



Programá
tu futuro



Municipalidad de
Tres de Febrero

www.tresdefebrero.gov.ar/tecno3f

#ProgramáTuFuturo



Programá
tu futuro



Municipalidad de
Tres de Febrero



Municipalidad de
Tres de Febrero



Programá
tu futuro

PYTHON INTERMEDIO

¡Les damos la bienvenida!

</>





Municipalidad de
Tres de Febrero



Programá
tu futuro



CONJUNTOS

CLASE 1

</>



¿QUÉ ES?

Un conjunto es una colección no ordenada de objetos únicos. Python provee este tipo de datos al igual que otras colecciones más convencionales como las listas, tuplas y diccionarios.

Los conjuntos son ampliamente utilizados en lógica y matemática, y desde el lenguaje podemos sacar provecho de sus propiedades para crear código más eficiente y legible en menos tiempo.

CREANDO UN CONJUNTO



Programá
tu futuro



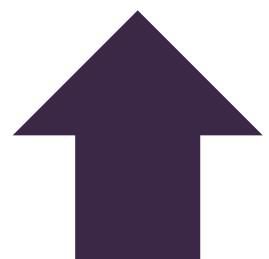
Municipalidad de
Tres de Febrero

Como se crean ?

Para crear un conjunto especificamos sus elementos entre llaves

```
► conjuntos.py > ...
1     x = {1, 2, 3, 4}
```

```
/ Tecno3f / Python / 2024
<class 'set'>
x = {1, 2, 3, 4}
```



Si utilizamos un `print(type(x))`, podremos ver que dice que es de la clase **set**

No obstante, un conjunto **no puede incluir objetos mutables** como listas, diccionarios, e incluso otros conjuntos.

Creando conjunto vacío

Para este caso, pensariamos que lo creariamos usando unas llaves vacias:

`p = {}`

Pero no.. si recuerdan las llaves vacias se utilizan para crear **diccionarios**, en este caso deberemos crear una instancia (ya veremos mas adelante que es) de la clase **set**

```
p = {}

print(type(p))
# <class 'dict'>

c = set()

print(type(c))
# <class 'set'>
```

Creando conjunto de otro objeto

Del mismo modo que el anterior debemos crear una instancia de la clase **set(OBJETO)** pasandole como argumento el objeto que deseamos convertir

```
lista = [1,2,3,4]
z = set(lista)

zz = set((1,2,3,4))

diccio = {'nombre': 'carlos', 'apellido': 'fernandez'}
zzz = set(diccio)
```

Para el caso de los **diccionarios** si lo convertimos directamente solo obtendremos las **keys** , para obtener los **valores** al convertirlo usaremos el metodo **.values()** , sobre nuestro diccionario

```
25 print(type(z))
26 print(type(zz))
27 print(type(zzz))

PROBLEMAS SALIDA CONSOL

garo-pc@garopc-MS-7C52:~/Tecno3F/Python/2024/1s
<class 'set'>
<class 'set'>
<class 'set'>
```



Programá
tu futuro



Municipalidad de
Tres de Febrero

ELEMENTOS

Agregar y Quitar Elementos

Al igual que las listas y diccionarios, los conjuntos son objetos mutables, utilizando los métodos `.add()` y `.discard()`, podremos agregar y eliminar elementos de un conjunto

```
31
32     conjunto1 = {1,2,3,4}
33     conjunto1.add(5)
34     print(conjunto1)
35     conjunto1.discard(2)
36     print(conjunto1)
```

PROBLEMAS SALIDA CONSOLA DE DE

```
/bin/python3.11 /home/garo-p
garo-pc@garopc-MS-7C52:~/Doc
/Tecno3F/Python/2024/ls-2024
{1, 2, 3, 4, 5}
{1, 3, 4, 5}
```

```
32     conjunto1 = {1,2,3,4}
33     # conjunto1.add(5)
34     # print(conjunto1)
35     conjunto1.discard(2)
36     # print(conjunto1)
37
38     conjunto1.discard(2)
39     print(conjunto1)
40     conjunto1.remove(2)
41     print(conjunto1)
```

PROBLEMAS SALIDA CONSOLA DE DE

```
bin/python3.11 /home/garo-p
garo-pc@garopc-MS-7C52:~/Doc
/Tecno3F/Python/2024/ls-2024
{1, 3, 4}
raceback (most recent call
  File "/home/garo-pc/Docume
    conjunto1.remove(2)
KeyError: 2
```

Limpiando el conjunto

Si queres vaciar el conjunto, podemos usar el metodo **.clear()**

```
conjunto1 = {1,2,3,4}  
print(conjunto1)  
  
conjunto1.clear()  
print(conjunto1)
```

```
{1, 2, 3, 4}  
set()
```

También podemos utilizar el metodo **.pop()** el cual nos devuelve y remueve un elemento aleatorio del conjunto

```
nombres = ['Ana', 'Marcos', 'Carlos', 'Mario', 'Pedro', 'Antonio', 'Pepe']  
print(nombres.pop())  
print(nombres)
```

```
['Ana', 'Marcos', 'Carlos', 'Mario', 'Pedro', 'Antonio', 'Pepe']  
Marcos  
['Mario', 'Carlos', 'Pepe', 'Ana', 'Antonio', 'Pedro']
```

OPERACIONES BÁSICAS



Programá
tu futuro



Municipalidad de
Tres de Febrero

Las principales: Union

La unión se realiza con el carácter | y retorna un conjunto que contiene los elementos que se encuentran en al menos uno de los dos conjuntos involucrados en la operación

```
1 conjunto = {1,2,4,5}
2 conjunto2 = {1,2,3,6,4}
3
4 print(conjunto | conjunto2)
5
```

- `garo-pc@garopc-MS-7C52:~/Docu/Tecno3F/Python/2024/ls -2024/ {1, 2, 3, 4, 5, 6}`
- `garo-pc@garopc-MS-7C52:~/Docu/Tecno3F/Python/2024/ls -2024/ {1, 2, 3, 4, 5, 6}`

Las principales: Intersección

La intersección se realiza con el operador `&`, y retorna un nuevo conjunto con los elementos que se encuentran en ambos

```
conjunto = {1,2,4,5}
conjunto2 = {1,2,3,6,4}

print(conjunto & conjunto2)
```

```
garo-pc@garopc-M
/Tecno3F/Python/
{1, 2, 4}
garo-pc@garopc-M
```

Las principales: Diferencia

La diferencia, que se realiza con el operador `-`, retorna un nuevo conjunto que contiene los elementos del **primero** que no están en el **segundo**.

```
conjunto = {1,2,4,5}  
conjunto2 = {1,2,3,6,4}  
  
print(conjunto - conjunto2)
```

```
/Tecno3F/Python/  
{5}  
garo-pc@garopc-M:
```

Las principales: Igualdad

Dos conjuntos son iguales **si y solo si** contienen los mismos elementos sin importar el orden. Se realiza con el operador ==

```
conjunto = {1,2,4,5}  
conjunto2 = {4,1,5,2}  
  
print(conjunto == conjunto2)
```

```
● garo-pc@gar...  
/Tecno3F/F...  
True  
○ garo-pc@gar...
```

MÁS OPERACIONES



Programá
tu futuro



Municipalidad de
Tres de Febrero

Es un sub-conjunto ?

Para saber si un conjunto es un subconjunto de otro podemos utilizar el método `.issubset()`, este nos dirá si el conjunto `x` puede ser un subconjunto de `y` devolviéndonos un **True** o **False** según corresponda.

```
nombres = {'Ana', 'Marcos', 'Carlos', 'Mario', 'Pedro', 'Antonio', 'Pepe'}  
nombres2 = {'Mario', 'Pedro'}  
  
print(nombres2.issubset(nombres))
```

```
> garo-pc@q  
s/Tecno3F  
True  
> garo-pc@q
```

Para que sea un subconjunto todos sus elementos deben formar parte del conjunto padre

Es el padre .. o super conjunto ?

Similar al anterior.. pero preguntamos de manera inversa si un conjunto es super de otro .. es decir si el conjunto x es super de y, esto significa que los elementos de y estan dentro del conjunto x, para esto utilizamos el metodo **.issuperset()**

```
nombres = {'Ana', 'Marcos', 'Carlos', 'Mario','Pedro', 'Antonio','Pepe'}  
  
nombres2 = {'Mario','Pedro'}  
  
print(nombres.issuperset(nombres2))
```

Al igual que el metodo anterior, nos devolvera un **booleano** depnediendo cual sea el resultado.

Obteniendo los distintos

Como vimos antes, con el operador **&** obteniamos los elementos que se repiten en ambos, y con el operador **-** solo los elementos que se encontraban en un primer conjunto, ahora si queremos obtener los elementos, que estan en al menos un conjunto (es decir elementos que no se repitan en ambos), podemos usar el metodo **.symmetric_difference()** , el cual nos retorna un nuevo conjunto con dichos elementos.

```
garo-pc@garopc-  
/Tecno3F/Python  
{3, 5, 6}  
garo-pc@garopc-
```

```
conjunto = {1,2,4,5}  
conjunto2 = {1,2,3,6,4}  
  
print(conjunto.symmetric_difference(conjunto2))
```

No conectados

Los conjuntos no conectados o desconexos, son aquellos que **no comparten** elementos entre si, y podemos averiguarlo utilizando el metodo **.isdisjoint()**, al igual que los primeros dos metodos que vimos en esta sección nos retornara un **booleano**

```
x ={1, 2, 3}
y ={3, 4, 5}
z ={5, 6, 7}

print(x.isdisjoint(y))
print(x.isdisjoint(z))
```

```
/Tecno3F/
False
True
○ garo-pc@g...
```

INMUTABILIDAD



Programá
tu futuro



Municipalidad de
Tres de Febrero

Libre soy.... Libre soy...

Como ya hemos visto en esta clase, la clase **set()** nos da un conjunto que es mutable, ahora bien, si necesitaramos crear un conjunto inmutable podemos utilizar la clase **frozenset()**, esta misma comparte todas las cualidades de la clase **set()**, salvo aquellas que hacen modificaciones, como add o discard , etc..

```
a = frozenset({1, 2, 3})  
b = frozenset({3, 4, 5})  
  
print(a | b)  
print(a-b)
```

```
garo-pc@garopc-MS-7C52:~/D00/  
/Tecno3F/Python/2024/ls-2024-  
frozenset({1, 2, 3, 4, 5})  
frozenset({1, 2})  
garo-pc@garopc-MS-7C52:~/D00/
```

Si recordán es similar a las tuplas y las listas

PARA PRACTICAR



Programá
tu futuro



Municipalidad de
Tres de Febrero

Practica.. Practica.. Practica

Dados dos conjuntos, A y B, escribe un programa en Python que imprima los elementos que se encuentran en A o en B, o en ambos.

Dados dos conjuntos, A y B, escribe un programa en Python que imprima los elementos que se encuentran en A y en B

Dados dos conjuntos, A y B, escribe un programa en Python que imprima el conjunto de los elementos que se encuentran en A o en B, pero no en ambos.

Dados un conjunto, A, escribe un programa en Python que imprima si el conjunto es un subconjunto de otro conjunto, B.

Dados un conjunto, A, escribe un programa en Python que imprima el número de elementos del conjunto.

¡MUCHAS GRACIAS!



Programá
tu futuro



Municipalidad de
Tres de Febrero