Conjuntos

Generalidades

Los conjuntos son colecciones **desordenadas** que **no aceptan valores repetidos**. Los conjuntos permiten cambiar su contenido pero dado que no poseen un orden, **no aceptan operaciones indexadas**.

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.

En Python, puedes crear conjuntos utilizando llaves {} o la función set().

```
conjunto1 = {1, 2, 3}
print (conjunto1)
print (type(conjunto1))
```

```
conjunto2 = set({4, 5, 6})
print (conjunto2)
print (type(conjunto2))
```

Un conjunto puede ser creado por un range

```
mi_conjunto = set (range (1,9))
print (mi_conjunto)
```

Un conjunto no puede incluir objetos mutables como listas, diccionarios, e incluso otros conjuntos.

```
mi_conjunto = {2,5,6,2,3,[4,2,6]}
print(mi_conjunto)
```

En los conjuntos el orden de inserción NO es necesariamente el orden de almacenamiento

```
mi_conjunto = {'P', 'H', 'A', 'Z'}
print (mi_conjunto)
```

Un conjunto en python no puede ser indexado, dado que no posee un orden

```
conj = {4,5,8,2}
print (conj.index(4))
```

Tampoco puedo acceder a un elemento del connjunto por su indice

```
lista1={6,9,1,2,3,5}
print(lista1[0])
```

Operaciones básicas con conjuntos

Adicción de Elementos

Para adicionar elementos al conjunto utilizamos el método **add()**

```
mi_conjunto = set (range (1,9))
print (mi_conjunto)
mi_conjunto.add(30)
print (mi_conjunto)
```

Si debemos agregar varios elemento a un conjunto lo realizamos con el método "update()"

```
conjunto1 = set(range(1, 9))
print("Conjunto inicial:", conjunto1)

conjunto1.update([30, 10, 23, 55, 2, 11])

print("Conjunto con elementos agregados:", conjunto1)
```

En la creación de conjuntos podemos observar que NO permite adicionar elementos repetido.

```
mi_conjunto = set()
mi_conjunto.add(8)
mi_conjunto.add(2)
mi_conjunto.add(8)
mi_conjunto.add(7)
mi_conjunto.add(3)
mi_conjunto.add(8)
mi_conjunto.add(8)
print (len(mi_conjunto))
print (mi_conjunto)
```

Comprobamos si un elemento se encuentra o no dentro de un conjunto con la palabra reservada **in**

```
```

```
mi_conjunto = {3,4,6,8,2}
print (mi_conjunto)
num = int(input("Cual número desea saber si está en el conjunto:"))
print (f"Está el nummero {num} en el conjunto {mi_conjunto}?: {num in mi_conjunto}")
```

Eliminación de elementos en un conjunto

Para eliminar elementos de un conjunto, se utilizan los métodos remove() o discard(). La diferencia principal es que remove() generará un error si el elemento no está presente, mientras que discard() no.

discard() quita un elemento del conjunto

```
c = {2,5,7,9,3,1}
print(c)
c.discard (5)
print(c)
c.discard (4)
print(c)
```

```
c = {2,5,7,9,3,1}
print(c)
c.remove (7)
print(c)
c.remove (6)
print(c)
```

Métodos en Conjuntos

El método clear() elimina todos los elementos del conjunto, dejándolo vacío.

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

copy(): Crea una copia del conjunto.

```
conjunto_original = {1, 2, 3}
print("Conjunto original ",conjunto_original)
copia_conjunto = conjunto_original.copy()
print("Copia 1del conjunto:", copia_conjunto)
```

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

```
conjunto = {"m", "b", "f", "i", "v", "n", "p"}
```

```
elemento = conjunto.pop()
print(elemento)  # Un elemento aleatorio
print(conjunto)
```

Operaciones con conjuntos

Los conjuntos en Python son una estructura de datos muy útil que nos permite almacenar colecciones de elementos únicos. Una de las características más poderosas de los conjuntos son las operaciones que podemos realizar con ellos, las cuales nos permiten combinar, comparar y manipular conjuntos de manera eficiente.

union() Une un conjunto a otro y devuelve el resultado en un nuevo conjunto:

```
c1 = {2,5,7}
c2 = {2,5,3}
c3 = c1.union(c2)
print (f"c1 unido con el c2 = {c3}")
```

intersection() Devuelve un conjunto con los elementos comunes en dos conjuntos:

```
c1 = {2,7,5}
c2 = {2,5,3}
print (f"Los elementos comunes son = {c1.intersection(c2)}")
```

difference() Encuentra los elementos no comunes entre dos conjuntos:

```
c1 = {2,5,7}
c2 = {2,5,3}
print (f"Los elementos que solo estan en c1 son = {c1.difference(c2)}")
print (f"Los elementos que solo estan en c2 son = {c2.difference(c1)}")
```

symmetric_difference() Devuelve los elementos simétricamente diferentes entre dos conjuntos, es decir, todos los elementos que no concuerdan entre los dos conjuntos:

```
c1 = {2,5,7}
c2 = {2,5,3}
print (f"Los elementos diferentes en los dos conjuntos son = {c1.symmetric_difference(c2)}")
```

issubset() Comprueba si el conjunto es subconjunto de otro conjunto, es decir, si sus ítems se encuentran todos dentro de otro:

```
c1 = {2,5,7}
c2 = {2,5}
print (f"c2 es subconjunto de c1 = {c2.issubset(c1)}")
```

issuperset() Comprueba si el conjunto es contenedor de otro subconjunto, es decir, si contiene todos los ítems de otro:

```
c1 = {2,5,7}
c2 = {2,5}
print (f"c1 contiene todos los elementos de c2? {c1.issuperset(c2)}")
```

isdisjoint() Comprueba si el conjunto es disjunto de otro conjunto, es decir, si no hay ningún elemento en común entre ellos:

```
c1 = {2,5,7}
c2 = {9,3,1}
print (f"c1 NO tiene elementos en comun con c2?  {c1.isdisjoint(c2)}")
#c3 = {9,3,5}
#print (f"c1 NO tiene elementos en comun con c3? {c1.isdisjoint(c3)}")
```

Recorrido de conjuntos

Puedes recorrer un conjunto utilizando un bucle for.

```
colores = {"rojo", "verde", "azul", "amarillo", "rosa", "rojo"}

for color in colores:
    print(color)
```

O convertir el conjunto en una lista y luego recorrer esa lista utilizando el bucle deseado

```
colores = {"rojo", "verde", "azul", "amarillo", "rosa", "rojo"}
indice = 0
color = list(colores)
while indice < len(colores):
    print(color[indice])
    indice += 1
```

Si se recorre con while el conjunto nos mostraria todos los elementos en cada iteración, al no ser indexado no se verian los elementos individualmente.

```
colores = {"rojo", "verde", "azul", "amarillo", "rosa", "rojo"}
indice = 0
cant = len(colores)
print (cant)
while indice<cant:
  print(colores)
  indice +=1
```

Apropiación

1. Desarrolla un programa para gestionar clientes de una empresa. Utiliza un conjunto para almacenar los correos electrónicos de los clientes registrados. Permite al usuario agregar nuevos clientes, verificar si un cliente ya está registrado y eliminar clientes.

2. Crea un sistema para controlar el inventario de productos de una tienda. Utiliza un conjunto para almacenar los códigos de barras de los productos disponibles. Permite al usuario agregar nuevos productos al inventario, verificar la disponibilidad de un producto y eliminar productos obsoletos.

3. Lee un archivo CSV con datos de ventas y analiza los productos más vendidos. Utiliza un conjunto para almacenar los códigos de los productos vendidos. Luego, muestra al usuario los productos más populares y su frecuencia de ventas.

4. Desarrolla un sistema de control de acceso a recursos en una empresa. Utiliza conjuntos para representar los permisos de acceso de los empleados a diferentes áreas o recursos. Permite al usuario asignar permisos a los empleados, verificar los permisos de un empleado y revocar permisos cuando sea necesario

5. Crea un programa para gestionar las suscripciones de un servicio en línea. Utiliza un conjunto para almacenar los correos electrónicos de los usuarios suscritos. Permite al usuario agregar nuevos suscriptores, verificar si un usuario ya está suscrito y cancelar suscripciones.

Diccionarios

Generalidades

Los diccionarios son un tipo de colección donde cada elemento almacenado contiene una estructura de tipo clave-valor.

Creacion de diccionarios

La definición al igual que los conjuntos es por medio de llaves **{}** o utlizando el contructor "dict()"

```
diccionario ={}
print(type(diccionario))
dic2= dict()
print(type(dic2))
```

En los diccionarios las claves deben ser únicas, pues es através de ellas que podemos acceder a los valores.

```
diccionario = {"tres":"three","dos":"two","cuatro":"four","cinco":"five"}
print (type (diccionario))
print (diccionario)
```

Cada elemento en un diccionario está asociado con una clave única que actúa como un identificador para acceder al valor correspondiente.

```
diccionario = {"tres":"three","dos":"two","cuatro":"four","cinco":"five"}
print (diccionario["tres"])
```

Los diccionarios pueden estar conformados por diferentes tipos de datos como cadenas de texto, numeros enteros, decimales y estructuras de datos como listas, y otros diccionarios.

```
dic = {"llave1":"a",2:[2,34,5],4:"c",5:8,("fila","columna"):[23,21], "tupla":(22,"ana","perez
print(type(dic))
print (dic)
print(dic["llave1"])
print(dic[2][1])
```

Acceso a un diccionario

Para acceder a un valor determinado del diccionario ya no lo hacemos por su posición sino por su clave. Si intentamos acceder a un elemento del diccionario a través de una clave que no existe obtendremos un error.

```
capitales = {"Caldas": "Manizales", "Risaralda": "Pereira", "Valle": "Cali"}
print (f"La capital de Caldas es {capitales['Caldas']}")#acceso por clave
```

```
x = {"3":2,"4":[1,"s",3],"5":2}
print (type(x))
print(x["4"][1])
```

Podemos acceder a un elemento utilizando el método get() partir de su clave y si no lo encuentra devuelve un valor por defecto que podemos especificar:

```
x = {"3":2,"4":[1,"s",3],"5":2}
#x.get('5','Lo que buscas no esta en el diccionario')
x.get(10,'Lo que buscas no esta en el diccionario')
```

RETO 1

Supongamos que tenemos un diccionario que almacena los precios de algunos productos en una tienda. Queremos que, dado el nombre de un producto, obtengamos su precio. Si el producto no está en el diccionario, queremos mostrar un mensaje predeterminado "El producto no esta disponible.

Para acceder a las claves de un diccionario podemos utilizar el método "keys()"

```
x = {"3":2,"4":[1,"s",3],"5":2}
x.keys()
```

Para acceder a los valores de un diccionario podemos utilizar el método "values()"

```
x = {"3":2,"4":[1,"s",3],"5":2}
x.values()
```

Podemos acceder a la clave-valor de un diccionario utilizando el método "items()"

Creando una lista de tuplas:

```
x = {"3":2,"4":[1,"s",3],"5":2}
x.items()
#print(x)
```

Creando dos variables para el almacenamiento de la clave y del valor

```
for clave, valor in x.items():
    print(clave, valor)
```

O creando una tupla con los elementos clave y valor

```
edades = {'Hector':27,'Juan':45,'Maria':34}
print (edades)
for item in edades.items():
    print(item) #obteniendo clave-valor en la misma variable
    print(item[0], "tiene", item[1], "años")
```

RETO 2

Tenemos un diccionarios con las materias y notas respectivamente de un estudiante, utilizando los metodos vistos, accede a las notas del estudiante y realiza el promedio

```
estudiante1 = {
    "matematicas": 8,
    "historia": 7,
    "ciencias": 9,
    "ingles": 8,
    "arte": 6
}
```

Los diccionarios son mutables, por lo tanto podemos cambiar los valores de sus elementos

```
capitales = {"Caldas": "Manizales", "Risaralda": "Pereira", "Valle": "Cali"}
print(capitales)
capitales["Valle"] = "Santiago de Cali"
print (capitales)
```

Adición de elementos.

Se puede agregar datos a un diccionario por asignacion directa

```
producto = {"nombre": "Laptop", "precio": 1200, "stock": 50}

producto["marca"] = "Acer"
print(producto)
```

Tambien lo podemos hacer mediante el método "update()", el cual nos permite agregar uno o varios elementos simultaneos al diccionario

```
producto = {"nombre": "Laptop", "precio": 1200, "stock": 50, "marca": "Acer"}

producto.update({"serial":"sn12en12","color":"plateado"})
print(producto)
```

Eliminación

Usamos la función **del()** para borrar un elemento del diccionario

```
capitales = {"Caldas": "Manizales", "Risaralda": "Pereira", "Valle": "Cali"}
del(capitales ["Valle"]) #se elimina el elemento con la clave dada
print (capitales)
```

Tambien podemos utilizar el método "pop"

```
capitales = {"Caldas": "Manizales", "Risaralda": "Pereira", "Valle": "Cali"}
print(capitales.pop("Risaralda"))
print (capitales)
```

RETO 3

Se ha organizado una rifa con 5 participantes. En los primeros 2 intento se elimina al azar un participante y se muestra quién fue el eliminado de la lista para que no pueda ser seleccionado nuevamente. El 3 seleccionado se anuncia quién como el ganador de la rifa.

```
import random
participantes = {
```

```
    'Juan': '123456',
    'María': '234567',
    'Pedro': '345678',
    'Laura': '456789',
    'Carlos': '567890'
}
```

Recorrido de diccionarios

Los diccionarios se pueden recorrer utilizando bucles for para acceder a todas las claves, valores o pares clave-valor.

```
edades = {'Hector':27,'Juan':45,'Maria':34}
print (edades)
for x in edades:
    print(x) #Acceso a las claves

for clave in edades:
    print(edades[clave]) #Acceso a los valores


for clave in edades:
    print(clave, edades[clave]) #Acceso a claves y los valores
```

Conversiones de colecciones

Diccionarios a Listas o Tuplas

Puedes convertir un diccionario en una lista de sus claves, una lista de sus valores o una lista de tuplas (clave, valor).

```
mi_diccionario = {'a': 1, 'b': 2, 'c': 3}

lista_claves = list(mi_diccionario.keys())
print("Lista de claves:", lista_claves)

lista_valores = list(mi_diccionario.values())
print("Lista de valores:", lista_valores)

lista_tuplas = list(mi_diccionario.items())
print("Lista de tuplas:", lista_tuplas)
```

Listas o Tuplas a Diccionarios

Puedes convertir una lista de tuplas (clave, valor) en un diccionario.

```
lista_tuplas = [('a', 1), ('b', 2), ('c', 3)]
```

```
nuevo_diccionario = dict(lista_tuplas)
print("Nuevo diccionario:", nuevo_diccionario)
```

Conjuntos a Listas o Tuplas

Puedes convertir un conjunto en una lista o una tupla.

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

lista_desde_conjunto = list(mi_conjunto)
print("Lista desde conjunto:", lista_desde_conjunto)

tupla_desde_conjunto = tuple(mi_conjunto)
print("Tupla desde conjunto:", tupla_desde_conjunto)
```

Listas o Tuplas a Conjuntos

Puedes convertir una lista o una tupla en un conjunto, eliminando así los duplicados.

```
mi_lista = [1, 2, 3, 4, 5, 3, 4]
conjunto_desde_lista = set(mi_lista)
print("Conjunto desde lista:", conjunto_desde_lista)

mi_tupla = (1, 2, 3, 4, 5, 3, 4)
conjunto_desde_tupla = set(mi_tupla)
print("Conjunto desde tupla:", conjunto_desde_tupla)
```

Apropiación

1. Crea un diccionario llamado empleados donde las claves sean los números de identificación y los valores sean diccionarios con el nombre y el cargo del empleado. Agrega al menos 5 empleados con su respectiva información. Permite al usuario buscar empleados por su número de identificación y mostrar toda su información. Permite al usuario eliminar un empleado por su número de identificación.

2. Crea un diccionario contactos_avanzado donde las claves sean los nombres de las personas y los valores sean diccionarios con información detallada como teléfono, correo electrónico y dirección.
    - Agrega al menos 3 contactos con su respectiva información detallada.
    - Muestra al usuario la información detallada de un contacto específico.
    - Permite al usuario actualizar la información de contacto de una persona existente.

3. Se te proporciona una lista de ventas mensuales de diferentes productos. Debes eliminar los duplicados, luego convertirlos en un diccionario donde las claves sean los nombres de los productos y los valores sean las ventas mensuales. ventas_mensuales = [("Producto A", 150), ("Producto B", 100), ("Producto A", 150),("Producto C", 180), ("Producto B", 100), ("Producto A",

150), ("Producto D", 200), ("Producto C", 150), ("Producto D", 200), ("Producto E", 400), ("Producto F", 200)]

Se requiere presentar un informe con la siguiente información:

```
a. El total de las ventas

b. El promedio de ventas mensual

c. cual fue la maxima venta y de que producto

d. cual fue la menor venta y de que producto
```

4. Crea un diccionario puntajes_juegos donde las claves sean los nombres de los juegos y los valores sean listas de puntajes obtenidos por los jugadores. Agrega al menos 3 juegos con sus respectivos puntajes. Muestra al usuario el promedio de puntajes de un juego específico. Permite al usuario agregar un nuevo puntaje a un juego existente.

5. Crea un diccionario llamado inventario donde las claves sean los nombres de los productos y los valores sean diccionarios con la cantidad y el precio del producto. Agrega al menos 10 productos con su respectiva información.

   a. El programa debe mostrar el precio total de todo el inventario

   b. Permite al usuario buscar un producto por su nombre y mostrar toda su información y el precio total del inventario de ese producto.

   c.Permite al usuario actualizar el precio de un producto existente.

6. Crea un diccionario inventario_libros donde las claves sean los codigos de los libros y los valores sean tuplas con el títulos,autor, el número de copias disponibles y valor del libro. Agrega al menos 5 libros con sus respectivos autores y copias disponibles.

   a. Muestra al usuario los libros disponibles de un autor específico.

   b.Permite al usuario comprar de un libro la cantidad si esta disponible y actualizar el número de copias disponibles de ese libro. Se debe mostrar al cliente el valor a pagar.