TUPLAS

Una tupla es una secuencia de valores similar a una lista.
Los valores guardados en una tupla pueden ser de cualquier tipo, y son indexados por números enteros.

Sintácticamente, una tupla es una lista de valores separados por comas

- Pueden ser de cualquier tipo
- Están indexadas

  Cuando decimos que las tuplas están ordenadas, significa que los elementos tienen un orden definido y ese orden no cambiará.
- Son inmodificables **(La principal diferencia es que las tuplas son *inmutables)***
- Permiten duplicados Dado que las tuplas están indexadas, pueden tener elementos con el mismo valor:

```
unaTupla = (“Junio”, “Julio”, “Agosto”,” Junio”, “Julio”)
print(unaTupla)
```

Me devolverá: (“Junio”, “Julio”, “Agosto”,” Junio”, “Julio”)

Longitud de la tupla

Para determinar cuántos elementos tiene una tupla, use la función **len()**:

```
unaTupla = (“Junio”, “Julio”, “Agosto”)
print(unaTupla)
```

Me devolverá: 3

Tupla con un elemento

Para crear una tupla con un solo elemento, debe agregar una coma después del elemento; de lo contrario, Python no lo reconocerá como una tupla.

```
T1 = ("Pedro",)
print(T1)
print("T1 tiene", len(T1), 'elementos')
```

Me devolverá:

('Pedro',)
T1 tiene 1 elementos

El constructor tuple (): También es posible usar el constructor **tuple ()** para hacer una tupla.

```
días = tuple(('lunes', 'martes', 'miércoles', 'jueves', 'viernes'))
print(dias)
```

Me devolverá:

'lunes', 'martes', 'miércoles', 'jueves', 'viernes'

Empaquetar y desempaquetar una tupla

Podemos generar una tupla asignando a una variable un conjunto de variables o valores separados por coma:

```
x=10
y=30
punto=x,y
print(punto)
```

Tenemos dos variables enteras x e y. Luego se genera una tupla llamada punto con dos elementos, esto se denomina "**empaquetar**" una tupla.
Pero, en Python, también podemos extraer los valores nuevamente en variables. Esto se denomina "**desembalaje**" o "**desempaquetar**"

```
fecha=(25, "diciembre", 2021)
print(fecha)
dd,mm,aa=fecha
print("Dia",dd)
print("Mes",mm)
print("Año",aa)
```

El desempaquetado de la tupla "fecha" se produce cuando definimos tres variables separadas por coma y le asignamos una tupla:

```
dd,mm,aa=fecha
```

Es importante tener en cuenta de definir el mismo número de variables que la cantidad de elementos de la tupla.

Uso de *

Si el número de variables es menor que el número de valores, puede agregar un *al nombre de la variable y los valores se asignarán a la variable como una lista

```
frutas = ("ananá", "banana", "naranja", "kiwi")
(blanco, amarillo, *naranja) = frutas
print(blanco)
print(amarillo)
print(naranja)
```

Me devolverá:

ananá
banana
['naranja', 'kiwi']

No permiten añadir, mover o eliminar elementos. (No existe append, extend o remove)

Permiten extraer formando una nueva Tupla.

Importante: Se pueden convertir listas a tuplas y viceversa

Ejemplo*:* *miLista=list(miTupla)*

miTupla=tuple(miLista)

print("Juan" in miTupla)…True

miTupla1 = (12,10,2020)
dia,mes,anio = miTupla1

Métodos de Tuplas

Python tiene dos métodos integrados que puede usar en tuplas.

MÉTODOS	DESCRIPCIÓN
Count()	Devuelve el número de veces que ocurre un valor especificado en una tupla
Index()	Busca en la tupla un valor especificado y devuelve la posición de donde se encontró.

Listas y Tuplas anidadas

La lista es una estructura mutable (es decir podemos modificar sus elementos, agregar y borrar) en cambio, una tupla es una secuencia de datos inmutable, es decir una vez definida no puede cambiar.

Podemos crear y combinar tuplas con elementos de tipo lista y viceversa, es decir listas con componente tipo tupla.

Definimos la lista llamada empleado con tres elementos: en el primero almacenamos su nombre, en el segundo su edad y en el tercero la fecha de ingreso a trabajar en la empresa (esta se trata de una tupla)

```
empleado=["juan", 53, (25, 11, 1999)]
print(empleado)
empleado.append((1, 1, 2016))
print(empleado)
alumno=("pedro",[7, 9])
print(alumno)
alumno[1].append(10)
print(alumno)
```

Me devolverá:

['juan', 53, (25, 11, 1999)]

['juan', 53, (25, 11, 1999), (1, 1, 2016)]

('pedro', [7, 9])

('pedro', [7, 9, 10])

Tenemos definida la tupla llamada alumno con dos elementos, en el primero almacenamos su nombre y en el segundo una lista con las notas que ha obtenido hasta ahora:

```
alumno=("Alexis",[7, 9])
print(alumno)
```

Podemos durante la ejecución del programa agregar una nueva nota a dicho alumno:

```
alumno[1].append(10)
print(alumno)
```

Variantes de la estructura repetitiva for para recorrer tuplas y listas

Siempre que recorremos una lista o una tupla utilizando un for procedemos de la siguiente manera:

```
lista=[2, 3, 50, 7, 9]
for x in range(len(lista)):
    print(lista[x])
```

Esta forma de recorrer la lista es utilizada obligatoriamente cuando queremos modificar sus elementos como podría ser:

```
lista=[2, 3, 50, 7, 9]
print(lista) # [2, 3, 50, 7, 9]

for x in range(len(lista)):
    if lista[x]<10:
        lista[x]=0
print(lista) # [0, 0, 50, 0, 0]
```

Ahora veremos una segunda forma de acceder a los elementos de una lista con la estructura repetitiva for sin indicar subíndices.

```
lista=[2, 3, 50, 7, 9]
for elemento in lista:
    print(elemento)
```

Como podemos ver la instrucción for requiere una variable (en este ejemplo llamada elemento), luego la palabra clave in y por último el nombre de la lista. El bloque del for se ejecuta tantas veces como elementos tenga la lista, y en cada vuelta del for la variable elemento almacena un valor de la lista.