- **Union**: `set1 | set2` or `set1.union(set2)` returns a new set containing all elements from both sets.
- **Intersection: `set1 & set2` or `set1.intersection(set2)` returns a new set containing elements common to both sets.
- **Difference**: `set1 - set2` or `set1.difference(set2)` returns a new set containing elements in `set1` that are not in `set2`.
- **Symmetric Difference**: `set1 ^ set2` or `set1.symmetric_difference(set2)` returns a new set containing elements that are in either `set1` or `set2`, but not both.
- **Subset**: `set1 <= set2` or `set1.issubset(set2)` returns `True` if `set1` is a subset of `set2`, `False` otherwise.
- **Superset**: `set1 >= set2` or `set1.issuperset(set2)` returns `True` if `set1` is a superset of `set2`, `False` otherwise.
- **Disjoint**: `set1.isdisjoint(set2)` returns `True` if `set1` and `set2` have no elements in common, `False` otherwise.
- **Remove element**: `set1.remove(element)` or `set1.discard(element)` removes an element from `set1`. `remove()` raises a `KeyError` if the element is not found, while `discard()` does not raise an error.

```python
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

print(set1 | set2)  # Union: {1, 2, 3, 4, 5, 6}
print(set1 & set2)  # Intersection: {3, 4}
print(set1 - set2)  # Difference: {1, 2}
print(set1 ^ set2)  # Symmetric Difference: {1, 2, 5, 6}
print(set1 <= set2)  # Subset: False
print(set1 >= set2)  # Superset: False
print(set1.isdisjoint(set2))  # Disjoint: False

set1.add(7)
print(set1)  # {1, 2, 3, 4, 7}

set1.remove(4)
print(set1)  # {1, 2, 3, 7}

set1.clear()
print(set1)  # {}
```

```python
import math

class Point2d(object):
    def __init__(self, x0=0, y0=0):
        self.x = x0
        self.y = y0
    def magnitude(self):
        return math.sqrt(self.x**2 + self.y**2)
    def dist(self, o):
        return math.sqrt((self.x - o.x)**2 + (self.y -
o.y)**2)
    def __sub__(self,o):
        return Point2d(self.x-o.x, self.y-o.y)
    def __mul__(self,s):
        return Point2d(s*self.x, s*self.y)
    def __eq__(self,o):
        return self.x==o.x and self.y==o.y
    def __lt__(self,o):
        """This is the less than operator"""
    def __str__(self):
        return "({},{})".format(self.x, self.y)
```

```python
# Find the values that are in exactly one of the three sets
s = (s1 ^ s2 ^ s3) - (s1 & s2) - (s1 & s3) - (s2 & s3)
s = (s1 - s2 - s3) | (s2 - s1 - s3) | (s3 - s1 - s2)
# Find the values that are in exactly two of the three sets
s = (s1 & s2 | s2 & s3 | s1 & s3) - (s1 & s2 & s3)
```

```python
s = {1, 2, 3}

s.remove(2) # Output: {1, 3}
s.discard(4) # No error raised
s.add(4) # Output: {1, 2, 3, 4}
s.update([4, 5, 6]) # Output: {1, 2, 3, 4, 5, 6}
```

```python
# Dictionary
d = {'name': 'John', 'age': 25, 'city': 'New York'}

d.keys()
d.values()
d.items()
d['city'] = 'New York'  # Adding a new key-value pair
d['age'] = 26  # Updating the value of an existing key
del d['age'] # Removing 'age': 25
city = d.pop('city')  # Removing 'city': 'New York' and retrieving
the value
print(d)  # Output: {'name': 'John'}
print(city)  # Output: 'New York'
```

```python
def merge_dict(D1, D2):
    D = {}

    # Merge keys from D1
    for name, numbers in D1.items():
        D[name] = numbers.copy()

    # Merge keys from D2
    for name, numbers in D2.items():
        if name in D:
            D[name] |= numbers
        else:
            D[name] = numbers.copy()

    return D
```

```python
a = True
b = False

print(a and b)   # False
print(a or b)    # True
print(not a)     # False
print(a ^ b)     # True
print(not (a and b))  # True (NAND)
print(not (a or b))   # False (NOR)
print(not (a ^ b))    # False (XNOR)
print(not a or b)     # False (Implication)
```

Syntax: `separator.join(list)`
Syntax: `[item for item in list if item != '']`
Syntax: `max(list)`
Syntax: `list.append(item)`
Syntax: `list.insert(index, item)`
Syntax: `list.remove(item)`
Syntax: `list.index(element)`
Syntax: `list.pop(index)`
Syntax: `string.find(substring)`
Syntax: `list.count(element)`
Syntax: `list[start:end:step]`
Syntax: `[i for i, x in enumerate(list) if x == element]`
Syntax: `list(range(start, stop, step))`

```python
a = [1, 2, 3, 4, 5, 6, 7, 8]
print(a[:5])    # prints [1, 2, 3, 4, 5]
print(a[2:])    # prints [3, 4, 5, 6, 7, 8]
print(a[2:5])    # prints [3, 4, 5]
print(a[2:7:2])    # prints [3, 5, 7]
print(a[::-1])    # prints [8, 7, 6, 5, 4, 3, 2, 1]
```

```python
# Deduplicate
list(set(L))
```

```python
def add_review(rest_reviews, new_review, rest_name):
    if rest_name in rest_reviews:
        rest_reviews[rest_name].append(new_review)
    else:
        rest_reviews[rest_name] = [new_review]
```

```python
def find_name( contacts, number):
    for name in contacts.keys():
        if number in contacts[name]:
            return name
    return 'Unknown'
```