Sets

A set is an unordered collection of items. Every element is unique (no duplicates) and must be immutable (which cannot be changed).However, the set itself is mutable. We can add or remove items from it.

A set is created by placing all the items (elements) inside curly braces {}, separated by comma or by using the built-in function `set()`.It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have a mutable element, like list, set or dictionary, as its element.

Example

```
# set of integers

my_set = {1, 2, 3}

print(my_set)

# set of mixed datatypes

my_set = {1.0, "Hello", (1, 2, 3)}

print(my_set)
```

Creating an empty set

Empty curly braces {} will make an empty dictionary in Python. To make a set without any elements we use the `set()` function without any argument.

Example

```
# initialize a with {}

a = {}

# check data type of a

# Output: <class 'dict'>

print(type(a))

# initialize a with set()

a = set()

# check data type of a

# Output: <class 'set'>

print(type(a))
```

How to change a set in Python?

Sets are mutable. But since they are unordered, indexing have no meaning.We cannot access or change an element of set using indexing or slicing. Set does not support it.We can add single element using the `add()` method and multiple elements using the `update()` method. The `update()` method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

Example

```
# initialize my_set
```

```
my_set = {1,3}
print(my_set)
# add an element
# Output: {1, 2, 3}
my_set.add(2)
print(my_set)
# add multiple elements
# Output: {1, 2, 3, 4}
my_set.update([2,3,4])
print(my_set)
# add list and set
# Output: {1, 2, 3, 4, 5, 6, 8}
my_set.update([4,5], {1,6,8})
print(my_set)
```

remove elements from a set

A particular item can be removed from set using methods, `discard()` and `remove()`.The only difference between the two is that, while using `discard()` if the item does not exist in the set, it remains unchanged. But `remove()` will raise an error in such condition.we can remove and return an item using the `pop()` method.Set being unordered, there is no way of determining which item will be popped. It is completely arbitrary.We can also remove all items from a set using `clear()`.

Example

```
# initialize my_set
my_set = {1, 3, 4, 5, 6}
print(my_set)
# discard an element
# Output: {1, 3, 5, 6}
my_set.discard(4)
print(my_set)
# remove an element
# Output: {1, 3, 5}
my_set.remove(6)
print(my_set)
# discard an element
# not present in my_set
# Output: {1, 3, 5}
```

```
my_set.discard(2)

print(my_set)

my_set.pop()

print(my_set)

# clear my_set

#Output: set()

my_set.clear()

print(my_set)
```

Python Set Operations

Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. We can do this with operators or methods.

Set Union

Union of *A* and *B* is a set of all elements from both sets. Union is performed using `|` operator. Same can be accomplished using the method `union()`.

Example

```
# initialize A and B

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

# use | operator

# Output: {1, 2, 3, 4, 5, 6, 7, 8}

print(A | B)

print(A.union(B))

print(B.union(A))
```

Set Intersection

Intersection of *A* and *B* is a set of elements that are common in both sets.Intersection is performed using & operator. Same can be accomplished using the method `intersection()`.

Example

```
# initialize A and B

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

# use & operator

# Output: {4, 5}

print(A & B)

print(A.intersection(B))

print(B.intersection(A))
```

Set Difference

Difference of *A* and *B* (*A* - *B*) is a set of elements that are only in *A* but not in *B*. Similarly, *B* - *A* is a set of element in *B* but not in *A*. Difference is performed using `-` operator. Same can be accomplished using the method `difference()`.

Example

```
# initialize A and B

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

# use - operator on A

# Output: {1, 2, 3}

print(A - B)

print(A.difference(B))

#Output:{8, 6, 7}

print(B-A)

print(B.difference(A))
```

Set Symmetric Difference

Symmetric Difference of *A* and *B* is a set of elements in both *A* and *B* except those that are common in both.Symmetric difference is performed using `^` operator. Same can be accomplished using the method `symmetric_difference()`.

Example

```
# initialize A and B

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

# use ^ operator

# Output: {1, 2, 3, 6, 7, 8}

print(A ^ B)

print(A.symmetric_difference(B))

print(B.symmetric_difference(A))
```

Built-in Functions with Set

all()	Return `True` if all elements of the set are true (or if the set is empty).
any()	Return `True` if any element of the set is true. If the set is empty, return `False.`
enumerate()	Return an enumerate object. It contains the index and value of all the items of set as a pair.
len()	Return the length (the number of items) in the set.
max()	Return the largest item in the set.
min()	Return the smallest item in the set.
sorted()	Return a new sorted list from elements in the set(does not sort the set itself).

sum()	Retrun the sum of all elements in the set.

Example

A = {1, 2, 3, 4, 5}

Example:

```
A = {1, 2, 3, 4, 5}
print(all(A))
print(any(A))
print(list(enumerate(A)))
print(len(A))
print(max(A))
print(min(A))
print(sorted(A))
print(sum(A))
```

Create a set by taking user input

```
myset=set()
n=int(input("Enter no. of elements"))
for i in range(1,n+1):
        myset.add(int(input("Enter any num")))
print(myset)
```