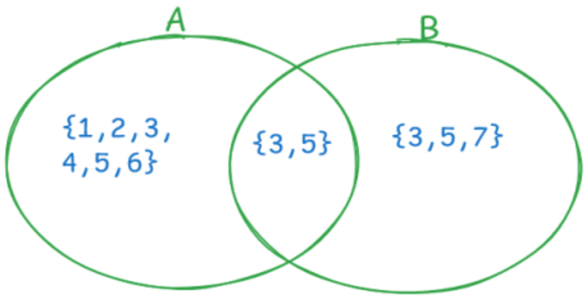
CN-Python-Lec09-Data Structures-II

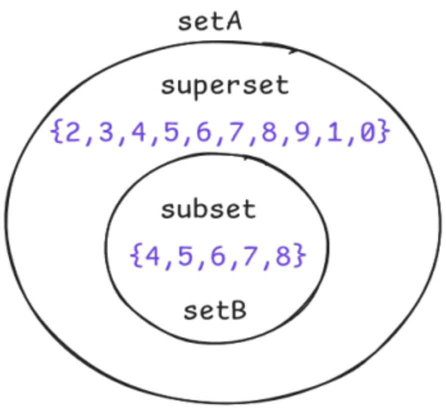
Cont. Data Structures-II

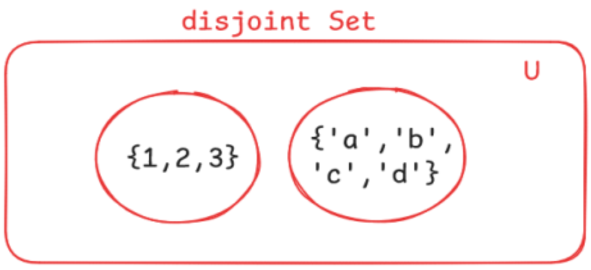
🎯 Session Objectives

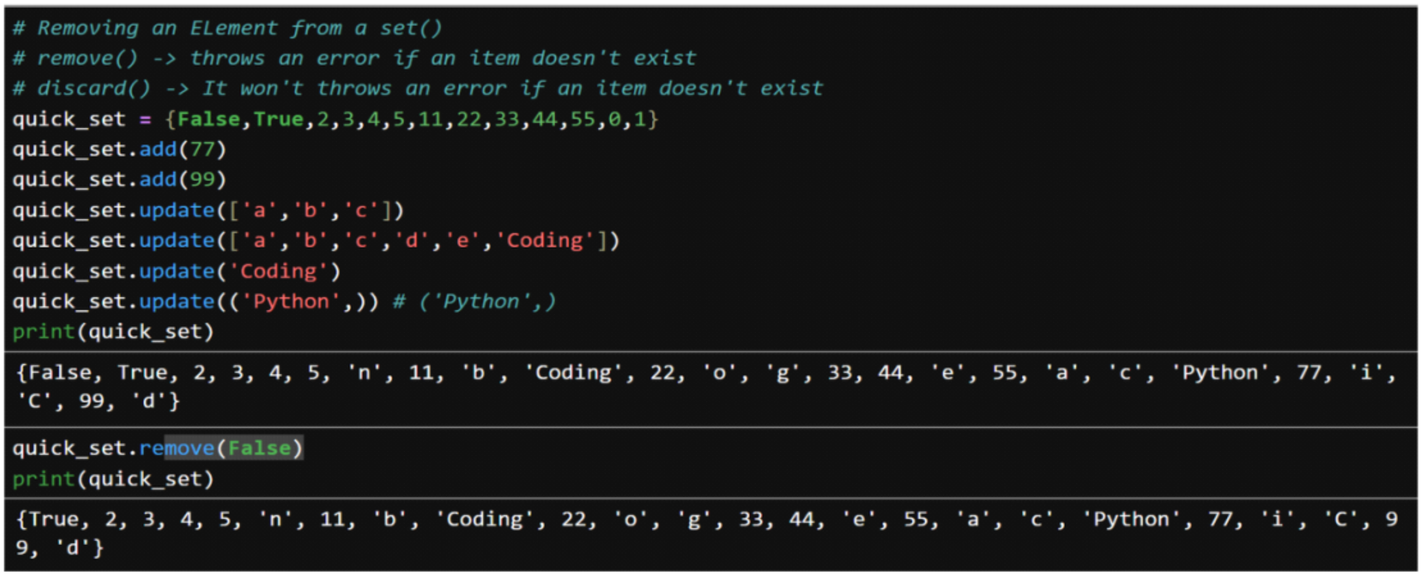
- 📚 Understand what sets are.
- ⚙️ Understand common methods and operations associated with sets.
- ⚖️ Understand the comparison between lists, tuples and sets.
- 🔑 Understand what dictionaries are.
- 🛠️ Understand common methods and operations associated with dictionaries.
- 🤝 Understand the comparison between lists, tuples, sets and dictionaries.

Joining Set







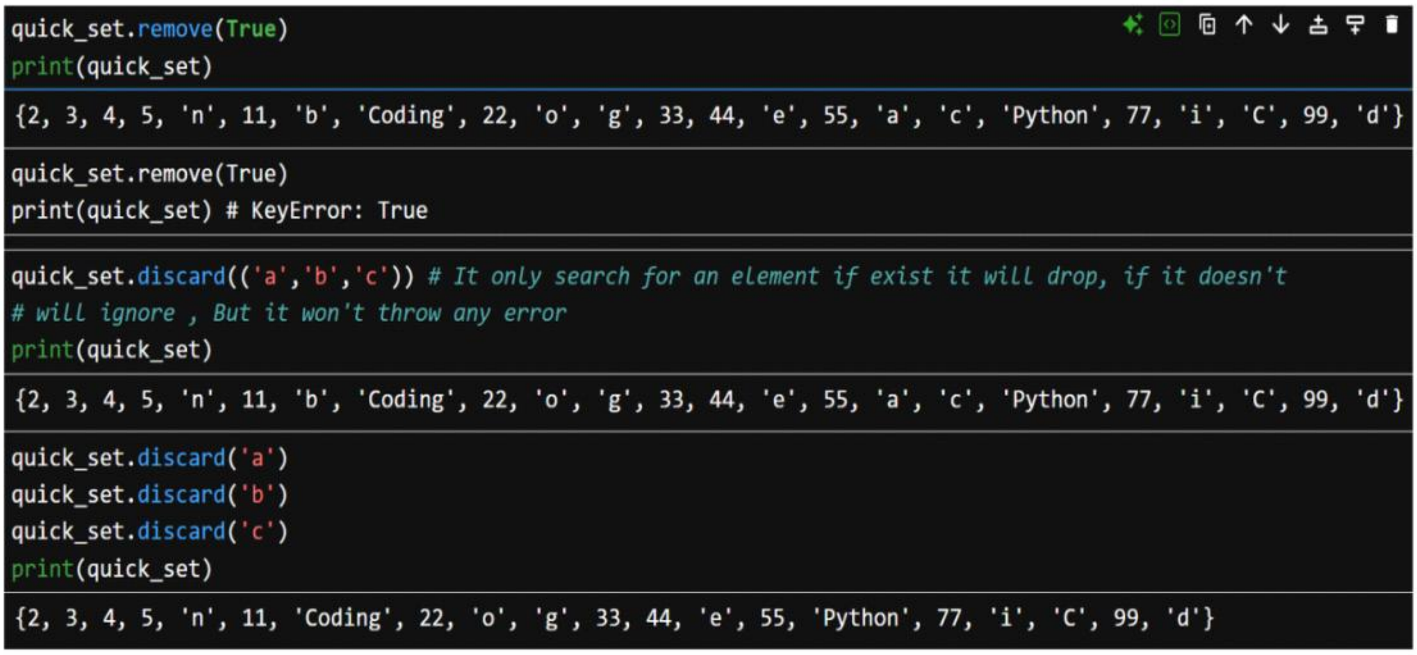


```
# Removing an ELement from a set()
# remove() -> throws an error if an item doesn't exist
# discard() -> It won't throws an error if an item doesn't exist
quick_set = {False,True,2,3,4,5,11,22,33,44,55,0,1}
quick_set.add(77)
quick_set.add(99)
quick_set.update(['a','b','c'])
quick_set.update(['a','b','c','d','e','Coding'])
quick_set.update('Coding')
quick_set.update(('Python',)) # ('Python',)
print(quick_set)
```

```
{False, True, 2, 3, 4, 5, 'n', 11, 'b', 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'a', 'c', 'Python', 77, 'i',
'C', 99, 'd'}
```

```
quick_set.remove(False)
print(quick_set)
```

```
{True, 2, 3, 4, 5, 'n', 11, 'b', 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'a', 'c', 'Python', 77, 'i', 'C', 9
9, 'd'}
```



```
quick_set.remove(True)
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'b', 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'a', 'c', 'Python', 77, 'i', 'C', 99, 'd'}
```

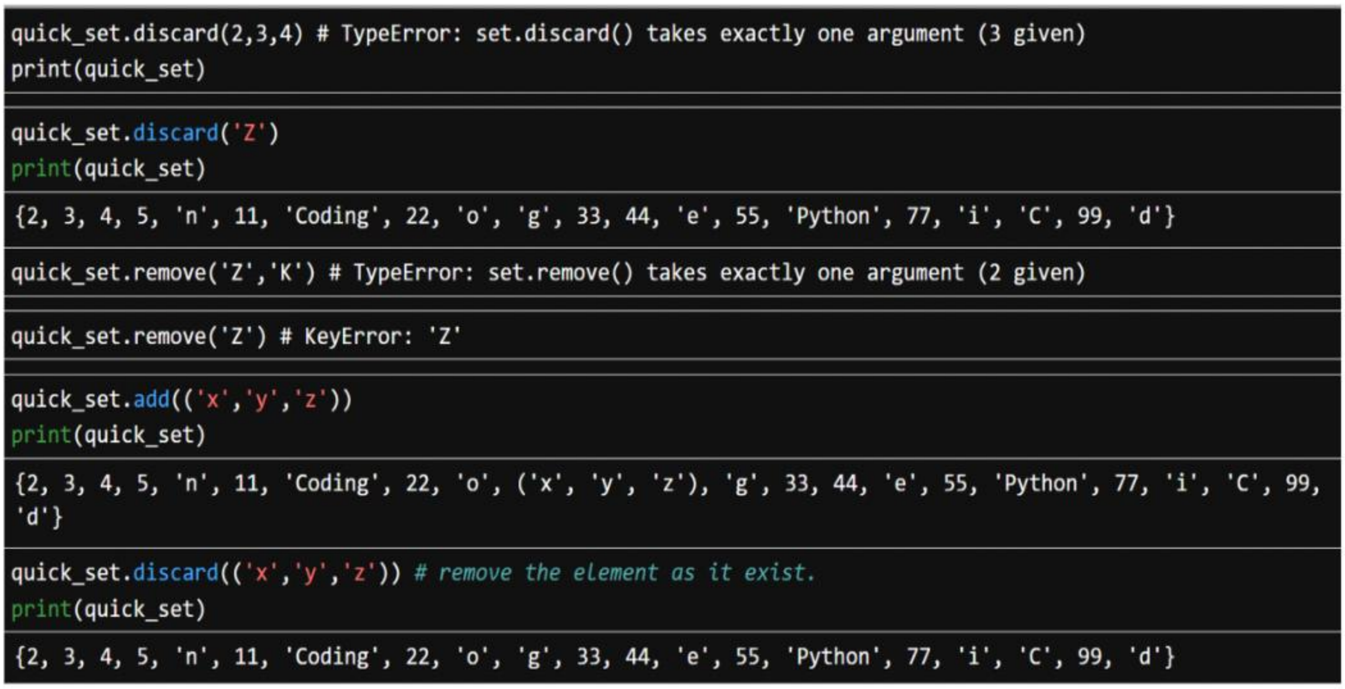
```
quick_set.remove(True)
print(quick_set) # KeyError: True
```

```
quick_set.discard(('a','b','c')) # It only search for an element if exist it will drop, if it doesn't
# will ignore , But it won't throw any error
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'b', 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'a', 'c', 'Python', 77, 'i', 'C', 99, 'd'}
```

```
quick_set.discard('a')
quick_set.discard('b')
quick_set.discard('c')
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```



```
quick_set.discard(2,3,4) # TypeError: set.discard() takes exactly one argument (3 given)
print(quick_set)
```

```
quick_set.discard('Z')
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

```
quick_set.remove('Z','K') # TypeError: set.remove() takes exactly one argument (2 given)
```

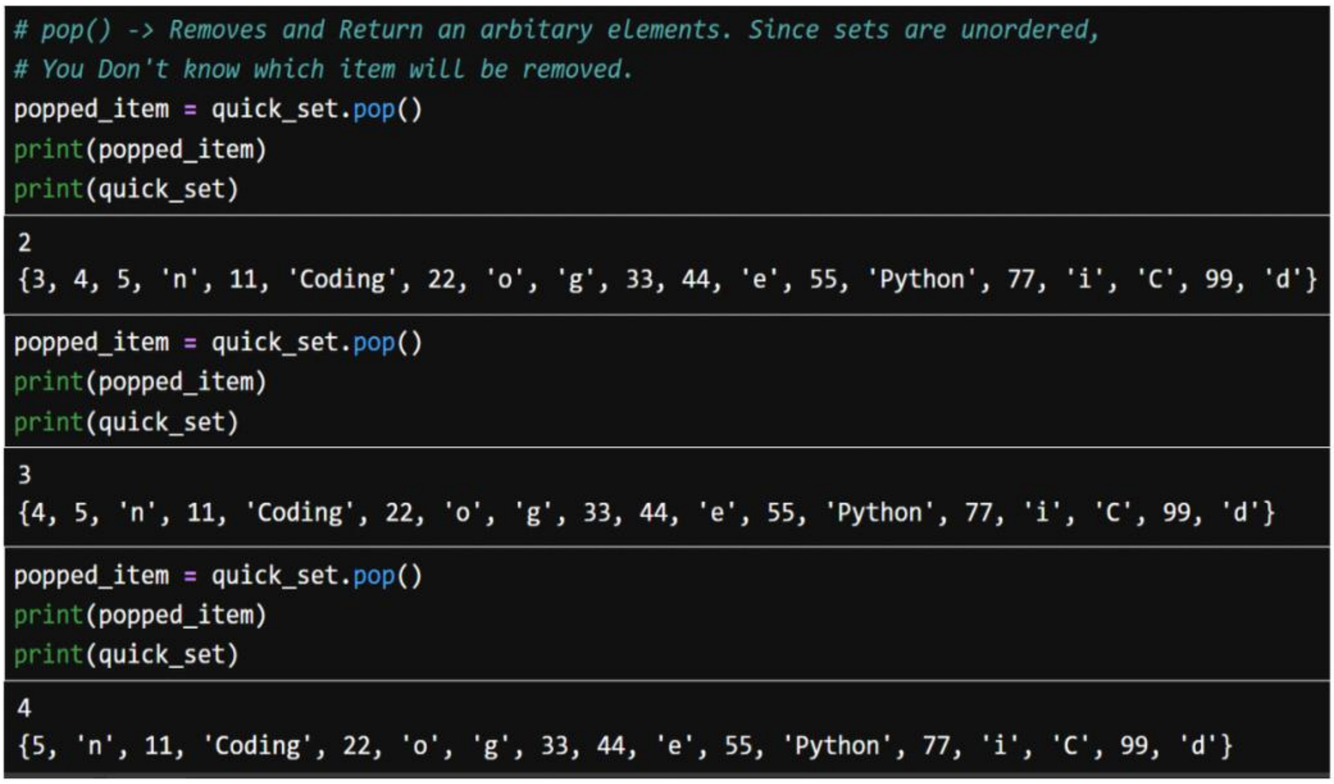
```
quick_set.remove('Z') # KeyError: 'Z'
```

```
quick_set.add(('x','y','z'))
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'Coding', 22, 'o', ('x', 'y', 'z'), 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99,
'd'}
```

```
quick_set.discard(('x','y','z')) # remove the element as it exist.
print(quick_set)
```

```
{2, 3, 4, 5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```



```python
# pop() -> Removes and Return an arbitary elements. Since sets are unordered,
# You Don't know which item will be removed.
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

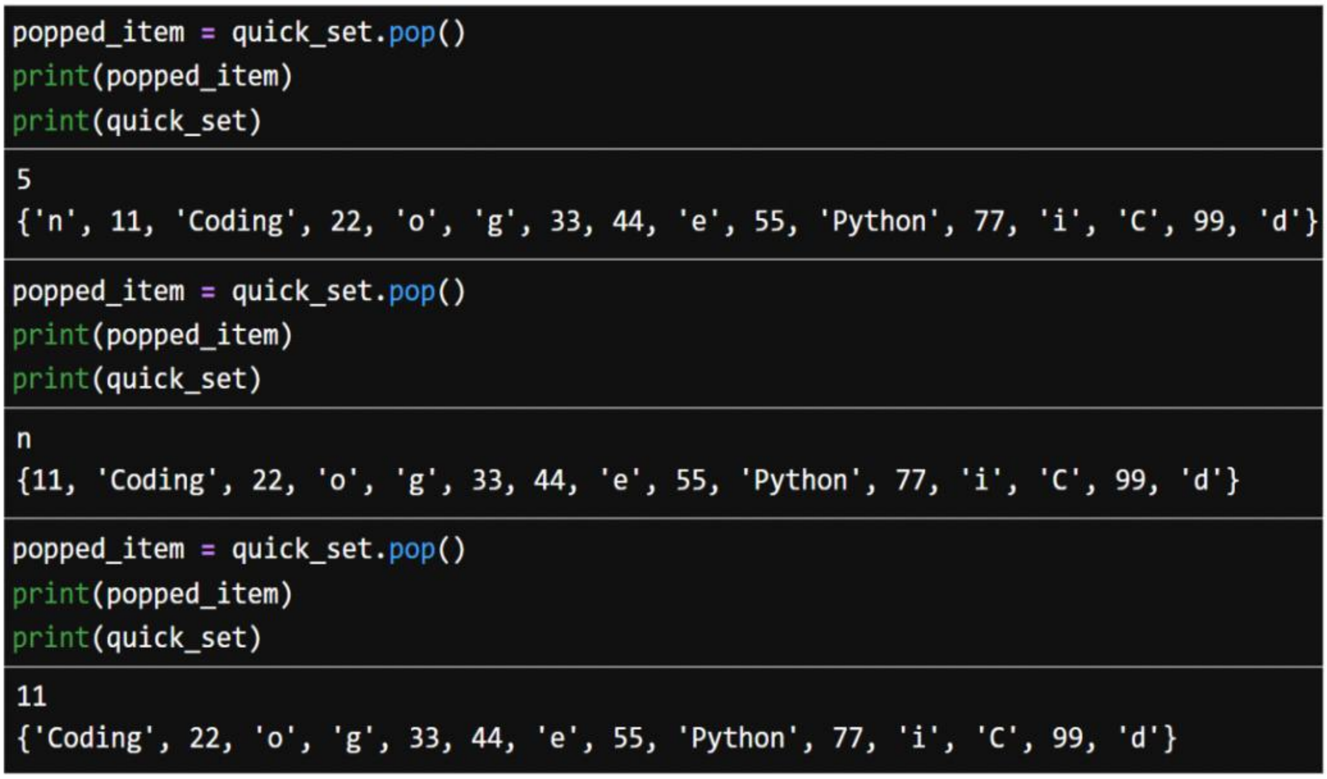
```
2
{3, 4, 5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

```
3
{4, 5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

```
4
{5, 'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```



```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

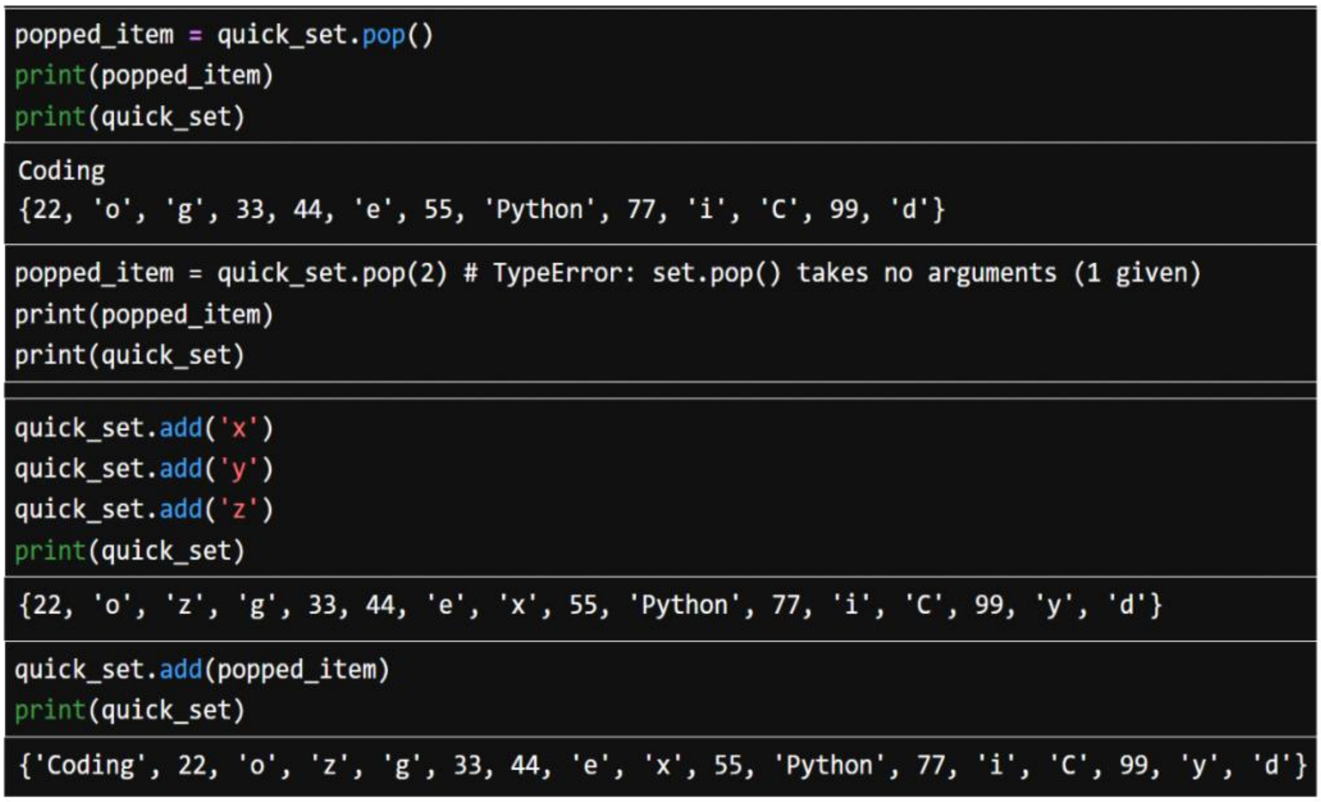
```
5
{'n', 11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

```
n
{11, 'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

```
11
{'Coding', 22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```



```python
popped_item = quick_set.pop()
print(popped_item)
print(quick_set)
```

```
Coding
{22, 'o', 'g', 33, 44, 'e', 55, 'Python', 77, 'i', 'C', 99, 'd'}
```

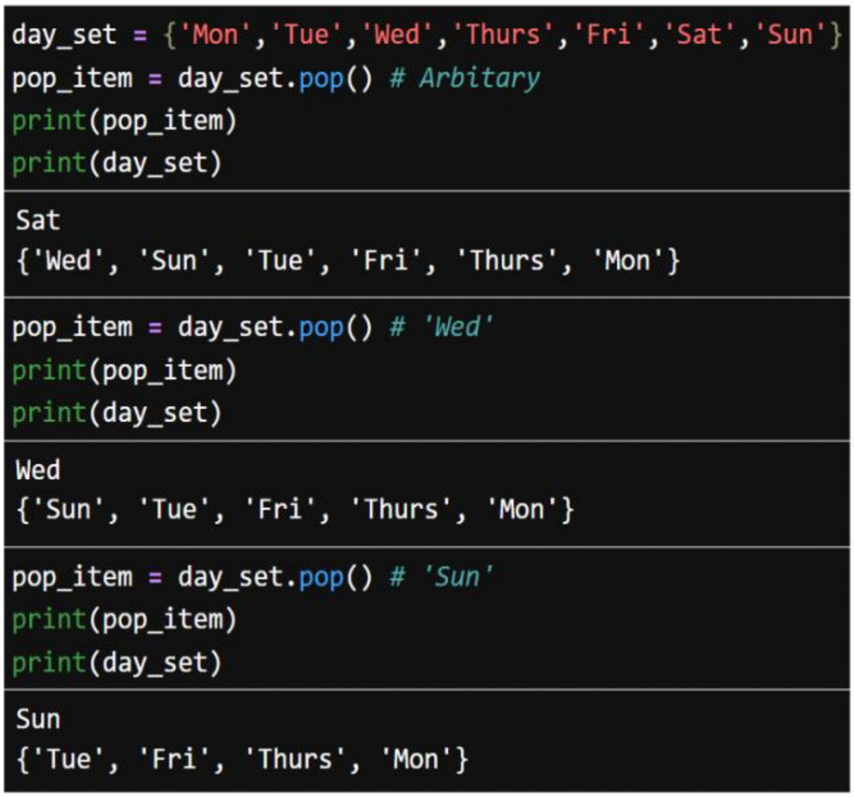
```python
popped_item = quick_set.pop(2) # TypeError: set.pop() takes no arguments (1 given)
print(popped_item)
print(quick_set)
```

```python
quick_set.add('x')
quick_set.add('y')
quick_set.add('z')
print(quick_set)
```

```
{22, 'o', 'z', 'g', 33, 44, 'e', 'x', 55, 'Python', 77, 'i', 'C', 99, 'y', 'd'}
```

```python
quick_set.add(popped_item)
print(quick_set)
```

```
{'Coding', 22, 'o', 'z', 'g', 33, 44, 'e', 'x', 55, 'Python', 77, 'i', 'C', 99, 'y', 'd'}
```



```python
day_set = {'Mon','Tue','Wed','Thurs','Fri','Sat','Sun'}
pop_item = day_set.pop() # Arbitary
print(pop_item)
print(day_set)
```

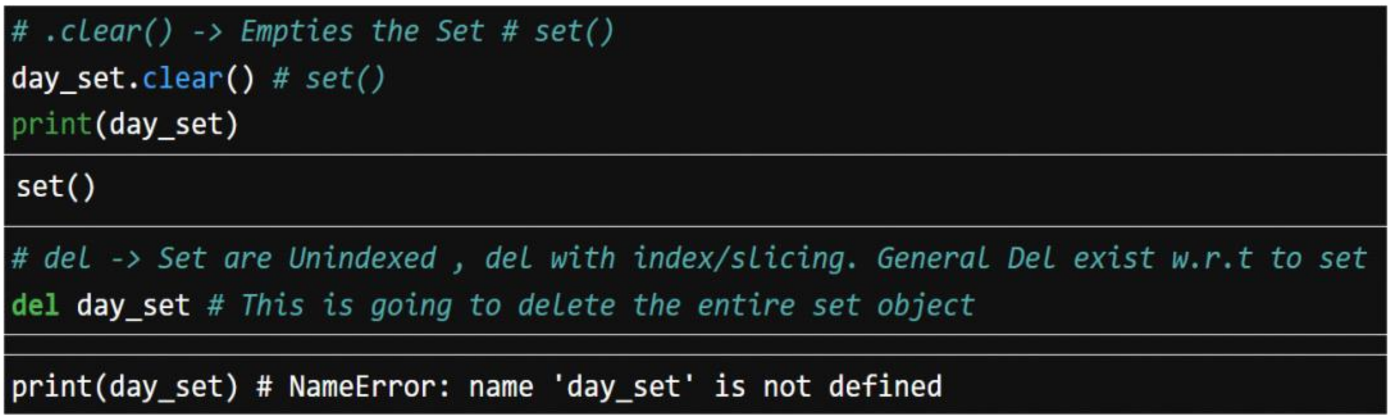
```
Sat
{'Wed', 'Sun', 'Tue', 'Fri', 'Thurs', 'Mon'}
```

```python
pop_item = day_set.pop() # 'Wed'
print(pop_item)
print(day_set)
```

```
Wed
{'Sun', 'Tue', 'Fri', 'Thurs', 'Mon'}
```

```python
pop_item = day_set.pop() # 'Sun'
print(pop_item)
print(day_set)
```

```
Sun
{'Tue', 'Fri', 'Thurs', 'Mon'}
```

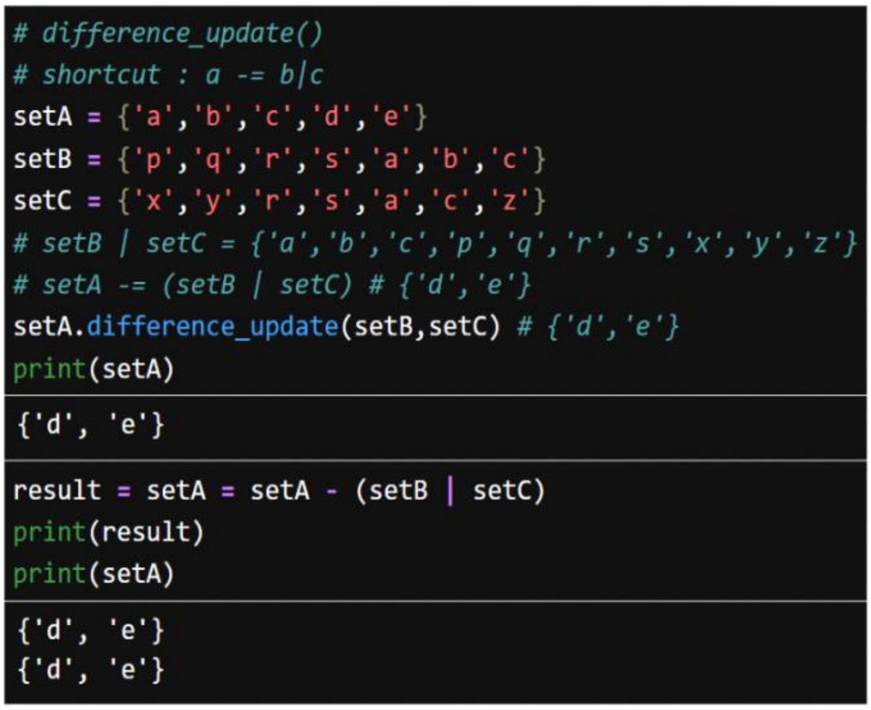


```
# .clear() -> Empties the Set # set()
day_set.clear() # set()
print(day_set)
```

```
set()
```

```
# del -> Set are Unindexed , del with index/slicing. General Del exist w.r.t to set
del day_set # This is going to delete the entire set object
```

```
print(day_set) # NameError: name 'day_set' is not defined
```

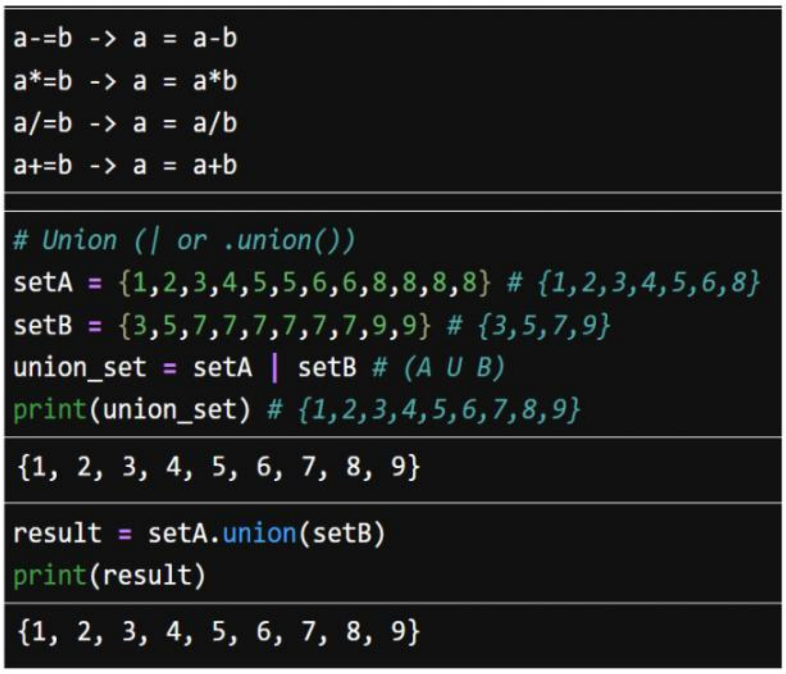


```
# difference_update()
# shortcut : a -= b|c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# setB | setC = {'a','b','c','p','q','r','s','x','y','z'}
# setA -= (setB | setC) # {'d','e'}
setA.difference_update(setB,setC) # {'d','e'}
print(setA)
```

```
{'d', 'e'}
```

```
result = setA = setA - (setB | setC)
print(result)
print(setA)
```

```
{'d', 'e'}
{'d', 'e'}
```



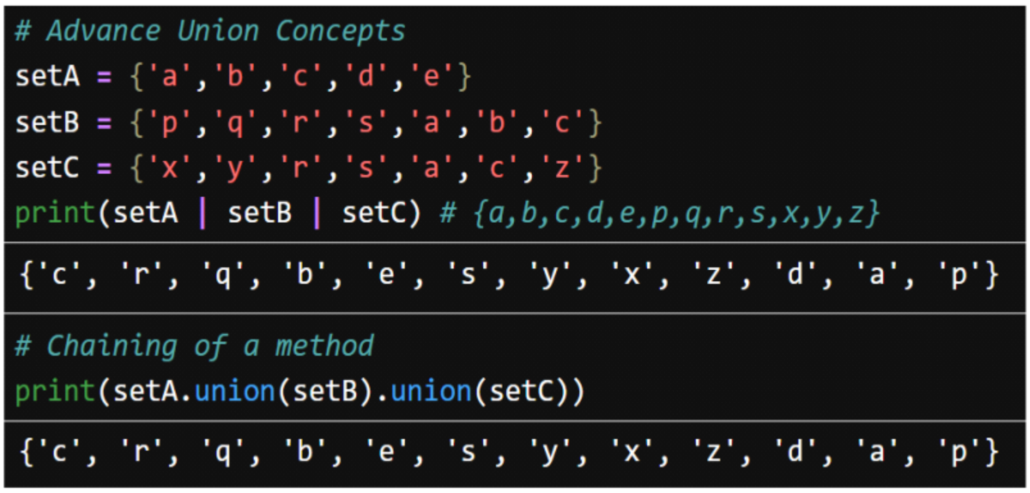
```
a-=b -> a = a-b
a*=b -> a = a*b
a/=b -> a = a/b
a+=b -> a = a+b
```

```
# Union (| or .union())
setA = {1,2,3,4,5,5,6,6,8,8,8,8} # {1,2,3,4,5,6,8}
setB = {3,5,7,7,7,7,7,7,9,9} # {3,5,7,9}
union_set = setA | setB # (A U B)
print(union_set) # {1,2,3,4,5,6,7,8,9}
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
result = setA.union(setB)
print(result)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

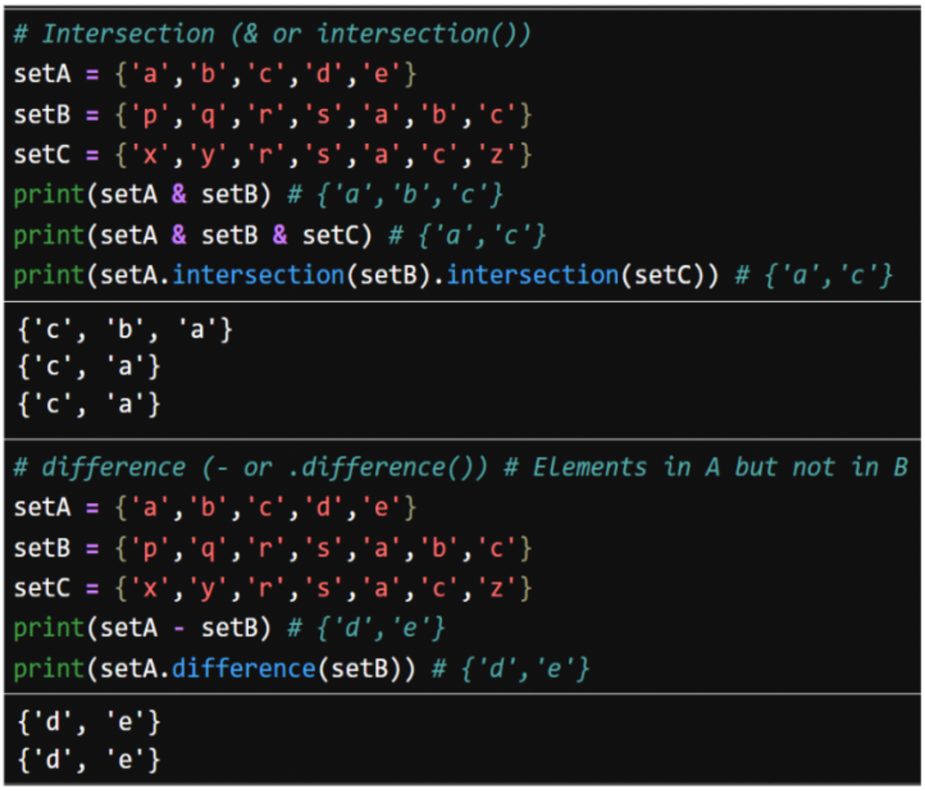


```python
# Advance Union Concepts
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA | setB | setC) # {a,b,c,d,e,p,q,r,s,x,y,z}
```

```
{'c', 'r', 'q', 'b', 'e', 's', 'y', 'x', 'z', 'd', 'a', 'p'}
```

```python
# Chaining of a method
print(setA.union(setB).union(setC))
```

```
{'c', 'r', 'q', 'b', 'e', 's', 'y', 'x', 'z', 'd', 'a', 'p'}
```

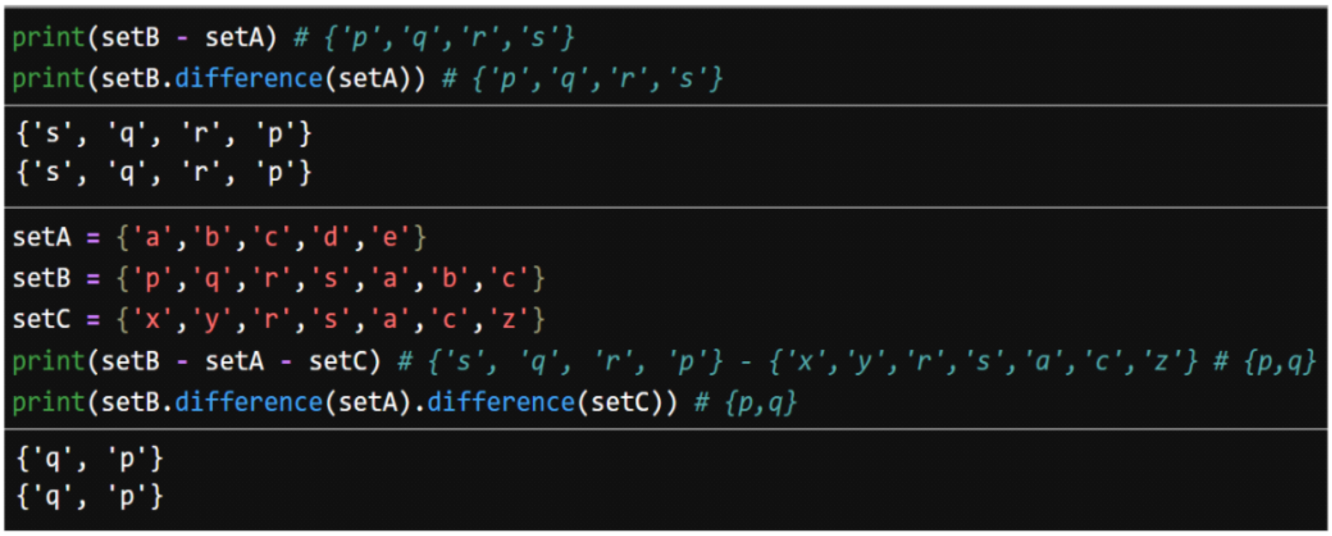


```python
# Intersection (& or intersection())
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA & setB) # {'a','b','c'}
print(setA & setB & setC) # {'a','c'}
print(setA.intersection(setB).intersection(setC)) # {'a','c'}
```

```
{'c', 'b', 'a'}
{'c', 'a'}
{'c', 'a'}
```

```python
# difference (- or .difference()) # Elements in A but not in B
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA - setB) # {'d','e'}
print(setA.difference(setB)) # {'d','e'}
```

```
{'d', 'e'}
{'d', 'e'}
```

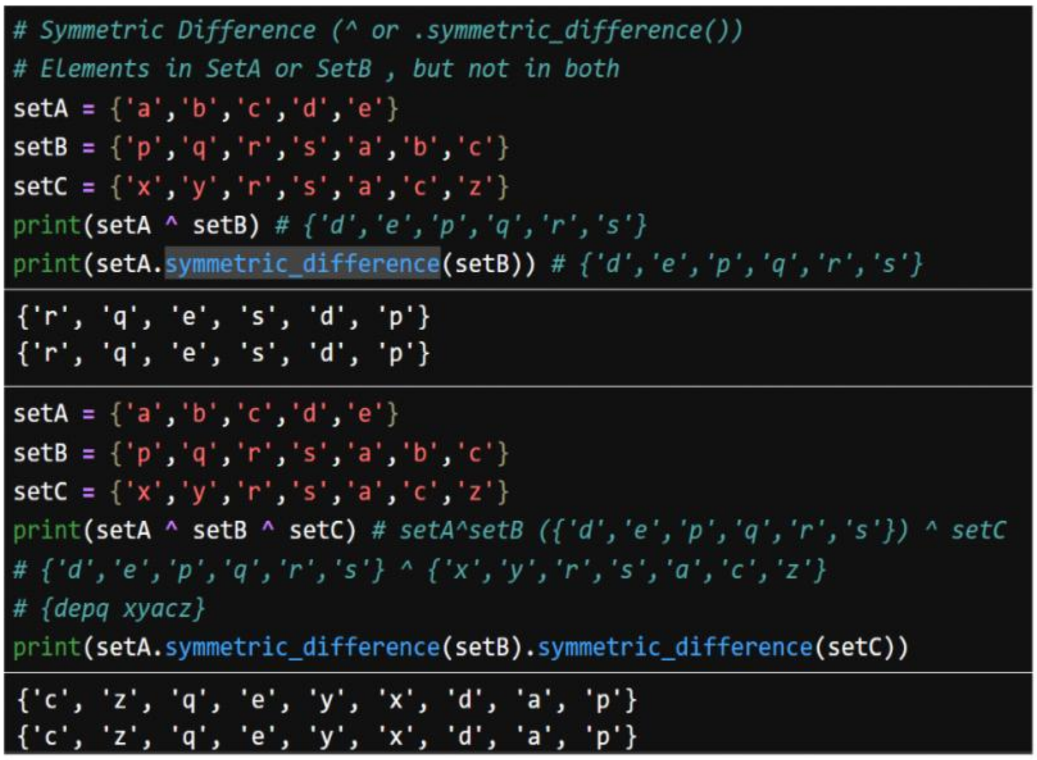


```python
print(setB - setA) # {'p','q','r','s'}
print(setB.difference(setA)) # {'p','q','r','s'}
```

```
{'s', 'q', 'r', 'p'}
{'s', 'q', 'r', 'p'}
```

```python
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setB - setA - setC) # {'s', 'q', 'r', 'p'} - {'x','y','r','s','a','c','z'} # {p,q}
print(setB.difference(setA).difference(setC)) # {p,q}
```

```
{'q', 'p'}
{'q', 'p'}
```

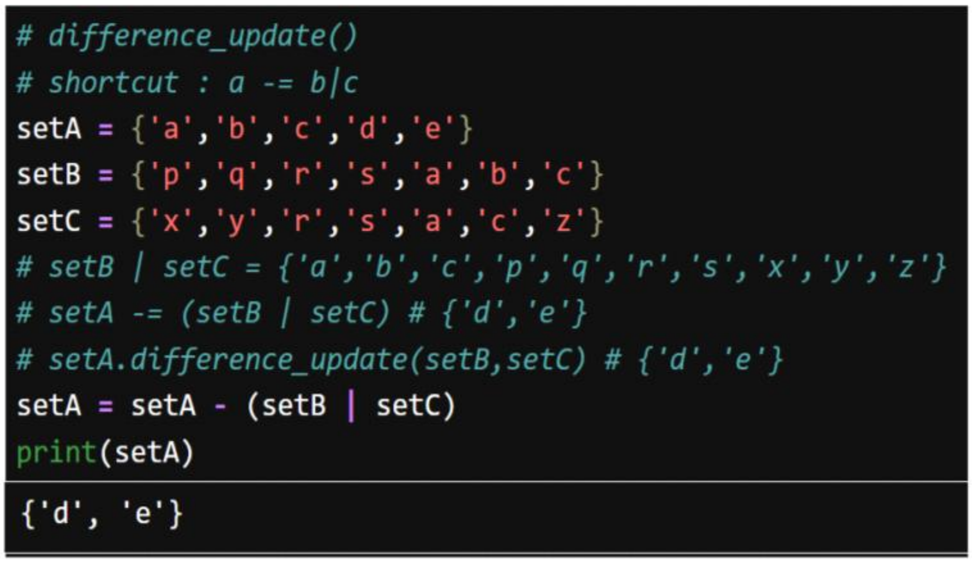


```
# Symmetric Difference (^ or .symmetric_difference())
# Elements in SetA or SetB , but not in both
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA ^ setB) # {'d','e','p','q','r','s'}
print(setA.symmetric_difference(setB)) # {'d','e','p','q','r','s'}
```

```
{'r', 'q', 'e', 's', 'd', 'p'}
{'r', 'q', 'e', 's', 'd', 'p'}
```

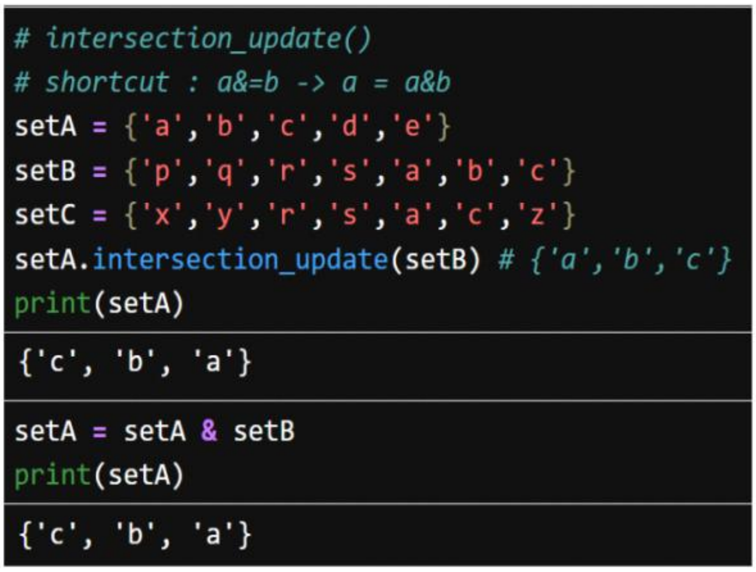
```
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
print(setA ^ setB ^ setC) # setA^setB ({'d','e','p','q','r','s'}) ^ setC
# {'d','e','p','q','r','s'} ^ {'x','y','r','s','a','c','z'}
# {depq xyacz}
print(setA.symmetric_difference(setB).symmetric_difference(setC))
```

```
{'c', 'z', 'q', 'e', 'y', 'x', 'd', 'a', 'p'}
{'c', 'z', 'q', 'e', 'y', 'x', 'd', 'a', 'p'}
```



```
# difference_update()
# shortcut : a -= b|c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# setB | setC = {'a','b','c','p','q','r','s','x','y','z'}
# setA -= (setB | setC) # {'d','e'}
# setA.difference_update(setB,setC) # {'d','e'}
setA = setA - (setB | setC)
print(setA)
```

```
{'d', 'e'}
```

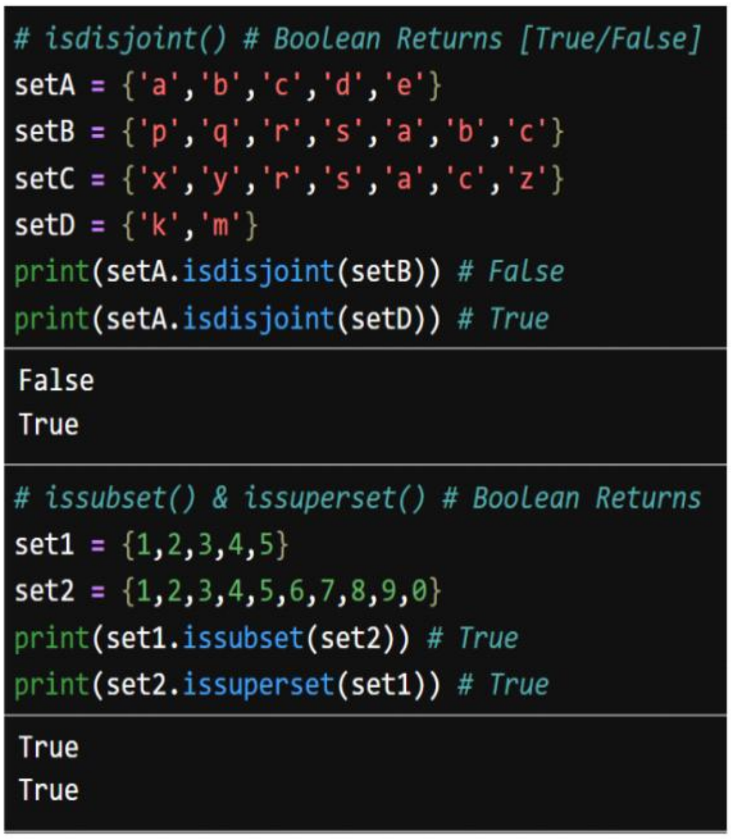


```
# intersection_update()
# shortcut : a&=b -> a = a&b
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA.intersection_update(setB) # {'a','b','c'}
print(setA)
```

```
{'c', 'b', 'a'}
```

```
setA = setA & setB
print(setA)
```

```
{'c', 'b', 'a'}
```

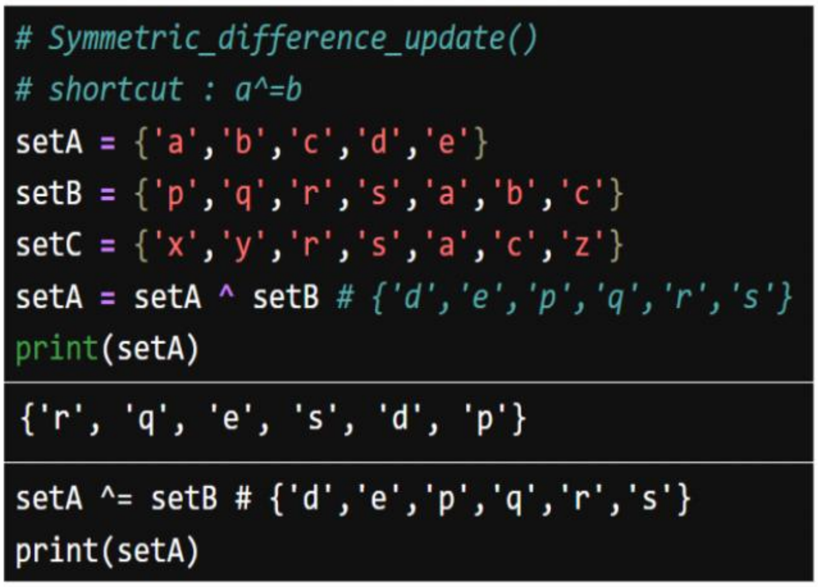


```python
# isdisjoint() # Boolean Returns [True/False]
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setD = {'k','m'}
print(setA.isdisjoint(setB)) # False
print(setA.isdisjoint(setD)) # True
```

```
False
True
```

```python
# issubset() & issuperset() # Boolean Returns
set1 = {1,2,3,4,5}
set2 = {1,2,3,4,5,6,7,8,9,0}
print(set1.issubset(set2)) # True
print(set2.issuperset(set1)) # True
```

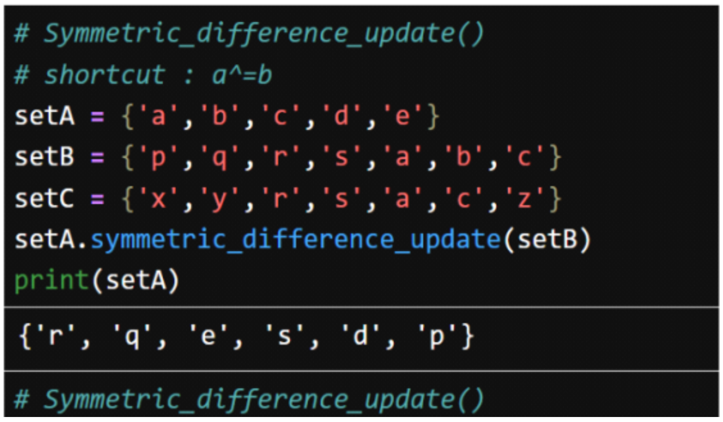
```
True
True
```



```python
# Symmetric_difference_update()
# shortcut : a^=b
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA = setA ^ setB # {'d','e','p','q','r','s'}
print(setA)
```

```
{'r', 'q', 'e', 's', 'd', 'p'}
```

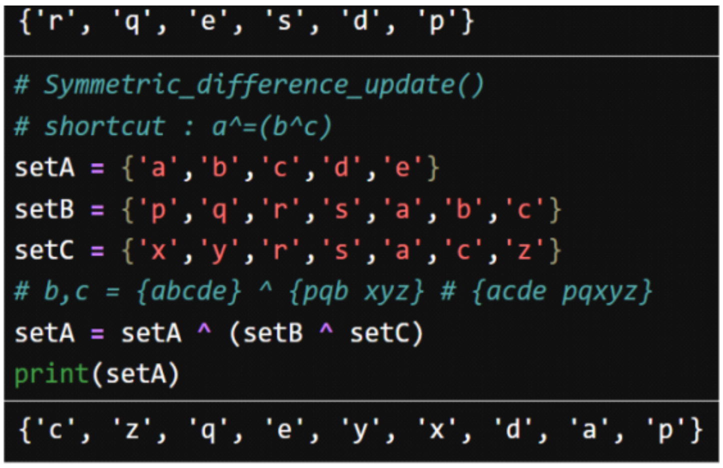
```python
setA ^= setB # {'d','e','p','q','r','s'}
print(setA)
```



```python
# Symmetric_difference_update()
# shortcut : a^=b
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA.symmetric_difference_update(setB)
print(setA)
```

```
{'r', 'q', 'e', 's', 'd', 'p'}
```

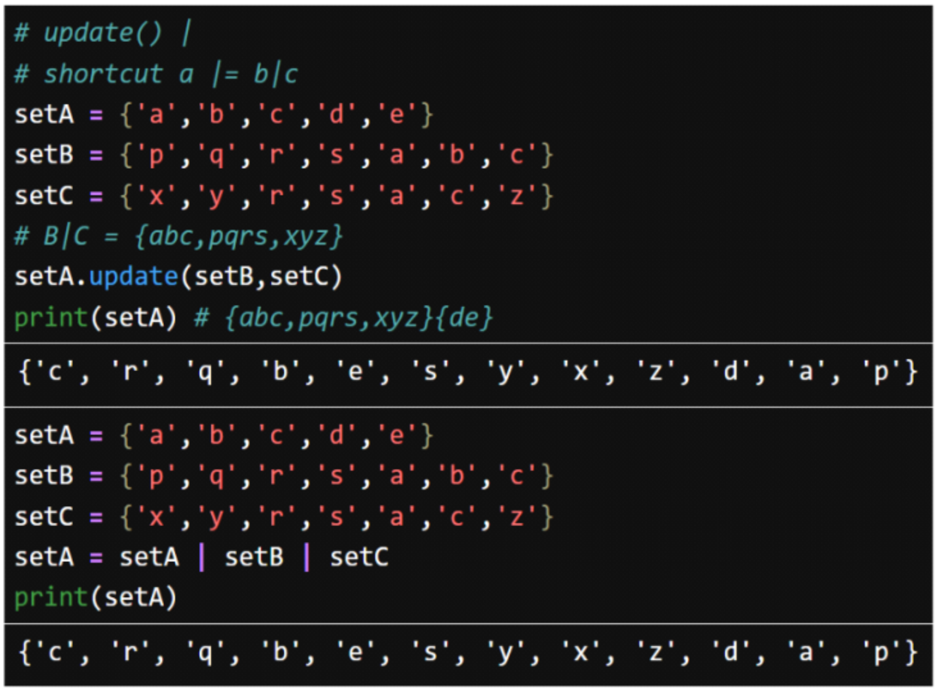
```python
# Symmetric_difference_update()
```



```
{'r', 'q', 'e', 's', 'd', 'p'}
```

```
# Symmetric_difference_update()
# shortcut : a^=(b^c)
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# b,c = {abcde} ^ {pqb xyz} # {acde pqxyz}
setA = setA ^ (setB ^ setC)
print(setA)
```

```
{'c', 'z', 'q', 'e', 'y', 'x', 'd', 'a', 'p'}
```

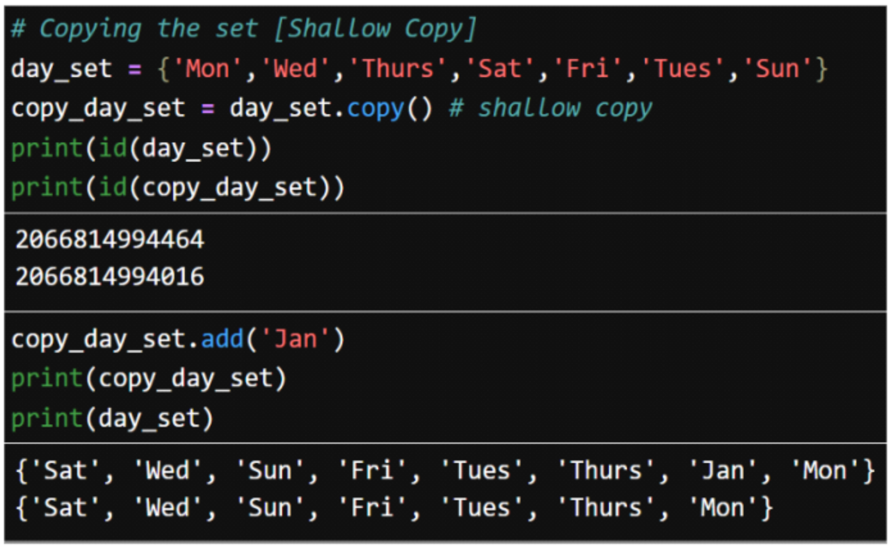


```
# update() |
# shortcut a |= b|c
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
# B|C = {abc,pqrs,xyz}
setA.update(setB,setC)
print(setA) # {abc,pqrs,xyz}{de}
```

```
{'c', 'r', 'q', 'b', 'e', 's', 'y', 'x', 'z', 'd', 'a', 'p'}
```

```
setA = {'a','b','c','d','e'}
setB = {'p','q','r','s','a','b','c'}
setC = {'x','y','r','s','a','c','z'}
setA = setA | setB | setC
print(setA)
```

```
{'c', 'r', 'q', 'b', 'e', 's', 'y', 'x', 'z', 'd', 'a', 'p'}
```

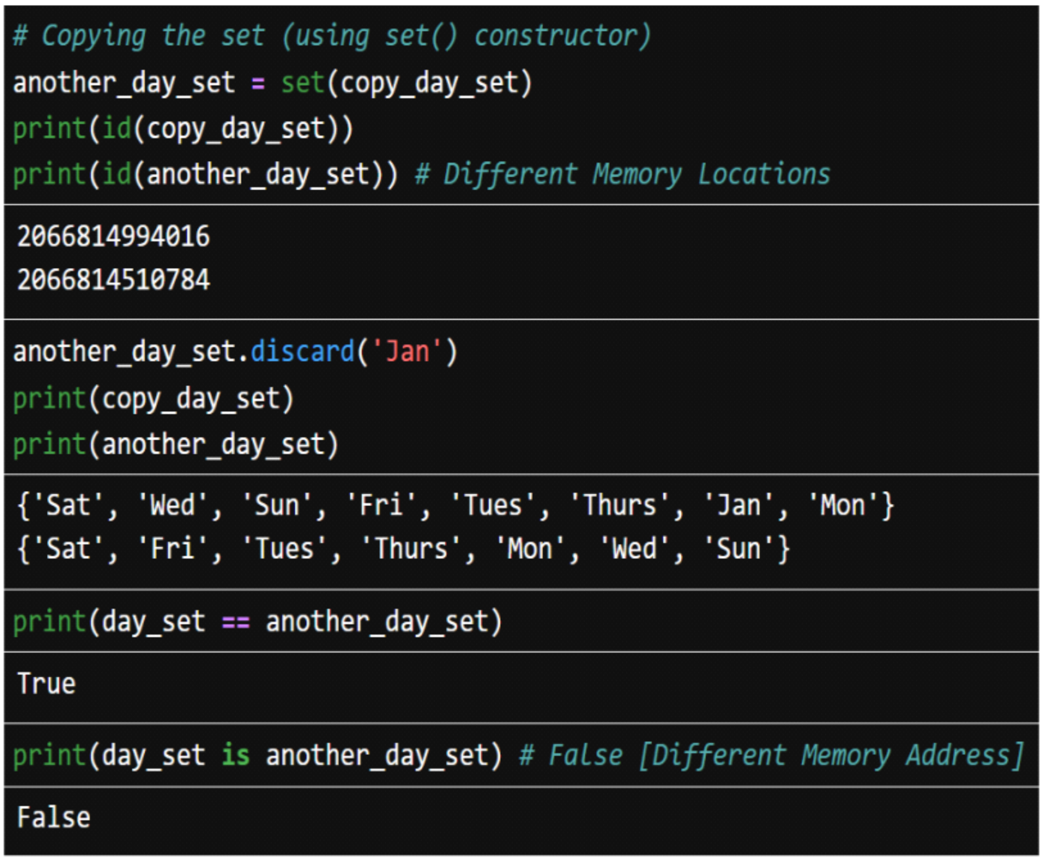


```
# Copying the set [Shallow Copy]
day_set = {'Mon','Wed','Thurs','Sat','Fri','Tues','Sun'}
copy_day_set = day_set.copy() # shallow copy
print(id(day_set))
print(id(copy_day_set))
```

```
2066814994464
2066814994016
```

```
copy_day_set.add('Jan')
print(copy_day_set)
print(day_set)
```

```
{'Sat', 'Wed', 'Sun', 'Fri', 'Tues', 'Thurs', 'Jan', 'Mon'}
{'Sat', 'Wed', 'Sun', 'Fri', 'Tues', 'Thurs', 'Mon'}
```



```
# Copying the set (using set() constructor)
another_day_set = set(copy_day_set)
print(id(copy_day_set))
print(id(another_day_set)) # Different Memory Locations
```

```
2066814994016
2066814510784
```

```
another_day_set.discard('Jan')
print(copy_day_set)
print(another_day_set)
```

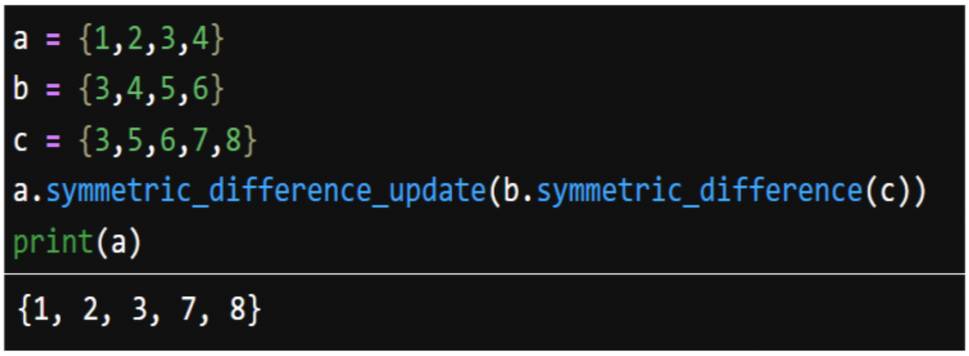
```
{'Sat', 'Wed', 'Sun', 'Fri', 'Tues', 'Thurs', 'Jan', 'Mon'}
{'Sat', 'Fri', 'Tues', 'Thurs', 'Mon', 'Wed', 'Sun'}
```

```
print(day_set == another_day_set)
```

```
True
```

```
print(day_set is another_day_set) # False [Different Memory Address]
```

```
False
```



```
a = {1,2,3,4}
b = {3,4,5,6}
c = {3,5,6,7,8}
a.symmetric_difference_update(b.symmetric_difference(c))
print(a)
```

```
{1, 2, 3, 7, 8}
```