Universidad de Guadalajara

CENTRO UNIVERSITARIO DE CIENCIAS ECONOMICO ADMINISTRATIVAS

Asignatura: Analisis de datos 2026-B

Profesor: Jose Antonio Origaza Trejo

Alumno: Martínez Velarde Jesús Santiago

Actividad 5: Este cuaderno contiene ejemplos prácticos y teóricos sobre Listas, Tuplas, Diccionarios y Conjuntos.

Listas

Las listas son las secuencias mutables.

- Los elementos tienen un orden definido (iniciando con el primero como 0).
- **Mutables:** Quiere decir que se le pueden agregar o eliminar elementos; son modificables una vez hayan sido declaradas.
- No necesitan ser del mismo tipo; una lista puede contener números, texto o incluso otras listas mezcladas.

In [41]:
```
# --- LISTAS  ---
#Se pueden mezclar tipos de datos (texto, numeros y booleanos)
mi_mochila = ["Laptop", 2, "Libros", True]
```

In [42]:
```
#Imprimir lista
print("Mi mochila tiene:", mi_mochila)
```

```
Mi mochila tiene: ['Laptop', 2, 'Libros', True]
```

In [43]:
```
#Acceder a un elemento
primer_objeto = mi_mochila[0]  # Accede a "Laptop"
print("El primer objeto es:", primer_objeto)
```

```
El primer objeto es: Laptop
```

In [44]:
```
# Modificar la lista (Mutabilidad)
# Cambiamos el segundo elemento (índice 1) que es el número 2
mi_mochila[1] = 5
print("Mochila actualizada (numero):", mi_mochila)
```

```
Mochila actualizada (numero): ['Laptop', 5, 'Libros', True]
```

In [45]:
```
#Para agregar un numero elemento AL FINAL DE LA LISTA se usa la funcion .append()
mi_mochila.append("Botella de agua")
print("Mochila final:", mi_mochila)
```

```
Mochila final: ['Laptop', 5, 'Libros', True, 'Botella de agua']
```

In [46]:
```
#Para agregar un elemento en una posición en especifico se usa .insert()
#Los demas elementos se recorren un lugare a la derecha
mi_mochila.insert(0, "Llaves de casa")
print("Después de insertar al inicio:", mi_mochila)

#Insertamos celular en el indice 2 (despues de laptop)
mi_mochila.insert(2, "Celular")
print("Después de insertar en medio:", mi_mochila)
```

```
Después de insertar al inicio: ['Llaves de casa', 'Laptop', 5, 'Libros', True, 'Bote
lla de agua']
Después de insertar en medio: ['Llaves de casa', 'Laptop', 'Celular', 5, 'Libros', T
rue, 'Botella de agua']
```

Tuplas

Las tuplas son secuencias inmutables, se utilizan para guardar multiples elementos en una sola variables

- Los elementos tienen un orden definido (iniciando con el primero como 0).
- **inmutables:** Quiere decir que NO SE PUEDEN MODIFICAR, son estaticas de solo lectura
- Son ideales para datos que no deberían cambiar durante la ejecución del programa, como coordenadas

In [47]:
```
# --- TUPLAS  ---
#Declaración de una tupla
coordenadas_gdl = (20.6597, -103.3496)
colores_primarios = ("Rojo", "Azul", "Amarillo")
```

In [48]:
```
#Imprimir
print("Ubicación:", coordenadas_gdl)
print(f"Tipo de dato: {type(coordenadas_gdl)}")
```

```
Ubicación: (20.6597, -103.3496)
Tipo de dato: <class 'tuple'>
```

In [49]:
```
#DEMOSTRACION QUE NO SON INMUTABLES
#Si se ejecuta esta linea, muestra un error.
#  coordenadas_gdl[0] = 19.4326  <-- ¡Esto no se permite!
```

In [50]:
```
#Desempaquetado
#Una caracteristica de las tuplas es que se le puede asignar sus variables a variab
lat, long = coordenadas_gdl
print(f"Latitud separada: {lat}")
print(f"Longitud separada: {long}")
```

```
Latitud separada: 20.6597
Longitud separada: -103.3496
```

Diccionarios

Definición oficial: Los diccionarios son una colección **desordenada**, **modificable** e **indexada**. En Python, los diccionarios se escriben con llaves `{}` y tienen claves y valores. A diferencia de las Listas y Tuplas que usan índices numéricos (0, 1, 2...), los diccionarios usan **Claves (Keys)**.

- **Clave-Valor:** Cada elemento se guarda como un par `clave : valor`.
- **Mutables:** Podemos cambiar el valor asociado a una clave en cualquier momento.
- **Sin duplicados:** No puede haber dos claves iguales (si repites una, la segunda sobrescribe a la primera).
- **Sintaxis:** `{"marca": "Ford", "modelo": "Mustang"}`

In [51]:
```python
#Declaración de un diccionario
#Representación de un alumno y sus datos.
alumno = {
    "nombre": "Santiago",      # Clave: "nombre", Valor: "Santiago"
    "edad": 22,
    "carrera": "Economía",
    "promedio": 9.5
}
```

In [52]:
```python
print("Datos del alumno:", alumno)
```

```
Datos del alumno: {'nombre': 'Santiago', 'edad': 22, 'carrera': 'Economía', 'promedi
o': 9.5}
```

In [53]:
```python
#Acceder a un valor, en este caso la carrera, se usa la clave como si fuera un indi
carrera_actual = alumno["carrera"]
print("El alumno estudia:", carrera_actual)
```

```
El alumno estudia: Economía
```

In [54]:
```python
#Modificar valores
alumno["edad"] = 23
alumno["promedio"] = 9.8
print("Datos actualizados:", alumno)
```

```
Datos actualizados: {'nombre': 'Santiago', 'edad': 23, 'carrera': 'Economía', 'prome
dio': 9.8}
```

In [55]:
```python
#Agregar claves y valores al diccionario
alumno["semestre"] = "6to"
print("Información alumno:", alumno)
```

```
Información alumno: {'nombre': 'Santiago', 'edad': 23, 'carrera': 'Economía', 'prome
dio': 9.8, 'semestre': '6to'}
```

In [56]:
```python
# Se puede obtener solo las claves o solo los valores
print("Claves disponibles:", alumno.keys())
print("Valores guardados:", alumno.values())
```

```
Claves disponibles: dict_keys(['nombre', 'edad', 'carrera', 'promedio', 'semestre'])
Valores guardados: dict_values(['Santiago', 23, 'Economía', 9.8, '6to'])
```

Hay una opcion para obntener una clave y no recibir un error si esta no existe, es el metodo llamado .get("clave") que es más seguro, pero con la sintaxis de corchetes ["clave"]

4. Conjuntos (Sets)

Un conjunto es una colección **desordenada** y **sin índice**. Su punto es la **unicidad**. Los conjuntos no permiten elementos duplicados. Si intentas agregar el mismo dato dos veces, Python simplemente ignorará la segunda vez.

- **Desordenados:** Los elementos no tienen un orden fijo. No le puede indicar "dame el elemento 1" porque su posición cambia.
- **No duplicados:** Garantiza que cada elemento sea único.

In [57]:
```
# --- CONJUNTOS  ---

#Ejemplo de una lsta de invitados donde hay dos datos repetidos "Ana" "Pedro"
invitados_lista = ["Ana", "Pedro", "Luis", "Ana", "Pedro", "Sofía"]
invitados_set = set(invitados_lista) #Elimina duplicados
```

In [58]:
```
# Lista de invitados con duplicados / sin duplicados
print("Lista original (sucia):", invitados_lista)
print("Conjunto limpio (sin repetidos):", invitados_set)
```

```
Lista original (sucia): ['Ana', 'Pedro', 'Luis', 'Ana', 'Pedro', 'Sofía']
Conjunto limpio (sin repetidos): {'Luis', 'Pedro', 'Sofía', 'Ana'}
```

In [59]:
```
#Para agregar elementos usamos .add()
invitados_set.add("Carlos")
print("Con nuevo invitado:", invitados_set)
```

```
Con nuevo invitado: {'Pedro', 'Luis', 'Ana', 'Carlos', 'Sofía'}
```

In [60]:
```
#Verificar pertenencia, es decir, si un elemento existe en el conjunto
print("¿Está Luis invitado?", "Luis" in invitados_set)
print("¿Está María invitada?", "María" in invitados_set)
```

```
¿Está Luis invitado? True
¿Está María invitada? False
```

In [61]:
```
#intersección
amigos_juan = {"Ana", "Pedro", "Luis"}
amigos_maria = {"Pedro", "Sofía", "Carlos"}
# Validar amigos en comun entre dos conjuntos
comunes = amigos_juan.intersection(amigos_maria)
print("Amigos en común:", comunes)

# Unión, no duplicados.
todos = amigos_juan.union(amigos_maria)
print("Todos los amigos juntos:", todos)
```

```
Amigos en común: {'Pedro'}
Todos los amigos juntos: {'Carlos', 'Sofía', 'Pedro', 'Luis', 'Ana'}
```

In [ ]: