



Facultad: Ingeniería
Escuela: Ingeniería en Computación
Asignatura: Autómatas y Compiladores

Conjuntos en Python

Introducción

En las ciencias de la computación, las notaciones formales tienen su principio en la teoría de conjuntos.

Durante el tiempo, se ha observado que una de las principales características de los seres humanos es la capacidad de agrupar objetos (entidades) de acuerdo a criterios específicos. La idea de agrupar estos elementos similares en grupos es una de los conceptos más fundamentales de la matemática moderna.

La teoría de conjuntos ha sido el marco unificador para todas las matemáticas desde que el matemático alemán Georg Cantor la formulara alrededor de 1870.

Ningún campo de la matemática podría describirse hoy en día sin hacer referencia a algún tipo de conjunto abstracto. En términos más generales, el concepto de membresía de un conjunto, se encuentra en el corazón de la teoría de conjuntos, explica cómo las sentencias con sustantivos y predicados son formulados en nuestro lenguaje, o en cualquier lenguaje abstracto como las matemáticas. Debido a esto, la teoría de conjuntos está íntimamente ligada a la lógica y sirve de base para todas las matemáticas.

Competencias

- Conoce las operaciones fundamentales de la teoría de conjuntos.
- Aplica lenguaje de programación para simular conjuntos.
- Implementa operaciones con conjuntos en lenguaje de programación.

Material y Equipo

- Guía de laboratorio N° 2.
- Computadora con Python 3.0 o superior.
- Memoria USB.

Introducción Teórica

¿Qué es un conjunto?

Un conjunto es una colección de objetos distintos, a menudo llamados elementos o miembros. Existen dos características que hacen a los conjuntos totalmente diferentes al resto de otras colecciones de elementos.

- Un conjunto está siempre bien definido, es decir, que si realizamos la pregunta: ¿este objeto particular se encuentra en esta colección?; siempre debe existir una respuesta clara por sí o por no basada en una regla o algunos criterios datos.
- No hay dos miembros del mismo conjunto que sean exactamente iguales, es decir, que no hay elementos repetidos.

Un conjunto puede obtener cualquier cosa imaginable, incluyendo números, letras, colores, incluso, otros conjuntos.

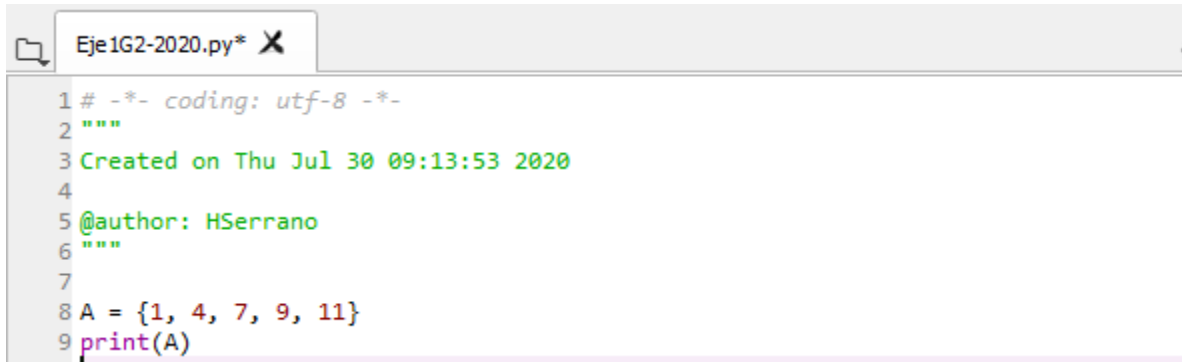
Además, utilizaremos el constructor `FiniteSet`, que proporciona la librería `sympy`, el cual tiene ciertas ventajas sobre la versión por defecto de Python.

Procedimiento

Conjuntos con Python

A continuación, se utilizarán unas funciones de Python para utilizar los conjuntos, ya que el lenguaje trae como una de sus estructuras de datos por defecto a los conjuntos.

1. Abrimos Spyder u otro IDE de su preferencia que le permita utilizar código Python.
2. Creando un conjunto en Python:



```

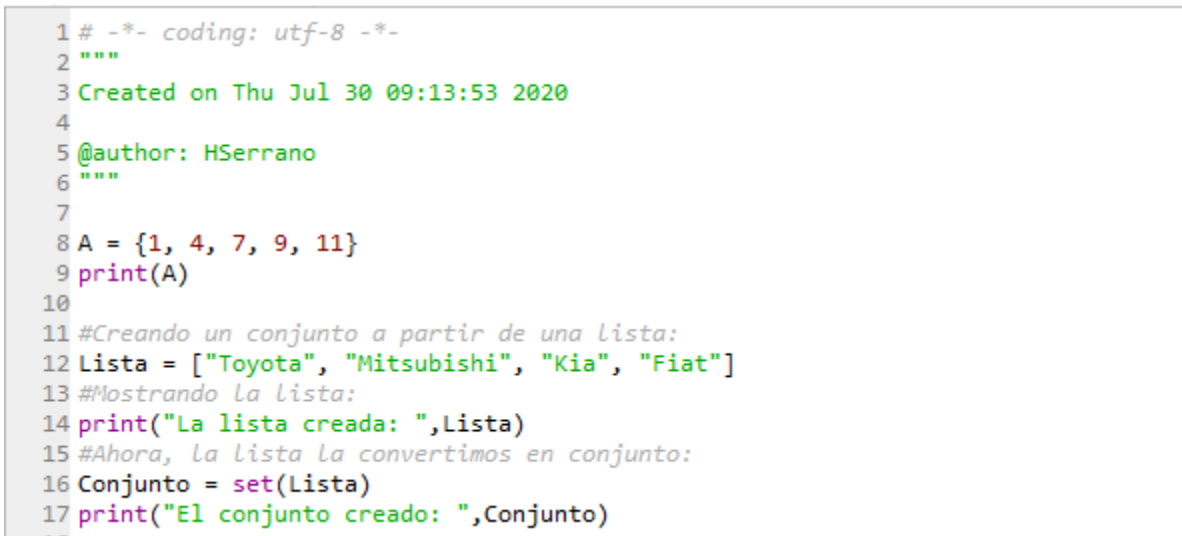
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Jul 30 09:13:53 2020
4
5 @author: HSerrano
6 """
7
8 A = {1, 4, 7, 9, 11}
9 print(A)

```

Resultado:

```
In [2]: runfile('C:/Users/HSerrano/Eje1G2-2020.py', wdir='C:/Users/HSerrano')
{1, 4, 7, 9, 11}
```

3. Creando un conjunto a partir de una lista. En Python, las listas se crean utilizando corchetes.



```

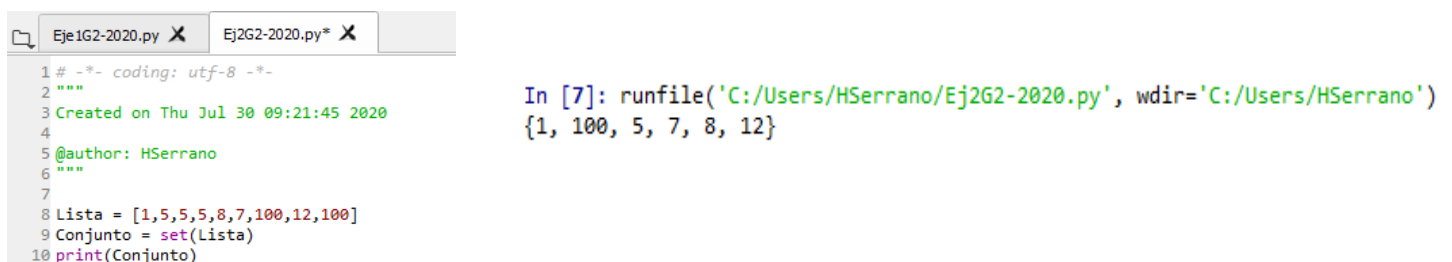
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Jul 30 09:13:53 2020
4
5 @author: HSerrano
6 """
7
8 A = {1, 4, 7, 9, 11}
9 print(A)
10
11 #Creando un conjunto a partir de una lista:
12 Lista = ["Toyota", "Mitsubishi", "Kia", "Fiat"]
13 #Mostrando la lista:
14 print("La lista creada: ",Lista)
15 #Ahora, la lista la convertimos en conjunto:
16 Conjunto = set(Lista)
17 print("El conjunto creado: ",Conjunto)

```

Resultado:

```
In [5]: runfile('C:/Users/HSerrano/Eje1G2-2020.py', wdir='C:/Users/HSerrano')
{1, 4, 7, 9, 11}
La lista creada: ['Toyota', 'Mitsubishi', 'Kia', 'Fiat']
El conjunto creado: {'Toyota', 'Mitsubishi', 'Fiat', 'Kia'}
```

4. Al establecer los conjuntos, esta estructura de datos elimina los elementos duplicados:



```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Jul 30 09:21:45 2020
4
5 @author: HSerrano
6 """
7
8 Lista = [1,5,5,5,8,7,100,12,100]
9 Conjunto = set(Lista)
10 print(Conjunto)

```

```
In [7]: runfile('C:/Users/HSerrano/Ej2G2-2020.py', wdir='C:/Users/HSerrano')
{1, 100, 5, 7, 8, 12}
```

5. El conjunto vacío se crea de esta manera:

```
Vacio = set()
```

6. Cardinalidad de un conjunto, es el número de elementos que tiene dicho conjunto:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 30 09:29:36 2020

@author: HSerrano
"""

A = [1,9,8,12,24,23,0]
print("La cardinalidad del conjunto es: ",len(A))

In [9]: runfile('C:/Users/HSerrano/Ej362-2020.py', wdir='C:/Users/HSerrano')
La cardinalidad del conjunto es: 7
```

7. Otras operaciones:

```
#Comprobando membresía, es decir, si un elemento pertenece al conjunto:
if 12 in A:
    print("True")

#Comprobando la igualdad de conjuntos:
B = [6,9,8,21,7]
B = set(B)
if A==B:
    print("True")
else:
    print("False")
```

Resultado:

True
False

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 30 09:40:02 2020

@author: HSerrano
"""

#Unión de conjuntos:
A = {1,2,3,4,5}
B = {4,5,6,7,8,9,10}
print("Unión de los conjuntos A y B: ",A.union(B))

#Intersección de conjuntos:
print("Intersección de conjuntos: ",A.intersection(B))

#Diferencia de conjuntos:
print("Diferencia de conjuntos: ", A-B)
```

Salida:

```
In [1]: runfile('C:/Users/HSerrano/Ej462-2020.py', wdir='C:/Users/HSerrano')
Unión de los conjuntos A y B: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Intersección de conjuntos: {4, 5}
Diferencia de conjuntos: {1, 2, 3}
```

Constructor FiniteSet de la librería Sympy:

SymPy es una biblioteca de Python para matemáticas simbólicas. Su propósito es llegar a ser un sistema de álgebra por computadora (CAS) completo manteniendo el código tan simple como sea posible para poder ser legible y extensible de manera fácil. SymPy está escrito en Python enteramente.

```
##Utilizando FiniteSet de Python
from sympy import FiniteSet
C = FiniteSet(1,2,3)
print("Utilizando FiniteSet: ",C)

#Generando el conjunto potencia
#Solo se puede hacer con FiniteSet
Potencia = C.powerset()
print("Conjunto potencia: ",Potencia)

#Cardinalidad:
print("La cardinalidad del conjunto potencia es: ",len(C.powerset()))

#Igualdad
A = FiniteSet("x","w","z","k")
B = FiniteSet("x","c","t")
if A==B:
    print("True")
else:
    print("False")

#Unión de dos conjuntos:
UnionConjuntos = A.union(B)
print("Unión de conjuntos con FiniteSet: ",UnionConjuntos)

#Intersección de conjuntos:
Interseccion = A.intersect(B)
print("Interseccion de conjuntos con FiniteSet: ",Interseccion)

#Diferencia
Diferencia = A-B
print("Diferencia de conjuntos con FiniteSet: ",Diferencia)
```

Salida:

```
In [1]: runfile('C:/Users/HSerrano/Sympy1.py', wdir='C:/Users/HSerrano')
Utilizando FiniteSet: {1, 2, 3}
Conjunto potencia: {EmptySet(), {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}
La cardinalidad del conjunto potencia es: 8
False
Unión de conjuntos con FiniteSet: {c, k, t, w, x, z}
Interseccion de conjuntos con FiniteSet: Union({x}, Intersection({c, t}, {k, w, z}))
Diferencia de conjuntos con FiniteSet: {k, w, z} \ {c, t}
```

Ejercicios propuestos

1. Sean $A = \{a, b, c, d, e\}$ y $B = \{b, d, e\}$

Hallar:

- $A \cup B$
- $B \cap B$
- $B - A$
- Conjunto potencia de A
- Cardinal del conjunto potencia de A

Investigación Complementaria

1. Cómo encontrar los subconjuntos (inclusión) en Python.
2. Cómo se trabaja el complemento de conjuntos en Python.

Bibliografía

- Python, Guido Van Rossum