¿Cuál es la diferencia entre una lista y una tupla en Python?

Las listas y las tuplas son estructuras de datos que permiten almacenar valores. La diferencia fundamental entra ambas es la posibilidad de modificar (o no) los valores almacenados.

Lista:

- Se puede modificar.
- Se escribe entre corchetes [ ]
- Es mas lenta que una tupla
- Tiene funciones útiles como .append(), .remove(), .sort() …

Ejemplo de lista:

usuarios = ['Ana', 'Luis', 'Pedro']

Añadimos un usuario:

usuarios.append('María')

Modificamos un usuario

usuarios[1] = 'Lucía'

Sacamos por pantalla:

print(usuarios)

Nos muestra por pantalla: ['Ana', 'Lucía', 'Pedro', 'María']

Tupla

- No se puede modificar, los datos están protegidos.
- Se escribe entre paréntesis ( )
- Es mas rápida que una lista
- Tiene funciones de solo lectura como .count () o .index()

Ejemplo de tupla:

datos = ('June', 39, 'Bilbao')

Desempaquetado / Unpacking: Es una forma de asignar valores a varias variables al mismo tiempo. Es una función de Phyton que se puede usar tanto en listas como en tuplas pero es mas seguro con tuplas, que no cambian los valores, el orden...

Ejemplo:

datos = ('June', 39, 'Bilbao')

nombre, edad, ciudad = datos

Luego se puede hacer print(nombre) para sacar por pantalla un solo valor, o print(datos) para sacarlos todos.

¿Cuál es el orden de las operaciones?

P -> Paréntesis ()

E -> Exponentes (potencias, **)

M -> Multiplicación (*)

D -> División (/)

A -> Adición (+)

S -> Sustracción (-)

Ejemplo:

resultado = (2 + 3) * 2 ** 2 - 4 / 2

Orden de resolución de Python:

- **Paréntesis:** (2 + 3) = 5
  resultado = 5 * 2 ** 2 - 4 / 2
- **Exponentes**: 2 ** 2 = 4
  resultado = 5 * 4 - 4 / 2
- **Multiplicación y división**: 5 * 4 = 20 y 4 / 2 = 2
  resultado = 20 – 2
- **Sustracción:** 20 - 2 = 18

  print(resultado)

  18

¿Qué es un diccionario Python?

Un diccionario es una estructura de datos que almacena pares clave-valor. Las claves deben ser únicas y permiten acceder a los valores. Se definen con llaves { } y los elementos se separan por comas. Las claves son case-sensitive.

Ejemplo de diccionario:

```
persona = {
        "nombre": "June",
        "edad": 39,
        "ciudad": "Bilbao"
}
```

Para acceder a los datos, se usa la clave entre corchetes:

```
print(persona["nombre"])
```

Para añadir información o modificarla. Es importante verificar que la clave no existe para no sobreescribir información por error:

Si no existe el dato, se añade:

```
persona["profesión"] = "Heldesk N2"
```

Si el dato existe, lo modifica:

```
persona["ciudad"] = "Madrid"
```

Y para borrar una clave-valor del diccionario:

```
del persona[“ciudad”]
```

Para acceder de forma segura con .get(). Se utiliza cuando no estas seguro si la clave existe.

```
Print(persona.get(“ciudad”, “Dato no disponible”))
```

Como hemos borrado el dato, mostraría el valor por defecto “Dato no disponible”

Y finalmente, para ver claves:

```
print(persona.keys())
```

Para ver valores:

```
print(persona.values())
```

Para verlo todo:

```
print(persona.items())
```

¿Cuál es la diferencia entre el método ordenado y la función de ordenación?

La diferencia entre el método sort() y la función sorted() en Python está en cómo afectan a los datos originales y cómo devuelven la información solicitada. Ambas funciones sirven para ordenar datos.

sort()

- Solo se puede usar sobre listas.
- Modifica la lista original de forma permanente.
- Ordena la lista
- No devuelve el resultado en una nueva variable. Si vuelves a sacar por pantalla los datos los devuelve modificados.
- Se utiliza cuando no se necesita conservar el orden original.

Ejemplo:

```
nombres = ['Lucas', 'Lur', 'Ambrosilla']

nombres.sort()

print(nombres)
```

Saca por pantalla los datos ordenados: ['Ambrosilla', 'Lucas', 'Lur']

sorted()

- Se puede usar sobre cualquier objeto iterable (listas, tuplas, diccionarios, …)
- No modifica la información original. Devuelve la información guardada como una **lista nueva**, aunque el elemento de origen sea otro.
- Se utiliza cuando se quiere mantener el original intacto.

Ejemplo:

```
nombres = ['Lucas', 'Lur', 'Ambrosilla']

ordenados = sorted(nombres)

print(ordenados)
```

Saca por pantalla los datos ordenados: ['Ambrosilla', 'Lucas', 'Lur']

¿Qué es un operador de reasignación?

Un operador de reasignación permite realizar una operación matemática y actualizar el valor de una variable en una sola línea. Es una forma más compacta y legible de escribir código. Es una buena práctica no usarlos en operaciones complejas y hay que tener en cuenta el tipo de dato (/ siempre devuelve float):

Ejemplo:

Creamos la variable total y le asignamos el valor de 100:

```
total = 100
```

Sumamos 10 al valor actual de total:

```
total += 10
```

total = 110

Se restan 5 al valor actual de total:

```
total -= 5
```

total = 105

Se multiplica por 2 el valor actual de total:

```
total *= 2
```

total = 210

Se divide entre 10 el valor actual de total:

```
total /= 10
```

total = 21.0 (siempre devuelve float)

Se divide redondeando hacia abajo:

```
total //= 2
```

total = 10

Se eleva a la potencia de 2:

```
total **= 2
```

total = 100

Aplicamos el % que devuelve el resto de la división de total entre 3:

```
total %= 3
```

total = 1