Actividad 5

Contenedores en Python

TIC

TALENTO TECH

Universidad Tecnológica de Pereira

faceIT

Contenedores en Python

Exploración de listas, tuplas, y diccionarios como contenedores fundamentales en Python. Ejemplos de manipulación de datos con estos contenedores.

Listas:

En Python, una lista es un contenedor flexible y mutable que te permite almacenar una colección ordenada de elementos.

Declaración de una Lista: Puedes declarar una lista utilizando corchetes [].

```
mi_lista = [1, 2, 3, 'a', 'b', 'c']
```

Contenedores en Python

Índices y Acceso a Elementos: Los elementos de una lista se numeran desde 0 en adelante. Puedes acceder a un elemento mediante su índice.

```
primer_elemento = mi_lista[0]  # Resultado: 1
tercer_elemento = mi_lista[2]  # Resultado: 3
```

Longitud de una Lista: Puedes obtener la longitud de una lista con la función len().

```
longitud = len(mi_lista)  # Resultado: 6
```

Modificación de Elementos: Las listas son mutables, lo que significa que puedes modificar sus elementos.

```
mi_lista[3] = 'x'
# Resultado: [1, 2, 3, 'x', 'b', 'c']
```

Contenedores en Python

Operaciones Comunes:

Puedes realizar diversas operaciones con listas, como agregar elementos, eliminar elementos, extender una lista con otra, etc.

```
# Agregar elementos al final de la lista
mi_lista.append(4)
# Resultado: [1, 2, 3, 'x', 'b', 'c', 4]
```

```
# Eliminar elemento por valor
mi_lista.remove('b')
# Resultado: [1, 2, 3, 'x', 'c', 4]
```

```
# Extender la lista con otra lista
otra_lista = [5, 6]
mi_lista.extend(otra_lista)
# Resultado: [1, 2, 3, 'x', 'c', 4, 5, 6]
```

Contenedores en Python

Slicing (Rebanado): Puedes obtener porciones de una lista utilizando la técnica de "slicing".

```
sublista = mi_lista[1:4]  # Resultado: [2, 3, 'x']
```

Inclusión de Elementos: Puedes verificar si un elemento está presente en una lista con el operador in.

```
existe = 'c' in mi_lista  # Resultado: True
```

Contenedores en Python

Las listas son una herramienta versátil en Python y se utilizan comúnmente para almacenar y manipular colecciones de datos.

```
# Lista
colores = ["rojo", "verde", "azul"]


# Acceder a elementos de la lista
primer_color = colores[0]
```

Contenedores en Python

Tuplas

En Python, una tupla es otro tipo de contenedor, similar a una lista, pero con la diferencia principal de que es inmutable, lo que significa que no puedes modificar su contenido después de haber sido creada.

Declaración de una Tupla: Puedes declarar una tupla utilizando paréntesis ().

```
mi_tupla = (1, 2, 3, 'a', 'b', 'c')
```

Contenedores en Python

Índices y Acceso a Elementos: Los elementos de una tupla también se numeran desde 0 en adelante. Puedes acceder a un elemento mediante su índice de la misma manera que con las listas.

```
primer_elemento = mi_tupla[0]  # Resultado: 1
tercer_elemento = mi_tupla[2]  # Resultado: 3
```

Longitud de una Tupla: Puedes obtener la longitud de una tupla con la función len() al igual que con las listas.

```
longitud = len(mi_tupla)  # Resultado: 6
```

Contenedores en Python

Inmutabilidad: La principal diferencia con las listas es que las tuplas son inmutables, lo que significa que no puedes agregar, eliminar o modificar elementos una vez que la tupla ha sido creada.

```
# Esto generará un error
mi_tupla[0] = 'x'
```

Empaquetado y Desempaquetado: Puedes "empaquetar" varios valores en una tupla y luego "desempaquetar" esos valores en variables individuales.

```
coordenadas = (10, 20)
x, y = coordenadas
# Ahora x es 10 y y es 20
```

Contenedores en Python

Uso en Iteraciones: Las tuplas se utilizan comúnmente en situaciones donde se necesita una estructura de datos inmutable, como claves en un diccionario o elementos en un conjunto.

Tuplas Anidadas: Puedes tener tuplas dentro de tuplas, creando así estructuras de datos más complejas.

```
tupla_anidada = ((1, 2), ('a', 'b'))
```

Las tuplas son útiles cuando necesitas asegurarte de que los datos no cambien durante la ejecución de tu programa, y se utilizan comúnmente en situaciones donde la inmutabilidad es esencial.

Contenedores en Python

Diccionario

En Python, un diccionario es un tipo de contenedor que almacena pares clave-valor. Cada valor en un diccionario está asociado a una clave única que actúa como identificador. Aquí te explico los aspectos clave de los diccionarios en Python:

Declaración de un Diccionario: Puedes declarar un diccionario utilizando llaves {}.

```
mi_diccionario = {'clave1': 'valor1', 'clave2': 'valor2', 'clave3': 'valor3'}
```

Contenedores en Python

Acceso a Elementos por Clave: Accedes a los elementos de un diccionario utilizando sus claves.

```
valor_asociado_a_clave1 = mi_diccionario['clave1']
```

```
# Resultado: 'valor1'
```

Modificación y Adición de Elementos: Puedes modificar el valor asociado a una clave existente o agregar nuevas parejas clave-valor.

```
mi_diccionario['clave1'] = 'nuevo_valor'
mi_diccionario['nueva_clave'] = 'valor_nuevo'
```

Contenedores en Python

Eliminación de Elementos: Puedes eliminar un par clave-valor utilizando la palabra clave del.

```
del mi_diccionario['clave1']
```

Verificación de Existencia de Claves: Puedes verificar si una clave existe en un diccionario utilizando el operador in.

```
existe_clave = 'clave1' in mi_diccionario
```

```
# Resultado: False (ya que la eliminamos)
```

Contenedores en Python

Longitud del Diccionario: Puedes obtener la cantidad de elementos en un diccionario con la función len().

```
cantidad_elementos = len(mi_diccionario)
```

```
# Resultado: 2 (luego de eliminar una clave)
```

Iteración a través de Claves, Valores o Elementos: Puedes iterar a través de las claves, valores o pares clave-valor en un diccionario.

```
for clave in mi_diccionario:
    valor = mi_diccionario[clave]
```

Contenedores en Python

Diccionarios Anidados: Puedes tener diccionarios dentro de diccionarios, creando así estructuras de datos más complejas.

```
diccionario_anidado = {'clave1': {'subclave1': 'valor_subclave1'},
                       , 'clave2': {'subclave2': 'valor_subclave2'}}
```

Los diccionarios son extremadamente útiles cuando necesitas almacenar datos estructurados y asociar información de manera eficiente utilizando claves únicas.

```
# Diccionario
persona = {"nombre": "Carlos", "edad": 30, "ciudad": "Madrid"}

# Acceder a valores en el diccionario
edad_persona = persona["edad"]
```

Ejercicios

Ejercicios relacionados con contenedores en Python, abarcando listas, tuplas y diccionarios. Estos ejercicios son útiles para practicar el manejo de estos tipos de datos y mejorar la comprensión de su funcionamiento:

Ejercicios sobre Listas:

Operaciones Básicas:

Crea una lista vacía llamada mi_lista.
Agrega los números del 1 al 5 a mi_lista.
Imprime mi_lista.
Elimina el número 3 de la lista.

Ejercicios

Rebanado de Listas:
Crea una lista llamada números del 1 al 10.
Imprime los primeros tres elementos de números.
Imprime los elementos desde el tercero hasta el sexto.
Imprime los últimos dos elementos.

Listas y Ciclos:
Utiliza un bucle for para imprimir cada elemento de mi_lista creado en el ejercicio 1.

Ejercicios sobre Tuplas:

Creación de Tuplas:
Crea una tupla llamada mi_tupla con tres elementos de tu elección.
Intenta modificar un elemento de mi_tupla. ¿Qué sucede?

Ejercicios

Desempaquetado de Tuplas:
Crea una tupla de dos elementos llamada coordenadas con valores de latitud y longitud.Utiliza desempaquetado de tuplas para asignar estos valores a dos variables distintas.

Ejercicios sobre Diccionarios:

Operaciones Básicas con Diccionarios:
Crea un diccionario llamado mi_diccionario con al menos tres pares clave-valor.
Agrega una nueva clave-valor a mi_diccionario.
Imprime solo las claves del diccionario.

Ejercicios

Diccionarios Anidados:
Crea un diccionario anidado llamado contactos con información de al menos dos personas (nombre, edad, etc.).Accede e imprime información específica de una persona utilizando el diccionario anidado.

Iteración y Actualización:
Utiliza un bucle for para imprimir todas las claves y valores de mi_diccionario.
Actualiza el valor de una clave en mi_diccionario.

Estos ejercicios son flexibles y pueden ser adaptados según el nivel de los estudiantes y los conceptos específicos que desees enfatizar.

TIC

TALENTO TECH

Universidad Tecnológica de Pereira

faceIT