Chapter-8 Python Sets

Contents

8.0 Python Sets

Set ဆိုသည်မှာ ဒေတာထည့်သွင်းသည့် ရှေ့နောက်နေရာ အစီအစဉ်တကျ မထားသည့် ဒေတာ အမျိုးအစား(unordered collection data type)ဖြစ်သည်။ ထို့ကြောင့် index နံပါတ်များဖြင့် ဖတ်ယူရန် မဖြစ် (unindexed)နိုင်ပါ။ လိုအပ်လျှင် set တစ်ခု အတွင်း ဒေတာများကို ပြောင်းလဲနိုင်သည်။ (mutable ဖြစ်သည်။)။ set တစ်ခု အတွင်း၌ တူညီသည့် ဒေတာများ ရှိနေခွင့်မပြု။ (no duplicate elements)။ Set အတွင်းမှ ဒေတာများကို လျှင်မြန်စွာ ထုတ်ယူနိုင်သည်။ သခ်ၤာမှ union, intersect စသည့် နှင့်သက်ဆိုင်သည့် operation များကို လုပ်နိုင်သည်။

```
my_set = {"apple", "banana", "cherry"}
```

8.1 Create a set

နည်းနှစ်မျိုးဖြင့် ပြုလုပ်နိုင်သည်။ နည်း(၂)မျိုးဖြင့် set များကို တည်ဆောက်နိုင်သည်။ curly braces {} ထဲတွင် element များကို ကော်မာခံ၍ ရေးခြင်းနှင့် built-in set function ကို သုံးခြင်းတို့ဖြစ်သည်။ Set နှင့် dictionary နှစ်မျိုးလုံးကို ဖြင့်ရေးသည်။ dictionary အတွင်းရှိ item များသည် key-value pair များဖြစ်သောကြောင့် ":" ပါသည်။

သတိပြုရန်အချက်မှာ တွန့်ကွင်း(curly braces { }) ကြောင့် empty set နှင့် empty dictionary ရောမှားနိုင်သည်။ Empty set တစ်ခုကို "a = { }" ဖြင့် ပြုလုပ်၍ ရနိုင်ပါ။ "a = { }" သည် dictionary သာဖြစ်သည်။ Empty set တစ်ခု ပြုလုပ်လိုပါက set() ကို သုံးရသည်။

```
# curly braces {} ထဲတွင် element များကို ကော်မာခံ၍ ရေးသည်။
my_set = {"apple", "banana", "cherry"}
print(my_set)

# or use the set function and create from an iterable, e.g. list, tuple
, string
my_set_2 = set(["one", "two", "three"])
print(my_set_2)

# careful: an empty set cannot be created with {}, as this is interpret
ed as dict
# use set() instead
a = {}
print(type(a))
a = set()                          #  set() ကို သုံး၍ empty set ပြုလုပ်သည်။
print(type(a))                     # set() ဖြစ်ကြောင်း စစ်သည်။
```

```
{'apple', 'banana', 'cherry'}
{'two', 'three', 'one'}
<class 'dict'>
<class 'set'>
```

8.1.1 တခြားသော ဒေတာများကို set အဖြစ်သို့ပေါင်းခြင်း

List, string, tuple, dictionary စသည့် data structure များကို set အဖြစ်သို့ပေါင်းနိုင်သည်။ တန်ဖိုးတူ ဒေတာများထဲမှ တစ်ခုကိုသာ သိမ်းဆည်းသည်။ (discarding any duplicate)

```
#'e' နှင့် 't' တို့သည် နှစ်လုံးစီပါနေသောကြောင့် တစ်ခုကိုသာ သိမ်းဆည်းသည်။
my_set4 = set( 'letters' )
my_set4
```

```
{'e', 'l', 'r', 's', 't'}
```

```
my_set_3 = set("aaabbbcccdddeeeeeffff")
print(my_set_3)
```

```
{'d', 'c', 'b', 'e', 'f', 'a'}
```

8.1.2 Set တစ်ခုအတွင်းသို့ များကို ထည့်ခြင်း(Adding Data to a Set)

set.add() နှင့် **update()** method ကို အသုံးပြုနိုင်သည်။ ရှိပြီးသား set တစ်ခုအတွင်းသို့ ဒေတာများ ထည့်လိုသည့်အခါ set.add() method ကို အသုံးပြုနိုင်သည်။

```
my_set = set()

# use the add() method to add elements
my_set.add(42)
my_set.add(True)
my_set.add("Hello")

# note: the order does not matter, and might differ when printed
print(my_set)

# nothing happens when the element is already present:
my_set.add(42)
print(my_set)
```

```
{True, 42, 'Hello'}
{True, 42, 'Hello'}
```

တစ်ခုထက် ပိုများသည့် iterable item များကို set တစ်ခုအတွင်းသို့ ထည့်လိုသည့်အခါ update() method ကို အသုံးပြုသည်။

```
a = {1,2,3}
print(a)

a.update([3,4,5,6])
print(a)
```

```
{1, 2, 3}
{1, 2, 3, 4, 5, 6}
```

8.2 "in" keyword သည် အသုံးပြု၍ set ထဲတွင် မိမိသိလိုသည့် item တစ်ခု ရှိ မရှိ စစ်ခြင်း

Set ထဲတွင် မိမိသိလိုသည့် item တစ်ခု ရှိ မရှိ (exist or not)ကို သိလိုလျှင် in key wordကို သုံး၍ စစ်နိုင်သည်။ in keyword သည် အသုံးအများဆုံး ဖြစ်သည်။

```
a = {1,2,3,4}
for num in a:
    print(num,end=' ')  # end=' ' ကို အတန်းလိုက်ပေါ်အောင် ထည့်ထားသည်။
```

```
1 2 3 4
```

အနည်းငယ်ခက်သည့် ဥပမာတစ်ခုဖြင့် ရှင်းပြမည်။ Drinks ဟုနာမည်ပေးထားသည့် dictionary တစ်ခုကို အသုံးပြု၍ ရှင်းပြမည်။ အရက်အမည်နှင့် ထိုထဲမှာ ပါဝင်သည့် အရာများကို dictionary တစ်ခုအဖြစ် တည်ဆောက်ထားသည်။ If statement နှင့် in key wordကို သုံး၍ set အတွင်း၌ ရှိနေသည့် အရက်အမည်များကို ဖော်ပြသည်။

```
drinks = {
    'martini': {'vodka', 'vermouth'},
    'black russian': {'vodka', 'kahlua'},
    'white russian': {'cream', 'kahlua', 'vodka'},
    'manhattan': {'rye', 'vermouth', 'bitters'},
    'screwdriver': {'orange juice', 'vodka'}
    }
drinks
```

```
{'martini': {'vermouth', 'vodka'},
 'black russian': {'kahlua', 'vodka'},
 'white russian': {'cream', 'kahlua', 'vodka'},
 'manhattan': {'bitters', 'rye', 'vermouth'},
 'screwdriver': {'orange juice', 'vodka'}}
```

```
# name  နှင့် contents တို့အစား မိမိကြိုက်သည့် အမည်ပေးနိုင်သည်။
for name, contents in drinks.items():
    if 'vodka' in contents:
        print(name)
```

```
martini
black russian
white russian
screwdriver
```

8.2.1 Check if element is in Set

```
my_set = {"apple", "banana", "cherry"}
if "apple" in my_set:
    print("yes")
```

```
yes
```

8.3 Element များကို ဖယ်ထုတ်ခြင်း

pop() method set တစ်ခု အတွင်းမှ တစ်ခုတည်းကိုသာ ဖယ်ရှားလိုလျှင် pop() method ကို သုံးသည်။ Use the set's pop() method to remove and return an item from the set. Items are removed from the beginning of the set, as follows:

```
a = {1, 2, 3, 4}
print(a.pop())
print(a)
```

```
1
{2, 3, 4}
```

remove() ကို သုံး၍ ထဲမှ item ကို ဖယ်ထုတ်နိုင်သည်။ ဖယ်ထုတ်မည့် item သည် set ထဲတွင် မရှိလျှင် KeyError ပေလိမ့်မည်။

```
a = {1,2,3}
a.remove(3)
a
```

```
{1, 2}
```

```
a.remove(3) #ဖယ်ထုတ်မည့် item သည် set ထဲတွင် မရှိလျှင် KeyError ပေလိမ့်မည်။
```

```
KeyError                  Traceback (most recent call last)
KeyError: 3
```

ထိုသို့ KeyError မဖြစ်ပေါ်စေရန်အတွက် discard()ကို အသုံးပြုသည်။ discard()သည် မရှိပါက KeyError မဖြစ်ပေါ်ပေ။ KeyError ဖြစ်ပေါ်ခြင်းကို ကျော်လွှားနိုင်သည်။

```
a = {1,2,3}
a.discard(2)
print(a)
# Set ထဲတွင် မရှိသည့် item ကို discard လုပ်ခိုင်းသည်။
a.discard("nonexistent item")
print(a)
```

```
{1, 3}
{1, 3}
```

remove() နှင့် discard() method သည် item တစ်ခုချင်းစီကိုသာ ဖယ်ရှားနိုင်သည်။ item များ အားလုံးကို တစ်ပြိုင်နက် ဖယ်ရှားလိုပါက clear() method ကို အသုံးပြုရသည်။

```
a = {1,2,3,4,5,6}
print(a)
a.clear()
print(a)                          # set() သည် empty set ဖြစ်သည်။
```

```
{1, 2, 3, 4, 5, 6}
```

```
set()
```

```
# remove(x): removes x, raises a KeyError if element is not present
my_set = {"apple", "banana", "cherry"}
my_set.remove("apple")
print(my_set)
```

```
{'banana', 'cherry'}
```

`my_set.remove("orange")` ဖြင့် set ထဲတွင် မရှိသည့် `"orange"` ကို ဖယ်ထုတ်လျှင် KeyError: ပေးလိမ့်မည်။

```
# discard(x): removes x, does nothing if element is not present
my_set.discard("cherry")
my_set.discard("blueberry")
print(my_set)

# clear() : remove all elements
my_set.clear()
print(my_set)
```

```
{'banana', 'cherry'}
```

```
# pop() : return and remove a random element
a = {True, 2, False, "hi", "hello"}
print(a.pop())
print(a)
```

```
{'banana'}
set()
```

```
# pop() : return and remove a random element
a = {True, 2, False, "hi", "hello"}
print(a.pop())
print(a)
```

```
False
{True, 2, 'hi', 'hello'}
```

8.4 Reading Data from a Set (Iterating)

Set object များအတွင်းရှိ item များကို ဖတ်လိုသည့်အခါ index နံပါတ်ကို အသုံးပြုရန်မဖြစ်နိုင်ပါ။ (Set objects do not support indexes)။ list နှင့် tuple တို့တွင် index နံပါတ်ကိုအသုံးပြု၍(indexing နည်းဖြင့်) item များကို access လုပ်နိုင်သည်။

```
# Iterating over a set by using a for in loop
# Note: order is not important
my_set = {"apple", "banana", "cherry"}
for i in my_set:
    print(i)
```

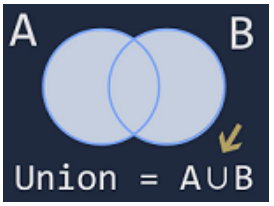
```
apple
```

```
banana
cherry
```

8.5 Set Operations

8.5.1 Union

Union လုပ်သည့် set နှစ်ခုစလုံးတွင် ပါဝင်နေသည့် item များ သို့မဟုတ် element များ အားလုံးကို ယူရသည်။ Union လုပ်ပုံကို အောက်တွင် Venn diagram ဖြင့် သရုပ်ဖော်ထားသည်။ Set A နှင့် set B ကို Union လုပ်လျှင် set A နှင့် set B နှစ်ခုစလုံးမှ item များအားလုံး ပါဝင်သည့် set တစ်ခုကို ပြန်ပေးသည်။

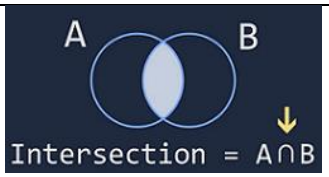



```
odds = {1, 3, 5, 7, 9}
evens = {0, 2, 4, 6, 8}
primes = {2, 3, 5, 7,11}

# union() : combine elements from both sets, no duplication
# note that this does not change the two sets
u = odds.union(evens)
print(u)
# Set (၃) ခု စလုံးဖြင့် union() လုပ်သည်။
u = odds.union(evens, primes)
print(u)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11}
```

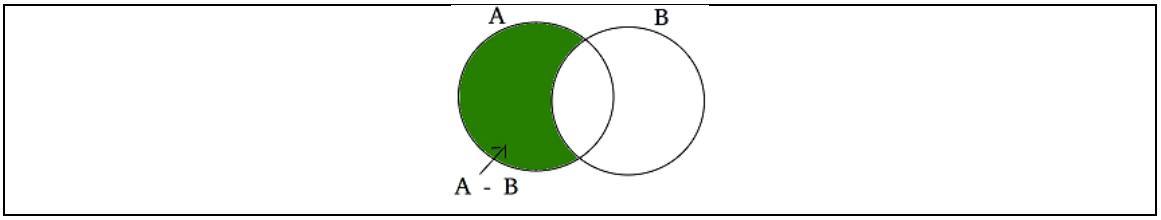
8.5.2 Intersection

Set A နှင့် set B ကို intersection လုပ်လျှင် set A နှင့် set B နှစ်ခုစလုံးထဲတွင် ပါဝင်နေသည့် item များ ပါဝင်သည့် set တစ်ခုကို ပြန်ပေးသည်။




```
# intersection(): take elements that are in both sets
i = odds.intersection(evens)
print(i)

i = odds.intersection(primes)
print(i)

i = evens.intersection(primes)
print(i)
```

```
set()
{3, 5, 7}
{2}
```



8.5.3 Difference of sets

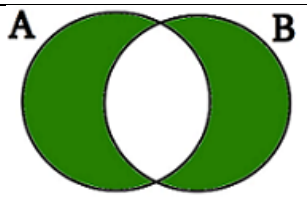
- A - B
  Syntax: **set_A.difference(set_B)** ကို သန်ျာဘာသာရပ်ဖြင့် (A - B)ဟုဖော်ပြသည်။ အဓိပ္ပာယ်မှာ set A ကို မူတည်ထား၍ set B ထဲမှ element များကို နှုတ်ပြီးကျန်သည့် set ဖြစ်သည်။(the elements present in A but not in B)
- B - A
  Syntax: **set_B.difference(set_A)** ကို သန်ျာဘာသာရပ်ဖြင့် (B - A)ဟုဖော်ပြသည်။ အဓိပ္ပာယ်မှာ set B ကို မူတည်ထား၍ set A ထဲမှ element များကို နှုတ်ပြီးကျန်သည့် set ဖြစ်သည်။(the elements present in B but not in A)

```
set_A = {1, 2, 3, 4, 5, 6, 7, 8, 9}
set_B = {1, 2, 3, 10, 11, 12}

# difference() : returns a set with all the elements from the setA that
 are not in setB.
diff_set = set_A.difference(set_B)
print(diff_set)

# A.difference(B) is not the same as B.difference(A)
diff_set = set_B.difference(set_A)
print(diff_set)
```

```
{4, 5, 6, 7, 8, 9}
{10, 11, 12}
```

8.5.4 symmetric_difference()

Python Set ၏ in-built function ဖြစ်သည်။ အဓိပ္ပာယ်မှာ set A နှင့် set B အတွင်းရှိ element များ အားလုံးထဲမှ set A နှင့် set B ထဲတွင် နှစ်ခုစလုံးထဲတွင်ပါဝင်နေသည့် element များကို နုတ်ထားသည့် set ဖြစ်သည်။ အစိမ်ရောင် ချယ်ထားသည့် set သည်symmetric difference set ဖြစ်သည်။ (Symmetric Difference is marked in Green)




```
# symmetric_difference() : returns a set with all the elements that are
 in setA and setB but not in both
diff_set = set_A.symmetric_difference(set_B)
print(diff_set)

# A.symmetric_difference(B) = B.symmetric_difference(A)
diff_set = set_B.symmetric_difference(set_A)
```

```
print(diff_set)
```

```
      {4, 5, 6, 7, 8, 9, 10, 11, 12}
{4, 5, 6, 7, 8, 9, 10, 11, 12}
```

8.6 Updating sets

update() - SetB ထဲမှ element များကို setA ထဲသို့ပေါင်းထည့်သည်။

```
setA = {1, 2, 3, 4, 5, 6, 7, 8, 9}
setB = {1, 2, 3, 10, 11, 12}

# update() : Update the set by adding elements from another set.
#setB ထဲမှ element များကို setA ထဲသို့ပေါင်းထည့်သည်။
setA.update(setB)
print(setA)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

8.6.1 intersection_update()

setA နှင့် setB တို့မှ နှစ်ခုစလုံးတွင် ပါသည့် element များကိုသာ ယူထားသည့် set ဖြစ်သည်။ (Update the set by keeping only the elements found in both)။ တစ်နည်းအားဖြင့်setA နှင့် setB တို့မှ နှစ်ခုစလုံးတွင် မပါသည့် element များကို နှုတ်ပြီးကျန်သည့် set ဖြစ်သည်။ (removes the items that is not present in both sets)

```
# intersection_update() : Update the set by keeping only the elements f
ound in both
# The intersection_update() method removes the items that is not presen
t in both sets
setA = {1, 2, 3, 4, 5, 6, 7, 8, 9}
setB = {1, 2, 3, 10, 11, 12}
setA.intersection_update(setB)
print(setA)
```

```
{1, 2, 3}
```

8.6.2 difference_update()

တခြား set တွင် ရှိသည့် element များကို မိမိ set ထဲမှ ဖယ်ထုတ်ခြင်းဖြင့် update လုပ်သည်။

```
# difference_update() : Update the set by removing elements found in an
other set.
setA = {1, 2, 3, 4, 5, 6, 7, 8, 9}
setA.difference_update(setB)
print(setA)
```

```
{4, 5, 6, 7, 8, 9}
```

8.6.3 symmetric_difference_update()

Set နှစ်ခုစလုံးတွင် ပါဝင်နေသည့် element များကို ဖယ်ထုတ်ပြီး set နှစ်ခုစလုံးမှ ကျန်သည့် element များကို ယူသည်။ Update လုပ်သည်။

```
# symmetric_difference_update() : Update the set by only keeping the el
ements found in either set, but not in both
setA = {1, 2, 3, 4, 5, 6, 7, 8, 9}
setB = {1, 2, 3, 10, 11, 12}
setA.symmetric_difference_update(setB)
print(setA)
```

```
{4, 5, 6, 7, 8, 9, 10, 11, 12}
```

```
# Note: all update methods also work with other iterables as argument,
e.g lists, tuples
setA.update([1, 2, 3, 4, 5, 6])
setA
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

8.6.4 Copying

ကော်ပီကူးနိုင်သည့် နည်းများမှာ .update(), .copy() တို့ဖြစ်သည်။

```
set_org = {1, 2, 3, 4, 5}

# this just copies the reference to the set, so be careful
set_copy = set_org
set_copy
```

```
{1, 2, 3, 4, 5}
```

```
# now modifying the copy also affects the original
set_copy.update([3, 4, 5, 6, 7])
print("Copy Set :", set_copy)
print("Original Set :", set_org)
```

```
Copy Set : {1, 2, 3, 4, 5, 6, 7}
Original Set : {1, 2, 3, 4, 5, 6, 7}
```

```
# use copy() to actually copy the set
set_org = {1, 2, 3, 4, 5}
set_copy = set_org.copy()
set_copy
```

```
{1, 2, 3, 4, 5}
```

```
# now modifying the copy does not affect the original
set_copy.update([3, 4, 5, 6, 7])
print("Copy Set :", set_copy)
```

```
print("Original Set :", set_org)
```

```
Copy Set : {1, 2, 3, 4, 5, 6, 7}
Original Set : {1, 2, 3, 4, 5}
```

8.7 Subset, Superset, and Disjoint

8.7.1 issubset() Method

Set တစ်ခုသည် တခြား set တစ်ခု၏ အစိတ်အပိုင်းတစ်ခု ဖြစ် မဖြစ်ကို .issubset() ဖြင့် စစ်နိုင်သည်။ set တစ်ခု element များအားလုံးသည် တခြား Set တစ်ခုထဲတွင် ပါဝင်နေလျှင် True ပေးလိမ့်မည်။

```
setA = {1, 2, 3, 4, 5, 6}
setB = {1, 2, 3}
# Return True if all items setB are present in set A
print(setA.issubset(setB))
# True - setB သည် setA ၏ အစိတ်အပိုင်းတစ်ခု ဖြစ်သည်။
print(setB.issubset(setA))
```

```
False
True
```

8.7.2 issuperset() Method

set x မှ item များ အားလုံးသည် set y ထဲတွင် ရှိနေလျှင် ပြန်ပေးသည်။ (return True if all items set x are present in set y:)

```
x = {"a", "b", "c"}
y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)
print(z)
```

```
True
```

```
# issuperset(setX): Returns True if the set contains setX
print(setA.issuperset(setB))            # True
print(setB.issuperset(setA))
```

```
True
False
```

8.7.3 isdisjoint() Method

set x ထဲမှ item တစ်ခုမျှ set y ထဲတွင် မရှိလျှင် True ပြန်ပေးသည်။ (Return True if no items in set x is present in set y)

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "facebook"}

z = x.isdisjoint(y)
```

```
# isdisjoint(setX) : Return True if both sets have a null intersection,
 i.e. no same elements
setC = {7, 8, 9}
print(setA.isdisjoint(setB))
print(setA.isdisjoint(setC))
```

```
False
True
```

8.8 Frozenset

Frozen set ဆိုသည်မှာ ပြင်၍ ပြောင်း၍(immutable) set ဖြစ်သည်။ Frozen set ကို တည်ဆောက်ပြီးသည့်နောက် ပြင်၍ ပြောင်း၍ မရတော့ပေ။

Frozen set is just an immutable version of normal set. While elements of a set can be modified at any time, elements of frozen set remains the same after creation. Creation with:

`my_frozenset = frozenset(iterable)`

```
a = frozenset([0, 1, 2, 3, 4])
odds = frozenset({1, 3, 5, 7, 9})
evens = frozenset({0, 2, 4, 6, 8})

print(odds.union(evens))
print(odds.intersection(evens))
print(odds.difference(evens))
```

```
frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
frozenset()
frozenset({1, 3, 5, 7, 9})
```

```
b = frozenset('asdfagsa')
print(b)
```

```
frozenset({'d', 'g', 'f', 's', 'a'})
```

```
frozenset({'f', 'g', 'd', 'a', 's'})
cities = frozenset(["Frankfurt", "Basel","Freiburg"])
print(cities)
```

```
frozenset({'Freiburg', 'Basel', 'Frankfurt'})
```

Passing a dictionary to the set() method will create a set of its key

```
print(dir(set))
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__',
 '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt_
_', '__hash__', '__iand__', '__init__', '__init_subclass__', '__ior__',
'__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne
__', '__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__
repr__', '__ror__', '__rsub__', '__rxor__', '__setattr__', '__sizeof__',
```

```
 '__str__', '__sub__', '__subclasshook__', '__xor__', 'add', 'clear', 'c
opy', 'difference', 'difference_update', 'discard', 'intersection', 'int
ersection_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remov
e', 'symmetric_difference', 'symmetric_difference_update', 'union', 'upd
ate']
```

8.9 Set Comprehensions

Set များကို for loop သုံး၍ တည်ဆောက်နိုင်သည်။

```
# အဖြေများကို ထည့်ရန် empty set တစ်ခု တည်ဆောက်သည်။
squares = set()
for i in range(5):
    squares.add(i * i) # squares set ထဲသို့ ကိန်းကို နှစ်ထပ်တင်၍ တစ်ခုချင်းစီထည့်သည်။

print(squares)              # ရလဒ်များပါသည့် squares list ကို ဖော်ပြသည်။
```

```
{0, 1, 4, 9, 16}
```

အထက်တွင် ဖော်ပြခဲ့သည့် ပို၍ ကျစ်လစ်သည့်နည်းဖြင့် ပြန်ရေးနိုင်သည်။ ist comprehension နည်းဖြစ်သည်။

```
new_list = [expression for member in iterable]
```

```
# better: use list comprehension
# new_list = [expression for member in iterable]
squares = {i * i for i in range(5)}
print(squares)
```

```
{0, 1, 4, 9, 16}
```

```
# set comprehension
quote = "hello everybody"
unique_vowels = {i for i in quote if i in 'aeiou'}
print(unique_vowels)

squares = {i: i * i for i in range(5)}
print(squares)
```

```
{'e', 'o'}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

```
#Set များကို for loop သုံး၍ တည်ဆောက်နိုင်သည်။
#{ expression for expression in iterable }
a_set = {number for number in range(1,6) if number % 3 == 1}
a_set
```

```
{1, 4}
```

-End of Set-