PROGRAMACIÓN

Guía de estudio 11: En conjunto, todo se puede lograr

Tuplas y conjuntos en Python

Nivel: Secundario - Modalidad Educación Técnico-Profesional.
Ciclo: Segundo ciclo.
Especialidades: Programación, Informática.

Introducción

Python ofrece varias formas de almacenar muchos datos. Anteriormente vimos la lista, pero en esta guía vamos a ver las “tuplas“ y los “conjuntos” y sus diferencias elementales con lo visto anteriormente.

¿Qué estamos aprendiendo? Qué son las tuplas y los conjuntos, como crearlos y utilizarlos y en qué se diferencian con las listas.

Recursos

Video nro 27 del curso: https://youtu.be/PjCBukZttG4	

inet | Instituto Nacional de Educación Tecnológica

Ministerio de Educación Argentina

Video nro 28 del curso: https://youtu.be/rmRrvol4XcM	
Video nro 29 del curso: https://youtu.be/UKD3CMINxik	

Secuencia didáctica

Comenzá viendo estos videos en los cuales se explican los temas introducidos. El primer video hablará sobre las tuplas y los dos siguientes sobre los conjuntos. Luego de que veas los videos te proponemos algunas preguntas y ejercicios para que respondas.

https://youtu.be/PjCBukZttG4	
https://youtu.be/rmRrvol4XcM	
https://youtu.be/UKD3CMINxik	

Actividades

Después de mirar el video del link o del QR, te invitamos a que reflexiones con las siguientes preguntas.

No olvides escribir tus respuestas, te facilitarán realizar trabajos prácticos y repasar los temas que hayas estudiado con esta guía.

1) Enumerá tres diferencias entre los conjuntos y las listas.

2) ¿Qué pasa si una tupla la convertís en conjunto y luego la volvés a convertir en tupla? Hacé este código y describí su comportamiento. (Sugerencia: que la tupla tenga elementos repetidos)

3) Desarrollá un programa en el cual se tenga un conjunto inicial llamado "nombres" lleno de distintos nombres (los que quieras). Luego el programa deberá pedir que se ingrese el nombre que se quiere buscar, y por último se deberá imprimir un mensaje que diga si se encontró o no. En este caso, ¿es mejor una lista, una tupla o un conjunto (frozen o normal)? ¿Por qué?

4) Escribí un código en el que se tenga una tupla inicial con números cualesquiera. Luego imprimí la cantidad total de números repetidos en la tupla. Deberás utilizar, sí o sí, la función **"len()"**. Lo que se pide (puesto en forma de ejemplo) es: si había tres "1" y cinco "4" en la tupla, el resultado que se debe imprimir sería 6, dado que hay dos "1" repetidos y cuatro "4" repetidos).

Si no te sale, revisá el ejercicio 2. Te puede ser de mucha ayuda.

5) Desarrollá un programa en el cual haya una lista de conjuntos de monstruos (una lista con varios conjuntos adentro). Cada conjunto estará compuesto de un nombre y un tipo de monstruo ("Nessi", "Monstruo del lago ness" por ejemplo). Se deberá comenzar con una lista vacía y utilizar la función "append()" para agregar los conjuntos. Deberán tener finalmente una lista de al menos 3 conjuntos de monstruos.

6) Teniendo el siguiente programa base, agregá el código correspondiente para que se muestren:

a) Los productos nuevos (fácil!)

b) Los disponibles (refácil!)

c) Los productos nuevos y disponibles (apa, se complicó! Desde ya que no tenés que escribir a manito el resultado...)

```
productosNuevos = set()
productosNuevos = {"Nutella", "Super DDL", "Leche de soja", "Garrapiñada de almendras"}

productosDisponibles = set()
productosDisponibles = {"Queso crema", "Leche de soja", "Super DDL", "Galletitas", "Papas fritas"}
```

Sobre Para seguir aprendiendo:

En otras guías vimos distintos operadores, y como se explicó en el video, ya vimos que algunos operadores como "+" y "|" cumplen algunas funciones cuando los usamos entre conjuntos. Probá cómo se comportan los operadores relacionales con los conjuntos. Algunos son bastante parecidos ("==" y "!=" ya se deben imaginar cómo funcionan) y otros tienen funciones totalmente diferentes ("<", "<=", ">=", ">").

Acá está un ejemplo que tal vez te ayude con estos últimos operadores (¡pero primero intentá solo! Las respuestas son fáciles cuando nos ayudan...)

```
conj = set()
conj = {"hola", 1, 13.5, "chau"}

conj2 = set()
conj2 = {"hola", 1, 13.5, "chau", "perro",5}

print(conj < conj2)
```

Sobre la guía anterior...

Como ya sabemos, podemos tener listas dentro de listas. Vamos a poner un ejemplo:

```
miSuperLista = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] , 0]
```

¿Cómo accedemos a los valores? Bueno, para acceder al "0" es fácil. sería cosa de usar **miSuperLista[3]** (porque está en la cuarta posici{on de la lista. ¿Pero para acceder al resto de los números? Ahí está la cuestión. Accedamos por ejemplo al "3", para eso deberíamos acceder a la primera lista, el 3 valor. Eso se escribe en código como **miSuperLista[0][2]**, para acceder al "7", **miSuperLista[2][0]**. Es decir que utilizamos el primer "[ ]" para acceder al elemento de la primer lista y el segundo "[ ]" para acceder a los elementos dentro de la lista contenida.