```
clear = """def clear() -> None
Remove all elements from this set."""

union = """def union(*s: Iterable[_T]) -> Set[_T]
Return the union of sets as a new set.

(i.e. all elements that are in either set.)"""

add = """def add(element: _T) -> None
Add an element to a set.

This has no effect if the element is already present."""

copy = """def copy() -> Set[_T]
Return a shallow copy of a set."""

remove = """def remove(element: _T) -> None
Remove an element from a set; it must be a member.

If the element is not a member, raise a KeyError."""

discard = """def discard(element: _T) -> None
Remove an element from a set if it is a member.

Unlike set.remove(), the discard() method does not raise
an exception when an element is missing from the set."""

pop = """def pop() -> _T
Remove and return an arbitrary set element.
Raises KeyError if the set is empty."""

update = """def update(*s: Iterable[_T]) -> None
Update a set with the union of itself and others."""

difference = """def difference(*s: Iterable[Any]) -> Set[_T]
Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)"""

difference_update = """def difference_update(*s: Iterable[Any]) -> None
Remove all elements of another set from this set."""

intersection = """def intersection(*s: Iterable[Any]) -> Set[_T]
Return the intersection of two sets as a new set.

(i.e. all elements that are in both sets.)"""
```

```
intersection_update = """def intersection_update(*s: Iterable[Any]) ->
None
Update a set with the intersection of itself and another."""

symmetric_difference = """def symmetric_difference(s: Iterable[_T]) ->
Set[_T]
Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)"""

symmetric_difference_update = """def symmetric_difference_update(s:
Iterable[_T]) -> None
Update a set with the symmetric difference of itself and another."""

issuperset = """def issuperset(s: Iterable[Any]) -> bool
Report whether this set contains another set."""

issubset = """def issubset(s: Iterable[Any]) -> bool
Report whether another set contains this set."""

isdisjoint = """def isdisjoint(s: Iterable[Any]) -> bool
Return True if two sets have a null intersection."""
```