MATERIAL TRABAJO EN CLASES

Actividad individual en laboratorio 210 para el alumno, leer analizar y poner en práctica los ejemplos de cada estructura de datos

Identifica diferencias entre tuplas, arreglos y diccionarios en función de las capacidades de almacenamiento.

1 Listas, tuplas y diccionarios

1.1. Listas (list)

Las listas son estructuras de datos que pueden almacenar cualquier otro tipo de dato, inclusive una lista puede contener otra lista, además, la cantidad de elementos de una lista se puede modificar removiendo o añadiendo elementos. Para definir una lista se utilizan los corchetes, dentro de estos se colocan todos los elementos separados por comas:

calificaciones = [10,9,8,7.5,9]

nombres = ["Ana","Juan","Sofía","Pablo","Tania"]

mezcla = [**True**, 10.5, "abc", [0,1,1]]

Las listas son *iterables* y por tanto se puede acceder a sus elementos mediante indexación:

nombres[2]

'Sofía'

nombres[-1]

'Tania'

Se tiene la posibilidad de agregar elementos a una lista mediante el método append:

nombres.append("Antonio")

nombres.append("Ximena")

print(nombres)

['Ana', 'Juan', 'Sofía', 'Pablo', 'Tania', 'Antonio', 'Ximena']

El método remove elimina un elemento de una lista:

nombres.remove("Ana")

print(nombres)

['Juan', 'Sofía', 'Pablo', 'Tania', 'Antonio', 'Ximena']

Sí el valor pasado al método remove no existe, Python devolverá un ValueError:

nombres.remove("Jorge")

---------------------------------------------------------------------------

ValueError Traceback (most recent call last)

**<ipython-input-91-d983d2559e2f>** in **<module>()**

----> 1 nombres.remove("Jorge")

ValueError: list.remove(x): x not in list

1.2. Tuplas (tuple)[¶](https://jorgedelossantos.github.io/apuntes-python/Listas%2C%20tuplas%20y%20diccionarios.html#tuplas-tuple)

Las tuplas son secuencias de elementos similares a las listas, la diferencia principal es que las tuplas no pueden ser modificadas directamente, es decir, una tupla no dispone de los métodos como append o insert que modifican los elementos de una lista.

Para definir una tupla, los elementos se separan con comas y se encierran entre paréntesis.

colores=("Azul","Verde","Rojo","Amarillo","Blanco","Negro","Gris")

Las tuplas al ser *iterables* pueden accederse mediante la notación de corchetes e índice.

colores[0]

'Azul'

colores[-1]

'Gris'

colores[3]

'Amarillo'

Si intentamos modificar alguno de los elementos de la tupla Python nos devolverá un TypeError:

colores[0] = "Café"

---------------------------------------------------------------------------

TypeError Traceback (most recent call last)

**<ipython-input-96-3502c7127536>** in **<module>()**

----> 1 colores[0] = "Café"

TypeError: 'tuple' object does not support item assignment

1.3. Diccionarios (dict)

Los diccionarios son estructuras que contienen una colección de elementos de la forma clave: valor separados por comas y encerrados entre llaves. Las claves deben ser objetos inmutables y los valores pueden ser de cualquier tipo. Necesariamente las claves deben ser únicas en cada diccionario, no así los valores.

Vamos a definir un diccionario llamado edades en el cual cada clave será un nombre y el valor una edad:

edades = {"Ana": 25, "David": 18, "Lucas": 35, "Ximena": 30, "Ale": 20}

Puede acceder a cada valor de un diccionario mediante su clave, por ejemplo, si quisieramos obtener la edad de la clave Lucas se tendría que escribir:

edades["Lucas"]

35

**Diferencia entre listas, tuplas y diccionarios en Python**

Después de aprender a utilizar los tipos de variables básicos en Python conviene aprender el uso de contenedores avanzados como las listas, tuplas y diccionarios que básicamente son un tipo de arreglo, es decir no guardan solamente un valor sino que podemos imaginarlos como un mueble con múltiples casillas y en cada casilla podemos meter la información que queramos, pero el mueble sigue siendo 1.

**Uso de tuplas en Python**

Una tupla es **una variable donde se pueden almacenar múltiples datos**, se caracteriza y se diferencia de las listas porque una vez se ha creado o llenado sus valores ya luego no pueden modificarse, pasemos de inmediato a un ejemplo de tupla para aprender cómo se crean.

tupla1 = ('valor1', 'valor2', 'otro valor mas')

Veamos que primero escribimos el nombre que tendrá nuestra tupla, posterior a utilizar el operador de asignación "=" colocamos entre paréntesis cada valor que estará encerrado entre comillas simples y una coma separará a cada valor.  
  
Otro dato importante a mencionar es que **en una tupla podemos almacenar distintos tipos de datos**, pueden ser texto, numéricos entre otros, todo dentro de una misma tupla.

tupla2 = ('8', 'mi texto', '50.1')

print (tupla2[1]) *# Para ver algun valor indicamos su indice.*

En el ejemplo previo utilizamos la función print() ingresando dentro de los paréntesis el nombre de nuestra tupla y entre paréntesis cuadrados el número del índice del dato que queremos leer, el índice viene a ser el número de casilla, en el ejemplo anterior ingresé un 1 por lo tanto se imprimirá el segundo valor, esto se debe a que en los arreglos se empieza a contar desde 0 no desde 1, así si queremos visualizar el primer valor de la tupla2 que sería el '8', tenemos que hacer esto print(tupla[0]).

**Definición de listas en Python**

Como ya mencioné en la definición de tupla vuelvo a enfatizar que la diferencia o importancia mayor de las listas es que **en estas a diferencia de las tuplas puede cambiarse sus valores**, vamos a pasar al ejemplo pero primero aclaro que obviamente es necesario que exista alguna diferencia en el código al momento de declarar una lista para que el intérprete de Python entienda que es una lista y no una tupla, esta diferencia es que las listas se encierran entre paréntesis cuadrados cuando se están creando, solo eso, veamos.

lista1 = ['primer valor', 'otro valor', '49', '8.2']

print(lista1[1]) *# Visualizar un valor es igual que con las tuplas*

**Modificando listas en Python**

lista1 = ['primer valor', 'otro valor', '49', '8.2']

print(lista1[1]) *# Imprimimos el valor de la segunda casilla*

lista1[1] = 'valor cambiado' *# Cambiamos los datos de la segunda casilla*

print(lista1[1]) *# Volvemos a imprimir el contenido de la segunda casilla fe nuestra tupla*

Quizá se esté preguntando si es necesario utilizar comillas simples 'ejemplo' , la respuesta es no, se puede también utilizar comillas dobles si así lo desea "ejemplo".  
  
Así mismo ha de preguntarse **¿para qué usar tuplas en Python si existen las listas?** donde además se puede hacer lo mismo que con las tuplas y también modificar los valores, y la verdad es que si, la recomendación es que si existe la posibilidad de llegar a necesitar la modificación de algún valor mejor siempre use las listas.

**Definición y uso de diccionarios en Python**

Tenemos una diferencia que en principio parece no muy útil y ciertamente para la mayoría de programas no es muy útil, los diccionarios son muy similares a las tuplas pero **tienen el beneficio de utilizar un nombre llamado clave, en lugar de solo utilizar un numero de índice**, veamos ya mismo el ejemplo.

**Ejemplo de diccionario en Python**

diccionario1={'edad':95, 'nombre': 'Codigazo'}

print(diccionario1['edad'])

Una clave es "edad", esto no es posible ni en listas ni en tuplas, su utilidad está en que es más fácil de recordar estas claves que recordar los números de índices, quizá para una lista o tupla con 5 casillas no es importante pero si en ella quiero almacenar 10 a 15 datos distintos será confuso tener que recordar en que numero tengo X dato, esta es además la solución al problema del uso de los [array en C++](https://www.codigazo.com/en-c/arreglos-c-ejemplos), donde puede ser confuso tener que recordar en que casillas esta la información deseada para consultarla.

*Para su Desarrollo*

## Ejercicio\_1

Escribir un programa que almacene las asignaturas de un curso (por ejemplo Matemáticas, Física, Química, Historia y Lengua) en una lista y la muestre por pantalla.

## Ejercicio\_2

Escribir un programa que almacene las asignaturas de un curso (por ejemplo Matemáticas, Física, Química, Historia y Lengua) en una lista y la muestre por pantalla el mensaje Yo estudio <asignatura>, donde <asignatura> es cada una de las asignaturas de la lista. (ocupar lista y un for)

## Ejercicio\_3

Escribir un programa que almacene las asignaturas de un curso (por ejemplo Matemáticas, Física, Química, Historia y Lengua) en una lista y la muestre por pantalla.

## Ejercicio\_4

Escribir un programa que almacene en una lista los números del 1 al 10 y los muestre por pantalla en orden inverso separados por comas.

TUPLAS

## Ejercicio\_5

Escribir un programa que pregunte al usuario su nombre, edad, dirección y teléfono y lo guarde en un diccionario. Despúes debe mostrar por pantalla el mensaje <nombre> tiene <edad> años, vive en <dirección> y su número de teléfono es <teléfono>.

## Ejercicio\_6

Escribir un programa que guarde en una variable el diccionario {'Euro':'€', 'Dolar':'$', 'Yen':'¥'}, pregunte al usuario por una divisa y muestre su símbolo o un mensaje de aviso si la divisa no está en el diccionario.