Chapter-9

Data Structure Questions and Answers

19.1 Python Data Structure မေးခွန်းများ

List နှင့် သက်ဆိုင်သည့် မေးခွန်း (၄၅)ခုကို စုစည်းဖော်ပြထားသည်။ မေးခွန်းများသည် အတွေးတချို့ လှုံ့ဆော်ပေးသည်။ နားလည်ထားမှုကို ပြန်လည်သုံးသပ်စေသည်။ Data structure တွင် list သည် အများဆုံး အသုံးပြုသည် အမျိုးအစားဖြစ်သောကြောင့် list မေးခွန်းများ အဓိကပါဝင်သော်လည်း tuple, dictionary နှင့် set ဒေတာအမျိုးအစားများအကြောင်းလည်း ပါဝင်သည်။

(၁) List တစ်ခုထဲ၌ မိမိသိလိုသည့် element တစ်ခု ပါ မပါကို စစ်ဆေးပါ။

"in" operator ကို အသုံးပြု၍ စစ်ဆေးနိုင်သည်။

```
my_list = [1,2,3, 'a', 'b', 'c']
'a' in my_list
```

```
True
```

၂။ (၂) ခုထက် ပိုများတဲ့ List တွေကို တပြိုင်နက် ဘယ်လို iterate မလဲ

List နှစ်ခု သို့မဟုတ် သုံးခု အတွင်းရှိ item များကို တစ်ပြိုင်နက် iterate လုပ်ရန်အတွက် **zip()** ရမည်။ အောက်တွင် list (၃)ခုကို **zip()** method ဖြင့် အတူတွဲပြီးနောက် for loop ဖြင့် iterate လုပ်သည်။

```
name = ['Snowball', 'Chewy', 'Bubbles', 'Gruff']
animal = ['Cat', 'Dog', 'Fish', 'Goat']
age = [1, 2, 2, 6]
z = zip(name, animal, age)
z
```

```
<zip at 0x5b1de48>
```

```
for name,animal,age in z:
    print("%s the %s is %s" % (name, animal, age))
```

```
Snowball the Cat is 1
Chewy the Dog is 2
Bubbles the Fish is 2
Gruff the Goat is 6
```

(၃) ဘယ်လိုနေရာမှာ dictionary ကို အသုံးပြုမလဲ? ဘယ်လိုနေရာမှာ list ကို အသုံးပြုမလဲ?

List - ဒေတာများကို သိမ်းထားသည့် နေရာ အစီအစဉ်(order)နှင့်တကွ သိမ်းဆည်းရန်လိုအပ်ပါက list ကို သုံးပါ။

Dictionary - ဒေတာများ၏ သိမ်းထားသည့် နေရာ အစီအစဉ်(order) ကို မလိုအပ်ဘဲ key နှင့် value များကို တွဲ၍ သိမ်းထားလိုပါက သုံးပြုပါ။ ဥပမာ-အိမ်မွေး နာမည်(key)နှင့် တိရစ္ဆာန် အကောင် အရေအတွက် (value) ကို တွဲ၍ သိမ်းထားရန် dictionary ကို သုံးသည်။

```
ids = [23,1,7,9]
print(ids)
pets = {'dogs':2,'cats':1,'fish':5}
print(pets)
```

```
[23, 1, 7, 9]
{'dogs': 2, 'cats': 1, 'fish': 5}
```

(၄) List ထဲတွင် ပါဝင်နေသည့် item များကို ပြင်ရန် ပြောင်းရန် ဖြစ်နိုင်ပါသလား?(Is a list mutable?)

Yes.

(၅) List ထဲတွင် မတူညီသည့် ဒေတာအမျိုးအစား ထည့်ထားရန် ဖြစ်နိုင်ပါသလား? တစ်မျိုးတည်း (homogeneous)သာ ဖြစ်ရပါမည်လား?

အမျိုးအစားများမတူသော ဒေတာများ(different types of object)ကို list တစ်ခုထဲတွင် အတူတကွ ရောထွေး သိမ်းဆည်းနိုင်ပါသည်။ List ထဲတွင် တခြားသော list များကို ထပ်ထည့်ထားလို့ရသည်။

```
# 1 သည် integer ဒေတာ အမျိုးအစား, 'a' သည် string , 3.517 သည် float ဖြစ်သည်။
a = [1, 'a', 3.517, []]
```

(၆) append နှင့် extend တို့၏ ခြားနားချက်ကို ရှင်းပြပါ။?

.append() သည် ထည့်ခိုင်းသည့် item ကို list ၏ နောက်ဆုံးနေရာတွင် ထည့်သည်။ (adds an object to the end of a list.)

```
a = [1,2,3]
a.append(4)
a
```

```
[1, 2, 3, 4]
```

တခြားသော list တစ်ခုကို single element အဖြစ်သတ်မှတ်ပြီး list တစ်ခု ထဲသို့ ထည့်နိုင်သည်။

```
a.append([5,6])
a
```

```
[1, 2, 3, 4, [5, 6]]
```

.extend() ကို သုံး၍ list တစ်ခု ထဲသို့ တခြားသော list တစ်ခုကို ထည့်သည့်အခါ (extend လုပ်သည့် အခါ) single element အဖြစ် ရောက်မသွားလဲ ဒုတိယ list မှ element များ အားလုံးသည် element တစ်ခုချင်းစီ အဖြစ် ဝင်ရောက်သွားသည်။

```
b = [1,2,3]
b.extend([5,6])
b
```

```
[1, 2, 3, 5, 6]
```

(၇) Python list သည် တန်ဖိုးများ(values)ကို သိမ်ဆည်းပါသလား? Memory Pointer ကို သိမ်းဆည်းပါလား?

Python list သည် တန်ဖိုးများကို မသိမ်းဆည်းပါ။ List ထဲတွင် value များကို သိမ်းဆည်းထားသည့် မှတ်ဉာဏ်(memory) နေရာကို ညွှန်ပြသည့် pointer များကိုသာ သိမ်းဆည်းထားသည်။ ထိုသို့ memory နေရာကို ညွှန်ပြသည့် pointer များကိုသာ သိမ်းဆည်းထားသောကြောင့် ပြင်နိုင် ပြောင်းနိုင်ခြင်းဖြစ်သည်။ (mutable ဖြစ်သည်။)

Value များ ဖြစ်သည့် integer 1 နှင့် 2 တို့၏ initialize လုပ်ပြီး identity (unique integer) ဖတ်ယူသည်။ ထို့နောက် integer 1 နှင့် 2 ပါသည့် list တစ်ခုကို တည်ဆောက်သည်။ List ၏ နေရာများ၏ identity (unique integer) ဖတ်ယူသည်။

```
#The id() function returns identity (unique integer) of an object.
a, b = 1, 2
print(id(a))
#object တစ်ခု၏ identity (unique integer)ကို သိလိုပါက id() function ကို အသုံးပြုရသည်။
print(id(b))
```

```
140732981420432
140732981420464
```

```
c = [1,2,3]
print( id(c) )
print( id(c[0]) )
print( id(c[1]) )
```

```
2944936958344
140732981420432
140732981420464
```

integer 1 သိမ်းထားသည့်နေရာ → id(c[0]) = id(a) = 140732981420432

integer 2 သိမ်းထားသည့်နေရာ → id(c[1]) = id(b) = 140732981420464

List တွင် ကိုယ်ပိုင် memory address ရှိသည်။ (list has its own memory address)။ List အတွင်းရှိ integer 1 နှင့် 2 သည် memory ပေါ်တွင် ရှိသည့် integer 1 နှင့် 2 သိမ်းထားသည့်နေရာကို ညွှန်ပြ(point လုပ်)သည်။ ထို့ကြောင့် list ထဲတွင် value များကို သိမ်းဆည်းထားသည့်မှတ်ဉာဏ်နေရာ (memory pointer) ကို သိမ်းဆည်း ထားသည်။

(၈) Del ကို မည်ကဲ့သို့ သုံးမည်နည်း?

List ထဲမှ ဖယ်ထုတ်လိုသည့် item များကို index နံပါတ်ပေး၍ ဖျက်ဖြစ်နိုင်သည်။ **del a[:]** သည် item အားလုံးကို ဖျက်ပေး (delete လုပ်) ပေးသည်။

```
a = ['w', 'X', 'y', 'z']
print(a)
del a[1]
```

```
print(a)
```

```
['w', 'X', 'y', 'z']
['w', 'y', 'z']
```

del သည် ဖယ်ထုတ်လိုက်သည့် item ကို ပြန်မပေးပါ။ (does not return the removed element)

(၉) .remove() နှင့် .pop() တို့၏ ကွာခြားချက်ကို ရှင်းပြပါ။

.remove() သည် ဖယ်ထုတ်လိုသည့် item များထဲမှ ပထမဆုံးတွေ့သည့် index နံပါတ် (အနည်းဆုံး) ကို ဖယ်ထုတ် ပေးသည်။(.remove() removes the first instance of a matching object) ဥပမာ - အောက်တွင် ပထမဆုံးတွေ့သည့် b ကို ဖယ်ထုတ်သည်။

```
a = ['a', 'a', 'b', 'c', 'c', 'b', 'd', 'e']
print(a)
a.remove('b')
print(a)
```

```
['a', 'a', 'b', 'c', 'c', 'b', 'd', 'e']
['a', 'a', 'c', 'c', 'b', 'd', 'e']
```

.pop() သည် index နံပါတ်အတိုင်း ဖယ်ရှားပေးပါသည်။ (removes an object by its index)။ “remove” နှင့် “pop” တို့၏ ကွာခြားချက်မှာ pop သည့် ဖယ်ရှားလိုက်သည့် element ကို ပြန်ပေးသည်။ ထို element ကို သိနိုင်သည်။ (pop returns the popped element)

```
a = ['a', 'a', 'b', 'b', 'c', 'c']
a.pop(4)
print(a)
print(a.pop(4))                # pop လုပ်ထားသည့် element ကို ပြန်ခေါ်ယူနိုင်သည်။
```

```
['a', 'a', 'b', 'b', 'c']
c
```

.pop()ကို သုံး၍ element ဖယ်ထုတ်သည့်အခါ index နံပါတ်ကို ထည့်မပေးပါက နောက်ဆုံးမှ element ကို ဖယ်ရှားသည်။ (By default, pop removes the last element from a list if an index isn’t specified.)

(၁၀) List တစ်ခု အတွင်းမှ တူညီနေသည့်(duplicate) item များကို ဖယ်ရှားရန် မည်သို့ပြုလုပ် မည်နည်း။

List ကို set အဖြစ် ပြောင်းပါ။ ထိုအခါ list အတွင်းမှ တူညီနေသည့်(duplicate) item များကို ဖယ်ရှား ပစ်လိမ့်မည်။ ထိုနောက် list အဖြစ်သို့ ပြန်ပြောင်းပါ။ (converting to a set and back to a list)။ List ၏ အစဉ် (order) ကို ထိန်းသိမ်းထားရန် မဖြစ်နိုင်ပါ။

```
li = [3, 2, 2, 1, 1, 1]
list(set(li))
```

```
[1, 2, 3]
```

(၁၁) List တစ်ခုအတွင်းသို့ ရှာလိုသည့် element တစ်ခု ထည့်ပေးပြီး ပထမဆုံးတွေ့သည့် element ၏ index နံပါတ်ကို ရှာပါ။ (Find the index of the 1st matching element)

.index() method ကိုသုံး၍ ရှာသည်။

```
fruit = ['pear', 'orange', 'apple', 'grapefruit', 'apple', 'pear']
fruit.index('apple') #''apple'နှစ်ခုတွင် ပထမဆုံးတွေ့သည့် index နံပါတ်ကို ပေးသည်။
```

```
2
```

```
fruit.index('pear') # pear' နှစ်ခုတွင် ပထမဆုံးတွေ့သည့် index နံပါတ်ကို ပေးသည်။
```

```
0
```

(၁၂) List တစ်ခုအတွင်း element များ အားလုံးကို ဖယ်ရှားပါ။

.clear() method ကိုသုံး၍ list တစ်ခုအတွင်း element များ အားလုံးကို ဖယ်ရှားနိုင်သည်။ Empty list အဖြစ် ကျန်ခဲ့လိမ့်မည်။

```
fruit = ['pear', 'orange', 'apple']
print( fruit )                # ['pear', 'orange', 'apple']
print( id(fruit) )            # list id()
```

```
['pear', 'orange', 'apple']
95541256
```

```
fruit.clear()
print(fruit)     # empty list
print(id(fruit)) # list id()
```

```
[]
95541256
```

Del ကို သုံး၍လည်း list တစ်ခုအတွင်း element များ အားလုံးကို ဖျက်နိုင်သည်။

```
fruit = ['pear', 'orange', 'apple']
print(fruit)
print(id(fruit))
```

```
['pear', 'orange', 'apple']
47386312
```

```
del fruit[:]
print(fruit)
print(id(fruit))
```

```
[]
47386312
```

(၁၃) List တစ်ခုအတွင်းရှိ item များအားလုံး၏ value နှင့် ၎င်းတို့၏ နံပါတ်များ(their indices)ကို ဖော်ပြပါ။ Iterate လုပ်ပါ။

enumerate() သည် list တစ်ခုအတွင်း သို့ဝင်ရောက်လာသည့် argument များကို ရေတွက်ပေးသည်။ List တစ်ခု၏ value နှင့် index နှစ်မျိုးလုံးကို ထည့်၍ iterate လုပ်သည်။

```
grocery_list = ['flour','cheese','carrots']
for idx,val in enumerate(grocery_list):
    print("%i : %s" % (idx, val))
```

```
0 : flour
1 : cheese
2 : carrots
```

(၁၄) List နှစ်ခုကို ဆက်ပါ။

+ operator ကိုသုံး၍ list နှစ်ခုကို ဆက်နိုင်သည်။ Concatenate လုပ်နိုင်သည်။

```
one = ['a', 'b', 'c']
two = [1, 2, 3]
one + two
```

```
['a', 'b', 'c', 1, 2, 3]
```

List နှစ်ခုတည်းသာမက နှစ်ခုထက်ပိုများသည့် list များကိုလည်း ဆက်နိုင်သည်။ Concatenate လုပ် နိုင်သည်။

```
one = ['a', 'b', 'c']
two = [1, 2, 3]
three = ['x', 'y', 'z']
one + two + three
```

```
['a', 'b', 'c', 1, 2, 3, 'x', 'y', 'z']
```

(၁၅) List comprehension နည်းဖြင့် list တစ်ခုအတွင်းရှိ element များကို ပေါင်း၊ နုတ်၊ မြှောက် စားလုပ်ပါ။ (manipulate ပါ။)

အောက်တွင် list တစ်ခုအတွင်းရှိ ကိန်းများကို တစ်ပေါင်း၍ list အသစ် တစ်ခုတည်ဆောက်ထားပုံကို ဥပမာ ပြထားသည်။

```
li = [0,25,50,100]
[i+1 for i in li]
```

```
[1, 26, 51, 101]
```

(၁၆) List တစ်ခုအတွင်းရှိ item တစ်ခု ပါဝင်နေသည့် အကြိမ်အရေအတွက်(occurrence of a specific item)ကို ရှာပါ။

count() method ဖြင့် list တစ်ခုအတွင်းရှိ item တစ်ခု ပါဝင်နေသည့် အကြိမ်အရေအတွက် (occurrence of a specific item)ကို ရှာနိုင်သည်။ Pets ဆိုသည့် list ထဲတွင် “fish” ဆိုသည့် item အကြိမ် အရေအတွက် မည်မျှ ပါဝင်နေသည်ကို ဖော်ပြသည်။

```
pets = ['dog','cat','fish','fish','cat']
pets.count('fish')
```

2

(၁၇) List တစ်ခုကို shallow copy ကော်ပီကူးပါ။

.copy() ကိုသုံး၍ list ကို shallow copy ကူးနိုင်သည်။ အောက်တွင် round2 သည် round1 ၏ shallow copy ဖြစ်သည်။ round2 ထဲမှ string sonny chiba. ကို ဖယ်ထုတ်(remove)သည်။ Shallow copy မရှိလျှင် round1 နှင့် round2 တို့သည် memory ပေါ်တွင်ရှိသည့် list တစ်ခုတည်း၏ နေရာကို ညွှန်ပြနေကြ လိမ့်မည်။ (round1 and round2 are just names pointing to the same list in memory.) အောက်တွင် round2 သည် round1 ၏ shallow copy ဖြစ်သည်။

```
round1 = ['chuck norris', 'bruce lee', 'sonny chiba']
round2 = round1
round2.remove('sonny chiba')          #(assignment operator)

print(round1)
print(round2)
```

```
['chuck norris', 'bruce lee']
['chuck norris', 'bruce lee']
```

(၁၈) List နှင့် tuple ရဲ့ ကွာခြားချက်က ဘာတွေလဲ

Tuple ကို update လုပ်လို့မရပါ။ Adding/removing/updating လုပ်လို့ မရပါ။

Lists တွေကို modifiy လုပ်လို့ရပါတယ်။ List နှင့် tuple နှစ်ခုစလုံးတို့သည် sequence များကြပြီး value တူသည့် item များကို လက်ခံထားနိုင်သည်။ (allow duplicate values)

(၁၉) List တစ်ခုရဲ့ အရှည် သို့မဟုတ် list တစ်ခုထဲမှာ ပါတဲ့ ကိန်းအရေအတွက်ကို ဘယ်လို ရှာမလဲ။

Len() ဖြင့် list တစ်ခုရဲ့ အရှည် သို့မဟုတ် list တစ်ခုထဲမှာ ပါတဲ့ item အရေအတွက်(length of a list)ကို ရှာနိုင်သည်။

```
my_list = ['a', 'b', 'c', 'd', 'e']
len(my_list)
```

5

Top level object များကိုသာ ရေတွက်ပေးသည်ကို သတိပြုပါ။ Nested list ဖြစ်လျှင် sub-list အတွင်းရှိ item များကို မရေတွက်ပါ။

```
li = [[1,2],[3,4,5]]
```

```
len(li)
```

```
2
```

(၂၀) List နှင့် set တို့၏ ကွာခြားချက်ကို ပြောပါ။

List တွင် order သည် အရေးကြီးသည်။ List ၏ အသက်သည် order ဖြစ်သည်။ Duplicate item များကို ခွင့်ပြုသည်။

Set တွင် order သည် အရေးကြီးသည့်အရာ မဟုတ်ပါ။ Unique value များကိုသာ သိမ်းဆည်းပေးသည်။ Duplicate item များကို ခွင့်မပြုပါ။

(၂၁) List တစ်ခုအတွင်း၌ မိမိသိလိုသည့် element တစ်ခု ရှိ မရှိ ဘယ်လိုစစ်မလဲ။

"**in**" operator ကို အသုံးပြု၍ စစ်နိုင်သည်။

```
li = [1,2,3,4]
print(5 not in li)
print(4 not in li)
```

```
True
False
```

(၂၂) List တစ်ခုအတွင်းရှိ element များအားလုံးကို (၅)ဆ တိုးထားသည့် list အသစ်တစ်ခု တည်ဆောက်ပါ။

```
numbers = [1, 2, 3, 4, 5]                # ပေးထားသည့် list
```

(၅)ဖြင့် မြှောက်ထားသည့်ကိန်းများ ထည့်ရန်(append လုပ်ရန်) တည်ဆောက်ထားသည့် empty list ပြုလုပ်သည်။ for loop ဖြင့် ကိန်းတစ်လုံးချင်းစီကို (၅)ဖြင့် မြှောက်၍ empty list ထဲ ထည့်သည်။

```
new_list = [ ]
for number in numbers:
    new_list.append(5 * number) # ကိန်းတစ်လုံးချင်းစီကို (၅)ဖြင့် မြှောက်သည်။
print(new_list)
```

```
[5, 10, 15, 20, 25]
```

(၂၃) List နှစ်ခုမှ item များကို တစ်ခုချင်းစီတွဲပြီး item နှစ်ခုပါသည့် tuple များတည်ဆောက်ပါ။ ထို့နောက် tuple များပါသည့် list အသစ်တစ်ခု တည်ဆောက်ပြပါ။

zip() သည် sequence များစွာထဲမှ item များကို တစ်ခုချင်းစီတွဲပြီး tuple တစ်ခု အဖြစ် တည်ဆောက် ပေးနိုင်သည်။(တူညီသည့် index နံပါတ်များမှ value များကိုတွဲပြီး)။

```
alphabet = ['a', 'b', 'c']
integers = [1, 2, 3]
list(zip( integers,  alphabet))
```

```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

```
list(zip(alphabet, alphabet))
```

```
[('a', 'a'), ('b', 'b'), ('c', 'c')]
```

```
list(zip(alphabet, integers))
```

```
[('a', 1), ('b', 2), ('c', 3)]
```

zip() သည် (၂)ခုထက် ပိုများသည့် list များကိုလည်း zip လုပ်နိုင်သည်။

```
alphabet = ['a', 'b', 'c']
integers = [1, 2, 3]
alphabet2 = ['x', 'y', 'z']
list(zip(integers,alphabet, alphabet2))
```

```
[(1, 'a', 'x'), (2, 'b', 'y'), (3, 'c', 'z')]
```

(၂၄) List တစ်ခုအတွင်းမှ မိမိအလိုရှိသည့် index နံပါတ်တစ်ခုတွင် မိမိအလိုရှိသည့် value ထည့်သွင်းပါ။

List တစ်ခုအတွင်းမှ မိမိအလိုရှိသည့် index နံပါတ်တစ်ခုတွင် မိမိအလိုရှိသည့် value ထည့်သွင်းနိုင်သည်။ ထိုသို့ ထည့်သွင်းလိုက်ခြင်းကြောင့် အရင်နေရာဟောင်းမှ value သည် နောက်သို့ ရွှေ့သွားသည်။ မိမိအလိုရှိသည့် value တန်ဖိုးကို list အတွင်းသို့ index နံပါတ်မှ တစ်ဆင့် ထည့်သည်။

insert() method ကို သုံးသည့်အခါ ထည့်မည့်နေရာ၏ index နံပါတ်နှင့် ထို့နေရာတွင်ထည့်မည့် value ကို ထည့်ပေးရသည်။ Overwritten လုပ်လိုက်ခြင်း မဟုတ်ပါ။ Insert လုပ်ခြင်းကြောင့် ကိန်းအလုံးရေ (length) ပိုတိုးလာလိမ့်မည်။

```
li = ['a','b','c','d','e']
li.insert(2, 'HERE')
li #=> ['a', 'b', 'HERE', 'c', 'd', 'e']
```

```
['a', 'b', 'HERE', 'c', 'd', 'e']
```

(၂၅) reduce() function ကို သုံး၍ list ထဲမှ value များကို subtract လုပ်ပါ။

```
from functools import reduce
def subtract(a,b):
    return a - b

numbers = [100,10,5,1,2,7,5]
reduce(subtract, numbers)
```

```
70
```

(၂၆) List တစ်ခုထဲမှ အနုတ်ကိန်းများကို ဖယ်ထုတ်ပါ။

Empty list တစ်ခု ပြုလုပ်ပြီး သုည(zero) ထက်ကြီးသည့် item များကိုသာ for loop ဖြင့် append လုပ်မည်။

```
numbers = [-10, 27, 1000, -1, 0, -30] # ပေးထားသည့် list
# ပေါင်းကိန်းများ ထည့်ရန်( append လုပ်ရန်) တည်ဆောက်ထားသည့် empty list
positive_number = [ ]
```

```
for number in numbers:
    if number >= 0:                    # ကိန်းတစ်လုံးချင်းကို ပေါင်းကိန်း ဖြစ် မဖြစ် စစ်သည်။
        positive_number.append(number) # ပေါင်းကိန်း ဖြစ်လျှင် positive_number
print(positive_number)
```

```
[27, 1000, 0]
```

(၂၇) List ထဲမှ element များကို key အဖြစ်ထားပြီး dictionary အဖြစ်သို့ ပြောင်းပါ။

Dictionary comprehension နည်းကို အသုံးပြုသည်။ Value များအားလုံးကို 1 အဖြစ်ထားမည်။

```
li = ['The', 'quick', 'brown', 'fox', 'was', 'quick']
d = {k:1 for k in li}
d
```

```
{'The': 1, 'quick': 1, 'brown': 1, 'fox': 1, 'was': 1}
```

(၂၈) Lambda function သုံး၍ ပေးထားသည့် list ထဲရှိ item များကို (၅)ဆတိုး၍ modify လုပ်ပါ။

map function ကို အသုံးပြုသည်။

```
a = [10,20,30,40,50]
list(map(lambda val:val*5, a))
```

```
[50, 100, 150, 200, 250]
```

(၂၉) သတ်မှတ်ထားသည့်(specific) index နံပါတ်၏ နောက်ပိုင်းမှ element များကို ဖယ်ရှားပစ်ပါ။ (Remove elements in a list after a specific index)

slicing နည်းကို သုံးသည်။ slice syntax ကို သုံး၍ list အသစ်တစ်ခု တည်ဆောက်သည်။ ထို list ထဲတွင် သတ်မှတ်ထားသည့်(specific) index နံပါတ်အထိသာ ပါသည်။ `li[ :10]` သည် အစဆုံးမှ item နံပါတ် 10 အထိကိုသာ ယူသည်။

```
li = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,10]
li[:10]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

(၃၀) သတ်မှတ်ထားသည့်(specific) index နံပါတ်၏ အရှေ့မှ element များကို ဖယ်ရှားပစ်ပါ။ (Remove elements in a list before a specific index)

Slice syntax ကို သုံး၍ list အသစ်တစ်ခုတည်ဆောက်သည်။ သတ်မှတ်ထားသည့်(specific) index နံပါတ်ကို start နေရာတွင်ထည့်ပေးရသည်။ index နံပါတ် 15 နောက်ပိုင်း " `li[15:]` " element များ အာလုံးကို ယူသည်။ Slice syntax သည် slice လုပ်လိုသည့် item များ ပါဝင်သည့် list အသစ်တစ်ခုကို ပြန်ပေးသည်။

```
li = [0, 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,10]
li[15:]
```

```
[15, 16, 17, 18, 19, 10]
```

(၃၁) ငယ်ရာမှ ကြီးရာသို့ စဉ်ပါ။ (Sort a list of integers in ascending order)

sort() method ကိုသုံး၍ ငယ်ရာမှ ကြီးရာသို့(ascending order) စဉ်သည်။

```
li = [10,1,9,2,8,3,7,4,6,5]
li.sort()
li
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

(၃၂) ကြီးရာမှ ငယ်ရာသို့ စဉ်ပါ။ (Sort a list of integers in descending order)

Argument (reverse=True) ကို သုံး၍ ပြောင်းပြန်းစဉ်(descending order) နိုင်သည်။ ငယ်ရာမှ ကြီးရာသို့(ascending order) သည် default ဖြစ်သောကြောင့် ထည့်ရေးရန် မလိုပါ။

```
li = [10,1,9,2,8,3,7,4,6,5]
li.sort(reverse=True)
li
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

(၃၃) List comprehension နည်းကို သုံး၍ မကိန်းများကို ယူပြီး list အသစ်တစ်ခု တည်ဆောက်ပါ။

မကိန်းများကိုသာ ယူရန် filter လုပ်သည်။ Filter လုပ်ရန်အတွက် list comprehension ထဲတွင် conditional logic ထည့်ရေးသည်။ (၂)ဖြင့်စားပြီး အကြွင်းသုည မဖြစ်လျှင်(2 != 0) ထိုကိန်းသည် မကိန်း ဖြစ်သည်။

```
li = [1,2,3,4,5,6,7,8,9,10]
[i for i in li if i % 2 != 0]
```

```
[1, 3, 5, 7, 9]
```

(၃၄) Value တစ်ခုချင်းစီ၏ ပါဝင်သည့် အကြိမ်အရေအတွက်ကို ရေတွက်ပါ။ (Count occurrences of each value in a list)

အလွယ်ဆုံးနည်းမှ **collections** module မှ **counter** class ကို import လုပ်၍ အသုံးပြုခြင်း ဖြစ်သည်။

```
li = ['blue', 'pink', 'green', 'green', 'yellow', 'pink', 'orange']
# import counter class from collections module
from collections import Counter
print(Counter(li))
```

```
Counter({'pink': 2, 'green': 2, 'blue': 1, 'yellow': 1, 'orange': 1})
```

(၃၅) Nested list အတွင်းရှိ list တိုင်း၏ index နံပါတ် သုည(first element)ကိုသာ ယူ၍ list အသစ်တစ်ခု တည်ဆောက်ပါ။

```
li = [[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15]]
[i[0] for i in li]
```

```
[1, 4, 7, 10, 13]
```

(၃၆) List အတွင်းရှိ element များကို string တစ်ခု အဖြစ် ပြောင်းပါ။ (Combine elements in a list into a single string.)

.join() ကို သုံးသည်။ ဆက်သည်အခါ(join လုပ်သည့်အခါ)element တစ်ခုနှင့် တစ်ခု အကြား space တစ်နေရာစာ ခြားလိုသောကြောင့် ' ' ထည့်ပေးသည်။

```
li = ['The','quick','brown', 'fox', 'jumped', 'over', 'the', 'lazy',
        'dog']
' '.join(li)
```

```
'The quick brown fox jumped over the lazy dog'
```

(၃၇) List တစ်ခုကို integer တစ်ခုနှင့် မြှောက်ပါ။

List တစ်ခုကို integer တစ်ခုနှင့် မြှောက်ခြင်းကို multiple concatenation ဟုခေါ်သည်။ List တစ်ခုကို (၅)နှင့် မြှောက်သည့်အခါ element တိုင်း၏ အရေအတွက် (၅)တိုးသွားသည်။ element တစ်ခု ချင်းစီ၏ value (5)ဆ တိုးခြင်း မဟုတ်ပါ။

```
['a','b'] * 5
```

```
['a', 'b', 'a', 'b', 'a', 'b', 'a', 'b', 'a', 'b']
```

```
# (၅)ခါပေါင်းခြင်းနှင့် တူညီသည်။
['a','b'] + ['a','b'] + ['a','b'] + ['a','b'] + ['a','b']
```

```
['a', 'b', 'a', 'b', 'a', 'b', 'a', 'b', 'a', 'b']
```

(၃၈) List တစ်ခု၏ item အစဉ် များကို ပြောင်းပြန်လှန်ပါ။ (Reverse the order of a list)

reverse() method ကို အသုံးပြု၍ item အစဉ် များကို ပြောင်းပြန်လှန် နိုင်သည်။ List အသစ် တခု ပြုလုပ်ပေးခြင်း မဟုတ်သည်ကို သတိပြုပါ။ Tuple , Set နှင့် dict တို့ကို reverse() လုပ်၍ မရပါ။ ၄င်းတို့တွင် 'reverse' attribute မရှိပါ။

```
li = [1,2,3,4,5,6,7,8,9,10]
li.reverse()
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

(၃၉) Reverse နှင့် reversed တို့၏ ခြားနားချက်

reverse() သည် item များကို ထို မူလ list အတွင်း၌သာ ပြန်စီသည်။ (reverse() reverses the list in place.) **reversed()** သည် ပြန်စီထားသည့် list အသစ်တစ်ခု ပေးသည်။ (returns an iterable of the list in reverse order.)

```
li = [1,2,3,4,5,6,7,8,9,10]
li.reverse()
li
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
li = [1,2,3,4,5,6,7,8,9,10]
list(reversed(li))
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

(၄၀) Sort နှင့် sorted တို့၏ ခြားနားချက်

sort() သည် item များကို ထို list အတွင်း၌သာ ပြန်စီ ပေးသည်။ ရှိပြီးသား မူလ list ကို modify လုပ်ပေးသည်။ (modifies the list in place)။ **sorted()** သည် ပြန်စီထားသည့် list အသစ်တစ်ခု ပေးသည်။ (returns a new list in reverse order.)

```
li = [10,1,9,2,8,3,7,4,6,5]
print(id(li))
```

```
2772155439816
```

```
li.sort()
print(li)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
print(id(li), "Same id")
```

```
2772155439816 Same id
```

```
li = [10,1,9,2,8,3,7,4,6,5]
sorted(li)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

(၄၁) List တစ်ခု အတွင်းရှိ အငယ်ဆုံးတန်ဖိုး(minimum value)ကို ဖော်ပြပါ။

min() function သည် list တစ်ခု အတွင်းရှိ အငယ်ဆုံးတန်ဖိုး(minimum value)ကို ရှာပေးသည်။ **max()** function သည် list တစ်ခု အတွင်းရှိ အကြီးဆုံးတန်ဖိုး(maximum value)ကို ရှာပေးသည်။ **sum()** function သည် ကိန်းများအားလုံး၏ ပေါင်းခြင်းတန်ဖိုး(sum of all values in a list)ကို ရှာပေးသည်။

```
li = [10,1,9,2,8,3,7,4,6,5]
print(min(li), "is minimum value of given list.")
```

```
1 minimum value of given list.
```

```
print(max(li), "is maximum value of given list.")
```

```
10 maximum value of given list.
```

```
print(sum(li), "is sum of all values in a given list.")
```

```
55 is sum of all values in a given list.
```

(၄၂) List နှစ်ခု၏ intersection ကို ရှာပါ။

List နှစ်ခုကို intersection လုပ်၍ မရပါ။ List နှစ်ခုကို set() အဖြစ် ပြောင်းလဲ၍ intersection လုပ်သည်။ Ampersand(&) ကို သုံးသည်။

```
li1 = [1,2,3]
li2 = [2,3,4]
set(li1) & set(li2)
```

```
{2, 3}
```

(၄၃) Set တစ်ခုမှ တခြား set တစ်ခုကို နုတ်(difference)ပါ။

List တစ်ခုနှင့် တခြား list တစ်ခုကို difference ပြုလုပ်၍ မရပါ။ ထို့ကြောင့် set အဖြစ်သို့ ပြောင်းသည်။ (we can subtract sets.)

```
li1 = [1,2,3]
li2 = [2,3,4]
print(set(li1) - set(li2))
print(set(li2) - set(li1))
```

```
{1}
{4}
```

(၄၄) Sublist (list of lists) အတွင်းမှ item များအားလုံးကို ထုတ်၍ Sublist မရှိအောင် လုပ်ပါ။

Sublist (list of lists) အတွင်းမှ item များအားလုံးကို ထုတ်၍ sublist မရှိအောင် လုပ်ပါ။ ထိုသို့ ပြုလုပ်ခြင်းကို flatten လုပ်သည်ဟုခေါ်သည်။

```
li = [[1,2,3],[4,5,6]]
[i for x in li for i in x]
```

```
[1, 2, 3, 4, 5, 6]
```

(၄၅) List နှစ်ခုကို ပေါင်း၍ dictionary ဖြစ်အောင်လုပ်ပါ။

zip() ကို သုံး၍ dictionary ဖြစ်အောင် ပြောင်းသည်။ List တစ်ခုမှ item များသည် key များ ဖြစ်ကြပြီး ကျန် list တစ်ခုမှ item များသည် value ဖြစ်လိမ့်မည်။

```
name = ['Snowball', 'Chewy', 'Bubbles', 'Gruff']
animal = ['Cat', 'Dog', 'Fish', 'Goat']
dict(zip(name, animal))
```

```
{'Snowball': 'Cat', 'Chewy': 'Dog', 'Bubbles': 'Fish', 'Gruff': 'Goat'}
```

-End-