2025-1

Tarea 2

CONJUNTOADT

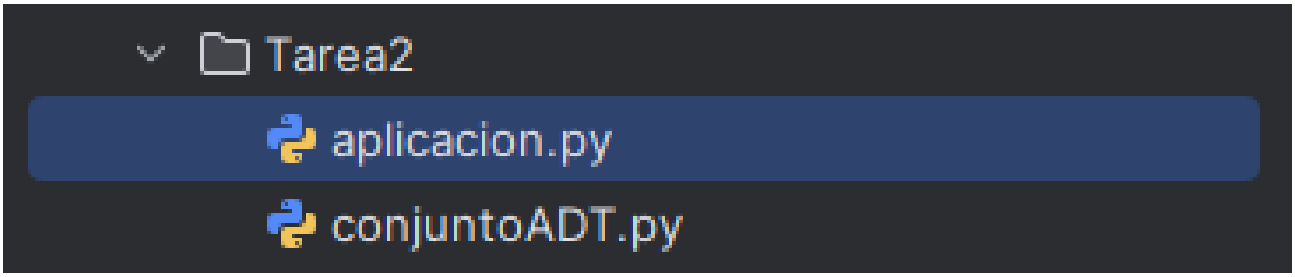
Estructura de Datos
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

REALIZADO POR:

Enrique Emiliano Cano García

22/08/2024

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



Se definen todos los métodos de la estructura de datos

```python
class ConjuntoADT:  8 usages  kno4
    def __init__(self):  kno4
        self._elementos = []

    def longitud(self):  5 usages (1 dynamic)  kno4
        return len(self._elementos)

    def contiene(self, elemento):  8 usages (4 dynamic)  kno4
        return elemento in self._elementos

    def agregar(self, elemento):  17 usages  kno4
        if not self.contiene(elemento):
            self._elementos.append(elemento)

    def eliminar(self, elemento):  1 usage  kno4
        if self.contiene(elemento):
            self._elementos.remove(elemento)

    def equals(self, otro_conjunto):  1 usage  kno4
        if self.longitud() != otro_conjunto.longitud():
            return False
        for elemento in self._elementos:
            if not otro_conjunto.contiene(elemento):
                return False
        return True
```

```python
    def es_subconjunto(self, otro_conjunto):  2 usages  kno4
        for elemento in self._elementos:
            if not otro_conjunto.contiene(elemento):
                return False
        return True

    def union(self, otro_conjunto):  1 usage  kno4
        nuevo_conjunto = ConjuntoADT()
        # Agregar todos los elementos del conjunto actual
        nuevo_conjunto._elementos = self._elementos.copy()
        # Agregar todos los elementos del otro conjunto si no están ya presentes
        for elemento in otro_conjunto._elementos:
            if not nuevo_conjunto.contiene(elemento):
                nuevo_conjunto.agregar(elemento)
        return nuevo_conjunto

    def interseccion(self, otro_conjunto):  1 usage  kno4
        nuevo_conjunto = ConjuntoADT()
        for elemento in self._elementos:
            if otro_conjunto.contiene(elemento):
                nuevo_conjunto.agregar(elemento)
        return nuevo_conjunto

    def diferencia(self, otro_conjunto):  1 usage  kno4
        nuevo_conjunto = ConjuntoADT()
        for elemento in self._elementos:
            if not otro_conjunto.contiene(elemento):
                nuevo_conjunto.agregar(elemento)
        return nuevo_conjunto

    def __str__(self):  kno4
        return "{" + ", ".join(str(elemento) for elemento in self._elementos) + "}"

```

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

```python
from Tareas.Tarea2.conjuntoADT import ConjuntoADT

def main():  1 usage  kno4
    #crea tres conjuntos
    conjunto_a = ConjuntoADT()
    conjunto_b = ConjuntoADT()
    conjunto_c = ConjuntoADT()

    #agrega elementos
    conjunto_a.agregar("manzana")
    conjunto_a.agregar(2.5)
    conjunto_a.agregar((1, 2))
    conjunto_a.agregar("pera")

    conjunto_b.agregar("pera")
    conjunto_b.agregar("naranja")
    conjunto_b.agregar(3.14)
    conjunto_b.agregar((3, 4))

    conjunto_c.agregar("manzana")
    conjunto_c.agregar(2.5)

    #muestra los conjuntos iniciales
    print(f"Conjunto A: {conjunto_a}")
    print(f"Conjunto B: {conjunto_b}")
    print(f"Conjunto C: {conjunto_c}")

    #comprueba que pertenece a un conjunto
    elemento = "pera"
    print(f"¿Conjunto A contiene '{elemento}'? {'Sí' if conjunto_a.contiene(elemento) else 'No'}")

    #elimina de un conjunto (en este caso el "a")
    conjunto_a.eliminar((1, 2))
    print(f"Conjunto A después de eliminar (1, 2): {conjunto_a}")

    #imprime la longitud de los conjuntos
    print(f"Longitud de Conjunto A: {conjunto_a.longitud()}")
    print(f"Longitud de Conjunto B: {conjunto_b.longitud()}")
    print(f"Longitud de Conjunto C: {conjunto_c.longitud()}")

    #verifica si dos conjutos son iguales
    conjunto_d = ConjuntoADT()
    conjunto_d.agregar("pera")
    conjunto_d.agregar("naranja")
    conjunto_d.agregar(3.14)
    conjunto_d.agregar((3, 4))
    print(f"Conjunto B: {conjunto_b}")
    print(f"Conjunto D: {conjunto_d}")
    print(f"¿Conjunto B es igual a Conjunto D? {'Sí' if conjunto_b.equals(conjunto_d) else 'No'}")

    #comprueba si un conjunto es subconjunto de otro
    print(f"¿Conjunto C es subconjunto de Conjunto A? {'Sí' if conjunto_c.es_subconjunto(conjunto_a) else 'No'}")
    print(f"¿Conjunto A es subconjunto de Conjunto C? {'Sí' if conjunto_a.es_subconjunto(conjunto_c) else 'No'}")

    #operaciones entre conjuntos
    union_ab = conjunto_a.union(conjunto_b)
    interseccion_ab = conjunto_a.interseccion(conjunto_b)
    diferencia_ab = conjunto_a.diferencia(conjunto_b)

    print(f"Unión de Conjunto A y Conjunto B: {union_ab}")
    print(f"Intersección de Conjunto A y Conjunto B: {interseccion_ab}")
    print(f"Diferencia de Conjunto A y Conjunto B: {diferencia_ab}")

if __name__ == "__main__":
    main()
```

Corriendo el archivo “aplicacion.py” podemos ver las operaciones y cambios que se van haciendo

```
Conjunto A: {manzana, 2.5, (1, 2), pera}
Conjunto B: {pera, naranja, 3.14, (3, 4)}
Conjunto C: {manzana, 2.5}
¿Conjunto A contiene 'pera'? Sí
Conjunto A después de eliminar (1, 2): {manzana, 2.5, pera}
Longitud de Conjunto A: 3
Longitud de Conjunto B: 4
Longitud de Conjunto C: 2
Conjunto B: {pera, naranja, 3.14, (3, 4)}
Conjunto D: {pera, naranja, 3.14, (3, 4)}
¿Conjunto B es igual a Conjunto D? Sí
¿Conjunto C es subconjunto de Conjunto A? Sí
¿Conjunto A es subconjunto de Conjunto C? No
Unión de Conjunto A y Conjunto B: {manzana, 2.5, pera, naranja, 3.14, (3, 4)}
Intersección de Conjunto A y Conjunto B: {pera}
Diferencia de Conjunto A y Conjunto B: {manzana, 2.5}

Process finished with exit code 0
```