Conjuntos - Lista de ejercicios - Soluciones en el curso

Lista de ejercicios que corresponden a los conjuntos. Las soluciones las dejo junto con este documento en un fichero zip.

1.- Crear un conjunto con elementos de la lista de la compra, con constructor y sin constructor. Comprobamos el tipo con type.

Elementos que podremos añadir:

- Macarrones
- Merluza
- Bloc de notas
- Pechugas de pollo
- Ordenador portátil
- Auriculares
- ...

1-extra. Hacer lo mismo pero en vez de añadir el conjunto completo, ir añadiendo elemento a elemento (mediante consola) en un cojunto hasta que introduzcamos “EXIT”

Ejemplo, introducimos solo “Anartz” y luego EXIT para salir

- Introduzca un valor. Para salir 'EXIT': Anartz
- Introduzca un valor. Para salir 'EXIT': EXIT
- {'Anartz'}
- <class 'set'>

2.- Realizar un programa que **añada un conjunto con valores de diferentes tipos**.

Lo haremos con diferentes tipos como

- int, float, complex, list,...

Recorremos los elementos uno a uno con un búcle.

3.- Crear un conjunto **de 4 números enteros.**

Ejemplo que usaremos: 4, 3, 4, 5

Una vez creada (si es posible, responder sin añadir el código):

- ¿Cuál es la longitud del conjunto?.
- Accedemos al elemento de la posición índice. ¿Qué información aparecerá?

- Recorrer el conjunto elemento a elemento.

4.- Convertimos una tupla en conjunto. Procurad añadir con por lo menos un elemento repetido para ver como funciona el descarte de duplicados

Ejemplo:

- tupla inicial = (‘a’, ‘n’, ‘a’, ‘r’, ‘t’, ‘z’)
- conjunto final ={‘a’, ‘n’, ‘r’, ‘t’, ‘z’} (El orden del conjunto final no tiene porque ser así, ya que NO ES ORDENADO)

5.- Teniendo un conjunto con estos colores y una lista, añadiremos los elementos de la lista en el conjunto **actualizándo el primer conjunto en una sola línea (sin que afecte a la lista en nada)**

- Conjunto: {“Amarillo”, “Naranja”, “Negro”}
- Lista: [“Azul”, “Verde”, “Rojo”]

Debemos de obtener lo siguiente (el orden no es el que tendrá, recordadlo, lo importante es que contenga esa información):

- {“Amarillo”, “Naranja”, “Negro”, “Azul”, “Verde”, “Rojo”}

6.- **Devuelve un nuevo conjunto de elementos después de añadir** desde los **dos conjuntos determinados sin modificarlos**

Tenemos:

- set1 = {10, 20, 30, 40, 50}
- set2 = {30, 40, 50, 60, 70}

Obtendremos:

- set1 = {10, 20, 30, 40, 50}
- set2 = {30, 40, 50, 60, 70}
- set3 = {10, 20, 30, 40, 50, 60, 70}

7.- Teniendo estos conjuntos con estos colores, mantendremos los elementos que están PRESENTES EN AMBOS CONJUNTOS modificando el conjunto que se encarga de llamar a la función, en mi caso lo haré con el primer conjunto.

- set1 = {“Amarillo”, “Negro”, “Azul”}
- set2 = {“Amarillo”, “Naranja”, “Negro”, “Rosa”}

Conjuntos - Lista de ejercicios - Soluciones en el curso

Debemos de obtener lo siguiente (el orden no es el que tendrá, recordadlo, lo importante es que contenga esa información):

- set1 = {“Amarillo”, “Negro”}
- set2 = {“Amarillo”, “Naranja”, “Negro”, “Rosa”}

8.- **Devuelve un nuevo conjunto de elementos después de obtener los elementos QUE ESTÁN PRESENTES EN AMBOS CONJUNTOS (valores identicos en los dos conjuntos)**. Recordad que al devolver el nuevo conjunto no modificaremos ninguno de los conjuntos iniciales

Tenemos:

- set1 = {10, 20, 30, 40, 50}
- set2 = {30, 40, 50, 60, 70}

Obtendremos:

- set1 = {10, 20, 30, 40, 50}
- set2 = {30, 40, 50, 60, 70}
- set3 = {30, 40, 50}

9.- Crear un conjunto a partir de una cadena de texto y recorrer todos sus carácteres, después de haber descartado los duplicados por estar dentro de un conjunto.

Teniendo esta cadena de texto: “python3”

El resultado debería de ser algo así (el orden no tiene nada que ver, pero los elementos que se muestran si):

- h
- 3
- p
- y
- t
- n
- o

10.- Ahora creamos un conjunto con 4-5 elementos (vamos a añadir sin repetirlos para que no se descarten) y debemos de recorrer uno a uno.

Teniendo este conjunto:

{“python3”, “java”, “javascript”, “graphql”, “typescript”}

El resultado debería de ser algo así (el orden no tiene nada que ver, pero los elementos que se muestran si):

- graphql
- typescript
- java
- javascript
- python3

11.- Obtener las longitudes de estos conjuntos que vamos a inicializar. Sin añadir el código, vamos a intentar responder a que longitud tiene cada uno de ellos y razonamos.

Una vez que lo hayamos hecho, añadimos el código y comprobamos.

Conjuntos a usar:

set1 = {5, 10, 3, 15, 2, 20}

set2 = {5, 5, 5, 5, 5, 5}

set3 = {5, 10, 5, 5, 5, 5, 5, 5}

set4 = {5, 10, 3, 5, 5, 5, 5, 5, 5}

12.- Usando un conjunto con números enteros, vamos a comprobar la pertenencia de los siguientes elementos, es decir, si un número existe o no existe en ese conjunto.

Ejemplo de uso:

nums = {1, 3, 5, 7, 9, 11}

Comprobar si existe / No existe:

6, 7, 11, 0

```
Conjunto 'nums' original:  {1, 3, 5, 7, 9, 11}

Comprobamos si 6 si existe en nums:
False
Comprobamos si 7 si existe en nums:
True
Comprobamos si 11 si existe en nums:
True
Comprobamos si 0 si existe en nums:
False

Comprobamos si 6 si NO existe en nums
True
Comprobamos si 7 si NO existe en nums
False
Comprobamos si 11 si NO existe en nums
False
Comprobamos si 0 si NO existe en nums
True
```