Status	Finished
Started	Monday, 2 December 2024, 6:07 PM
Completed	Monday, 2 December 2024, 6:13 PM
Duration	5 mins 56 secs
Marks	4.00/4.00
Grade	**10.00** out of 10.00 (**100**%)

Information

El tipo *set*

El tipo *set* de Python se usa para representar colecciones de elementos **sin orden** y **sin repetición**; básicamente, implementa el concepto matemático de conjunto. Un conjunto vacío se crea usando la función constructora *set()*:

```
s = set()
```

Un conjunto no vacío se puede crear escribiendo sus elementos encerrados entre llaves, separados por coma:

```
elementos = {"agua", "aire", "fuego", "tierra"}
print(elementos)
```

```
{'agua', 'tierra', 'aire', 'fuego'}
```

>>> Nótese que el orden de los elementos no está definido y que, por tanto, su visualización no tiene por qué coincidir con el orden en que se escribieron al crear el conjunto.

Los elementos no pueden ser de un tipo mutable. En el siguiente ejemplo, el tercer elemento es una lista, lo que supone un error:

```
elementos = {"agua", "aire", ["fuego", "tierra"]}
```

Traceback (most recent call last):
  File "/home/p11049/p1.py", line 1, in <module>
    elementos = {"agua", "aire", ["fuego", "tierra"]}
TypeError: unhashable type: 'list'

Otra forma de crear un conjunto es pasando una colección de elementos (tupla, lista, string, diccionario, otro conjunto) a la función constructora:

```
elementos = set(["agua", "aire", "fuego", "tierra"])
print(elementos)
caracteres = set("esto es una string")
print(caracteres)
```

{'agua', 'fuego', 'aire', 'tierra'}
{'a', 's', 'r', 'e', 'g', ' ', 't', 'o', 'i', 'n', 'u'}
>>> Nótese que en el conjunto caracteres no hay elementos repetidos, a pesar de que sí los había en la string que se pasó para crearlo.

En caso de que se pase un diccionario, se crea un conjunto con las claves del mismo (no las parejas clave/valor). Si se pasa un conjunto, se crea una copia del mismo.

Otra forma de copiar un conjunto es usando el método *.copy()*:

```
elementos = set(["agua", "aire", "fuego", "tierra"])
copia = elementos.copy()
```

Su cardinalidad (número de elementos de un conjunto) puede averiguarse usando la función *len()*, como en cualquier otra colección de Python:

```
elementos = set(["agua", "aire", "fuego", "tierra"])
print(len(elementos))
```

```
4
```

Question 1

Complete

Mark 1.00 out of 1.00

¿Qué valor muestra el siguiente código?

```
elementos = set([21, 13, 23, 45, 67, 22, 13])
print(len(elementos))
```

Answer: 6

Information

Modificación de conjuntos

Los conjuntos de Python son mutables. A un conjunto existente se le pueden añadir elementos individuales usando la operación *.add()*:

```
elementos = {"agua", "aire", "fuego", "tierra"}
elementos.add("eter")
print(elementos)
```

```
{'tierra', 'eter', 'agua', 'aire', 'fuego'}
```

También se le pueden añadir elementos de una colección usando la operación *.update()*:

```
elementos = {"agua", "aire"}
elementos.update(["fuego", "tierra"])
print(elementos)
```

```
{'fuego', 'aire', 'tierra', 'agua'}
```

Por supuesto, ni *.add()* ni *.update()* añaden elementos repetidos.

Para eliminar un elemento concreto, se pueden usar las operaciones *.remove()* o *.discard()*:

```
elementos = {'fuego', 'aire', 'tierra', 'agua'}
elementos.remove("agua")
print(elementos)
```

```
{'tierra', 'fuego', 'aire'}
```

```
elementos.discard("aire")
print(elementos)
```

```
{'tierra', 'fuego'}
```

La diferencia entre *.remove()* y *.discard()* es que el primero produce un error si el elemento a eliminar no está en el conjunto, mientras que el segundo lo ignora y no hace nada:

```
elementos.remove("eter")
```

```
Traceback (most recent call last):
  File "p1.py", line 6, in <module>
    elementos.remove("eter")
KeyError: 'eter'
```

El método *.pop()* elimina un elemento al azar:

```
elementos = {'fuego', 'aire', 'tierra', 'agua'}
elementos.pop()
print(elementos)
```

```
{'agua', 'tierra', 'fuego'}
```

El método *.clear()* vacía el objeto conjunto en cuestión, eliminando todos sus elementos:

```
elementos = {'fuego', 'aire', 'tierra', 'agua'}
elementos.clear()
print(elementos)
```

```
set()
```

Obsérvese la especial visualización de un conjunto vacío. Nótese que {} representaría un diccionario vacío.

Question 2

Complete

Mark 1.00 out of 1.00

¿Qué método debemos usar para borrar un elemento de un *set* (conjunto) si queremos evitar que ocurra un error en caso de que el elemento que queremos borrar no esté en el conjunto?

- .remove()
- .discard()

Information

Pertenencia e inclusión

Para comprobar si un valor pertenece a un conjunto (es uno de los elementos del conjunto) usamos el operador **`in`**:

```
elementos = {'fuego', 'aire', 'tierra', 'agua'}
print('aire' in elementos)
```

```
True
```

```
print('eter' in elementos)
```

```
False
```

Los operadores relacionales <, <=, >, >= permiten saber si un conjunto está incluido en otro (es un subconjunto del otro):

```
alfabeto = set("abcdefghijklmnñopqrstuvwxyz")
vocales = set("aeiou")
print(vocales < alfabeto)
```

```
True
```

Los operadores < y > requieren que sea un subconjunto estricto (se corresponden con las relaciones matemáticas ⊂ y ⊃), mientras que <= y >= (que se corresponden con ⊆ y ⊇) admiten también los "subconjuntos" que son iguales al conjunto que los "contiene" (subconjuntos impropios).

Entre conjuntos también se pueden usar los operadores relacionales == y !=, que determinan si dos conjuntos son iguales o distintos; dos conjuntos son iguales si tienen exactamente los mismos elementos (recuérdese que los elementos no tienen orden).

Question 3

Complete

Mark 1.00 out of 1.00

¿Cuál es el resultado del siguiente código?

```
set1 = set([12, 23, 45, 67, 89])
set2 = set([23, 45, 89, 67, 12])
print(set2 <= set1)
```

- True
- False

Information

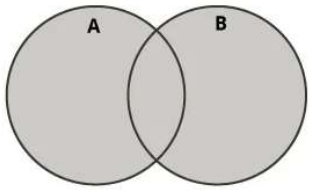
Álgebra de conjuntos

Los conjuntos disponen de cuatro operaciones básicas: unión, intersección, diferencia y diferencia simétrica.

La **unión** de dos conjuntos resulta en un conjunto que tiene todos los elementos que estén en alguno(s) de los conjuntos originales, sin repetición. En Python, la unión se hace con el operador **|** (barra vertical; para escribirla, normalmente la compondremos con la tecla del 1):

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
unión = set1 | set2
print(unión)
```

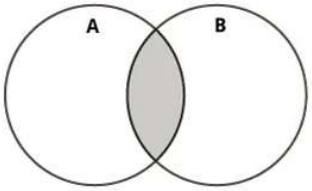
```
{1, 2, 3, 4, 5}
```



La **intersección** contiene únicamente los elementos comunes a ambos conjuntos y se hace con el operador **&**:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersección = set1 & set2
print(intersección)
```

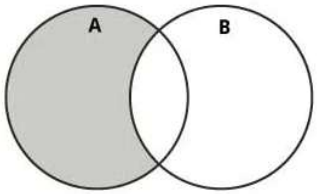
```
{3}
```



Para la **diferencia** se utiliza el operador **–** y su resultado es el conjunto formado por todos los elementos del primer conjunto que no están en el segundo:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
diferencia = set1 - set2
print(diferencia)
```

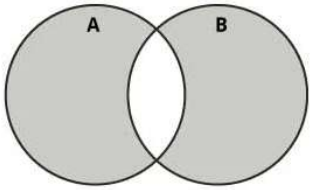
```
{1, 2}
```



La **diferencia simétrica** comprende todos los elementos del primer conjunto que no están en el segundo y los del segundo que no están en el primero (los elementos no comunes), y se hace con el operador ^:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
diferencia = set1 ^ set2
print(diferencia)
```

```
{1, 2, 4, 5}
```



Question 4

Complete

Mark 1.00 out of 1.00

Dados los siguientes conjuntos:

```
set1 = {'a', 'x', 'v', 'b', 'r', 't'}
set2 = {'x', 't', 'k', 'o', 'h', 'r'}
```

Empareje cada operación con su resultado.

`set1 | set2`	{'k', 'a', 'b', 'r', 'v', 'o', 'x', 'h', 't'}
`set1 & set2`	{'x', 't', 'r'}
`set1 - set2`	{'b', 'a', 'v'}
`set1 ^ set2`	{'k', 'a', 'b', 'v', 'o', 'h'}