Clase 07: Tuplas y Conjuntos

Parte 1. Tuplas: Inmutabilidad, Creación y Acceso

1.1 Definición

Una **tupla** es una estructura de datos en Python que permite almacenar una colección **ordenada** e **inmutable** de elementos. A diferencia de las listas, una vez creada una tupla, no se pueden modificar sus elementos (no se pueden agregar, eliminar ni cambiar).

Características principales:

- **Inmutables**: No se pueden modificar después de su creación
- **Ordenadas**: Mantienen el orden de inserción
- **Permiten duplicados**: Pueden contener elementos repetidos
- **Heterogéneas**: Pueden almacenar diferentes tipos de datos
- **Indexables**: Se accede a sus elementos mediante índices (desde 0)

1.2 Sintaxis

```
1 # Creación con paréntesis
2 tupla = ("elemento1", "elemento2", "elemento3")
```

```
1 # Creación sin paréntesis (empaquetado)
2 tupla = "elemento1", "elemento2", "elemento3"
```

```
1 # Tupla vacía
2 tupla_vacia = ()
```

```
1 # Tupla de un solo elemento (requiere coma)
2 tupla_unitaria = ("elemento",)
```

```
1 # Creación con el constructor tuple()
2 tupla = tuple([1, 2, 3])
```

```
1 # Acceso por índice
2 elemento = tupla[0]  # Primer elemento
3 elemento = tupla[-1]  # Últi  elemento
```

```
# Slicing (rebanado)
sub_tupla = tupla[0:2:1]

print(tupla)
```

1.3 Ejemplos Básicos

Ejemplo 1: Coordenadas geográficas

```
# Almacenar las coordenadas de una ciudad (latitud, longitud)
madrid = (40.4168, -3.7038)

print(f"Coordenadas de Madrid: {madrid}")
print(f"Latitud: {madrid[0]}")
print(f"Longitud: {madrid[1]}")

# Desempaquetado
latitud, longitud = madrid
print(f"Latitud: {latitud}, Longitud: {longitud}")
```

Ejemplo 2: Información de estudiante

```
# Tupla con datos de un estudiante (nombre, edad, calificación)
estudiante = ("Ana García", 22, 9.5)

nombre, edad, calificacion = estudiante

print(f"Estudiante: {nombre}")
print(f"Edad: {edad} años")
print(f"Calificación: {calificacion}")

# Intentar modificar la tupla (esto genera error)
# estudiante[2] = 10  # TypeError: 'tuple' object does not support item ass
```

Ejemplo 3: Días de la semana

```
# Tupla inmutable de días laborables
dias_laborables = ("Lunes", "Martes", "Miércoles", "Jueves", "Viernes")

print(f"Total de días laborables: {len(dias_laborables)}")
print(f"Primer día: {dias_laborables[0]}")
print(f"Último día: {dias_laborables[-1]}")

# Verificar si un día está en la tupla
dia = "Sábado"
if dia in dias_laborables:
    print(f"{dia} es laborable")
else:
    print(f"{dia} no es laborable")
```

```
14
15 # Slicing: obtener días del medio de la semana
16 medio_semana = dias_laborables[1:4]
17 print(f"Días del medio de semana: {medio_semana}")
```

1.4 Ejercicios Propuestos

Ejercicio 1: Datos de producto

Crea una tupla que almacene información de un producto: nombre, precio y cantidad en stock. Luego:

- Imprime cada dato por separado
- Usa desempaquetado para extraer los valores
- Calcula el valor total del inventario (precio × cantidad)

```
1 # Tu código aquí
2
```

Ejercicio 2: Colores RGB

Crea una tupla con los valores RGB de un color (por ejemplo, rojo = (255, 0, 0)). Luego:

- Accede a cada componente (rojo, verde, azul)
- Verifica si algún componente es mayor a 200
- Imprime el color en formato "RGB(r, g, b)"

```
1 # Tu código aquí
2
```

Ejercicio 3: Ranking de puntuaciones

Crea una tupla con las 5 puntuaciones más altas de un videojuego. Luego:

- Muestra la puntuación más alta y la más baja del top 5
- Obtén las 3 primeras puntuaciones usando slicing
- Verifica si una puntuación específica está en el ranking

```
1 # Tu código aquí
2
```

Parte 2. Conjuntos (set): Eliminación de Duplicados y Operaciones Básicas

2.1 Definición

Un **conjunto (set)** es una estructura de datos en Python que almacena una colección **no ordenada** de elementos **únicos**. Los conjuntos son ideales para eliminar duplicados y realizar operaciones matemáticas como unión, intersección y diferencia.

Características principales:

- **Sin duplicados**: Cada elemento aparece solo una vez
- **No ordenados**: No mantienen orden de inserción
- **Mutables**: Se pueden agregar o eliminar elementos
- **No indexables**: No se puede acceder por índice
- **Elementos inmutables**: Solo pueden contener tipos de datos inmutables (int, str, tuple, etc.)

2.2 Sintaxis

```
# Creación con llaves
conjunto = {1, 2, 3}
```

```
# Conjunto vacío (debe usar constructor)
conjunto_vacio = set()
```

```
# Creación desde una lista (elimina duplicados)
conjunto = set([1, 2, 2, 3, 3, 3])  # Resultado: {1, 2, 3}
print(f"Conjunto desde lista: {conjunto}")
```

```
# Agregar elementos
conjunto.add(4)
print(f"Después de agregar 4: {conjunto}")
```

```

# Eliminar elementos
conjunto.discard(2)  # No genera error si no existe
print(f"Después de eliminar 2: {conjunto}")
```

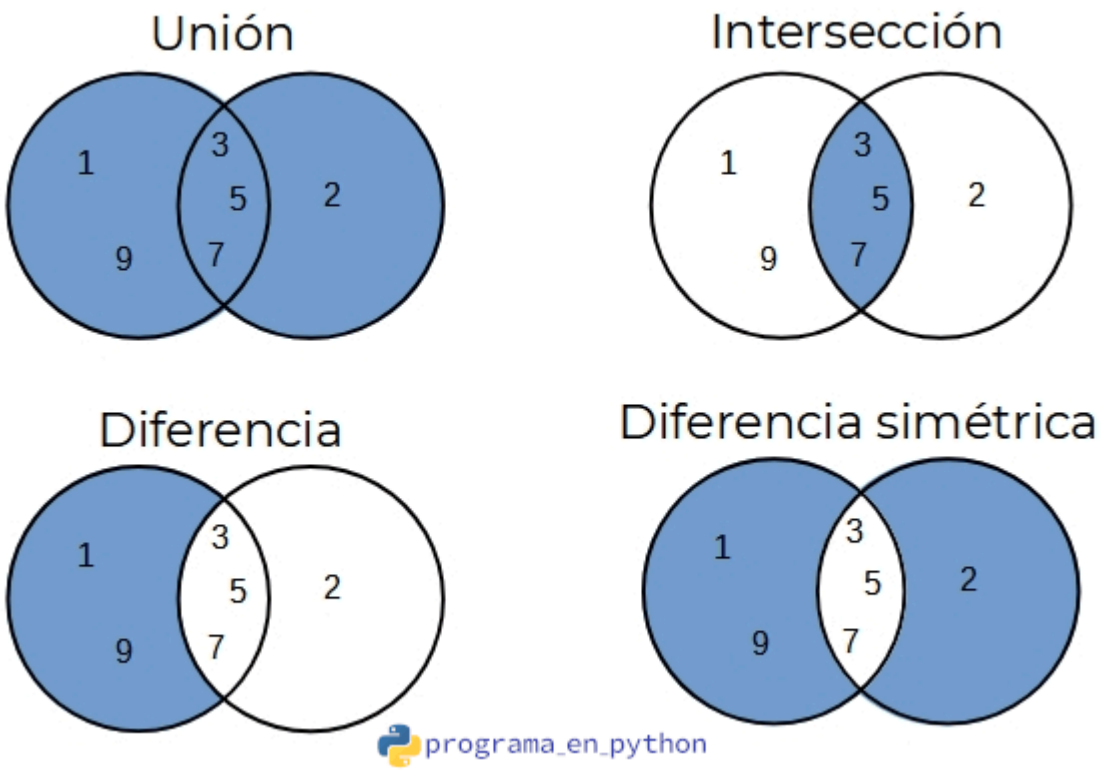
```
# Operaciones entre conjuntos
conjunto1 = {1, 2, 3}
conjunto2 = {3, 4, 5}

union = conjunto1 | conjunto2
interseccion = conjunto1 & conjunto2
diferencia = conjunto1 - conjunto2
diferencia_simetrica = conjunto1 ^ conjunto2

print(f"Unión: {union}")
print(f"Intersección: {interseccion}")
```

```
print(f"Diferencia: {diferencia}")
print(f"Diferencia simétrica: {diferencia_simetrica}")
```

Union	Interseccion	Diferencia	Diferencia Simetrica
			
Las personas que cuenten con un medio de transporte: Auto o Biciclieta	Las personas que cuenten Auto y Biciclieta	Las personas que cuenten Bicicleta pero no con Auto	Las personas que cuentan con Auto o Bicicleta, pero no los dos
Hugo,Paco, Luis	Paco	Hugo	Hugo, Luis



2.3 Ejemplos Básicos

Ejemplo 1: Eliminar números duplicados

```
# Lista con números duplicados
numeros = [1, 2, 2, 3, 4, 4, 4, 5, 1, 3]

print(f"Lista original: {numeros}")
print(f"Total de elementos: {len(numeros)}")
```

```
6
7 # Convertir a conjunto para eliminar duplicados
8 numeros_unicos = set(numeros)
9
10 print(f"Números únicos: {numeros_unicos}")
11 print(f"Total de únicos: {len(numeros_unicos)}")
12
13 # Convertir de nuevo a lista ordenada
14 lista_limpia = sorted(numeros_unicos)
15 print(f"Lista limpia y ordenada: {lista_limpia}")
```

Ejemplo 2: Operaciones entre conjuntos de estudiantes

```
1 # Estudiantes inscritos en diferentes cursos
2 python = {"Ana", "Luis", "María", "Carlos", "Elena"}
3 javascript = {"María", "Pedro", "Carlos", "Sofía"}
4
5 print(f"Estudiantes de Python: {python}")
6 print(f"Estudiantes de JavaScript: {javascript}")
7
8 # Estudiantes en ambos cursos (intersección)
9 ambos_cursos = python & javascript
10 print(f"Estudiantes en ambos cursos: {ambos_cursos}")
11
12 # Todos los estudiantes (unión)
13 todos = python | javascript
14 print(f"Total de estudiantes: {todos}")
15
16 # Solo en Python (diferencia)
17 solo_python = python - javascript
18 print(f"Solo en Python: {solo_python}")
19
20 # En solo uno de los cursos (diferencia simétrica)
21 solo_uno = python ^ javascript
22 print(f"En solo un curso: {solo_uno}")
```

Ejemplo 3: Gestión dinámica de un conjunto

```
1 # Conjunto de frutas favoritas
2 frutas = {"manzana", "banana", "naranja"}
3 print(f"Frutas iniciales: {frutas}")
4
5 # Agregar una fruta
6 frutas.add("uva")
7 print(f"Después de agregar uva: {frutas}")
8
9 # Intentar agregar duplicado (no tendrá efecto)
10 frutas.add("manzana")
11 print(f"Después de intentar agregar manzana duplicada: {frutas}")
12
13 # Eliminar una fruta
14 frutas.discard("banana")
```

```
print(f"Después de eliminar banana: {frutas}")

# Verificar si una fruta está en el conjunto
if "naranja" in frutas:
    print("La naranja está en el conjunto")

# Número de elementos
print(f"Total de frutas: {len(frutas)}")
```

2.4 Ejercicios Propuestos

Ejercicio 1: Letras únicas en una palabra

Crea un programa que reciba una palabra y muestre:

- Todas las letras únicas que contiene
- El número de letras únicas
- Las vocales que aparecen en la palabra

```
# Tu código aquí

```

Ejercicio 2: Comparación de listas de compras

Tienes dos listas de compras (semana pasada y esta semana). Usando conjuntos:

- Encuentra los productos que compraste ambas semanas
- Encuentra productos que solo compraste la semana pasada
- Encuentra todos los productos diferentes que has comprado

```
# Tu código aquí

```

Ejercicio 3: Tags de artículos

Crea dos conjuntos con tags (etiquetas) de dos artículos de blog. Luego:

- Encuentra los tags comunes entre ambos artículos
- Encuentra tags que tiene el primer artículo pero no el segundo
- Crea un conjunto con todos los tags únicos

```
# Tu código aquí

```

---

3. Caso de Uso 1: Sistema de Gestión de Coordenadas

📍 Problema

Desarrolla un sistema para gestionar ubicaciones de puntos de interés en una ciudad. Cada ubicación debe almacenarse como una tupla inmutable con formato `(nombre, latitud, longitud, tipo)`. El sistema debe permitir:

1. Registrar nuevas ubicaciones
2. Buscar ubicaciones por nombre
3. Calcular la distancia entre dos ubicaciones (usando fórmula simplificada)
4. Listar todas las ubicaciones de un tipo específico (ej: "restaurante", "museo", "parque")
5. Obtener la ubicación más cercana a unas coordenadas dadas

💡 Requisitos

- Usa tuplas para almacenar cada ubicación (aprovechar inmutabilidad)
- Implementa al menos 5 ubicaciones de prueba
- Crea funciones para cada operación solicitada
- Maneja casos donde no se encuentre una ubicación

🎯 Funcionalidades esperadas

```
# Ejemplo de uso esperado:
ubicaciones = [
    ("Museo del Prado", 40.4138, -3.6921, "museo"),
    ("Restaurante Botín", 40.4141, -3.7077, "restaurante"),
    # ... más ubicaciones
]
```

```
1 # Tu código aquí
2
```

---

4. Caso de Uso 2: Análisis de Datos de Encuestas

📊 Problema

Desarrolla un programa que analice los resultados de encuestas sobre preferencias de lunguajes de programación de diferentes grupos de personas. Debes usar conjuntos para:

1. Identificar preferencias únicas en cada grupo
2. Encontrar preferencias comunes entre todos los grupos
3. Detectar preferencias exclusivas de cada grupo
4. Encontrar preferencias que aparecen en al menos 2 grupos

5. Generar estadísticas sobre diversidad de opiniones

💡 Requisitos

- Simula al menos 3 grupos diferentes (ej: Grupo A, B, C)
- Cada grupo tiene una lista de preferencias que puede contener duplicados
- Usa operaciones de conjuntos (unión, intersección, diferencia, etc.)
- Presenta resultados de forma clara y organizada
- Calcula porcentajes de coincidencia entre grupos

🎯 Funcionalidades esperadas

```
# Ejemplo de estructura:
grupo_a = ["opción1", "opción2", "opción1", "opción3"]
grupo_b = ["opción2", "opción4", "opción5"]
grupo_c = ["opción1", "opción2", "opción6"]

# Análisis esperado:
- Preferencias únicas totales
- Preferencias compartidas por todos
- Preferencias exclusivas de cada grupo
- Nivel de consenso (% de coincidencia)
- Grupo con mayor diversidad de opiniones
```

```
1 # Tu código aquí
2
```