

Tarea 2

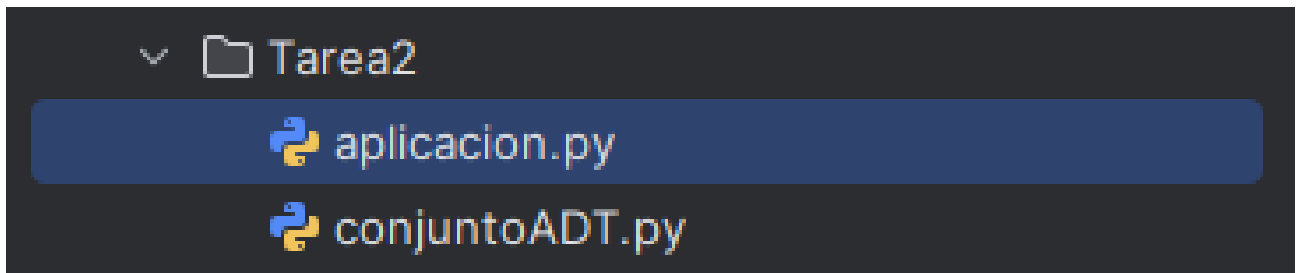
CONJUNTO ADT

Estructura de Datos
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

REALIZADO POR:

**Enrique Emiliano Cano
García**

Se divide en 2 archivos para una mejor visualización (conjuntoADT.py , aplicacion.py)



Se definen todos los métodos de la estructura de datos

```
1  class ConjuntoADT: 8 usages  👤 kno4
2      def __init__(self):  👤 kno4
3          self._elementos = []
4
5      def longitud(self): 5 usages (1 dynamic)  👤 kno4
6          return len(self._elementos)
7
8      def contiene(self, elemento): 8 usages (4 dynamic)  👤 kno4
9          return elemento in self._elementos
10
11     def agregar(self, elemento): 17 usages  👤 kno4
12         if not self.contiene(elemento):
13             self._elementos.append(elemento)
14
15     def eliminar(self, elemento): 1 usage  👤 kno4
16         if self.contiene(elemento):
17             self._elementos.remove(elemento)
18
19     def equals(self, otro_conjunto): 1 usage  👤 kno4
20         if self.longitud() != otro_conjunto.longitud():
21             return False
22         for elemento in self._elementos:
23             if not otro_conjunto.contiene(elemento):
24                 return False
25         return True
```

```

27     def es_subconjunto(self, otro_conjunto): 2 usages  ⓘ kno4
28         for elemento in self._elementos:
29             if not otro_conjunto.contiene(elemento):
30                 return False
31         return True
32
33     def union(self, otro_conjunto): 1 usage  ⓘ kno4
34         nuevo_conjunto = ConjuntoADT()
35         # Agregar todos los elementos del conjunto actual
36         nuevo_conjunto._elementos = self._elementos.copy()
37         # Agregar todos los elementos del otro conjunto si no están ya presentes
38         for elemento in otro_conjunto._elementos:
39             if not nuevo_conjunto.contiene(elemento):
40                 nuevo_conjunto.agregar(elemento)
41         return nuevo_conjunto
42
43     def interseccion(self, otro_conjunto): 1 usage  ⓘ kno4
44         nuevo_conjunto = ConjuntoADT()
45         for elemento in self._elementos:
46             if otro_conjunto.contiene(elemento):
47                 nuevo_conjunto.agregar(elemento)
48         return nuevo_conjunto
49
50     def diferencia(self, otro_conjunto): 1 usage  ⓘ kno4
51         nuevo_conjunto = ConjuntoADT()
52         for elemento in self._elementos:
53             if not otro_conjunto.contiene(elemento):
54                 nuevo_conjunto.agregar(elemento)
55         return nuevo_conjunto
56
57     def __str__(self): ⓘ kno4
58         return "{" + ", ".join(str(elemento) for elemento in self._elementos) + "}"
59

```

El archivo "aplicacion.py" contiene la aplicación con unos conjuntos básicos

```
1 from Tareas.Tarea2.conjuntoADT import ConjuntoADT
2
3 def main(): 1 usage ± kno4
4     #crea tres conjuntos
5     conjunto_a = ConjuntoADT()
6     conjunto_b = ConjuntoADT()
7     conjunto_c = ConjuntoADT()
8
9     #agrega elementos
10    conjunto_a.agregar("manzana")
11    conjunto_a.agregar(2.5)
12    conjunto_a.agregar((1, 2))
13    conjunto_a.agregar("pera")
14
15    conjunto_b.agregar("pera")
16    conjunto_b.agregar("naranja")
17    conjunto_b.agregar(3.14)
18    conjunto_b.agregar((3, 4))
19
20    conjunto_c.agregar("manzana")
21    conjunto_c.agregar(2.5)
22
23    #muestra los conjuntos iniciales
24    print(f"Conjunto A: {conjunto_a}")
25    print(f"Conjunto B: {conjunto_b}")
26    print(f"Conjunto C: {conjunto_c}")
27
28    #comprueba que pertenece a un conjunto
29    elemento = "pera"
30    print(f"¿Conjunto A contiene '{elemento}'? {'Si' if conjunto_a.contiene(elemento) else 'No'}")
31
32    #elimina de un conjunto (en este caso el "a")
33    conjunto_a.eliminar((1, 2))
34    print(f"Conjunto A después de eliminar (1, 2): {conjunto_a}")
35
36    #imprime la longitud de los conjuntos
37    print(f"Longitud de Conjunto A: {conjunto_a.longitud()}")
38    print(f"Longitud de Conjunto B: {conjunto_b.longitud()}")
39    print(f"Longitud de Conjunto C: {conjunto_c.longitud()}")
40
41    #verifica si dos conjuntos son iguales
42    conjunto_d = ConjuntoADT()
43    conjunto_d.agregar("pera")
44    conjunto_d.agregar("naranja")
45    conjunto_d.agregar(3.14)
46    conjunto_d.agregar((3, 4))
47
48    print(f"Conjunto B: {conjunto_b}")
49    print(f"Conjunto D: {conjunto_d}")
50    print(f"¿Conjunto B es igual a Conjunto D? {'Si' if conjunto_b.equals(conjunto_d) else 'No'}")
51
52    #comprueba si un conjunto es subconjunto de otro
53    print(f"¿Conjunto C es subconjunto de Conjunto A? {'Si' if conjunto_c.es_subconjunto(conjunto_a) else 'No'}")
54    print(f"¿Conjunto A es subconjunto de Conjunto C? {'Si' if conjunto_a.es_subconjunto(conjunto_c) else 'No'}")
55
56    #operaciones entre conjuntos
57    union_ab = conjunto_a.union(conjunto_b)
58    interseccion_ab = conjunto_a.interseccion(conjunto_b)
59    diferencia_ab = conjunto_a.diferencia(conjunto_b)
60
61    print(f"Unión de Conjunto A y Conjunto B: {union_ab}")
62    print(f"Intersección de Conjunto A y Conjunto B: {interseccion_ab}")
63    print(f"Diferencia de Conjunto A y Conjunto B: {diferencia_ab}")
64
65 if __name__ == "__main__":
66     main()
```