Conjuntos

Los conjuntos (sets) son un tipo de datos, iterable, semejante a las tuplas, listas y strings. La diferencia radica en:

- No permite información duplicada
- Se pueden agregar y eliminar elementos, pero no modificarlos. En ese sentido es "inmutable".

Los conjuntos son útiles para almacenar información en donde se requiere que no haya datos duplicados. Un conjunto se define con {} de la forma:

In [1]:

```
conjunto = {1, 2, 3, 4, 5}
type(conjunto)
```

```
set
```

Como y se menciono, un conjunto descarta los elementos repetidos:

In [3]:

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

```
{1, 2, 3, 4, 5}
```

Se puede observar, además, que son almacenados en el conjunto de forma ordenada, aunque esto solo es así con los números:

In [6]:

```
conjunto = {'a', 'c', 'd', 'b'}
print(conjunto)
```

```
{'c', 'd', 'b', 'a'}
```

Los conjuntos tienen un conjunto de métodos asociados:

In [7]:

```
dir(conjunto)
```

```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

Estos métodos se pueden clasificar en aquellos que editan un conjunto y los que realizan operaciones con los conjuntos.

In [32]:

```
mamiferos = {'gato', 'pero', 'caballo', 'jirafa', 'delfin'}
reptiles = {'caiman', 'lagarto', 'piton', 'tortuga'}
insectos = {'mosca', 'avispa', 'abeja', 'araña'}
# Se agrega un elemento nuevo en la lista de reptiles
reptiles.add('boa')
# Se elimina un elemento de insectos (araña no es un insecto!)
# Es prefereible utlizar discard a utilizar remove
insectos.discard(-1)
# Se agrega "hormiga" a la lista de insectos
insectos.add('hormiga')
# Se agrega un elemento que ya exista
insectos.add('abeja')
# Se imprimen las listas
print('mamiferos =', mamiferos, '\tnum elem = ', len(mamiferos))
print('reptiles =', reptiles, '\tnum elem =', len(reptiles))
print('insectos =', insectos, '\tnum ele =', len(insectos))
```

```
mamiferos = {'delfin', 'gato', 'caballo', 'pero', 'jirafa'}     num elem =  5
reptiles = {'piton', 'tortuga', 'boa', 'lagarto', 'caiman'}     num elem = 5
insectos = {'araña', 'abeja', 'hormiga', 'mosca', 'avispa'}     num ele = 5
```

Asi también se puede eliminar un conjunto con clear. Hay que recordar que aunque se pueden hacer varias modificaciones en un conjunto, no se puede editar sus elementos:

In [33]:

```
conjunto = {'juan', 'maria', 'elmer', 'elba'}
conjunto[0] = {'ernesto'}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-86be303cefc2> in <module>()
      1 conjunto = {'juan', 'maria', 'elmer', 'elba'}
----> 2 conjunto[0] = {'ernesto'}

TypeError: 'set' object does not support item assignment
```

In [34]:

```
conjunto.clear()    # Se elimina este conjunto
```

Cuando se trata de las operaciones que se pueden realizar sobre los conjuntos (+ y * no estan soportados) hay que pensar en los Digramas de Venn. Las operaciones hacen referencia a que relación tienen los conjuntos entre si

In [53]:

```
herbivoros = {'vaca', 'caballo', 'jirafa', 'tortuga'}
carnivoros = {'perro', 'gato', 'lagarto', 'caiman', 'piton', 'boa'}

# Union de conjuntos -> a.union(b)
animales = mamiferos | reptiles | insectos
print(animales)

# Intersección de conjuntos -> a.intersection(b)
reptiles_herbivoros = reptiles & herbivoros
print(reptiles_herbivoros)

# Diferencia entre conjuntos-> a.difference(b)
reptiles_carnivoros = reptiles - reptiles_herbivoros
print(reptiles_carnivoros)

# Diferencia simetrica (Union sin Interseccion)  -> a.simetric_difference(b)
print(reptiles ^ herbivoros)
```

```
{'gato', 'piton', 'abeja', 'caballo', 'jirafa', 'boa', 'hormiga', 'mosca', 'araña', 'delfin', 'tortuga', 'pero', 'lagarto', 'caiman', 'avispa'}
{'tortuga'}
{'caiman', 'piton', 'boa', 'lagarto'}
{'piton', 'caballo', 'vaca', 'jirafa', 'boa', 'lagarto', 'caiman'}
```

El tipo de dato set es ideal para hacer una colección de información y se desea evitar que haya información repetida. Por ejemplo, se requiere genera una lista de datos aleatorios para simular un sorteo y como cada número solo puede salir una sola vez, un conjunto es un tipo de dato ideal.

Solo hay que tener una precaución. Exste otro tipo de datos llamado "diccionario" que también se especifica con {}, por lo que si se crea un conjunto vacio *conj = {}* lo que se esta creando es una diccionario y no un conjunto. Para crear un conjunto vacio hay que utilizar la instrucción *set()*

In [63]:

```
import random

num_sorteo = set()
num_bolillas = 6
contador = 0

# Vamos a generar numeros mientras no hallan suficientes numeros del sorteo
while len(num_sorteo) < num_bolillas:
    # Se genera un numero entre 1 y 12
    # Se guarda en el conjunto. Si el numero ya salio este no se agregara
    num_sorteo.add(random.randrange(1, 12))
    contador += 1

print("Bolillas del sorteo: ")
for bolillas in num_sorteo:
    print('{:4}'.format(bolillas), end='')

print("\nNum Veces Num Generados: ", contador)
```

```
Bolillas del sorteo:
   1   2   6   7   9  10
Num Veces Num Generados:  8
```

Ejecute el programa anterior varias veces y observará no solo que las bolillas del sorteo siempre serán diferentes, sino que el numero de veces que se han generado números aleatorios puede ser mayor que 6, lo que significará que se han generado número repetidos pero al intentar agregarse al conjunto este número es descartado de forma automática.