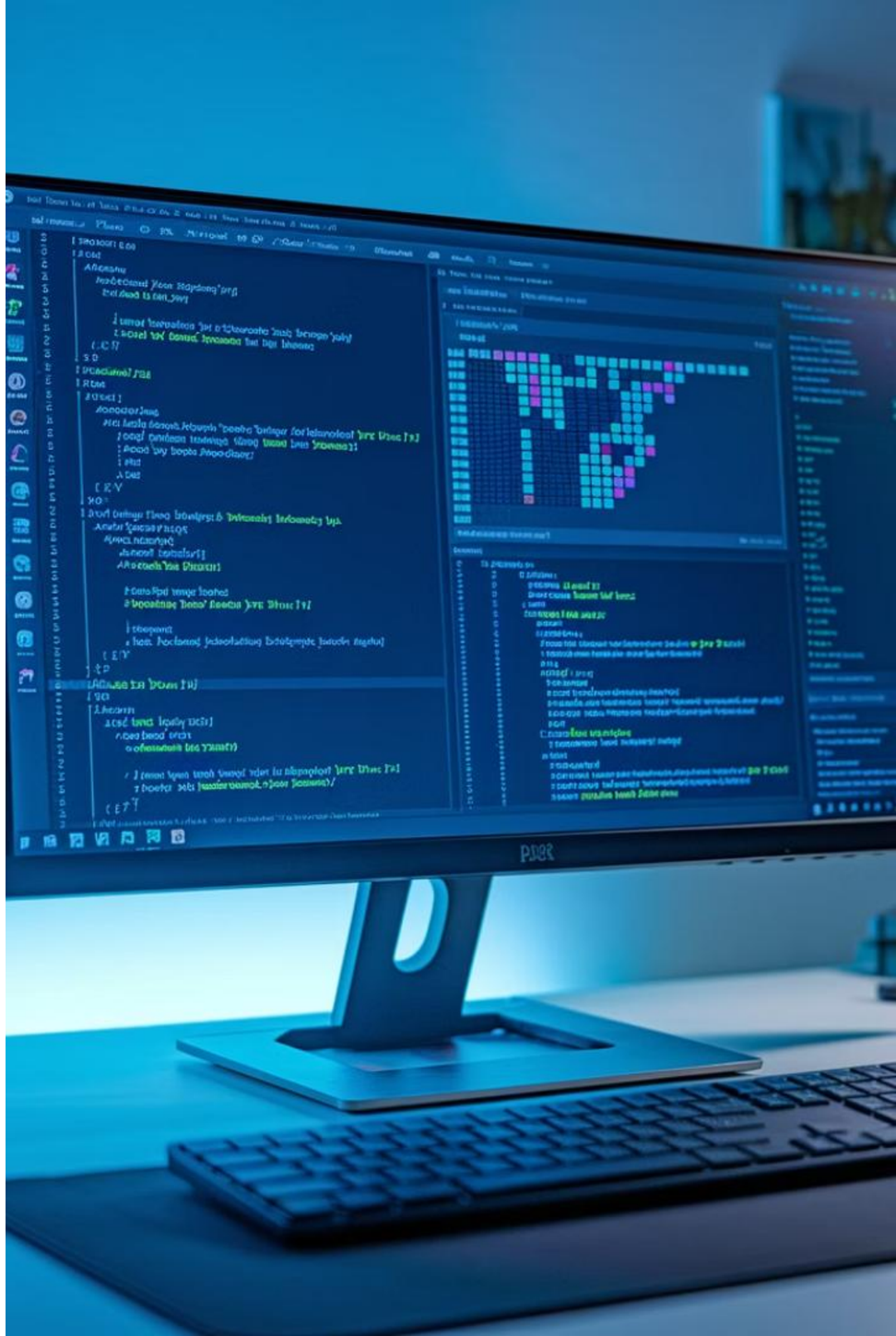
Estructuras de Datos y Sentencias Iterativas en Python

En esta presentación exploraremos las estructuras de datos fundamentales en Python y cómo trabajar con ellas mediante sentencias iterativas. Aprenderemos a organizar información de manera eficiente utilizando listas, diccionarios, tuplas y sets, así como a recorrer estos datos con bucles while y for.

Cada estructura tiene características únicas que la hacen más adecuada para ciertas tareas. Comprenderlas y saber cuándo utilizarlas es esencial para escribir código limpio, claro y eficiente.

por Kibernum Capacitación

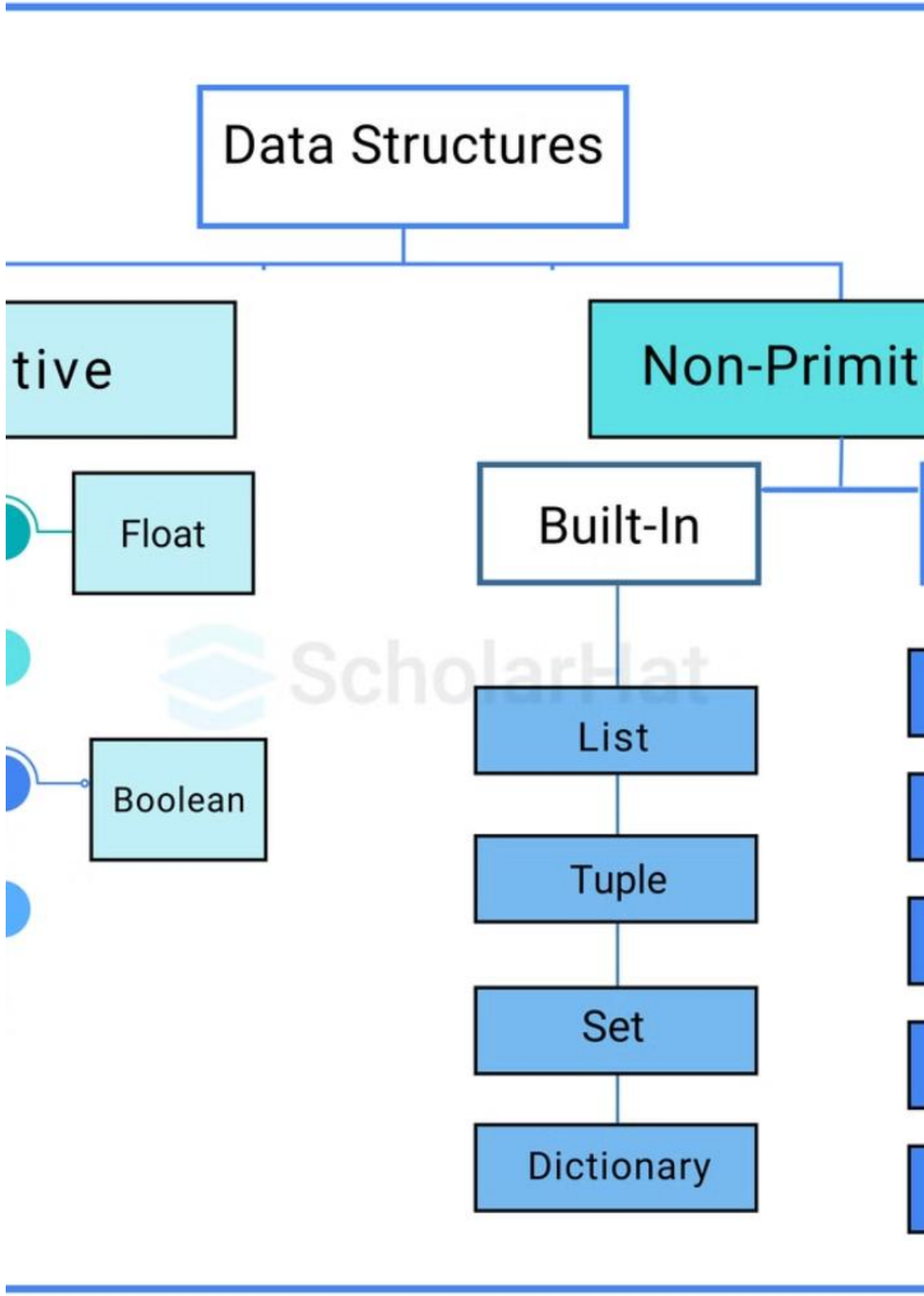


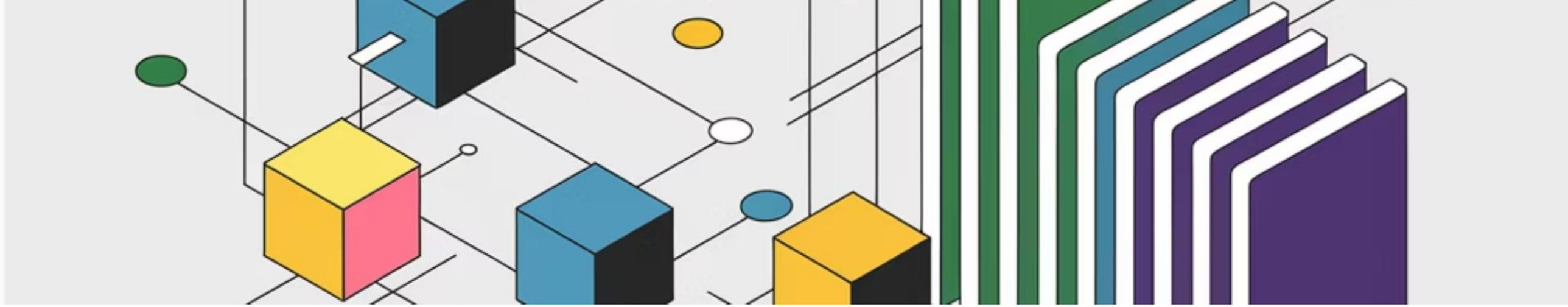
Preguntas de Activación de Contenidos

¿Has utilizado antes listas o diccionarios en otros lenguajes de programación? ¿En qué contexto?

¿Qué ventajas crees que tiene usar estructuras como sets o tuplas en lugar de listas?

¿Qué tareas repetitivas en el mundo real podrían automatizarse con una estructura iterativa?





¿Qué son las Estructuras de Datos?

Definición

Una estructura de datos es una forma de organizar y almacenar información para que pueda ser utilizada de manera eficiente en programación.

Importancia

Permiten trabajar con diferentes tipos de colecciones: secuencias, asociaciones clave-valor, agrupaciones inmutables y conjuntos no ordenados.

En Python

Python proporciona varias estructuras incorporadas que facilitan el manejo de datos: listas, diccionarios, tuplas y sets.

Listas en Python

Características

- Colección ordenada de elementos
- Mutable (modificable después de creada)
- Permite duplicados
- Usa corchetes [] para su declaración

Las listas son una de las estructuras más utilizadas gracias a su versatilidad. Pueden contener números, cadenas, otras listas e incluso objetos personalizados.

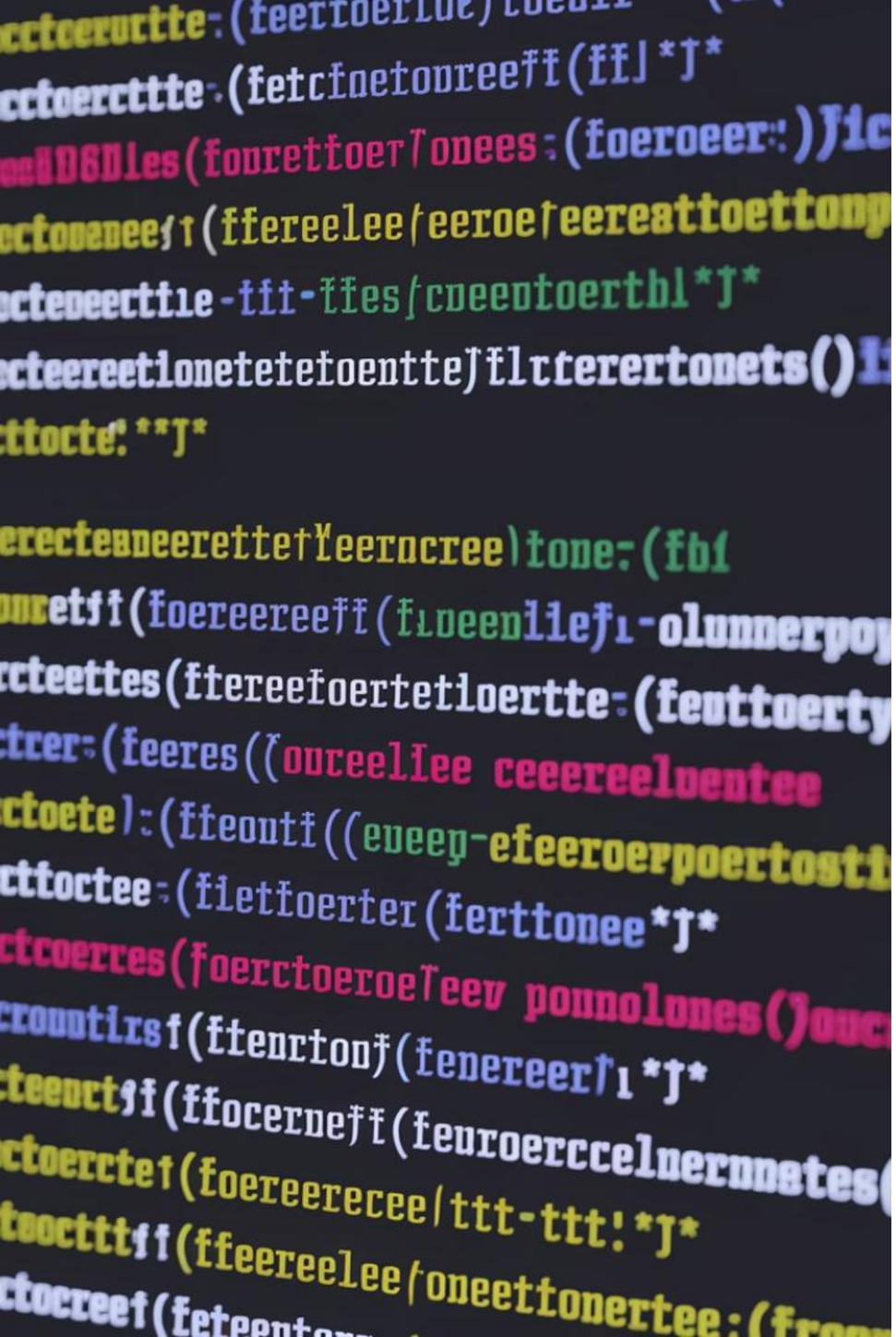
Ejemplos Básicos

Crear una lista:

```
frutas = ['manzana', 'banana', 'naranja']
print(frutas)
```

Acceder a elementos (índice desde 0):

```
print(frutas[0])     # manzana
print(frutas[-1])    # naranja (último)
```



Operaciones con Listas

Agregar elementos

```
frutas.append('pera')        # Agrega al final
frutas.insert(1, 'uva')      # Inserta en la posición 1
```

Modificar elementos

```
frutas[0] = 'durazno'
```

Eliminar elementos

```
matriz = [[1, 2], [3, 4], [5, 6]]
print(matriz[1][0])  # 3
```

Listas anidadas (matrices)

```
matriz = [[1, 2], [3, 4], [5, 6]]
print(matriz[1][0])  # 3
```

Diccionarios en Python

Colección de pares clave-valor

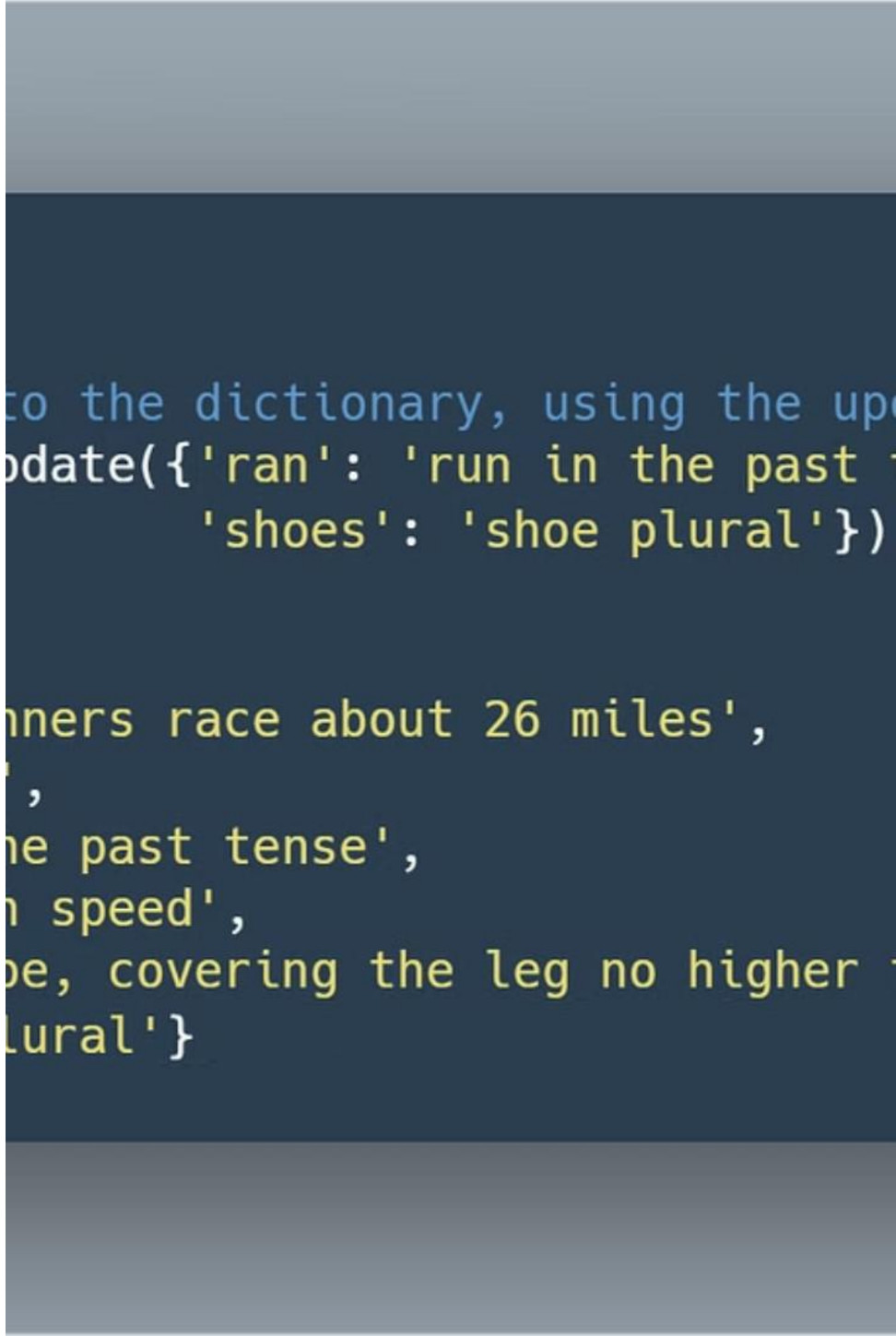
Útil cuando necesitas acceder a información por nombre o etiqueta en lugar de un índice numérico.

Características clave

No ordenado (hasta Python 3.6), clave única e inmutable, estructura mutable, usa llaves {}.

Ejemplo básico

```
persona = {'nombre': 'Ana', 'edad': 30}
```



Operaciones con Diccionarios

Acceder a valores

```
print(persona['nombre'])      # Ana
print(persona.get('edad'))    # 30
```

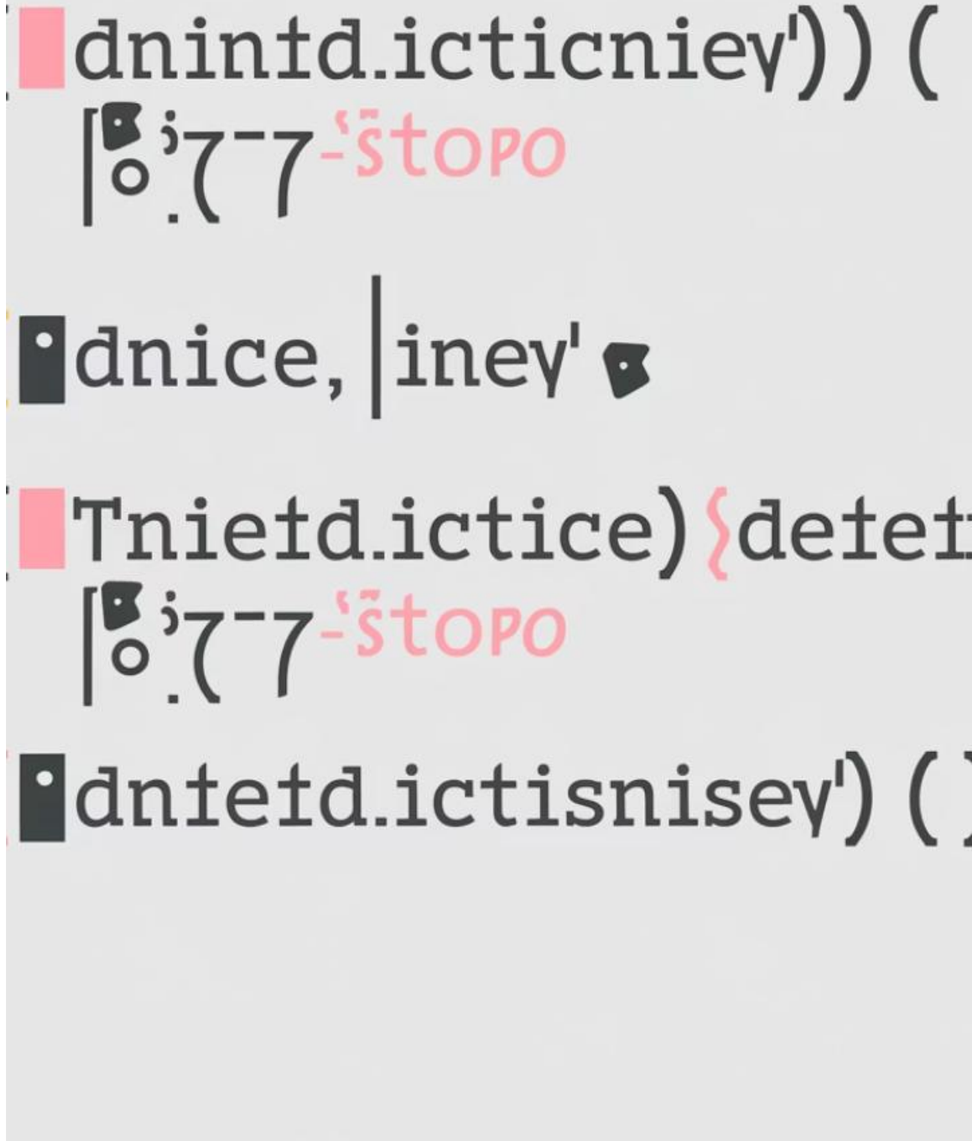
Si la clave no existe, get() devuelve None o un valor predeterminado, mientras que usar corchetes lanzará un error.

Agregar/modificar claves

```
persona['ciudad'] = 'Santiago'
persona['edad'] = 31
```

Eliminar claves

```
del persona['nombre']
```



Diccionarios Anidados

Estructura

Permiten almacenar diccionarios dentro de otros diccionarios, creando estructuras de datos jerárquicas.

Aplicaciones

Ideal para representar datos jerárquicos como organizaciones, catálogos o configuraciones complejas.

Ejemplo

```
alumno = {
    'nombre': 'Carlos',
    'notas': {'matemáticas': 6.5, 'historia': 5.8}
}
print(alumno['notas']['historia'])  # 5.8
```

Tuplas en Python

¿Qué es una tupla?

Una tupla es similar a una lista, pero inmutable. Es útil para almacenar datos que no deberían cambiar durante la ejecución del programa, como coordenadas o constantes.

Características clave

- Ordenada
- Inmutable (no se puede modificar)
- Permite duplicados
- Usa paréntesis ()

Ejemplos prácticos

Crear una tupla:

```
coordenadas = (10.5, 20.3)
```

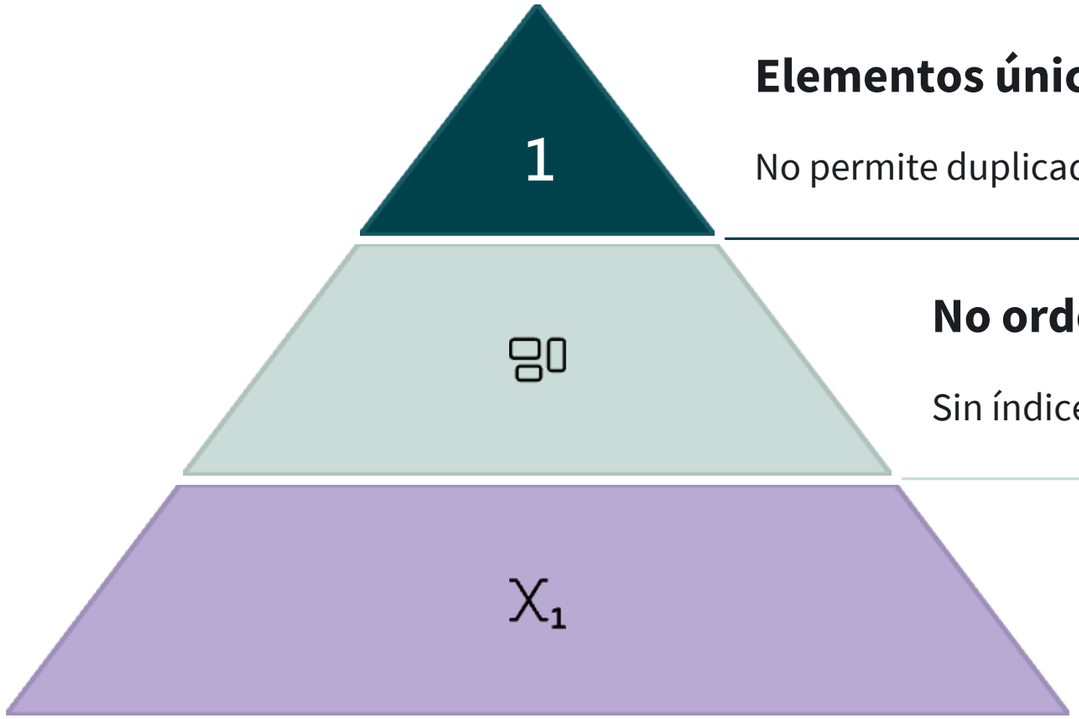
Acceder a elementos:

```
print(coordenadas[0])  # 10.5
```

Empaquetado y desempaquetado:

```
persona = ('Luis', 28, 'Chile')
nombre, edad, pais = persona
print(nombre)  # Luis
```

Sets (Conjuntos) en Python



Elementos únicos

No permite duplicados

No ordenado

Sin índices de acceso

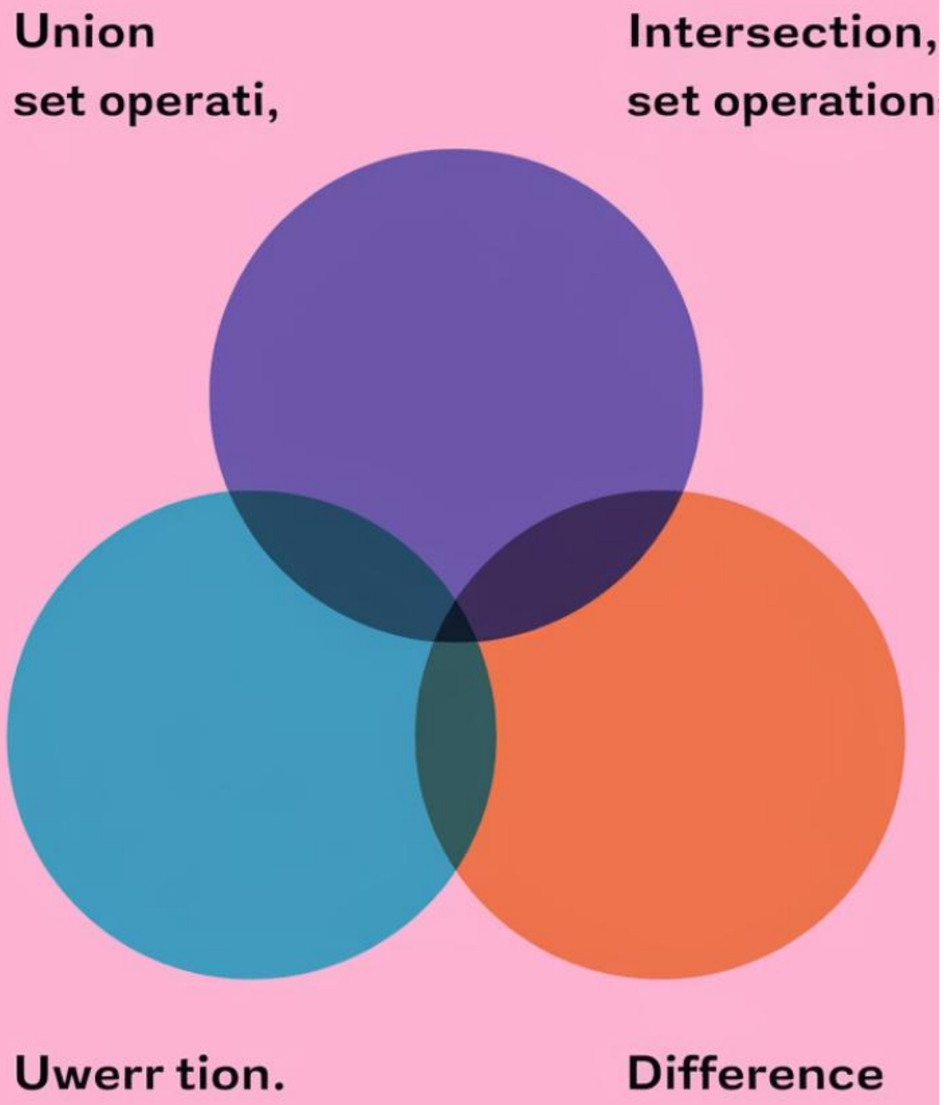
Mutable

Se puede modificar

Un set es una colección no ordenada de elementos únicos. Es útil para realizar operaciones matemáticas de conjuntos como unión, intersección y diferencia. Se declara usando llaves {} o la función set().

Ejemplo:

```
colores = {'rojo', 'verde', 'azul'}
```



Operaciones con Sets

Agregar elementos

```
colores.add('amarillo')
```

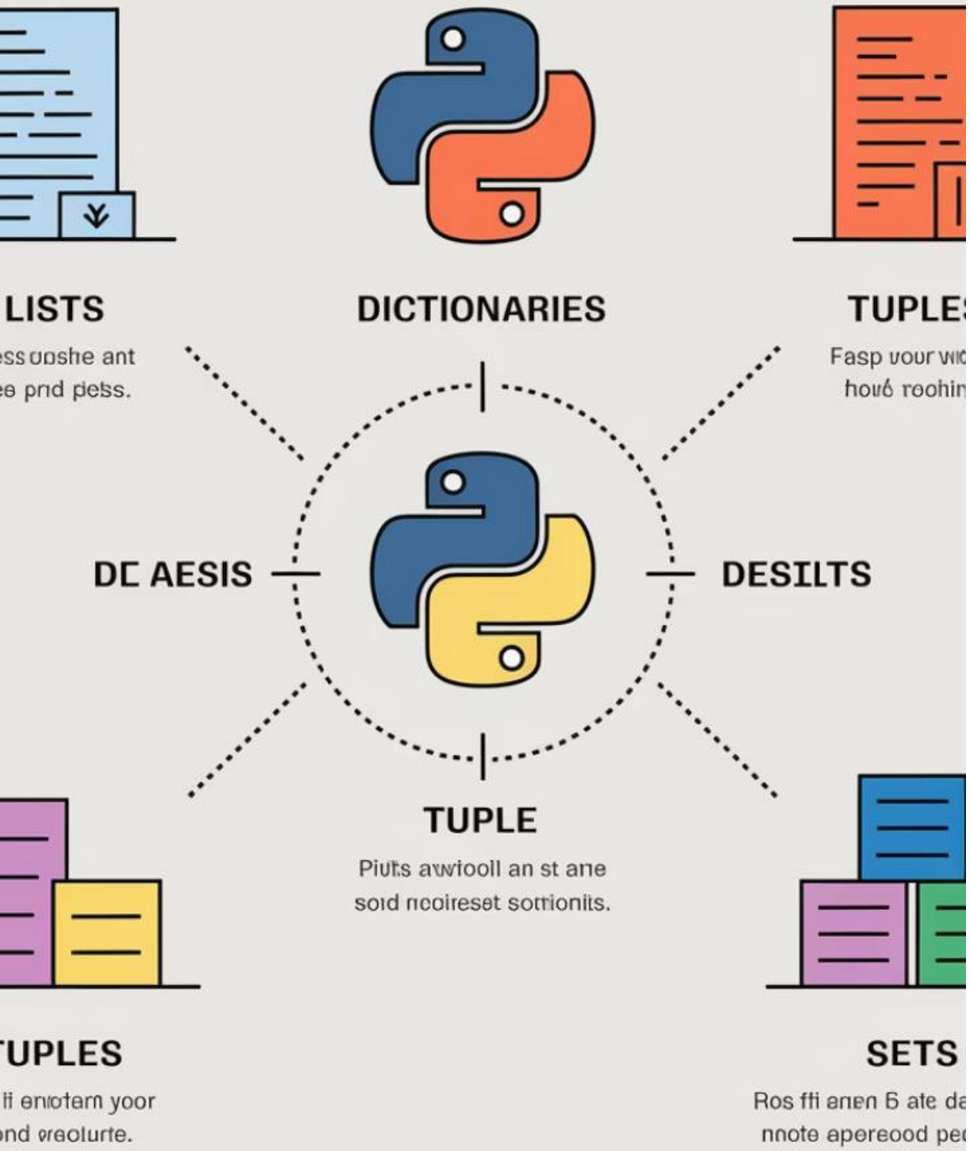
Eliminar elementos

```
colores.discard('verde')
```

Operaciones de de conjuntos

```
a = {1, 2, 3}
b = {3, 4, 5}

print(a | b)   # Unión → {1, 2, 3, 4, 5}
print(a & b)   # Intersección → {3}
print(a - b)   # Diferencia → {1, 2}
```



Comparativa de Estructuras de Datos

Estructura	Ordenada	Mutable	Permite duplicados	Sintaxis
Lista	✅	✅	✅	[]
Diccionario	❌ (3.6-)	✅	❌ (claves únicas)	{}
Tupla	✅	❌	✅	()
Set	❌	✅	❌	{} / set()

Sentencias Iterativas en Python

¿Qué son?

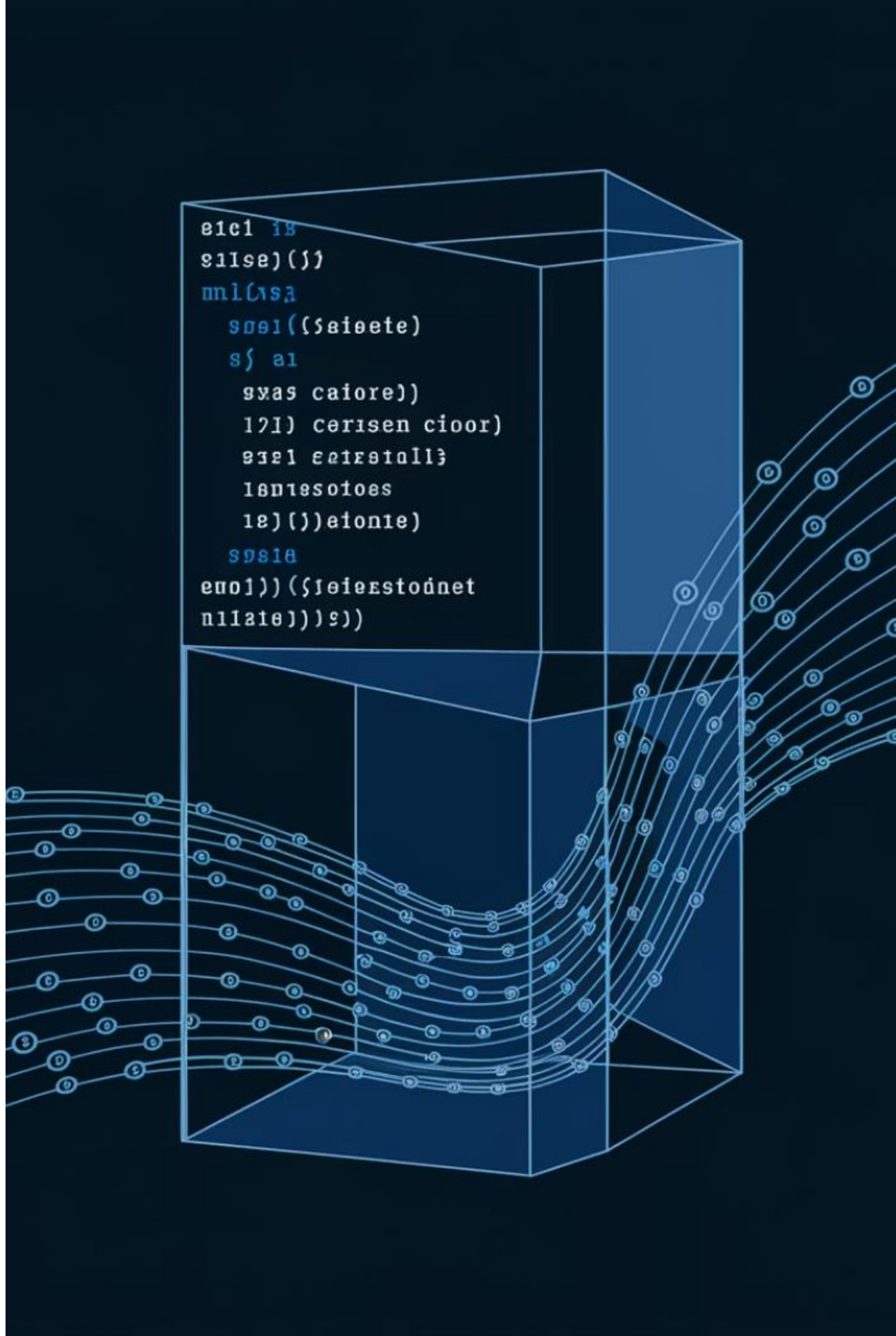
Las sentencias iterativas son instrucciones que ejecutan repetidamente un bloque de código mientras se cumpla una condición, o por cada elemento de una colección.

¿Por qué se necesitan?

Automatizan tareas repetitivas, evitan código duplicado y permiten recorrer colecciones como listas o diccionarios de manera eficiente.

Tipos principales

El bucle while, que se repite mientras una condición sea verdadera, y el bucle for, ideal para recorrer colecciones de datos.



Bucle While

?

Definición

Ejecuta un bloque de código mientras una condición sea verdadera

Sintaxis

while condición: # código a ejecutar

!

Precaución

Debe modificarse la condición dentro del bucle para evitar bucles infinitos

Ejemplo:

```
contador = 0
limite = 5

while contador < limite:
    print(f"Contador: {contador}")
    contador += 1
```

Bucle For y Range()

1

Bucle For

Se utiliza para recorrer elementos de una secuencia como una lista, tupla, cadena o diccionario.

2

Range()

Genera una secuencia de números que puedes usar en un bucle for. Útil cuando necesitas repetir algo un número fijo de veces.

3

Enumerate()

Permite iterar obteniendo tanto el índice como el valor de cada elemento en la secuencia.

Ejemplo recorriendo una lista:

```
frutas = ['manzana', 'banana', 'naranja']

for fruta in frutas:
    print(f"Me gusta la {fruta}")
```

Iterar con range()

```
# De 0 a 4
for i in range(5):
    print(f"Iteración {i}")

# De 1 a 5
for i in range(1, 6):
    print(f"Número: {i}")

# De 10 a 2, en pasos de -2
for i in range(10, 1, -2):
    print(i)
```

Iterar con índice usando enumerate:

```
for index, lenguaje in enumerate(lenguajes):
    print(f"{index + 1}. {lenguaje}")
```

Iterando Estructuras de Datos

Iterar listas con for

```python
lenguajes = ['Python', 'JavaScript', 'C#']

for lenguaje in lenguajes:
    print(f"Lenguaje: {lenguaje}")
```

Clave y valor con items():

```python
for clave, valor in persona.items():
    print(f"{clave.capitalize()}: {valor}")
```

Recorriendo claves de un diccionario:

```python
persona = {'nombre': 'Ana', 'edad': 30}

for clave in persona:
    print(clave)
```

Buenas prácticas (Clean Code):

- Usa nombres descriptivos (nunca x, y, data si puedes evitarlo).
- Evita repetir código dentro del bucle.
- Controla condiciones para evitar bucles infinitos.
- Prefiere for cuando iteras colecciones y while cuando no sabes cuántas veces se repetirá.

Actividad Práctica – Explorando Estructuras de Datos e Iteraciones en Python

Objetivo:

Aplicar estructuras de datos (listas, diccionarios, tuplas, sets) y sentencias iterativas (for, while, range) mediante la implementación de un sistema simple para registrar estudiantes, calcular promedios y clasificar resultados.

Paso a paso detallado:

1. Crear una lista vacía llamada estudiantes

Esta lista almacenará los datos de cada estudiante.

2. Crear un bucle while que permita ingresar los datos de varios estudiantes (nombre y 3 notas)

Usa un diccionario para almacenar los datos de cada estudiante:

Actividad Práctica – Explorando Estructuras de Datos e Iteraciones en Python

3. Al finalizar cada ingreso, preguntar si se desea agregar otro estudiante.

4. Una vez finalizado el ingreso, iterar la lista de estudiantes con for y mostrar:

- Nombre del estudiante
- Notas
- Promedio
- Mensaje:
- “Aprobado” si el promedio ≥ 4.0
- “Reprobado” si el promedio < 4.0

5. Opcional: mostrar totales y estadísticas al final

- Total de estudiantes ingresados
- Porcentaje de aprobados y reprobados

Esto permite practicar el uso de len(), contadores y condiciones dentro del ciclo.

Material Complementario

https://www.youtube.com/watch?v=v25-m1LOUiU

🎥 **"Estructuras de Datos en Python – Curso desde cero"**.
Este video explica de forma visual, paso a paso, cómo funcionan las listas, diccionarios, tuplas y sets en Python, junto con ejemplos de iteraciones usando for y while.

Preguntas de Reflexión Final

1. ¿En qué situaciones usarías una lista, un diccionario, una tupla y un set? Describe un ejemplo concreto para cada una.
2. ¿Qué diferencias observaste entre usar for y while en Python? ¿En qué casos te parece más claro uno que otro?
3. ¿Qué estructura te pareció más útil o interesante y por qué?