

**UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA**

**Facultad de Ciencias Químicas e Ingeniería**

Programas de Ingeniero en Computación e Ingeniero en Software y Tecnologías Emergentes

**INFORMACIÓN DE LA MATERIA**

Nombre de la materia y clave: Lenguaje de Programación Python (36305).

Grupo y periodo: 532 (2022-2)

Profesor: Manuel Castañón Puga.

**INFORMACIÓN DE LA ACTIVIDAD**

Nombre de la actividad: Práctica de laboratorio 1.3.2 Conjuntos

Lugar y fecha: A 4 de septiembre de 2022 en el Edificio 6E, Salón 204.

Carácter de la actividad: Individual/En equipo.

Participante(es): Emmanuel Alberto Gómez Cárdenas

**REPORTE DE ACTIVIDADES**

1. Utilice el repositorio en GitHub con el portafolio de prácticas de laboratorio que creó en la Meta 1.2.
2. Clone el repositorio en su computadora y agregue una carpeta de código para la Actividad de taller 1.3.2. Puede hacerlo utilizando una IDE (Visual Studio Code, PyCharm, etc.).
3. Consideremos  $U = \{a, b, c, d, e\}$  como conjunto universal y los subconjuntos  $A = \{a, b, d\}$ ,  $B = \{b, d, e\}$  y  $C = \{a, b, e\}$ .

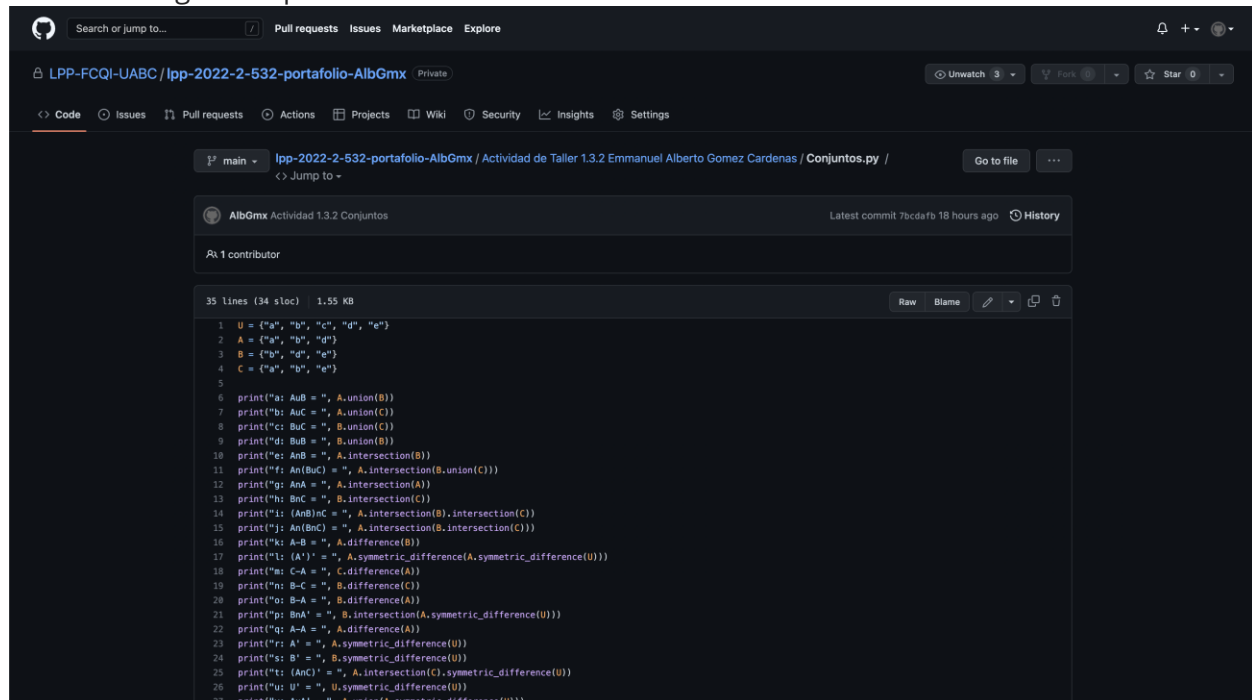
4. Haga un programa en Python que haga las siguientes operaciones de conjuntos:

- |                        |                  |                  |
|------------------------|------------------|------------------|
| a. $A \cup B$          | k. $A - B$       | u. $U'$          |
| b. $A \cup C$          | l. $(A')'$       | v. $A \cup A'$   |
| c. $B \cup C$          | m. $C - A$       | w. $A \cap A'$   |
| d. $B \cup B$          | n. $B - C$       | x. $\emptyset'$  |
| e. $A \cap B$          | o. $B - A$       | y. $A' \cup C'$  |
| f. $A \cup (B \cup C)$ | p. $B \cap A'$   | z. $(A \cup B)'$ |
| g. $A \cap A$          | q. $A - A$       | aa. $A' \cap B'$ |
| h. $B \cap C$          | r. $A'$          | bb. $(B - C)'$   |
| i. $(A \cap B) \cap C$ | s. $B'$          | cc. $A \cup B'$  |
| j. $A \cap (B \cap C)$ | t. $(A \cap C)'$ | dd. $B' - A'$    |

```

1  U = {"a", "b", "c", "d", "e"}
2  A = {"a", "b", "d"}
3  B = {"b", "d", "e"}
4  C = {"a", "b", "e"}
5
6  print("a: A ∪ B = ", A.union(B))
7  print("b: A ∪ C = ", A.union(C))
8  print("c: B ∪ C = ", B.union(C))
9  print("d: B ∪ B = ", B.union(B))
10 print("e: A ∩ B = ", A.intersection(B))
11 print("f: A ∩ (B ∪ C) = ", A.intersection(B.union(C)))
12 print("g: A ∩ A = ", A.intersection(A))
13 print("h: B ∩ C = ", B.intersection(C))
14 print("i: (A ∩ B) ∩ C = ", A.intersection(B).intersection(C))
15 print("j: A ∩ (B ∩ C) = ", A.intersection(B.intersection(C)))
16 print("k: A - B = ", A.difference(B))
17 print("l: (A')' = ", A.symmetric_difference(A.symmetric_difference(U)))
18 print("m: C - A = ", C.difference(A))
19 print("n: B - C = ", B.difference(C))
20 print("o: B - A = ", B.difference(A))
21 print("p: B ∩ A' = ", B.intersection(A.symmetric_difference(U)))
22 print("q: A - A = ", A.difference(A))
23 print("r: A' = ", A.symmetric_difference(U))
24 print("s: B' = ", B.symmetric_difference(U))
25 print("t: (A ∩ C)' = ", A.intersection(C).symmetric_difference(U))
26 print("u: U' = ", U.symmetric_difference(U))
27 print("v: A ∪ A' = ", A.union(A.symmetric_difference(U)))
28 print("w: A ∩ A' = ", A.intersection(A.symmetric_difference(U)))
29 print("x: ∅ = ", U.symmetric_difference(U))
30 print("y: A' ∪ C' = ", A.symmetric_difference(U).union(C.symmetric_difference(U)))
31 print("z: (A ∪ B)' = ", A.union(B).symmetric_difference(U))
32 print("aa: A' ∩ B' = ", A.intersection(B.symmetric_difference(U)))
33 print("ab: (B - C)' = ", B.difference(C).symmetric_difference(U))
34 print("ac: A ∪ B' = ", A.union(B.symmetric_difference(U)))
35 print("ad: B' - A' = ", B.symmetric_difference(U).difference(A.symmetric_difference(U)))
  
```

5. Respalde (*commit*) y suba (*push*) su código en el repositorio de GitHub para hacer la entrega de la práctica.



```
1 U = {"a", "b", "c", "d", "e"}
2 A = {"a", "b", "d"}
3 B = {"b", "d", "e"}
4 C = {"a", "b", "e"}
5
6 print("a: AuB = ", A.union(B))
7 print("b: AuC = ", A.union(C))
8 print("c: BuC = ", B.union(C))
9 print("d: BuB = ", B.union(B))
10 print("e: AnB = ", A.intersection(B))
11 print("f: An(BuC) = ", A.intersection(B.union(C)))
12 print("g: BnA = ", B.intersection(A))
13 print("h: BnC = ", B.intersection(C))
14 print("i: (AnB)nC = ", A.intersection(B).intersection(C))
15 print("j: An(BnC) = ", A.intersection(B.intersection(C)))
16 print("k: A-B = ", A.difference(B))
17 print("l: (A')' = ", A.symmetric_difference(A.symmetric_difference(U)))
18 print("m: C-A = ", C.difference(A))
19 print("n: B-C = ", B.difference(C))
20 print("o: B-A = ", B.difference(A))
21 print("p: BnA' = ", B.intersection(A.symmetric_difference(U)))
22 print("q: A-A = ", A.difference(A))
23 print("r: A' = ", A.symmetric_difference(U))
24 print("s: B' = ", B.symmetric_difference(U))
25 print("t: (AnC)' = ", A.intersection(C).symmetric_difference(U))
26 print("u: U' = ", U.symmetric_difference(U))
27 print("v: AuA' = ", A.union(A.symmetric_difference(U)))
```

## RESUMEN/REFLEXIÓN/CONCLUSIÓN

Al momento de realizar un programa que involucre una colección de datos, es necesario saber elegir el tipo de dato a utilizar ya que Python nos ofrece 4 opciones diferentes:  
Lista, Tupla, Conjunto y Diccionario.  
Cada uno con ofrece unas propiedades únicas, las cuales pueden ser clave al momento de realizar dicho programa. Entender las propiedades de cada colección y utilizarlas de manera correcta puede ofrecernos un aumento de eficiencia o simplemente un código más limpio.

Doy fe de que toda la información dada es completa y correcta.

Nombre y firma del alumno.

Gómez, Emmanuel A.