Python - 14/03/2024 - Conjuntos

Fecha de creación	@14 de marzo de 2024 7:48
Clase	Python - Complementario
Tipo	Clase
Trimestre	2

Conjuntos:

Son colecciones desordenadas (El orden en que los ingrese no significa que en ese mismo orden se almacenen los datos) y que no aceptan valores repetidos.

Se crean con llaves {} llamando la función set.
Permiten cambiar su contenido. Dado que no poseen un orden.

Objetivo:

El objetivo es saber que elementos pertenecen al conjunto.

Para adicionar elementos al conjunto utilizamos el metodo add().
Porque cuando no puedo garantizar de que vaya quedar al final, ya que varía el orden del conjunto.

Método add():

Es para añadir algo al conjunto.

```
#EL METODO ADD SE UTILIZA SE LA SIGUIENTE MANERA:

mi_conjunto = {2,3,4,5,6,8}
```

```
    mi_conjunto.add(30)


    print (mi_conjunto) # Se va a añadir el 30 a cualquier pa
```

Método set:

El método sirve para determinar que es un conjunto:

```
mi_conjunto = set(range(1,4)) #El set es para que se vuelva u
print (mi_conjunto)
```

Método para saber si un número está dentro del conjunto:

Es una palabra observada, sirve para conocer si un número o string está dentro del conjunto

```
mi_conjunto = {2,5,6,2,3,4,2,6}
print(mi_conjunto)

num1 = (int(input("Ingrese un numero")))
print(num1)

print(f"Está el numero {num1} en el conjunto {mi_conjunto}?:
# Dirá si es falso o true, conociendo si el número está dentr
```

Recorrido de conjuntos.

Se hace de la misma forma, pero con otro tipo de colección.

Se va moviendo en la posición empieza desde 0.

```
mi_conjunto = {2,43,5,67,8,23,45}
for valor in mi_conjunto:
print(valor)
```

Métodos en conjuntos:

- Método discard():

El método discard sirve para quitar un elemento del conjunto:

```
c = {5,3,4,5,6,7,8}
c.discard(5)
print(c)
# Me va a quitar el número 5
# Ejecución: {3, 4, 6, 7, 8}
```

Método isdisjoint():

El método sirve para mirar que tiene un elemento, y que no tiene el otro. (La intersección de conjuntos es vacía):

- Si es False entonces es porque tienen elementos en común.
- Si es True entonces es porque no tienen elementos en común

```


# Cuando tienen ningún elemento repetido es False.
# Cuando no tienen algún elemento en común es True.

c1 = {2,5,7}
c2 = {9,3,1}
print(f"c1 disjoint c2 = {c1.isdisjoint(c2)}")
c3 = {9,3,5}
print(f"c1 disjoint c3 = {c1.isdisjoint(c3)}")


```

```
# Ejecución:
# c1 disjoint c2 = True
# c1 disjoint c3 = False
```

Método issubet():

El método sirve para comprobar si el conjunto es subconjunto de otro conjuntos, es decir, so sis items se encuentran todos dentro de otro.

Issubet → Es para por debajo. El que tenga menor cantidad de elementos.

Tienen que estar ciertos elementos si o sí, para poder considerar True como subconjunto.

```
a1 = {2,5,7}
a2 = {2,5}

print(f"A2 es Subconjunto de A1 {a2.issubset(a1)}")

#Ejecución:
# A2 es Subconjunto de A1 True

#Pero si se coloca al revés da False:
# A1 es Subconjunto de A2 False.
```

Método issuperset():

El métodos es para comprobar si es contenedor de otro subconjunto. Es el que tenga mayor cantidad de elementos dará siempre True.

```
a1 = {2,5,7}
a2 = {2,5}

print(f"A es Subconjunto de A2 {a1.issuperset(a2)}")
```

```
#Ejecución
# A es Subconjunto de A2 True
```

Método union():

El método es para unir un conjunto a otro y devuelve el resultado del nuevo conjunto:

```
a1 = {2,5,7}
a2 = {2,5,4}
print(f"La unión de los dos es: {a1.union(a2)}")

#Ejecución:
# La unión de los dos es: {2, 4, 5, 7}
```

Método difference():

El método encuentra los elementos no comunes entre dos conjuntos:

```
a1 = {2,5,7}
a2 = {2,5,4}

print(f"La diferencia de A2 Y A1 {a2.difference(a1)}")
# Básicamente sería ¿Qué hay en A2 que no esté en A1? = 4.
print(f"La diferencia de A1 y A2  {a1.difference(a2)}")
# O también ¿Qué hay en A1 que no esté en A2? = 7.

#Ejecución:
# La diferencia de A1 Y A2 {4}
```

Método symmetric_difference():

Devuelve los elementos simétricamente diferentes entre dos conjuntos, es decir, todos los elementos que no concuerdan entre los dos conjuntos:

```
c1 = {2,5,7}
c2 = {2,5,3}

print(f"C1 symmetric_differences C2 = {c1.symmetric_differenc

#Ejecución:
# C1 symmetric_differences C2 = {3, 7}
```

Método intersection():

Devuelve un conjunto con los elementos comunes en dos conjuntos:

```
t1 = {2,7,5}
t2 = {2,5,3}
print(f"c1 INTERSECTION C2 = {t1.intersection(t2)}")


#Ejecución:
#c1 INTERSECTION C2 = {2, 5}
```