FCC Tool Kit

Teoría de conjuntos

ITESO, Universidad Jesuita de Guadalajara

Equipo: M&M

Lilia Arceli Lobato Martínez, 706937

Laura Griselda González Camacho, 734049

Fundamentos de Ciencias Computacionales

Profesor: Fernando Velasco Loera

Periodo: Primavera 2021

Fecha de entrega: 05/04/2021

Link de github: https://github.com/LiliaLobato/FCC_Practicas/tree/main/P2

Funcionamiento a nivel usuario:

La calculadora de teoría de conjuntos lee hasta 3 diferentes conjuntos ingresados por el usuario (separados por comas), y dependiendo de la operación que se desee hacer, regresa el nuevo conjunto. Las operaciones que hace son:

- Unión
- Intersección
- Diferencia
- Diferencia simétrica

La cardinalidad máxima de cada conjunto es de 10 elementos de tipo alfanumérico.

El usuario deberá comenzar por definir los elementos de los tres conjuntos, llamados 'A', 'B' y 'C' con los parámetros indicados anteriormente. Después, deberá seleccionar cuál de las cuatro operaciones desea realizar, o bien, si desea terminar el programa. Una vez hecho esto, se desplegarán los diferentes acomodos para realizar la operación elegida y el usuario deberá seleccionar uno. El conjunto de respuesta se despliega en pantalla y el proceso se repite hasta que el usuario seleccione la opción de salir.

Descripción del trabajo:

UNIÓN:

```
def UNION_func(A,B):
    U = list(set(A) | set(B))
    return U
```

Esta es la función para la operación de unión. La unión son todos aquellos elementos que se encuentran en los conjuntos. En la función del código se usó la operación 'or' ('|') para definir la variable 'U', que es el conjunto respuesta, y regresarla.

```
if opcion == 1:
    #UNION
    print ("1. A U B")
    print ("2. B U C")
    print ("3. A U C")
    print ("4. A U (B U C)")
    sub_opcion = int(input("Introduce los conjuntos a operar:
"))
    if sub_opcion == 1:
        print("A U B:" ,A," U ", B)
        print(UNION_func(A,B))
    elif sub_opcion == 2:
```

```
      print("B U C:" ,B," U ", C)
      print(UNION_func(B,C))
   elif sub_opcion == 3:
      print("A U C:" ,A," U ", C)
      print(UNION_func(A,C))
   else :
      print("A U (B U C):" ,A," U ( ", B," U ",C," )")
      print(UNION_func(A,UNION_func(B,C)))
```

Aquí se le ofrecen al usuario los diferentes acomodos de los tres conjuntos para la operación Unión. Después, el código imprime la opción seleccionada y los conjuntos participes en la operación, e invoca la función explicada previamente con esos dos conjuntos.

Ejemplo:

INTERSECCIÓN:

```
def INTERSEC_func(A,B):
   N = list(set(A) & set(B))
   return N
```

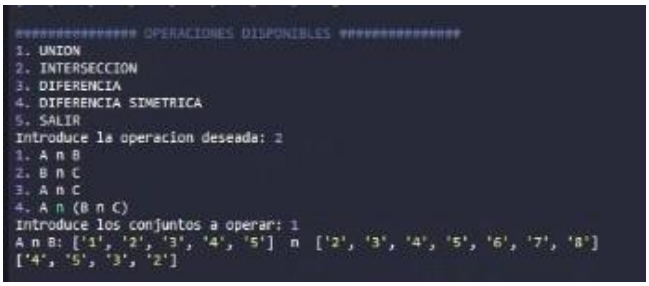
Esta es la función para la operación de intersección. La intersección son solo aquellos elementos que se encuentran en ambos conjuntos. En la función del código se usó la operación ‘and’ (‘&’) para definir la variable ‘N’, que es el conjunto respuesta, y regresarla.

```
elif opcion == 2:
   #INTERSECCION
   print ("1. A n B")
   print ("2. B n C")
   print ("3. A n C")
   print ("4. A n (B n C)")
   sub_opcion = int(input("Introduce los conjuntos a operar:
"))
   if sub_opcion == 1:
      print("A n B:" ,A," n ", B)
      print(INTERSEC_func(A,B))
```

```
    elif sub_opcion == 2:
        print("B n C:" ,B," n ", C)
        print(INTERSEC_func(B,C))
    elif sub_opcion == 3:
        print("A n C:" ,A," n ", C)
        print(INTERSEC_func(A,C))
    else :
        print("A n (B n C):" ,A," n ( ", B," n ",C," )")
        print(INTERSEC_func(A,INTERSEC_func(B,C)))
```

Aquí se le ofrecen al usuario los diferentes acomodos de los tres conjuntos para la operación Intersección. Después, el código imprime la opción seleccionada y los conjuntos participes en la operación, e invoca la función explicada previamente con esos dos conjuntos.

Ejemplo:



DIFERENCIA:

```
def DIFF_func(A,B):
    D = list(set(A)-set(B))
    return D
```

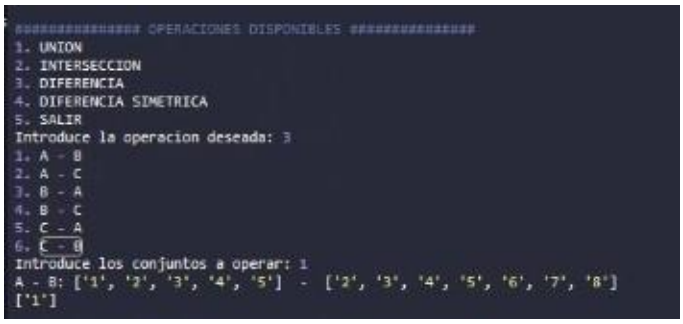
Esta es la función para la operación de diferencia. La diferencia son aquellos elementos que se encuentran en el primer conjunto que no se encuentran en el segundo. En la función del código se usó la operación de resta (‘-‘) para definir la variable ‘D’, que es el conjunto respuesta, y regresarla.

```
elif opcion == 3:
      #DIFERENCIA
      print ("1. A - B")
      print ("2. A - C")
      print ("3. B - A")
      print ("4. B - C")
      print ("5. C - A")
      print ("6. C - B")
      sub_opcion = int(input("Introduce los conjuntos a
operar: "))
      if sub_opcion == 1:
```

```
            print("A - B:" ,A," - ", B)
            print(DIFF_func(A,B))
        elif sub_opcion == 2:
            print("A - C:" ,A," - ", C)
            print(DIFF_func(A,C))
        if sub_opcion == 3:
            print("B - A:" ,B," - ", A)
            print(DIFF_func(B,A))
        elif sub_opcion == 4:
            print("B - C:" ,B," - ", C)
            print(DIFF_func(B,C))
        elif sub_opcion == 5:
            print("C - A:" ,C," - ", A)
            print(DIFF_func(C,A))
        else :
            print("C - B:" ,C," - ", B)
            print(DIFF_func(C,B))
```

Aquí se le ofrecen al usuario los diferentes acomodos de los tres conjuntos para la operación Diferencia. Después, el código imprime la opción seleccionada y los conjuntos participes en la operación, e invoca la función explicada previamente con esos dos conjuntos.

Ejemplo:



DIFERENCIA SIMÉTRICA:

```
def DIFF_SIM_func(A,B):
    D = list(list(set(A)-set(B)) + list(set(B)-set(A)))
    return D
```

Esta es la función para la operación de diferencia simétrica. La diferencia simétrica son aquellos elementos que se encuentran en el primer conjunto que no se encuentran en el segundo, o que se encuentran en el segundo conjunto pero no se encuentran en el primero. En la función del código se usó la operación de resta (‘-‘) entre los dos conjuntos y después la operación de suma (‘+’) entre ambas respuestas para definir la variable ‘D’, que es el conjunto respuesta, y regresarla.

```
elif opcion == 4:
      #DIFERENCIA SIMETRICA
      print ("1. A Δ B")
      print ("2. A Δ C")
      print ("3. B Δ A")
      print ("4. B Δ C")
      print ("5. C Δ A")
      print ("6. C Δ B")
      sub_opcion = int(input("Introduce los conjuntos a operar: "))
      if sub_opcion == 1:
         print("A - B:" ,A," - ", B)
         print(DIFF_func(A,B))
         print("B - A:" ,B," - ", A)
         print(DIFF_func(B,A))
         print("A Δ B:" ,A," Δ ", B)
         print(DIFF_SIM_func(A,B))
      elif sub_opcion == 2:
         print("A - C:" ,A," - ", C)
         print(DIFF_func(A,C))
         print("C - A:" ,C," - ", A)
         print(DIFF_func(C,A))
         print("A Δ C:" ,A," Δ ", C)
         print(DIFF_SIM_func(A,C))
      if sub_opcion == 1:
         print("B - A:" ,B," - ", A)
         print(DIFF_func(B,A))
         print("A - B:" ,A," - ", B)
         print(DIFF_func(A,B))
         print("B Δ A:" ,B," Δ ", A)
         print(DIFF_SIM_func(B,A))
      elif sub_opcion == 2:
         print("B - C:" ,B," - ", C)
         print(DIFF_func(B,C))
         print("C - B:" ,C," - ", B)
         print(DIFF_func(C,B))
         print("B Δ C:" ,B," Δ ", C)
         print(DIFF_SIM_func(B,C))
      elif sub_opcion == 2:
         print("C - A:" ,C," - ", A)
         print(DIFF_func(C,A))
         print("A - C:" ,A," - ", C)
         print(DIFF_func(A,C))
         print("C Δ A:" ,C," Δ ", A)
         print(DIFF_SIM_func(C,A))
      else :
         print("C - B:" ,C," - ", B)
         print(DIFF_func(C,B))
         print("B - C:" ,B," - ", C)
         print(DIFF_func(B,C))
         print("C Δ B:" ,C," Δ ", B)
         print(DIFF_SIM_func(C,B))
```

Aquí se le ofrecen al usuario los diferentes acomodos de los tres conjuntos para la operación Diferencia Simétrica. Después, el código imprime la opción seleccionada y los conjuntos participes en la operación, e invoca la función explicada previamente con esos dos conjuntos.

Ejemplo:

Relación de la materia con mi carrera:

Los fundamentos de las ciencias computacionales se relacionan con nuestras carreras (Ingeniería en Sistemas Computacionales e Ingeniería en Electrónica) ya que al ser ingenierías, debemos conocer las bases de los pensamientos lógico-matemáticos. Los conocimientos adquiridos en este curso nos ayudarán a desarrollar nuestros proyectos de manera profesional, ya que estaremos mejor preparadas para dar solución a problemas de lógica de manera creativa y estructurada, con la ayuda de las proposiciones, las reglas de inferencia y las operaciones aprendidas. Los podremos utilizar para desarrollar códigos y circuitos con las operaciones básicas (and, or, etc.) con un mayor entendimiento de cómo funcionan y de los resultados que podemos esperar de estas.

Los temas vistos durante este curso nos ayudan a practicar la resolución de problemas de nuestra vida diaria también, con bases científicas y con una mejor organización. El conocer estos temas nos ayuda a comprender otros conceptos básicos para ambas carreras. La lógica proposicional, así como la teoría de conjuntos son las bases para comprender el funcionamiento de las herramientas que utilizamos para desarrollar proyectos en nuestros ámbitos de estudio. Para los sistemas computacionales, la lógica es la base de los lenguajes de programación y del desarrollo de códigos con funciones implementadas correctamente. Para la electrónica, constituyen la base de la implantación de circuitos por medio de lenguajes de descripción de hardware como Verliog y programación de FPGA. Finalmente, esta materia nos ha permitido ejercitar nuestras habilidades de programación, y también nos va a dejar con una serie de herramientas que nos serán útiles para materias y/o proyectos laborales futuros.

Referencias bibliográficas:

1. Lenka, C. (2018). *Python | Union of two or more lists*. Recuperado de: https://www.geeksforgeeks.org/python-union-two-lists/
2. Lenka, C. (2020). *Python | Difference between two lists*. Recuperado de: https://www.geeksforgeeks.org/python-difference-two-lists/
3. Lenka, C. (2018). *Python | Intersection of two lists*. Recuperado de: https://www.geeksforgeeks.org/python-intersection-two-lists/