Set examine methods

isdisjoint(*other*)
Return True if the set has no elements in common with *other*. Sets are disjoint if and only if their intersection is the empty set.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> A.isdisjoint(B)
False
>>> C={10,20,30}
>>> A.isdisjoint(B)
False
>>> A.isdisjoint(C)
True
>>> python_stud={"naresh","ramesh","kishore"}
>>> java_stud={"rajesh","kiran","karan"}
>>> python_stud.isdisjoint(java_stud)
True
>>> java_stud.add("naresh")
>>> print(java_stud)
{'kiran', 'naresh', 'rajesh', 'karan'}
>>> python_stud.isdisjoint(java_stud)
False
```

issubset(*other*)
set <= other
Test whether every element in the set is in *other*.

```
>>> A={1,2,3}
>>> B={1,2,3,4,5}
>>> A.issubset(B)
True
>>> C={5,6,7,8,9}
>>> A.issubset(C)
False
```

issuperset(*other*)
set >= other
Test whether every element in *other* is in the set.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3}
>>> A.issuperset(B)
True
>>> C={1,2,5,6}
>>> A.issuperset(C)
False
>>> A>=B
True
```

https://www.hackerrank.com/challenges/py-check-subset/problem?isFullScreen=false

```
T=int(input())
for i in range(T):
    n=int(input())
    A=set(map(int,input().split()[:n]))
    m=int(input())
    B=set(map(int,input().split()[:m]))
    print(A.issubset(B))
```

https://www.hackerrank.com/challenges/py-check-strict-superset/problem?isFullScreen=false

```
A=set(map(int,input().split()))
N=int(input())
result=True
for i in range(N):
    B=set(map(int,input().split()))
    if not A.issuperset(B):
        result=False
        break


print(result)
```

https://www.hackerrank.com/challenges/py-the-captains-room/problem?isFullScreen=false

```
K=int(input())
A=list(map(int,input().split()))
B=set(A)

for x in B:
    c=A.count(x)
    print(x,c)
    if c==1:
        print(x)
        break
```

frozenset

frozenset is an immutable set, after creating frozenset changes cannot done (OR) frozenset does not support the following mutable operations.

1. add()
2. remove()
3. discard()
4. clear()
5. update()
6. pop()
7. difference_update()
8. intersection_update()
9. symmetric_difference_update()

In application development frozenset is used,

1. To represent an immutable set
2. To represent nested set

How to create frozenset?

Frozenset is created in different ways,

1. frozenset() : Create empty frozenset
2. frozenset(iterable): Create frozenset using elements of iterable/collection

```
>>> A=frozenset()
>>> print(A)
frozenset()
>>> A.add(10)
Traceback (most recent call last):
```

```
  File "<pyshell#27>", line 1, in <module>
    A.add(10)
AttributeError: 'frozenset' object has no attribute 'add'
>>> A=frozenset(range(10,60,10))
>>> print(A)
frozenset({40, 10, 50, 20, 30})
>>> A.remove(10)
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    A.remove(10)
AttributeError: 'frozenset' object has no attribute 'remove'
B=frozenset("PYTHON")
print(B)
frozenset({'O', 'Y', 'P', 'N', 'T', 'H'})
>>> C=frozenset("JAVA")
>>> print(C)
frozenset({'J', 'A', 'V'})
>>> D=frozenset([1,2,3,4,5])
>>> print(D)
frozenset({1, 2, 3, 4, 5})
>>>
```

Example:

```
A=frozenset(range(10,110,10))
for value in A:
    print(value,end=' ')
```

Output

```
100 70 40 10 80 50 20 90 60 30
```

Example:

```
A={frozenset({1,2,3}),frozenset({4,5,6})}
print(A)
for x in A:
    print(x)

for x in A:
    for y in x:
        print(y,end=' ')
    print()
```

Output
{frozenset({1, 2, 3}), frozenset({4, 5, 6})}
frozenset({1, 2, 3})
frozenset({4, 5, 6})
1 2 3
4 5 6

What is difference between list and set?

List	**Set**
List is ordered collection	Set is unordered collection
List insertion is preserved	Set insertion order is not preserved
List is linear data structure	Set is non linear data structure (hashing)
List allows duplicate values/elements	Set does not allows duplicate values/elements
List support indexing and slicing	Set does not support indexing and slicing
List allows any type of objects/elements (mutable, immutable)	Set allows only immutable objects/elements
List is created using []	Set is created using {}
In application development list is used to group individual objects where duplicates are allowed, reading is done sequentially and randomly	In application development set is used to group individual objects where duplicates are not allowed, mathematical set operations, membership testing and removing duplicates from sequences

all(iterable)
Return True if all elements of the iterable are true (or if the iterable is empty).

any(iterable)
Return True if any element of the iterable is true. If the iterable is empty, return False.

```
>>> A=[1,2,3,4,5]
```

```
>>> all(A)
True
>>> B=[1,2,3,0]
>>> all(B)
False
>>> C=[1,2,3,4]
>>> any(C)
True
>>> D=[1,0,0,0,10]
>>> any(D)
True
```