**Conjuntos (Sets) de Python**

myset = {"apple", "banana", "cherry"}

**Set**

Los conjuntos se utilizan para almacenar varios elementos en una sola variable.

Set es uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos, los otros 3 son List, Tuple y Dictionary, todos con diferentes calidades y usos.

Un conjunto es una colección desordenada y no indexada.

Los conjuntos se escriben con llaves.

Example

Create a Set:

thisset = {"apple", "banana", "cherry"}  
print(thisset)

**Nota**: la lista de conjuntos está desordenada, lo que significa que los elementos aparecerán en un orden aleatorio.

**Nota**: Los conjuntos están desordenados, por lo que no puede estar seguro de en qué orden aparecerán los elementos.

**Establecer elementos**

Los elementos del conjunto no están ordenados, no se pueden cambiar y no permiten valores duplicados.

**Desordenado**

Desordenado significa que los elementos de un conjunto no tienen un orden definido.

Los elementos del conjunto pueden aparecer en un orden diferente cada vez que los use y no se puede hacer referencia a ellos por índice o clave.

**Inmutable**

Los conjuntos no se pueden cambiar, lo que significa que no podemos cambiar los elementos una vez creado el conjunto.

Una vez que se crea un conjunto, no puede cambiar sus elementos, pero puede agregar elementos nuevos.

**No se permiten duplicados**

Los conjuntos no pueden tener dos elementos con el mismo valor.

Example

Duplicate values will be ignored:

thisset = {"apple", "banana", "cherry", "apple"}  
  
print(thisset)

**Obtenga la longitud de un conjunto**

Para determinar cuántos elementos tiene un conjunto, use el método len().

Example

Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}  
  
print(len(thisset))

**Establecer elementos: tipos de datos**

Los elementos del conjunto pueden ser de cualquier tipo de datos:

Example

String, int and boolean data types:

set1 = {"apple", "banana", "cherry"}  
set2 = {1, 5, 7, 9, 3}  
set3 = {True, False, False}

print(set1)

print(set2)

print(set3)

Un conjunto puede contener diferentes tipos de datos:

Example

A set with strings, integers and boolean values:

set1 = {"abc", 34, True, 40, "male"}

print(set1)

**type()**

Desde la perspectiva de Python, los conjuntos se definen como objetos con el tipo de datos 'set':

<class 'set'>

Example

What is the data type of a set?

myset = {"apple", "banana", "cherry"}  
print(type(myset))

**El constructor set()**

También es posible usar el constructor set() para hacer un conjunto.

Example

Using the set() constructor to make a set:

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets  
print(thisset)

**Colecciones de Python (Arrays)**

Hay cuatro tipos de datos de recopilación en el lenguaje de programación Python:

* List (lista) es una colección ordenada y modificable. Permite miembros duplicados.
* Tuple (tupla) es una colección ordenada e inmutable. Permite miembros duplicados.
* Set (conjunto) es una colección desordenada y no indexada. No hay miembros duplicados.
* Dictionary (diccionario) es una colección ordenada\* y modificable. No hay miembros duplicados.

\* A partir de la versión 3.7 de Python, los diccionarios están ordenados. En Python 3.6 y versiones anteriores, los diccionarios están desordenados.

Al elegir un tipo de colección, es útil comprender las propiedades de ese tipo. Elegir el tipo correcto para un conjunto de datos en particular podría significar la retención del significado y podría significar un aumento en la eficiencia o la seguridad.

**Python: acceso a elementos del conjunto**

**Acceso a Elementos**

No puede acceder a los elementos de un conjunto consultando un índice o una clave.

Pero puede recorrer los elementos del conjunto utilizando un ciclo for, o preguntar si un valor especificado está presente en un conjunto, utilizando la palabra clave in.

### Example

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}  
  
for x in thisset:  
  print(x)

### Example

Check if "banana" is present in the set:

thisset = {"apple", "banana", "cherry"}  
  
print("banana" in thisset)

**Cambiar elementos**

Una vez que se crea un conjunto, no puede cambiar sus elementos, pero puede agregar elementos nuevos.

**Python - Agregar elementos al conjunto**

**Agregar elementos**

Una vez que se crea un conjunto, no puede cambiar sus elementos, pero puede agregar elementos nuevos.

Para agregar un elemento a un conjunto, use el método add().

### Example

Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.add("orange")  
  
print(thisset)

**Agregar conjuntos**

Para agregar elementos de otro conjunto al conjunto actual, utilice el método update().

### Example

Add elements from tropical into thisset:

thisset = {"apple", "banana", "cherry"}  
tropical = {"pineapple", "mango", "papaya"}  
  
thisset.update(tropical)  
  
print(thisset)

**Agregar cualquier iterable**

El objeto en el método update() no tiene que ser un conjunto, puede ser cualquier objeto iterable (tuplas, listas, diccionarios, etc.).

### Example

Add elements of a list to at set:

thisset = {"apple", "banana", "cherry"}  
mylist = ["kiwi", "orange"]  
  
thisset.update(mylist)  
  
print(thisset)

**Python - Eliminar elementos del conjunto**

**Remover un elemento**

Para eliminar un elemento de un conjunto, utilice el método remove() o discard().

### Example

Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.remove("banana")  
  
print(thisset)

**Nota**: Si el elemento a eliminar no existe, remove() generará un error.

### Example

Remove "banana" by using the discard() method:

thisset = {"apple", "banana", "cherry"}  
  
thisset.discard("banana")  
  
print(thisset)

**Nota**: Si el elemento a eliminar no existe, discard()NO generará un error.

También puede utilizar el método pop() para eliminar un elemento, pero este método eliminará el último elemento. Recuerde que los conjuntos están desordenados, por lo que no sabrá qué elemento se quita.

El valor de retorno del método pop() es el elemento eliminado.

### Example

Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}  
  
x = thisset.pop()  
  
print(x)  
  
print(thisset)

**Nota**: Los conjuntos están desordenados, por lo que cuando se utiliza el método pop(), no sabe qué elemento se elimina.

### Example

El método clear() vacía el conjunto:

thisset = {"apple", "banana", "cherry"}  
  
thisset.clear()  
  
print(thisset)

### Example

La palabra clave del borrará el conjunto por completo:

thisset = {"apple", "banana", "cherry"}  
del thisset  
print(thisset) # esto generará un error porque el conjunto ya no existe

**Python – Bucles en Conjuntos**

**Elementos de bucle**

Puede recorrer los elementos configurados utilizando un bucle for:

### Example

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}  
  
for x in thisset:  
  print(x)

**Python – Unión de Conjuntos**

**Unir dos conjuntos**

Hay varias formas de unir dos o más conjuntos en Python.

Puede usar el método union() que devuelve un nuevo conjunto que contiene todos los elementos de ambos conjuntos, o el método update() que inserta todos los elementos de un conjunto en otro:

Example

The union() method returns a new set with all items from both sets:

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
  
set3 = set1.union(set2)  
print(set3)

Example

The update() method inserts the items in set2 into set1:

set1 = {"a", "b" , "c"}  
set2 = {1, 2, 3}  
  
set1.update(set2)  
print(set1)

Nota: Tanto union() como update() excluirán cualquier elemento duplicado.

**Mantener SOLO los duplicados**

El método intersection\_update() mantendrá solo los elementos que están presentes en ambos conjuntos.

Example

Keep the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.intersection\_update(y)  
  
print(x)

El método intersection() devolverá un nuevo conjunto, que solo contiene los elementos que están presentes en ambos conjuntos.

Example

Return a set that contains the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.intersection(y)  
  
print(z)

**Conserve todo, pero NO los duplicados**

El método symmetric\_difference\_update() mantendrá solo los elementos que NO están presentes en ambos conjuntos.

Example

Keep the items that are not present in both sets:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.symmetric\_difference\_update(y)  
  
print(x)

El método symmetric\_difference() devolverá un nuevo conjunto, que contiene solo los elementos que NO están presentes en ambos conjuntos.

Example

Return a set that contains all items from both sets, except items that are present in both:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.symmetric\_difference(y)  
  
print(z)

**Python – Métodos de conjuntos**

Python tiene un conjunto de métodos integrados que puede usar en conjuntos.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Agrega un elemento al conjunto |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Elimina todos los elementos del conjunto |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Devuelve una copia del conjunto |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Devuelve un conjunto que contiene la diferencia entre dos o más conjuntos. |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Elimina los elementos de este conjunto que también están incluidos en otro conjunto especificado. |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Eliminar el elemento especificado. |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Devuelve un conjunto, que es la intersección de otros dos conjuntos. |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Elimina los elementos de este conjunto que no están presentes en otros conjuntos especificados. |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Devuelve si dos conjuntos tienen una intersección o no. |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Devuelve si otro conjunto contiene este conjunto o no. |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Devuelve si este conjunto contiene otro conjunto o no. |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Elimina un elemento del conjunto. |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Elimina el elemento especificado |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Devuelve un conjunto con las diferencias simétricas de dos conjuntos. |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserta las diferencias simétricas de este conjunto y otro. |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Devuelve un conjunto que contiene la unión de conjuntos. |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Actualiza el conjunto con la unión de este conjunto y otros. |

Actividades del 23sept2021

**Python Set método add ()**

### Example

Add an element to the fruits set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.add("orange")  
  
print(fruits)

**Definición y uso**

El método add() agrega un elemento al conjunto.

Si el elemento ya existe, el método add() no agrega el elemento.

**Sintaxis**

set.add(elmnt)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| elmnt | Requerido. El elemento para agregar al conjunto. |

**Más ejemplos**

### Example

Try to add an element that already exists:

fruits = {"apple", "banana", "cherry"}  
  
fruits.add("apple")  
  
print(fruits)

**Python Set método clear()**

Example

Remove all elements from the fruits set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.clear()  
  
print(fruits)

**Definición y uso**

El método clear() elimina todos los elementos de un conjunto.

**Sintaxis**

set.clear()

**Valores paramétricos**

Sin parámetros

**Python Set copy() Método**

Example

Copy the fruits set:

fruits = {"apple", "banana", "cherry"}  
  
x = fruits.copy()  
  
print(x)

**Definición y uso**

El método copy() copia el conjunto.

**Sintaxis**

set.copy()

**Valores paramétricos**

Sin parámetros

**Método Python Set difference()**

Example

Return a set that contains the items that only exist in set x, and not in set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.difference(y)  
  
print(z)

**Definición y uso**

El método difference() devuelve un conjunto que contiene la diferencia entre dos conjuntos.

Significado: el conjunto devuelto contiene elementos que existen solo en el primer conjunto y no en ambos conjuntos.

**Definición y uso**

set.difference(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para comprobar las diferencias en los  Conuntos. |

**Más ejemplos**

### Example

Reverse the first example. Return a set that contains the items that only exist in set y, and not in set x:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = y.difference(x)  
  
print(z)

**Método Python Set difference\_update ()**

### Example

Remove the items that exist in both sets:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.difference\_update(y)  
  
print(x)

**Definición y uso**

El método difference\_update() elimina los elementos que existen en ambos conjuntos.

El método difference\_update() es diferente del método difference(), porque el método difference() devuelve un nuevo conjunto, sin los elementos no deseados, y el método difference\_update() elimina los elementos no deseados del conjunto original.

**Sintaxis**

set.difference\_update(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para comprobar las diferencias en |

**Método Python Set discard()**

### Example

Remove "banana" from the set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.discard("banana")  
  
print(fruits)

**Definición y uso**

El método discard() elimina el elemento especificado del conjunto.

Este método es diferente del método remove(), porque el método remove() generará un error si el elemento especificado no existe y el método discard() no.

**Sintaxis**

et.discard(value)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Requerido. El elemento para buscar y eliminar. |

**Método Python Set intersection()**

### Example

Return a set that contains the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.intersection(y)  
  
print(z)

**Definición y uso**

El método intersection() devuelve un conjunto que contiene la similitud entre dos o más conjuntos.

Significado: el conjunto devuelto contiene solo elementos que existen en ambos conjuntos, o en todos los conjuntos si la comparación se realiza con más de dos conjuntos.

**Sintaxis**

set.intersection(set1, set2 ... etc)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set1 | Requerido. El conjunto para buscar elementos iguales. |
| set2 | Opcional. El otro conjunto para buscar elementos iguales.  Puede comparar tantos conjuntos como desee.  Separe los conjuntos con una coma. |

**Más ejemplos**

### Example

Compare 3 sets, and return a set with items that is present in all 3 sets:

x = {"a", "b", "c"}  
y = {"c", "d", "e"}  
z = {"f", "g", "c"}  
  
result = x.intersection(y, z)  
  
print(result)

**Python Set método intersection\_update()**

### Example

Remove the items that is not present in both x and y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.intersection\_update(y)  
  
print(x)

**Definición y uso**

El método intersection\_update() elimina los elementos que no están presentes en ambos conjuntos (o en todos los conjuntos si la comparación se realiza entre más de dos conjuntos).

El método intersection\_update() es diferente del método intersection(), porque el método intersection() devuelve un nuevo conjunto, sin los elementos no deseados, y el método intersection\_update() elimina los elementos no deseados del conjunto original.

**Sintaxis**

set.intersection\_update(set1, set2 ... etc)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set1 | Requerido. El conjunto para buscar elementos iguales. |
| set2 | Opcional. El otro conjunto para buscar elementos iguales.  Puede comparar tantos conjuntos como desee.  Separe los conjuntos con una coma. |

**Más ejemplos**

### Example

Compare 3 sets, and return a set with items that is present in all 3 sets:

x = {"a", "b", "c"}  
y = {"c", "d", "e"}  
z = {"f", "g", "c"}  
  
x.intersection\_update(y, z)  
  
print(x)

**Python Set método isdisjoint ()**

Example

Devuelve True si no hay elementos en el conjunto x presentes en el conjunto y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "facebook"}  
  
z = x.isdisjoint(y)  
  
print(z)

**Definición y uso**

El método isdisjoint() devuelve True si ninguno de los elementos está presente en ambos conjuntos; de lo contrario, devuelve False.

**Sintaxis**

set.isdisjoint(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para buscar elementos iguales. |

**Más ejemplos**

### Example

¿Qué pasa si no hay elementos presentes en ambos conjuntos?

Devuelve False si uno o más elementos están presentes en ambos conjuntos:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.isdisjoint(y)  
  
print(z)

**Python Set método issubset()**

### Example

Return True if all items in set x are present in set y:

x = {"a", "b", "c"}  
y = {"f", "e", "d", "c", "b", "a"}  
  
z = x.issubset(y)

**Definición y uso**

El método issubset() devuelve True si todos los elementos del conjunto existen en el conjunto especificado; de lo contrario, devuelve False.

**Sintaxis**

set.issubset(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para buscar elementos iguales. |

**Más ejemplos**

### Example

¿Qué pasa si no todos los elementos están presentes en el conjunto especificado?

Devuelve False si no todos los elementos del conjunto x están presentes en el conjunto y:

x = {"a", "b", "c"}  
y = {"f", "e", "d", "c", "b"}  
  
z = x.issubset(y)  
  
print(z)

**Python Set método issuperset()**

Example

Return True if all items set y are present in set x:

x = {"f", "e", "d", "c", "b", "a"}  
y = {"a", "b", "c"}  
  
z = x.issuperset(y)  
  
print(z)

**Definición y uso**

El método issuperset() devuelve True si todos los elementos del conjunto especificado existen en el conjunto original; de lo contrario, devuelve False.

**Sintaxis**

set.issuperset(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para buscar elementos iguales. |

**Más ejemplos**

### Example

What if not all items are present in the specified set?

Return False if not all items in set y are present in set x:

x = {"f", "e", "d", "c", "b"}  
y = {"a", "b", "c"}  
  
z = x.issuperset(y)  
  
print(z)

**Método Python Set pop()**

### Example

Remove a random item from the set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.pop()  
  
print(fruits)

**Definición y uso**

El método pop() elimina un elemento aleatorio del conjunto.

Este método devuelve el elemento eliminado.

**Sintaxis**

set.pop()

**Valores paramétricos**

Sin valores de parámetro.

**Más ejemplos**

### Example

Return the removed element:

fruits = {"apple", "banana", "cherry"}  
  
x = fruits.pop()  
  
print(x)

**Nota**: El método pop() devuelve el valor eliminado.

**Python Set método remove()**

### Example

Remove "banana" from the set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.remove("banana")  
  
print(fruits)

**Definición y uso**

El método remove() elimina el elemento especificado del conjunto.

Este método es diferente del método discard(), porque el método remove() generará un error si el elemento especificado no existe y el método discard() no.

**Sintaxis**

set.remove(item)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| item | Requerido. El elemento para buscar y eliminar. |

**Método Python Set symmetric\_difference ()**

### Example

Return a set that contains all items from both sets, except items that are present in both sets:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.symmetric\_difference(y)  
  
print(z)

**Definición y uso**

El método symmetric\_difference() devuelve un conjunto que contiene todos los elementos de ambos conjuntos, pero no los elementos que están presentes en ambos conjuntos.

Significado: el conjunto devuelto contiene una combinación de elementos que no están presentes en ambos conjuntos.

**Sintaxis**

set.symmetric\_difference(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para comprobar si hay coincidencias. |

**Python Set método symmetric\_difference\_update()**

### Example

Remove the items that are present in both sets, AND insert the items that is not present in both sets:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.symmetric\_difference\_update(y)  
  
print(x)

**Definición y uso**

El método symmetric\_difference\_update() actualiza el conjunto original eliminando elementos que están presentes en ambos conjuntos e insertando los otros elementos.

**Sintaxis**

set.symmetric\_difference\_update(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. El conjunto para comprobar si hay coincidencias. |

**Método Python Set union()**

### Example

Return a set that contains all items from both sets, duplicates are excluded:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.union(y)  
  
print(z)

**Definición y uso**

El método union() devuelve un conjunto que contiene todos los elementos del conjunto original y todos los elementos de los conjuntos especificados.

Puede especificar tantos conjuntos como desee, separados por comas.

No tiene que ser un conjunto, puede ser cualquier objeto iterable.

Si un elemento está presente en más de un conjunto, el resultado contendrá solo una apariencia de este elemento.

**Sintaxis**

set.union(set1, set2...)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set1 | Requerido. El iterable para unificar. |
| set2 | Opcional. El otro iterable con el que unificarse.  Puede comparar tantos iterables como desee.  Separe cada iterable con una coma. |

**Más ejemplos**

### Example

Unify more than 2 sets:

x = {"a", "b", "c"}  
y = {"f", "d", "a"}  
z = {"c", "d", "e"}  
  
result = x.union(y, z)  
  
print(result)

**Método Python Set update()**

### Example

Insert the items from set y into set x:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
x.update(y)  
  
print(x)

**Definición y uso**

El método update() actualiza el conjunto actual, agregando elementos de otro conjunto (o cualquier otro iterable).

Si un elemento está presente en ambos conjuntos, solo una apariencia de este elemento estará presente en el conjunto actualizado.

**Sintaxis**

set.update(set)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| set | Requerido. La inserción iterable en el conjunto actual. |