # Añadir un elemento
print(frutas)

frutas.insert(1, "fresa")  # Insertar en la posición 1
print(frutas)

frutas.remove("mango")  # Eliminar "mango"
print(frutas)
```

- El programa mostrará:

```
['manzana', 'banana', 'naranja', 'mango', 'uva', 'piña']
['manzana', 'fresa', 'banana', 'naranja', 'mango', 'uva',
'piña']
['manzana', 'fresa', 'banana', 'naranja', 'uva', 'piña']
```

Listas dentro de listas

Puedes tener listas dentro de otras listas, lo que es útil para organizar datos más complejos, como una tabla o matriz.

Ejemplo:

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matriz[1][2])  # Accede al tercer elemento de la segunda fila
```

- El programa mostrará:
  6

Diccionarios

Los **diccionarios** en Python son estructuras de datos que almacenan pares de **clave**

. Son útiles cuando necesitas asociar valores con claves únicas, como un teléfono con un nombre o un producto con su precio.

Crear un diccionario

Un diccionario se define con llaves { } y cada par clave

se separa por comas.

Ejemplo:

```
telefono_contactos = {
    "Ana": "555-1234",
    "Carlos": "555-5678",
    "Laura": "555-9876"
}
```

Acceder a valores

Puedes acceder al valor asociado a una clave utilizando la clave entre corchetes [ ].

Ejemplo:

```
print(telefono_contactos["Carlos"])
```

- El programa mostrará:
  555-5678

Añadir y modificar elementos en un diccionario

Puedes añadir un nuevo par clave

o modificar el valor de una clave existente.

Ejemplo:

```
telefono_contactos["Miguel"] = "555-6543"  # Añadir nuevo
contacto
telefono_contactos["Ana"] = "555-4321"  # Modificar el
número de Ana
print(telefono_contactos)
```

- El programa mostrará:
  {'Ana': '555-4321', 'Carlos': '555-5678', 'Laura': '555-9876', 'Miguel': '555-6543'}

Eliminar elementos

Puedes eliminar una clave y su valor usando la palabra clave **del**.

Ejemplo:

```
del telefono_contactos["Carlos"]
print(telefono_contactos)
```

- El programa mostrará:
  {'Ana': '555-4321', 'Laura': '555-9876', 'Miguel': '555-6543'}

Conjuntos (set)

Un **conjunto** en Python es una colección desordenada de elementos únicos, es decir, no permite duplicados. Los conjuntos son útiles cuando quieres asegurarte de que no haya elementos repetidos en una colección.

Crear un conjunto

Un conjunto se define usando llaves `{}`.

Ejemplo:

```
numeros = {1, 2, 3, 4, 5}
print(numeros)
```

- El programa mostrará:
  {1, 2, 3, 4, 5}

Operaciones con conjuntos

Los conjuntos permiten realizar operaciones como **uniones** y **intersecciones**.

- **Union**: Combina dos conjuntos, eliminando duplicados.
- **Intersección**: Devuelve los elementos que están en ambos conjuntos.

Ejemplo:

```
conjunto1 = {1, 2, 3}
conjunto2 = {3, 4, 5}

union = conjunto1.union(conjunto2)
interseccion = conjunto1.intersection(conjunto2)

print("Unión:", union)
print("Intersección:", interseccion)
```

- El programa mostrará:
  Unión: {1, 2, 3, 4, 5}
  Intersección: {3}

Tuplas

Una **tupla** es similar a una lista, pero es **inmutable**, lo que significa que no puedes cambiar sus elementos después de haberla creado. Las tuplas se usan cuando quieres asegurarte de que los datos no cambien.

Crear una tupla

Las tuplas se definen usando paréntesis `()`.

Ejemplo:

```
dias_semana = ("Lunes", "Martes", "Miércoles", "Jueves",
"Viernes")
print(dias_semana)
```

- El programa mostrará:
  ('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes')

Acceder a elementos de una tupla

Puedes acceder a los elementos de una tupla igual que en las listas, usando su índice.

Ejemplo:

```
print(dias_semana[2])
```

- El programa mostrará:
  Miércoles

Por qué usar tuplas

Las tuplas son útiles cuando tienes un conjunto de datos que no debe cambiar, como los días de la semana o los meses del año. También pueden ser más eficientes en términos de memoria que las listas.

Ejemplos prácticos con estructuras de datos

Ejemplo 1: Recorrer un diccionario con un bucle

```
telefono_contactos = {
    "Ana": "555-1234",
    "Carlos": "555-5678",
    "Laura": "555-9876"
}

for nombre, telefono in telefono_contactos.items():
    print(nombre, ":", telefono)
```

- El programa mostrará:

```
Ana : 555-1234
Carlos : 555-5678
Laura : 555-9876
```

Ejemplo 2: Comprobar si un elemento está en un conjunto

```
frutas = {"manzana", "banana", "cereza"}

if "banana" in frutas:
    print("La banana está en el conjunto de frutas.")
else:
    print("La banana no está en el conjunto.")
```

- El programa mostrará:
  La banana está en el conjunto de frutas.

Las **estructuras de datos complejas** como las **listas**, **diccionarios**, **conjuntos** y **tuplas** son herramientas poderosas que te permiten organizar y manejar grandes

cantidades de información de manera eficiente en Python. Cada una tiene sus características únicas y se usan en diferentes situaciones según lo que necesites hacer.