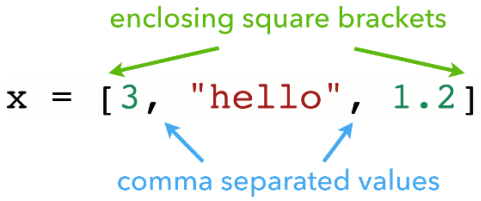
Chapter-5 Python Lists

Contents

Chapter-5 Python Lists

List သည် data structure အမျိုးအစား တစ်ခု ဖြစ်သည်။ Java, C, C++ စသည့် တခြားသော programming language များတွင် list များကို array များဟု သတ်မှတ်ကြသည်။ Python list သည် တခြားသော programming language တွင်ရှိသည် traditional array ထက်ပို အစွမ်းထက်(powerful) သည်။ (A Python list is more than just a list.)



List ထဲတွင် ပါဝင်သည့် အရာများကို item သို့မဟုတ် element ဟုခေါ်သည်။ List ထဲတွင် ပါဝင်သည့် အရာများကို လေးထောင့်ကွင်း [ ] ထဲတွင် ကော်မာခံ၍ ထည့်ရေးသည်။ Python list ၏ အဓိက အချက်များမှာ

- List ထဲတွင် ပါဝင်သည့် item သို့မဟုတ် element များ၏ ရှေ့နောက် အစီအစဉ်(ordered)သည် အရေးကြီးသည့် အချက်ဖြစ်သည်။
- List ထဲရှိ element များကို index နံပါတ်များသုံး၍ access လုပ်နိုင်သည်။ လှမ်းဖတ်နိုင်သည်။ ကော်ပီကူးနိုင်သည်။ ပြောင်းလဲနိုင်သည်။
- List များ ထဲတွင် တခြား list များ ထပ်ခါ ထပ်ခါ ထည့်နိုင်(arbitrarily nestable)သည်။ List များ ထဲတွင် တခြား list များ sublist အဖြစ် ရှိနေနိုင်သည်။
- List များ၏ အရွယ်အစား ပြောင်းလဲနိုင်သည်။ (variable sizes)
- List တစ်ခုသည် value များပါဝင်သည့် sequence တစ်ခုဖြစ်သည်။ (A list is a sequence of values.)
- List တွင် index နံပါတ်သည် 0 မှ စသည်။ (zero indexing)
- Python list ကို အမျိုးအစားမတူသည့် item မျိုးစုံကို ထည့်နိုင်သည့် container တစ်ခု အဖြစ် နားလည် မှတ်ယူနိုင်သည်။ List ထဲမှ element များကို ပြောင်းနိုင် ပြင်နိုင်သည်။ (Mutable ဖြစ်သည်။)
- List ထဲတွင် ရှိနေသည့် item များအားလုံးသည် Python object များ ဖြစ်ကြသည်။ List သည်လည်း Python object ဖြစ်သည်။
- List ထဲတွင် arbitrary အမျိုးအစား(type)များ ပါဝင်နိုင်သည်။

5.1- List တစ်ခုတည်ဆောက်ခြင်း(Declare a list)

နည်းအမျိုးမျိုးဖြင့် List များကို တည်ဆောက်နိုင်သည်။ အောက်တွင် integer များ ပါသည် list တစ်ခုကို တည်ဆောက်သည်။ လေးထောင့်ကွင်း[ ] ထဲတွင် item များကို comma ခံ၍ ထည့်ရေး၍ list တစ်ခု

တည်ဆောက် ခြင်းသည် အလွယ်ကူဆုံးနည်း ဖြစ်သည်။ Element တစ်ခုမှ မပါသည့် list ကို empty list ဟု ခေါ်သည်။ အောက်တွင် ဘာမှ မပါသည့် empty list တစ်ခု တည်ဆောက်ပြထားသည်။

```
my_list = []
my_list
```

```
[]
```

အောက်တွင် ဥပမာများအဖြစ် နံပါတ်များ(numbers)ပါသည့် list တစ်ခု(list containing numbers)ကို တည်ဆောက်ပြထားသည်။ လေးထောင့်ကွင်း [ ] ထဲတွင် item များကို comma ခံ၍ ထည့်ရေးသည်။

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

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

```
type(L)     # type()function ဖြင့် list L ၏ data type အမျိုးအစားကို စစ်သည်။
```

```
list
```

```
type(L[0])  # list L ထဲတွင် ရှိနေသည့် item ၏ data type အမျိုးအစားကို စစ်နိုင်သည်။
```

```
int
```

range() function ကို သုံး၍ 0 မှ 9 အထိပါသည် list တစ်ခုကို တည်ဆောက်သည်။ range(10) သည် digit 0 မှ 9 အထိ ကိန်း (၁၀) လုံး ထုတ်ပေးသည်။

```
L = list(range(10))
L
```

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

အပေါ်က list ထဲရှိ integer ကို string type ပြောင်းသည်။ List “L” ထဲမှ item တစ်ခုစီကို **str()** ဖြစ် string type ပြောင်းသည်။

```
L_str = [str(c) for c in L]
L_str
```

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

```
type(L_str[0])   # L ထဲတွင် ရှိနေသည့် item ၏ data type အမျိုးအစားကို စစ်သည်။
```

```
str
```

Python သည် dynamic typing ဖြစ်သောကြောင့် အမျိုးအစားမတူသည့် data type များကို list တစ်ထဲတွင် ထည့်ထားနိုင်သည်။ Python data type ထဲမှ မည်သည့် အမျိုးအစားမဆို ဖြစ်နိုင်သည်။ အောက်တွင် bool, str, float, int စသည်တို့ ပါဝင်သည့် Python list တစ်ခုကို ဥပမာအဖြစ် ဖော်ပြ ထားသည်။

```
L3 = [True, '2', 3.0, 4]
#  L3 ထဲတွင် ရှိနေသည့် item ၏ တစ်ခုချင်းစီ၏ အမျိုးအစားကို စစ်သည်။
[type(item) for item in L3]
```

```
[bool, str, float, int]
```

5.1.1 Nested sublists

List တစ်ခုထဲတွင် တခြားသော list သည် sublist အဖြစ် ရှိနေနိုင်သည်။

```
L4 = ['Bob', 40.0, ['dev', 'mgr']]
#  L4 ထဲတွင် ရှိနေသည့် item ၏ တစ်ခုချင်းစီအမျိုးအစားကို စစ်သည်။
[type (item) for item in L4]
```

```
[str, float, list]
```

'Python' ဆိုသည့် string တစ်ခုသည် iterable's item ဖြစ်သောကြောင့် character များပါသည့် list တစ်ခု တည်ဆောက်နိုင်သည်။ String တစ်ခုသည် character အစဉ်အတန်း တစ်ခုဖြစ်သည်။ (A string is a sequence of characters.)

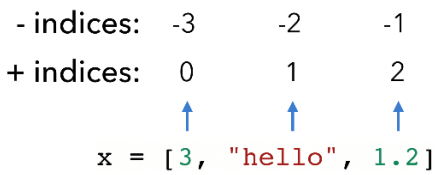
```
L5 = list('Python')
L5
```

```
['P', 'y', 't', 'h', 'o', 'n']
```

L5 ထဲတွင် ရှိနေသည့် item ၏ တစ်ခုချင်းစီအမျိုးအစားကို စစ်သည်။

```
[type (item) for item in L5]
```

```
[str, str, str, str, str, str]
```



5.1.2 List ၏ index နံပါတ်များ

```
L[i]     # Index
L[i][j]  # index of index
L[i:j]   # slice
len(L)   # length
```

List ၏ index နံပါတ်များ အလုပ်လုပ်ပုံသည် string ၏ index နံပါတ်များ အလုပ်လုပ်ပုံမှ အတူတူပင် ဖြစ်သည်။ (List indices work the same way as string indices)

- မည်သည့် integer ကို မဆို အသုံးပြုနိုင်သည်။ (Any integer expression can be used as an index.)

- List သို့မဟုတ် string ထဲတွင် မရှိသည့် index နံပါတ်ကို ရည်ညွှန်းလျင်၊ ဖတ်လျှင် ၊ ရေးလျှင် IndexError ပေးလိမ့်မည်။(If you try to read or write an element that does not exist, you get an IndexError.)
- Index နံပါတ်တွင် အနုတ်လက္ခဏာ ပါရှိနေပါက အနောက်မှ စ၍ ပြန်ရေတွက်သည်ဟု မှတ်ပါ။(If an index has a negative value, it counts backward from the end of the list.)။ နောက်ဆုံး item သည် [-1] ဖြစ်သည်။

5.1.3 Mapping

List ကို index နံပါတ်များ နှင့် element များ၏ ဆက်သွယ်ချက် အဖြစ်လည်း မှတ်ယူနိုင်သည်။ (a list as a relationship between indices and elements.)။ ထိုဆက်သွယ်ချက် mapping ဟုခေါ်သည်။ List တွင် **in** operator ကို သုံးနိုင်သည်။ List ထဲတွင် သိလိုသည့် item ပါဝင်နေလျှင် True ပြန်ပေးသည်။ မပါဝင်လျှင် False ပြန်ပေးသည်။

```
cheeses = ['Cheddar', 'Edam', 'Gouda']
print('Edam' in cheeses)
print('Brie' in cheeses)
```

```
True
False
```

5.1.4 Traversing a List

List အတွင်း element များကို တစ်လုံးချင်းစီ ထုတ်ယူရန်(traverse လုပ်ရန်) for loop ကို သုံးသည်။ String များအတွက်လည်း အတူတူပင် ဖြစ်သည်။

```
for cheese in cheeses:
    print(cheese)
```

```
Cheddar
Edam
Gouda
```

5.1.5 List Operations(+ operator)

+ operator ဖြင့် list နှစ်ခုကို ဆက်နိုင်သည်။ (concatenates lists)

```
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b        #+ operator ဖြင့် list နှစ်ခုကို ဆက်သည်။
print (c)
```

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

```
d = str([1, 2]) + "34"     # Same as "[1, 2]" + "34"
print(d)
```

```
e = [1, 2] + list("34")    # Same as [1, 2] + ["3", "4"]
print(e)
```

```
[1, 2]34
[1, 2, '3', '4']
```

5.1.6 Lists and Strings

String သည် အက္ခရာစဉ်အတန်း(sequence of characters)ဖြစ်သည်။ List သည် value စဉ်အတန်း(sequence of values)ဖြစ်သည်။

```
s = 'spam'
t = list(s)
print (t)
```

```
['s', 'p', 'a', 'm']
```

String တစ်ခုကို စာလုံးများ(word)အဖြစ် ပိုင်းခြားလိုလျှင် **.split()** method ကို အသုံးပြုသည်။

```
s = 'pining for the fjords'
t = s.split()                    # စာလုံးများ(word)အဖြစ် ပိုင်းခြားရန်
print (t)
```

```
['pining', 'for', 'the', 'fjords']
```

အောက်က ဥပမာတွင် hyphen ကို delimiter အဖြစ်သုံး၍ ပိုင်းခြားသည်။

```
s = 'spam-spam-spam'
delimiter = '-'          # hyphen ကို delimiter အဖြစ် အသုံးပြုသည်။
s.split(delimiter)
```

```
['spam', 'spam', 'spam']
```

.split() လုပ်ထားသည့် item များကို ပြန်ပေါင်းစပ်(Join)သည်။ (Join is a string method.)

```
t = ['pining', 'for', 'the', 'fjords']
delimiter = ' '
# space delimiter အဖြစ် သုံးပြီး list တစ်ခု အတွင်းရှိ element များကို ဆက်သည်။
delimiter.join(t)
```

```
'pining for the fjords'
```

5.1.7 List Operations(* operator)

***** operator ဖြင့် list အတွင်းရှိ element များကို အဆများစွာ ပွားနိုင်သည်။(repeats a list a given number of times)။

```
f = ['Ni!'] * 4          # repeats ['Ni!'] four times
print(f)
```

```
['Ni!', 'Ni!', 'Ni!', 'Ni!']
```

```
g = [1, 2, 3] * 3        #repeats the list [1, 2, 3] three times.
print(g)
```

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

5.1.8 List Iteration and Comprehensions

For loop သုံး၍ iteration လုပ်သည်။ Loop မှ ထွက်သည့် item များ အလျားလိုက်ပေါ် စေရန် **end=' '** ကို အသုံးပြုထားသည်။

```
for x in [1, 2, 3]:
    print(x, end=' ')          # Iteration
```

```
1 2 3
```

အောက်တွင် 'SPAM' ထဲမှ အက္ခရာ တစ်လုံးချင်းစီကို (၄) ဆပွားပြီး list တစ်ခု ဆောက်သည်။

```
res = [c * 4 for c in 'SPAM'] # List comprehensions
res
```

```
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

List ထဲမှာ item တစ်ခုချင်းစီကို (၄)ခု ဖြစ်အောင် မြှောက်ပြီးမှ **.append()** လုပ်၍ empty list ထဲသို့ ထည့်သည်။

```
res = []
for c in 'SPAM':           # List comprehension equivalent
    res.append(c * 4)

res
```

```
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

Slice operator ကို အသုံးပြု၍ list များကို slice လုပ်နိုင်သည်။

```
t = ['a', 'b', 'c', 'd', 'e', 'f']
t[1:3]                      # 1 မှ 3 မတိုင်ခင်အထိ ( 3 မပါ)
```

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

စမည့် index နံပါတ်ကို မထည့်ပေးလျှင် 0 မှ စရမည်။ ဥပမာ - t[:4]

ဆုံးမည့် index နံပါတ်ကို မထည့် ပေးလျှင် နောက်ဆုံးအထိ ယူရမည်။ ဥပမာ - t[3:]

စမည့် index နံပါတ် နှင့် ဆုံးမည့် index နံပါတ်ကို မထည့် ပေးလျှင် အစမှ အဆုံးအထိ ဖြစ်သည်။

```
t[:4]                # စမည့် index နံပါတ်ကို မထည့်ပေးလျှင် 0 မှ စရမည်။
```

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

```
t[3:]                 # ဆုံးမည့် index နံပါတ်ကို မထည့်ပေးလျှင် နောက်ဆုံးအထိယူရမည်။
```

```
['d', 'e', 'f']
```

```
t[:] #စမည့် index နံပါတ်နှင့်ဆုံးမည့် index နံပါတ်ကို မထည့်ပေးလျှင် အစမှအဆုံးအထိ ဖြစ်သည်။
```

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

5.2- List တစ်ခု အတွင်း သို့ item များထည့်ခြင်း

5.2.1 List ၏ နောက်ဆုံးနေရာ၌ item များထည့်ခြင်း

Itemတစ်ခု(single item)ကို list ၏ နောက်ဆုံးနေရာတွင် ထည့်သွင်းလိုသည့်အခါ **.append()** ကို သုံးသည်။

```
programming_languages = ["Python","Java","R","Ruby and Rails"]
programming_languages.append("C++")     # C++" ကို ထည့်သည်။
programming_languages
```

```
['Python', 'Java', 'R', 'Ruby and Rails', 'C++']
```

5.2.2 List ၏ သတ်မှတ်ထားသည့်နေရာတွင်item များထည့်ခြင်း

Add item at particular place of a list / insert items in a list

List ၏ သတ်မှတ်ထားသောနေရာတွင် item ကို ထည့်ပေး(insert လုပ်ပေး)သည်။ ထည့်ပေးစေလိုသည့် နေရာ၏ index နံပါတ် ထည့်ပေးရသည်။

```
programming_languages.insert(2,"C#") # ထည့်ပေးစေလိုသည့်နေရာ၏ index နံပါတ်= 2
programming_languages
```

```
['Python', 'Java', 'C#', 'R', 'Ruby and Rails', 'C++']
```

5.3 List ထဲမှာ item များကို ဖျက်ခြင်း(Delete items from the list)

5.3.1 List ထဲမှာနောက်ဆုံး item ကို ဖျက်ခြင်း(delete item at the end of a list)

ထည့်ပေးလိုက်သည့် index နံပါတ်နှင့် ကိုက်ညီသည့် item ကိုဖယ်ထုတ်ပေးသည်။ အကယ်၍ ဖယ်ထုတ် ရမည့် item နံပါတ်ကို မထည့်ပေးပါက list ထဲမှာ နောက်ဆုံး(last) item ကို ဖယ်ထုတ်ပေးသည်။

```
programming_languages.pop()
```

```
'C++'
```

```
programming_languages
```

```
['Python', 'Java', 'C#', 'R', 'Ruby and Rails']
```

```
programming_languages.pop(3)
```

```
'R'
```

```
programming_languages
```

```
['Python', 'Java', 'C#', 'Ruby and Rails']
```

5.3.2 List ၏ သတ်မှတ်ထားသည့်နေရာ item အား ဖျက်ခြင်း

Delete item from particular position of a list

ထည့်ပေးလိုက်သည့် index နံပါတ်နှင့် ကိုက်ညီသည့် item ကို ဖျက်ပေးသည်။ အောက်က ဥပမာတွင် index နံပါတ်(2) နေရာမှ element ကို ဖျက်သည်။

```
del(programming_languages[2])
programming_languages
```

```
['Python', 'Java', 'Ruby and Rails']
```

5.3.3 List ၏ သတ်မှတ်ထားသည့် value အား ဖျက်ခြင်း

Delete item of particular value in a list

ထည့်ပေးလိုက်သည့် value နှင့် ကိုက်ညီသည့် item ကို ဖျက်ပေးသည်။ ထည့်ပေးလိုက်သည့် value နှင့် ကိုက်ညီသည့် item မရှိပါက ValueError ပေးသည်။

```
programming_languages.remove("Ruby and Rails")
programming_languages
```

```
['Python', 'Java']
```

Deleting Elements (.pop() , del , .remove())

```
t = ['a', 'b', 'c']
x = t.pop(1)        #.pop() ဖြင့် list ရှိ index နံပါတ်(1)မှ element ကို ဖယ်ထုတ်သည်။
print(t)
print(x)
```

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

```
t = ['a', 'b', 'c']
del t[1]          # del ဖြင့် list ရှိ index နံပါတ်(1)မှ element ကို delete လုပ်သည်။
print(t)
```

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

```
t = ['a', 'b', 'c']
t.remove('b')         # element 'b' ကို remove လုပ်သည်။
```

```
print(t)
```

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

```
t = ['a', 'b', 'c', 'd', 'e', 'f']
del t[1:5]     # Index နံပါတ် (၁) မှ (၅) မတိုင်ခင်အထိ ဖျက်မည်။ (Index နံပါတ် (၅) မပါ။)
print(t)
```

```
['a', 'f']
```

5.4- List အတွင်း ရှိ item များအား Update လုပ်ခြင်း (Update item)

Index နံပါတ်ပေး၍ list ထဲမှာ item များကို update လုပ်နိုင်သည်။

```
programming_languages = ['Python', 'Fortran']
programming_languages[1] = 'C++'
programming_languages
```

```
['Python', 'C++']
```

5.4.1- Get a particular item

Index နံပါတ် ဖြင့် access လုပ်နိုင်သည်။

```
programming_languages[1]  # from start
```

```
'C++'
```

```
programming_languages[-1]  # from end
```

```
'C++'
```

List ၏ နောက်ဆုံးနေရာမှ ထပ်ထည့်လိုသည့်အခါ .append() ကို သုံးသည်။

```
programming_languages.append("Java")
programming_languages.append("R")
programming_languages.append("Ruby and Rails")
programming_languages
```

```
['Python', 'C++', 'Java', 'R', 'Ruby and Rails']
```

5.4.2- ရှေ့နောက် အစီအစဉ်ပြောင်းပြန်လှန်ခြင်း (Reverse a list)

.reverse() ကို သုံး၍ list အတွင်းရှိ element များ၏ အစဉ်ကို ပြောင်းပြန်လှန်သည်။

```
programming_languages.reverse()
programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

5.4.3- Finding no.of items in a list/ Length of a list

.len() ဖြင့် list အတွင်းရှိ element များ၏ အရေအတွက်ကို ရှာနိုင်သည်။ element များ၏ အရေအတွက်ကို list ၏ length ဟု သတ်မှတ်သည်။

```
len(programming_languages)
```

```
5
```

5.5- ရှေ့နောက်စဉ်ခြင်း (Sorting a list)

5.5.1 Permenantly sort in ascending order

Integers list ဖြစ်လျှင် ကိန်းတန်ဖိုးအတိုင်း စီပေးသည်။ String list ဖြစ်လျှင် အက္ခရာစဉ်တိုင်း စီပေးသည်။ (reverse = False) ဆိုလျှင် ငယ်ရာမှ ကြီးရာသို့ နည်းရာမှ များရာသို့(ascending) စီသည်။ (reverse = True) ဆိုလျှင် ပြောင်းပြန်(descending) စီသည်။

```
numbers = [1, 3, 4, 2]
numbers.sort()            # Sorting list of Integers in ascending
print(numbers)
```

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

```
strs = ["geeks", "code", "ide", "practice"]
strs.sort()                    # Sorting list of Integers in
ascending
print(strs)
```

```
['code', 'geeks', 'ide', 'practice']
```

```
numbers = [1, 3, 4, 2]
numbers.sort(reverse = True) # Sorting list of Integers in
descending
print(numbers)
```

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

5.5.2 Permenantly sort in descending order

```
programming_languages.sort(reverse = True)
programming_languages
```

```
['Ruby and Rails', 'R', 'Python', 'Java', 'C++']
```

5.5.3 Temporary sort (sorted())

```
programming_languages = ['Ruby and Rails','R','Java','C++','Python']
sorted(programming_languages)
```

```
['C++', 'Java', 'Python', 'R', 'Ruby and Rails']
```

မူလ(original) list ကို ဘာမှ မပြောင်းလဲစေပါ။ အောက်တွင် မူလ(original) list ကို ပြန်ကြည့်သည်။

```
programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

5.6 Accessing list items one by one through looping

For loop ဖြင့် list အတွင်းရှိ item များကို တစ်ခု ချင်းစီ access လုပ်နိုင်သည်။

```
for language in programming_languages:
    print(language)
```

```
Ruby and Rails
R
Java
C++
Python
```

```
programming_languages[1:4]      # [1:4] သည် items 1 မှ 3 အထိ access လုပ်သည်။
```

```
['R', 'Java', 'C++']
```

```
programming_languages[1:]    # [1:] သည် items 1 မှ အဆုံး အထိ(end) ဖြစ်သည်။
```

```
['R', 'Java', 'C++', 'Python']
```

```
programming_languages[:4] # [:4] သည် အစဆုံး items မှ 3 အထိ access လုပ်သည်။
```

```
['Ruby and Rails', 'R', 'Java', 'C++']
```

```
programming_languages[-1:] #အဆုံးမှ အစ အထိ ဖြစ်သည်။ - ver သည် အဆုံးမှာ စရေတွက်သည်။
```

```
['Python']
```

```
programming_languages[:-4]              # အဆုံးမှ စရေတွက်လျှင် 4th item
```

```
['Ruby and Rails']
```

```
programming_languages[:]                # [:] အစမှ အဆုံးအထိ။ အားလုံးကို ဆိုလိုသည်။
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

```
programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

5.7 ကော်ပီကူးခြင်း (Copy a list)

5.7.1 List တစ်ခုလုံးကို ကော်ပီကူးခြင်း (Copy whole list)

```
new_programming_languages = programming_languages[:]
new_programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

```
programming_languages
```

```
 ['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

5.7.2 Item များ တစ်ခုချင်းစီကို ကော်ပီကူးခြင်း(Copy list items one by one)

ပထမ ဦးစွာ new_programming_languages ဆိုသည့် empty list တစ်ခု တည်ဆောက်သည်။ ထို့နောက် programming_languages ဆိုသည့် list ထဲက element များကို .pop()ဖြင့် ဖယ်ထုတ်ပြီး empty list ထဲသို့ .append()ဖြင့် ထည့်သည်။

```
programming_languages = ['Ruby and Rails', 'R', 'Java',
                         'C++', 'Python']
new_programming_languages = []

new_programming_languages.append(programming_languages.pop(0))

print(new_programming_languages)
print(programming_languages)
```

```
['Ruby and Rails']
['R', 'Java', 'C++', 'Python']
```

5.8 if Statement With a List

5.8.1 List ထဲတွင် မိမိသိလိုသည့် value တစ်ခု ရှိ မရှိ စစ်ဆေးခြင်း

```
if 'R' in programming_languages:
    print('exist')
```

```
exist
```

5.8.2 List ထဲတွင် မိမိသိလိုသည့် value တစ်ခု မရှိခြင်းကို စစ်ဆေးခြင်း (Checking whether a value not in a list.)

not in keyword ကိုသုံး၍ programming_languages list ထဲတွင် မိမိသိလိုသည့် language ရှိမရှိ စစ်သည်။

```
if 'R' not in programming_languages:
    print('not exist')
else:
    print('exist')
```

```
exist
```

5.8.3 List ထဲတွင် ဘာမှ ရှိမရှိ စစ်ဆေးခြင်း (Checking whether list is empty or not)

List တစ်ခုထဲတွင် element သို့မဟုတ် item များ ရှိ မရှိကို bool() ဖြင့် စစ်ဆေးနိုင်သည်။ List ထဲတွင် item တစ်ခုခု ရှိလျှင် True ပြန်ပေးလိမ့်မည်။ မရှိလျှင် `False` ပြန်ပေးလိမ့်မည်။

```
bool(programming_languages)
```

```
False
```

```
programming_languages = [ ]
if programming_languages:
    print('There exist item or items')
else:
    print('List is empty')
```

```
List is empty
```

OR

```
programming_languages = [ ]
if not programming_languages:
    print('List is empty')
else:
    print('There exist item or items')
```

```
List is empty
```

5.9 Loops with List

5.9.1 For loop နှင့် pop item ကို သုံး၍ List ထဲမှ item များ ကို တစ်ခုချင်း ဖယ်ထုတ်ခြင်း

(pop item one by one from one list to another (using for loop))

```
programming_languages = ['Ruby and Rails',
'R','Java','C++','Python']
new_programming_languages = []

for i in range(0,len(programming_languages)):
    new_programming_languages.append(programming_languages.pop())

programming_languages
```

```
[]
```

```
new_programming_languages
```

```
['Python', 'C++', 'Java', 'R', 'Ruby and Rails']
```

5.9.2 List တစ်ခုထဲမှ item များကို ဖယ်ထုတ်ပြီး တခြား List ထဲသို့ ထည့်ခြင်း

While loop ဖြင့် append item နှင့် pop item ကို သုံး၍ list တစ်ခုထဲမှ item များကို ဖယ်ထုတ်ပြီး တခြား list ထဲသို့ ထည့်ခြင်း (pop item one by one from one list to another (using while loop))

```
while new_programming_languages:
```

```
    programming_languages.append(new_programming_languages.pop())
new_programming_languages
```

```
[]
```

```
programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'C++', 'Python']
```

5.9.3 Value ဖြင့် list ထဲမှ instance များကို ဖယ်ထုတ်ခြင်း (Removing all instances of particular value from a list)

If statement ကို သုံး၍ list တစ်ခုထဲမှ instance(item) တစ်ခုကို ဖယ်ထုတ်သည်။ အောက်က ဥပမာတွင် programming_languages ထဲ၌ 'C++' ရှိ မရှိကို အရင်စစ်သည်။ ရှိလျှင် ဖယ်ထုတ်သည်။

```
programming_languages = ['Ruby and Rails', 'R', 'Java', 'C++',
                         'Python','C++','R','C++']

if 'C++' in programming_languages:
    programming_languages.remove('C++')
programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'Python', 'C++', 'R', 'C++']
```

For loop ကို သုံး၍ ရှိသမျှ instance(item) အားလုံးကို ဖယ်ထုတ်သည်။ ဖယ်ထုတ်ရမည့် instance မကုန် မခြင်း for loop သည် ရပ်လိမ့်မည် မဟုတ်ပေ။

```
programming_languages = ['Ruby and Rails', 'R', 'Java', 'C++',
                         'Python','C++','R','C++']
while 'C++' in programming_languages:
    programming_languages.remove('C++')

programming_languages
```

```
['Ruby and Rails', 'R', 'Java', 'Python', 'R']
```

5.10 Set Operation On a List

List ကို set() ဖြင့် set အဖြစ်သို့ ပြောင်းသည်။ List ထဲတွင် element တစ်ခုသည် အကြိမ်ကြိမ် အခါခါ ပါဝင်နေပါက တစ်ခုသာယူရန် ဖြစ်သည်။ အောက်က ဥပမာတွင် 'C++' သည် နှစ်ခါပါဝင် နေသော်လည်း output တွင် တစ်ခုသာ ပါဝင်သည်။

```
programming_languages = ['Ruby and Rails', 'R', 'Java', 'Python',
                         'C++', 'R', 'C++']
print(set(programming_languages))
```

```
{'Ruby and Rails', 'R', 'C++', 'Java', 'Python'}
```

```
type(set(programming_languages))      # Set ဖြစ် မဖြစ် စစ်သည်။
```

set

```
programming_languages=['Ruby and Rails','R','Java','Python',
                       'C++','R', 'C++']
set_programming_languages = set(programming_languages)
print(set_programming_languages)
```

```
{'Ruby and Rails', 'R', 'C++', 'Java', 'Python'}
```

5.11 List ကို Tuple အဖြစ်သို့ ပြောင်းခြင်း(Convert List into a Tuple)

tuple() ကို အသုံးပြု၍ list ကို tuple အဖြစ်သို့ ပြောင်းသည်။

```
programming_languages = ['Ruby and Rails', 'R', 'Java',
                         'Python', 'C++', 'R', 'C++']
tuple_programming_languages = tuple(programming_languages)
print(tuple_programming_languages)
```

```
('Ruby and Rails', 'R', 'Java', 'Python', 'C++', 'R', 'C++')
```

```
type(tuple_programming_languages)  # tuple ဖြစ် မဖြစ် စစ်သည်။
```

```
tuple
```

5.12 - List နှင့် Nested For Loop ကို တွဲသုံးခြင်း

`name` ဆိုသည့် empty list တစ်ခု ပြုလုပ်သည်။ List ထဲမှ `first_names` နှင့် `last_names` နာမည်များကို nested for loop ဖြင့် တွဲ၍ ထို empty list ထဲသို့ ဖြင့် .append() လုပ်ပြီး ထည့်သည်။

```
first_names = ['Maung',' U','Ko']
last_names = ['Ba Kaung',' Myint Soe','Sai']
name = []
for f_name in first_names:
    for l_name in last_names:
        name.append(f_name+" "+l_name \n)
name
```

```
['Maung Ba Kaung',
 'Maung  Myint Soe',
 'Maung Sai',
 'U Ba Kaung',
 'U  Myint Soe',
 'U Sai',
 'Ko Ba Kaung',
 'Ko  Myint Soe',
 'Ko Sai']
```

5.13 List Methods in Python

Python list method များကို ရှင်းပြထားသည်။

```
my_string = ["first"]
```

```
print(dir(my_string))
```

```
['__add__',      '__class__',      '__contains__',      '__delattr__',
'__delitem__',   '__dir__',    '__doc__',    '__eq__',    '__format__',
'__ge__',  '__getattribute__',  '__getitem__',  '__gt__',  '__hash__',
'__iadd__',  '__imul__',  '__init__',  '__init_subclass__',  '__iter__',
'__le__',   '__len__',   '__lt__',   '__mul__',   '__ne__',   '__new__',
'__reduce__',     '__reduce_ex__',     '__repr__',     '__reversed__',
'__rmul__',  '__setattr__',  '__setitem__',  '__sizeof__',  '__str__',
'__subclasshook__',  'append',  'clear',  'copy',  'count',  'extend',
'index',  'insert',  'pop',  'remove',  'reverse',  'sort']
```

help(list)ကို သုံး၍ list နှင့် သက်ဆိုင်သည့်အချက်အလက်များကို ဖတ်နိုင်သည်။(The **help()** method calls the built-in Pyathon help system)။ dir(list) အသုံးများသည့် built-in method များ

- `list.append(item)`
- `list.extend(iterable)`
- `list.insert(index, item)`
- `list.remove(item)`
- `list.pop([index])`
- `list.clear()`
- `list.index(item [, start [, end]])`
- `list.count(item)`
- `list.sort(key=None, reverse=False)`
- `list.reverse()`
- `list.copy()`

5.13.1 list.append(item)

list.append(item) method ကို အသုံးပြု၍ itemတစ်ခု(single item)ကို list ၏ နောက်ဆုံးနေရာတွင် ထည့်သွင်း နိုင်သည်။ ထိုသို့ထည့်သွင်းလိုက်ခြင်းဖြင့် list အသစ်တစ်ခု ဖြစ်ပေါ်မလာပါ။ ဥပမာ-

```
things = ["first"]
things.append("another thing")
things
```

```
['first', 'another thing']
```

5.13.2 list.extend(iterable)

list.extend(iterable) သည် iterable data type argument တစ်ခုကို လက်ခံယူနိုင်သည်။ Iterable data type argument သည် တခြားlist တစ်ခုလည်း ဖြစ်နိုင်သည်။ ထို iterable data type argument ကို မူလ list ၏ အဆုံးနေရာတွင် ထည့်သွင်း(append လုပ်)သည်။ List တစ်ခုကို တခြား list တစ်ခုနှင့် extend လုပ်သည်ဟု ခေါ်သည်။ (extend a list with another list)။ List နှစ်ခုကို ပေါင်းလိုသည့်အခါ သို့မဟုတ် list တစ်ခုကို တခြား list တစ်ခုထဲသို့ ထည့်လိုသည့်အခါအတွင် `.extend()`ကို အသုံးပြုသည်။

```
things = ['first', 'another thing']
others = ["third", "fourth"]
things.extend(others)
```

```
things
```

```
['first', 'another thing', 'third', 'fourth']
```

list.append(item) နှင့် list.extend(iterable)တို့ မတူညီပုံကို ဖော်ပြထားသည်။

```
things = ["first"]
things.append("another thing")
things
```

```
['first', 'another thing']
```

List တစ်ခု extend method ကို သုံး၍ string တစ်ခု ထည့်ပေးလိုက်သည့်အခါ(passed a string to the extend method) ရလာမည် ရလဒ်ကို အောက်တွင် လေ့လာနိုင်သည်။ String သည် iterable data type ဖြစ်သောကြောင့် character တစ်ခုချင်းသည် element တစ်ခု ဖြစ် ဝင်ရောက်သည်။

```
things.extend("another thing")
print(things)
```

```
['first', 'another thing', 'a', 'n', 'o', 't', 'h', 'e', 'r', ' ',
't', 'h', 'i', 'n', 'g']
```

String ကို extend method ဖြင့် သုံးသည့်အခါ string ၏ character များသည် iterable ဖြစ်သောကြောင့် character အားလုံးနှင့် တကွ space ကိုပါ ထည့်သွင်း(append လုပ်) လိမ့်မည်။

5.13.3 list.insert(index, item)

list.insert(index, item) method သည် list ၏ သတ်မှတ်ထားသော နေရာတွင် item ကို ထည့်ပေး(insert လုပ်ပေး)သည်။ ထည့်ပေး စေလိုသည့်နေရာ(index)နှင့် item ဟူ၍ argument နှစ်ခု ပေးရသည်။ Index နံပါတ်သည် ထည့်ပေးစေလိုသည့် နေရာ ဖြစ်သည်။ Argument နှစ်ခုလုံး ထည့်ပေး ရမည်။

```
things = ["first","second", "fourth"]
things
```

```
['first', 'second', 'fourth']
```

```
things.insert(2, "third")  # index နံပါတ် 2 နေရာတွင် "third" ကိုထည့်မည်။
things
```

```
['first', 'second', 'third', 'fourth']
```

5.13.4 list.remove(item)

list.remove(item) method သည် ထည့်ပေးလိုက်သည့်တန်ဖိုး(passed in value) တူသည့်နေရာ(match ဖြစ်သည့်နေရာ)မှ ပထမဆုံးတွေ့သည်(first) item ကို list မှ ဖယ်ရှားပေးသည်။

```
my_list = ['first', 'second', 'first', 'second']
```

```
['first', 'second']
```

```
my_list.remove("second")  # second" ဆိုသည့် item ကို ဖယ်ရှားရန်
my_list
```

```
['first', 'first', 'second']
```

ပထမဆုံးတွေ့သည် "second" ဆိုသည့် item ကို ဖယ်ရှားပစ်သည်။

5.13.5 list.pop([index])

list.pop([index]) method သည် list ထဲမှ ပေးထားသည့် index နံပါတ်နှင့် ကိုက်ညီသည့် item ကို ဖယ်ထုတ်ပြီး return အဖြစ် ပြန်ပေးသည်။ မဖြစ်မနေ index နံပါတ်ထည့်ပေးရန် မလိုပါ။ (The index is an optional argument)။ အကယ်၍ ဖယ်ထုတ်ရမည့် item နံပါတ်ကို မထည့်ပေးပါက list ထဲမှာ နောက်ဆုံး (last) item ကို ဖယ်ထုတ်ပေးသည်။

```
things = ['first', 'second', 'third']
things
```

```
['first', 'second', 'third']
```

```
second_item = things.pop(1) # index နံပါတ် 1 နေရာမှ item ကို .pop() လုပ်သည်။
second_item
```

```
'second'
```

```
things
```

```
['first', 'third']
```

5.13.6 list.clear()

list.clear() method သည် list ထဲမှာ item များအားလုံးကို ဖယ်ရှားရှင်းလင်း(clear)ပေးသည်။ **del** a[:] နှင့် တူညီသည်။ del a[:] သည် အားလုံးကို ဖျက်ပေး(delete လုပ်) ပေးသည်။ **del** ကို သုံးလျှင် index နံပါတ် ထည့်ပေးရသည်။

5.13.7 list.index(item [, start [, end]])

list ထဲမှ item များ၏ index နံပါတ်ကို သိလိုသည့်အခါ **list.index()**ကို သုံးသည်။ **list.index(item [, start [, end]])** method သည် ထည့်ပေးလိုက်သည့် value နှင့် တူညီသည့် item ၏ index နံပါတ်ကို ပြန်ပေးသည်။ တူညီသည့် value များစွာရှိနေလျှင် ပထမဆုံးတွေ့သည့် item ၏ index နံပါတ်ကို ပြန်ပေးသည်။

မိမိရှာလိုသည် နေရာအစ နှင့် အဆုံးကို ထည့်ပေး၍လည်း index နံပါတ်ကို ရယူနိုင်သည်။ Index zero အစဦးဆုံး item ဖြစ်သည်။ List ၏ နောက်ဆုံး item ၏ index နံပါတ်သည် list အရှည်(length) အရေအတွက်ထက် (၁)လျော့သည်။ (The index at the end of the list is one less than the length of the list.)

```
things = ['first', 'second', 'third']
```

```
things.index('second')              # 'second'၏ index နံပါတ်ကို ရှာသည်။
```

```
1
```

ရှာလိုသည့် အစနေရာနှင့် အဆုံးနေရာကိုပေး၍ ထိုအစနှင့် အဆုံးအတွင်း၌သာ ရှာနိုင်သည်။ အောက်က ဥပမာတွင် 1 မှ 3 အတွင်း၌သာ third ဆိုသည့် string ကို ရှာလိမ့်မည်။

```
things = ['first', 'second', 'third', 'fourth']
things.index('third', 1, 3)   # index နံပါတ် 1 မှ 3 အတွင်း၌သာ 'third' ကို ရှာရန်
```

```
2
```

ရှာခိုင်းသည့် item(specified item)သည် list ထဲတွင် ရှိမနေပါက Python သည် **ValueError** ထုတ်ပေးလိမ့်မည်။

```
things = ['first', 'second', 'third', 'fourth']
things.index('fifth')               # ValueError ထုတ်ပေးလိမ့်မည်။
```

```
ValueError                Traceback (most recent call last)
ValueError: 'fifth' is not in list
```

5.13.8 list.count(item)

List ထဲတွင် ရှိသည့် element တစ်မျိုးသည် အကြိမ်မည်မျှ ပါဝင်နေသည်ကို သိရန် **.count()** ကို သုံးသည်။ **list.count(item)** method list ထဲတွင် ရှိနေသည့် အရေအတွက်ကို ဖော်ပြပေးသည်။ (number of occurrences of an element in a list.)။

```
myList=["a","a","a","b","c","c"]
myList.count('a')                  # list ထဲတွင် 'a' ဘယ်နှစ်ခါပါသည်ကို ရှာရန်
```

```
3
```

5.13.9 list.sort(key=None, reverse=False)

sort() method သည် ဘာမှ မသတ်မှတ်ပေးလျှင် ငယ်ရာမှ ကြီးရာသို့(ascending)စဉ်ပေးသည်။ (sorts the list ascending by default.)။ မိမိ အလိုရှိသည့် sorting criteria ကို ထည့်ပေးနိုင်သည်။ ဘာမှ မပြောလျှင် ငယ်ရာမှ ကြီးရာသို့ စီသည်။ Keyword argument ကိုသုံး၍ စီနိုင်သည်။ **reverse** argument ကို သုံး၍ ကြီးရာမှ ငယ်ရာသို့ ပြောင်းပြန်(descending order)စီနိုင်သည်။

- ကိန်းတန်ဖိုး အလိုက် စီနိုင်သည်။
- String အတိုအရှည်အလိုက် စီနိုင်သည်။ (စာလုံးရေ အနည်းအများ)
- အက္ခရာ ရှေ့နောက် စီနိုင်သည်။
- အက္ခရာ အကြီးစာလုံး အသေးစာလုံးအလိုက် စီနိုင်သည်။
- နှစ်၊ လ၊ အချိန်(time) အလိုက် စီနိုင်သည်။

5.13.10 list.sort(key=None, reverse=False) method

```
cars = ['Ford', 'BMW', 'Volvo']
```

```
cars.sort(reverse=True)            # ပြောင်းပြန် စီသည်။ (V, F, B)
print(cars)
```

```
['Volvo', 'Ford', 'BMW']
```

```
L = ['abc', 'ABD', 'aBe']
L.sort()                           # Sort with mixed case
L
```

```
['ABD', 'aBe', 'abc']
```

အောက်တွင် **str.lower** ကို keyword argument ကိုသုံး၍ အသေးစာလုံးကို ဦးစားပေး **စီသည်**။

```
my_list = ['abc', 'aBD','ABD', 'aBe']
my_list.sort(key=str.lower) # အသေးစာလုံး(lowercase)ကို ဦးစားပေး၍ စီသည်။
my_list
```

```
['abc', 'aBD', 'ABD', 'aBe']
```

```
my_list = ['abc', 'aBD','ABD', 'aBe']
# အသေးစာလုံးကို ဦးစားပေး၍ ပြောင်းပြန် စီသည်။(Change sort order)
my_list.sort(key=str.lower, reverse=True)
my_list
```

```
['aBe', 'aBD', 'ABD', 'abc']
```

5.13.11 list.reverse()

list.reverse() method သည် list အတွင်း element များ၏ အစဉ်ကို ပြောင်းပြန်လှန်ပေးသည်။ (reverses the elements of the list.)။

5.13.12 list.copy()

list.copy() method သည် list တစ်ခုလုံးကို ကော်ပီကူးပေးသည်။(returns a shallow copy of a list)။

```
fruits = ['apple', 'banana', 'cherry', 'orange']
x = fruits.copy()
print(x)
```

```
['apple', 'banana', 'cherry', 'orange']
```

5.14 method များကို အသုံးပြုပုံနှင့် သတိပြုရန်အချက်များ

5.14.1 append() method, extend() method နှင့် inert() method

List နောက်ဆုံးတွင် item များ တစ်ခုပြီးတစ်ခုထည့်(adding items one by one to the end) လိုသည့်အခါ append()ကို သုံးသည်။ **append()** method ကို သုံးရန်အတွက် သတိပြုရမည့်အချက်မှာ element တစ်ခုတည်းကို list နောက်ဆုံးနေရာ(single element to the end)တွင် ထည့်နိုင်သည်။ ထည့်မည့် အရေအတွက်နှင့် ထည့်မည့်နေရာသည် သတိပြုရမည့်အချက် ဖြစ်သည်။

တစ်ခုထက်ပိုများသည့် element များကို list ၏ နောက်ဆုံး၌ ထည့်လိုပါက extend() method ကို အသုံးပြုသည်။ List ၏ နောက်ဆုံးမဟုတ်သည့် နေရာတွင် element များ ထည့်လိုပါက inert() method ကို အသုံး ပြုသည်။

```
fib = [1, 1, 2, 3, 5, 8]
fib.append(13)           # list ၏ နောက်ဆုံးတွင် 13 ကို append() ဖြင့် ထည့်သည်။
fib
```

```
[1, 1, 2, 3, 5, 8, 13]
```

```
fib += [89, 144]             # (+=) operator ကို သုံးသည်။
fib
```

```
[1, 1, 2, 3, 5, 8, 13, 89, 144]
```

```
fib.extend([21, 34, 55])
fib
```

```
[1, 1, 2, 3, 5, 8, 13, 89, 144, 21, 34, 55]
```

5.14.2 assignment (=) နှင့် del operators

assignment(=) operator နှင့် del operator တို့ကို ဥပမာဖြင့် အောက်တွင် ရှင်းပြထားသည်။ စတုတ္ထမြောက်ကိန်း နေရာတွင် "whoops" တန်ဖိုးကို assignment (=)ဖြင့် အစားထိုးသည်။ (Set the fourth element of the fib list to whoops.)

```
fib[3] = "whoops" # fourth element ကို whoop ဖြင့် assign လုပ်သည်။
fib
```

```
[1, 1, 2, 'whoops', 5, 8, 13, 89, 144, 21, 34, 55]
```

ပထမဆုံးကိန်း(၅)လုံးကို ဖျက်သည်။ ဖယ်ရှားသည်။(Remove the first five elements)

```
del fib[:5]
fib
```

```
[8, 13, 89, 144, 21, 34, 55]
```

Index နံပါတ် မဂဏန်း ဖြစ်သည့်နေရာရှိ element အားလုံးကို -1 ဖြင့် အစားထိုးသည်။ (Assign -1 to each odd element)

```
fib[1::2] = [-1, -1, -1]
fib
```

```
[8, -1, 89, -1, 21, -1, 55]
```

5.14.3 Lists of Lists

List ထဲတွင် list များသည် element အဖြစ် ရှိနေနိုင်သည်။ ထိုသို့ list ထဲတွင် list များသည် element အဖြစ် ရှိနေသည့်အခါ access လုပ်ပုံကို လေ့လာနိုင်၍ရန် အောက်တွင် ဥပမာဖြင့် ပြထားသည်။

```
small_birds = ['hummingbird', 'finch']
extinct_birds = ['dodo', 'passenger pigeon', 'Norwegian Blue']
carol_birds = [3, 'French hens', 2, 'turtledoves']
all_birds = [small_birds, extinct_birds, 'macaw', carol_birds]
all_birds
```

```
[['hummingbird', 'finch'],
 ['dodo', 'passenger pigeon', 'Norwegian Blue'],
 'macaw',
 [3, 'French hens', 2, 'turtledoves']]
```

ပထမဆုံး item သည် list တစ်ခု ဖြစ်သည်။ all_birds ဆိုသည့် list ၏ first item သည် small_birds ဆိုသည့် list ဖြစ်သည်။

```
all_birds[0]
```

```
['hummingbird', 'finch']
```

List ၏ second item သည် extinct_birds ဆိုသည့် list ဖြစ်သည်။ ထို extinct_birds ထဲမှ item များကို access လုပ်ရန် index နံပါတ်ခု ပေးရမည်။

```
all_birds[1]
```

```
['dodo', 'passenger pigeon', 'Norwegian Blue']
```

```
all_birds[1][0]
```

```
'dodo'
```

[1] သည် all_birds ထဲမှ second item ဖြစ်သည့် extinct_birds ကို ရည်ညွှန်းသည်။ [0] သည် inner list ဖြစ်သည့် extinct_birds ထဲမှ first item ကို ရည်ညွှန်းသည်။

5.14.5 Multiplication by an Integer

List တစ်ခုကို integer ဖြင့် မြှောက်လျှင် element အရေအတွက် အဆပွားများသည်။

```
[1, 2, 3] * 6          # 1, 2, 3 ကို (၆)ဆ ပွားသည်။
```

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

List တစ်ခုကို နာမည် (၂)မျိုး၊ (၃)မျိုး(two or more variable names)ပေးနိုင်သည်။ နာမည်တစ်မျိုးတွင် element တစ်ခုကို ပြောင်းလိုက်လျှင် ကျန်နာမည်အားလုံးနှင့် သက်ဆိုင်သည်။

Python statement x = y = [ ] ၏အဓိပ္ပာယ်မှာ empty list အသစ်တစ်ခုတွင် နာမည်နှစ်မျိုးရှိသည်။ (one new empty list with two names (x and y))။ List တစ်ခုကို နာမည်(၂)မျိုး ပေးထားခြင်း ဖြစ်သည်။

```
x = [3, 2, 1, "blast off!"]
y = x
y[1] = "TWO"
print(x)
```

```
[3, 'TWO', 1, 'blast off!']
```

5.14.6 List ထဲရှိ item တစ်ခုချင်းစီကို ပြောင်းနိုင်သည်။(Change an Item)

```
marxes = ['Groucho', 'Chico', 'Harpo']
marxes[2] = 'Wanda'       # index နံပါတ် 2 ကို 'Wanda' ဖြင့် assign လုပ်သည်။
marxes
```

```
['Groucho', 'Chico', 'Wanda']
```

အပေါ်ကကဲ့သို့ list offset လုပ်ခြင်းမျိုးကို string အတွင်းရှိ character တစ်ခုခုကို ပြောင်းရန် အတွက် အသုံးမပြုနိုင်ပါ။ အဘယ်ကြောင့်ဆိုသော် string များသည် ပြောင်းလဲနိုင်သည့် data structure မဟုတ်သောကြောင့် ဖြစ်သည်။ (strings are immutable)

5.14.7 Get a Slice to Extract Items

Slice လုပ်ခြင်းဖြင့် list ထဲမှ element များကို access လုပ်နိုင်သည်။ Extract လုပ်နိုင်သည်။

```
marxes = ['Groucho', 'Chico', 'Harpo']
marxes[0:2]
```

```
['Groucho', 'Chico']
```

List တစ်ခုကို slice လုပ်သည့်အခါ list တစ်ခု ဖြစ်ပေါ်လာသည်။ (element တစ်ခုထက် ပိုများသည့်အခါ)

```
test = marxes[::2]
print(test, type(test ))
```

```
['Groucho', 'Harpo'] <class 'list'>
```

```
# start at the end and go left by 2:
marxes[::-2]
```

```
['Harpo', 'Groucho']
```

-1 သည် list ကို ပြောင်းပြန်(reverse) လုပ်ခြင်းဖြစ်သည်။ အစမှာအဆုံးအထိ ပါဝင်သောကြောင့် list တစ်ခုလုံးကို ပြောင်းပြန်ပြန်စီသည်။

```
marxes[::-1]
```

```
['Harpo', 'Chico', 'Groucho']
```

```
thisList=["apple","banana","cherry","orange","kiwi","melon","mango"]
print(thisList)
```

```
['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon', 'mango']
```

နောက်ဆုံး item ကို print ထုတ်ရန် ။ [-1] နောက်ဆုံး item ဖြစ်သည်။

```
print("last element in the list--->", thisList[-1])
```

```
last element in the list---> mango
```

[2:5] သည် index နံပါတ် 2 မှ 4 အထိ ဖြစ်သည်။ (index နံပါတ် 5 မပါ)။ (return elements from index 2 to 4(index 5 is not included))

```
print(thisList[2:5])
```

```
['cherry', 'orange', 'kiwi']
```

[:4])သည် index နံပါတ် 0 မှ 3 အထိ ဖြစ်သည်။ (index နံပါတ် 4 မပါ)။ (returns the elements from 0 to 3 (prints first four elements))

```
print(thisList[:4])
```

```
['apple', 'banana', 'cherry', 'orange']
```

[2:] သည် နံပါတ် 2 မှ အဆုံးအထိ ဖြစ်သည်။ (returns the elements from index 2 to end)

```
print(thisList[2:])
```

```
['cherry', 'orange', 'kiwi', 'melon', 'mango']
```

[-4:-1] index နံပါတ် **-4** မှ -2 အထိ ဖြစ်သည်။ (index နံပါတ် -1 မပါ)။ (return elements index -4 to -1(not included))

```
print(thisList[-4:-1])
```

```
['orange', 'kiwi', 'melon']
```

5.14.8 Combine Lists by Using extend() or +=

.extend() သို့မဟုတ် **+=** ကို သုံး၍ list နှစ်ခုကို ပေါင်း(merge လုပ်)နိုင်သည်။

```
marxes = ['Groucho', 'Chico', 'Harpo', 'Zeppo']
others = ['Gummo', 'Karl']
marxes.extend(others)
marxes
```

```
['Groucho', 'Chico', 'Harpo', 'Zeppo', 'Gummo', 'Karl']
```

```
# Alternatively, you can use +=:
marxes = ['Groucho', 'Chico', 'Harpo', 'Zeppo']
others = ['Gummo', 'Karl']
marxes += others
marxes
```

```
['Groucho', 'Chico', 'Harpo', 'Zeppo', 'Gummo', 'Karl']
```

append() ကို သုံးလျှင် list နှစ်ခုကို ပေါင်း(merge လုပ်)ခြင်း မဟုတ်တော့ဘဲ append လုပ်ခြင်း ခံရသည့် list သည် item တစ်ခု (single item) အနေဖြင့် ဝင်ရောက်သွားလိမ့်မည်။

```
marxes = ['Groucho', 'Chico', 'Harpo', 'Zeppo']
others = ['Gummo', 'Karl']
marxes.append(others)
marxes
```

```
['Groucho', 'Chico', 'Harpo', 'Zeppo', ['Gummo', 'Karl']]
```

5.15 Integer ကို နှစ်ထပ်တင်ခြင်း

1 မှ 10 အထိပါသည့် list တစ်ခုကို range(1, 11) တည်ဆောက်သည်။

```
squares = [num for num in range(1, 11)]
squares
```

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

```
squares = [num**2 for num in range(1, 11)] # num**2 ဖြင့် နှစ်ထပ်ကိန်းတင်
squares
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

1 မှ 10 အထိကိန်းများ(`range(1, 11)`)ကို နှစ်ထပ်ကိန်းတင်သည်။ ၎င်းကိန်းများထဲမှ (၂)ဖြင့် စား၍ ပြတ်သည့် စုံကိန်းများ(num%2 == 0)ကိုသာ ယူ၍ list တစ်ခု တည်ဆောက်သည်။

```
squares = [num**2 for num in range(1, 11) if num%2 == 0]
squares
```

```
[4, 16, 36, 64, 100]
```

ပေးထားသည့် string တစ်ခုကို unicode သို့ပြောင်းပြီး list တစ်ခု ပြုလုပ်သည်ရန်

```
symbols = '$¢£¥€¤'
codes = []
for symbol in symbols:
    codes.append(ord(symbol))

codes
```

```
[36, 162, 163, 165, 8364, 164]
```

5.16 List ဥပမာများ

List အတွင်းရှိ element တစ်ခုချင်းစီကို for loop သုံး၍ print လုပ်သည်။

```
# looping through the list
thisList =['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon',
           'mango']
for x in thisList:
```

```
    print(x, end ='')  # list အတွင်းရှိ element တစ်ခုချင်းစီကို print လုပ်သည်။
 # end ='' ကို element တစ်ခုချင်းစီ print ထုတ်သည့်အခါ အလျားလိုက် ဖော်ပြရန် သုံးသည်။
```

```
apple banana cherry orange kiwi melon mango
```

```
'avogado' in thisList       # in keyword ဖြစ် ရှိ မရှိ စစ်ဆေးသည်။
```

```
False
```

```
thisList
```

```
['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon', 'mango']
```

မိမိသိလိုသည့် list အတွင်းရှိ၌ avogado ရှိ မရှိ if else: သုံး၍ စစ်ဆေးသည်။ မရှိပါက ထည့်ပေး (append() လုပ်ပေး)ရန်

```
if 'avogado' in thisList:                              #'avogado' ရှိလျှင်
    print('True, avogado in thisList')                 # print ထုတ်ရန်
else:
    thisList.append('avogado')          #'avogado' မရှိလျှင်.append() လုပ်ရန်
    print(thisList)
```

```
['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon', 'mango',
'avogado']
```

sorted() method ကို သုံး၍ list အတွင်းရှိ item များအား function ဖြင့် စာလုံးအရေအတွက် အတိုအရှည်ကို လိုက် စီခြင်းဖြစ်သည်။ Function အခန်းကို မဖတ်ရသေးသူများ ကျော်သွားပါ။

```
# to sort a list by creating another list by using sorted() method
# စာလုံးအရေအတွက် အတိုအရှည်ကို လိုက်၍ sort လုပ်ခြင်း
def Sorting(lst):
    lst2 = sorted(lst, key=len)
    return lst2

lst = ["maung", "lin", "thuzar", "thinzar", "zaw",
       "juju", "soe"]
print(Sorting(lst))
print(lst)          #sort မလုပ်ရသေးသည့် မူလ list ကို print ထုတ်သည်။
```

```
['lin', 'zaw', 'soe', 'juju', 'maung', 'thuzar', 'thinzar']
['maung', 'lin', 'thuzar', 'thinzar', 'zaw', 'juju', 'soe']
```

5.16.1 Python program to demonstrate sorting by user's choice

List1 ထဲတွင် ကိန်း(၂)လုံးပါသည့် အတွဲ (၃)တွဲ ရှိသည်။ ပထမ ဥပမာတွင် ထိုကိန်းအတွဲထဲမှ ပထမ

ကိန်း(index နံပါတ် 0) တန်ဖိုးကို လိုက်၍ စီသည်။ ဒုတိယ ဥပမာတွင် ထိုကိန်းအတွဲထဲမှ ဒုတိယကိန်း(index နံပါတ် 1) တန်ဖိုးကို လိုက်၍ စီသည်။

```
# Function to return the second element of the two elements passed
as the parameter
def sortFirst(val):             # Function တည်ဆောက်သည်။
    return val[0]               # index နံပါတ် 0 ကို ရွေးပေးသည်။

# list1 to demonstrate the use of sorting using using second key
list1 = [(2, 6), (3, 6), (1, 6)]

# sorts the array in ascending according to  second element
# ပထမကိန်း(index နံပါတ် 0) တန်ဖိုးကို လိုက်၍ ငယ်ရာမှ ကြီးရာသို့ စီသည်။
list1.sort(key = sortFirst)     # sortFirst မှ index နံပါတ် 0 အတိုင်းစီရန်
print(list1)

# sorts the array in descending according to second element
# ပထမကိန်း(index နံပါတ် 0) တန်ဖိုးကို လိုက်၍ ကြီးရာမှ ငယ်ရာသို့ စီသည်။
list1.sort(key = sortFirst, reverse = True)
print(list1)
```

```
[(1, 6), (2, 6), (3, 6)]
[(3, 6), (2, 6), (1, 6)]
```

```
# Function to return the second element of the two elements passed as
the parameter
def sortSecond(val): # Function တည်ဆောက်သည်။
    return val[1]

# list1 to demonstrate the use of sorting using using second key
list1 = [(7, 2), (7, 3), (7, 1)]

# sorts the array in ascending according to second element
# ဒုတိယကိန်း(index နံပါတ် 1) တန်ဖိုးကို လိုက်၍ ငယ်ရာမှ ကြီးရာသို့ စီသည်။
list1.sort(key = sortSecond)
print(list1)
```

```
[(7, 1), (7, 2), (7, 3)]
```

```
# sorts the array in descending according to second element
# ဒုတိယကိန်း(index နံပါတ် 1) တန်ဖိုးကို လိုက်၍ ကြီးရာမှ ငယ်ရာသို့ စီသည်။
list1.sort(key = sortSecond, reverse = True)
print(list1)
```

```
[(7, 3), (7, 2), (7, 1)]
```

5.17 List Comprehensions

Comprehension ဆိုသည်မှာ Python data structure များ ပြုလုပ်ခြင်း၊ တည်ဆောက်ခြင်း ဖြစ်သည်။ List comprehension ဆိုသည်မှာ list များ ပြုလုပ်ခြင်း၊ တည်ဆောက်ခြင်း ဖြစ်သည်။ တစ်နည်း အားဖြင့် list ပြုလုပ်သည့် နည်းလမ်းများဟု မှတ်ယူနိုင်သည်။ Python တွင် ပြောဆိုလေ့ရှိသည့် စကားလုံး ဖြစ်သည်။ ထိုသို့ Python ဝေါဟာရများ ပြောဆိုခြင်း၊ Python စတိုင် code များ ရေးခြင်းကို Pythonic ဟု တင်စားကြသည်။ 1 မှ 5 အထိ ကိန်းများ ပါသည့် list တစ်ခုကို iterator and နှင့် range() function ကို အသုံးပြုပြီး အောက်ပါနည်း (၃)မျိုးဖြင့် တည်ဆောက်နိုင်သည်။

ပထမနည်း - empty list တစ်ခုကို တည်ဆောက်သည်။ ထို empty list ထဲသို့ **.append()** method ကို သုံး၍ 1 မှ 5 အထိ ကိန်းများကို တစ်ခုချင်းစီထည့်သည်။

```
number_list = []
number_list.append(1)
number_list.append(2)
number_list.append(3)
number_list.append(4)
number_list.append(5)
number_list
```

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

ဒုတိယနည်း- iterator နှင့် range() function ကို သုံး၍ တည်ဆောက်သည်။

```
number_list = []
for number in range(1, 6):
    number_list.append(number)
number_list
```

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

တတိယနည်း range() ရဲ့ output ကို **list()** ထဲသို့ထည့်ပြီး တစ်ကြောင်းတည်းဖြင့် ရေးနိုင်သည်။

```
number_list = list(range(1, 6))
number_list
```

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

အထက်တွင် ဖော်ပြခဲ့သည့် နည်း(၃)မျိုး စလုံးသည် စနစ်တကျ ရေးထားသည့် Python code များ ဖြစ်ကြပြီး တူညီသည့် အဖြေကို ထုတ်ပေးသည်။ သို့သော် list တည်ဆောက်ခြင်း(comprehension)အတွက် Pythonic နည်း မှာ `[ expression for item in iterable ]` ဖြစ်သည်။ range(1, 6) သည် 1 မှ 5 အတွင်းသာ ဖြစ်သည်။ range(1, 6) သည် list တစ်ခု မဟုတ်ပါ။ ထို့ကြောင့် for keyword ဖြင့် digit တစ်လုံးစီကို ထုတ်၍ list တစ်ခု ဖြစ်အောင် ပြုလုပ်သည်။

```
# Pythonic way to build a list
number_list = [number for number in range(1,6)]
```

```
number_list
```

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

လေးထောင်းကွင်း[ ] ထဲတွင် number variable နှစ်ခုပါဝင်သည်။ ပထမ number variable သည် list ထဲသို့ ဝင်ရောက်လာမည့် ကိန်းများကို ကိုယ်စားပြုသည်။ ဒုတိယ number variable သည် for loop အတွက် ဖြစ်သည်။ အောက်တွင် နောက်ထပ် ဥပမာတစ်ခုကို လေ့လာကြရအောင်။
1 မှ 5 အထိ ကိန်းများကို နှစ်ထပ်ကိန်းတင်ထားသည့်ကိန်းများ ပါသည့် list တစ်ခုကို တည်ဆောက်နိုင်သည်။

```
number_list = [number ** 2 for number in range(1,6)]
number_list
```

```
[1, 4, 9, 16, 25]
```

နောက်ထပ် အနည်းငယ်ခက်ခဲသည့် ဥပမာတစ်ခုမှ square bracket ထဲတွင် loop နှင့် if statement ကို ထည့်ထားသည်။ [`expression for item in iterable if condition`] 1 မှ 5 အထိ ပါဝင်သည့် ကိန်းများ အတွင်းမှ မကိန်း(odd number)များသာပါသည့် list တစ်ခုတည်ဆောက်ပါ။ `number % 2 == 1` ဖြင် စစ်၍ သည် True ဖြစ်လျှင် မကိန်း(odd number) ဖြစ်သည်။ False ဖြစ်လျှင် စုံကိန်း(even number) ဖြစ်သည်။

```
a_list = [number for number in range(1,6) if number % 2 == 1]
a_list
```

```
[1, 3, 5]
```

အထက်တွင် ဖော်ပြခဲ့သည့် ဥပမာကို သမာရိုးကျနည်းဖြင့် ရေးပါက အောက်ပါအတိုင်း ရေးနိုင်သည်။ လုပ်ခြင်းသည့် အနည်းငယ်ပိုကျစ်လစ်သည်။ အလေ့အကျင့်များသည့်အခါ အလွယ်တကူ ဖတ်နိုင်ပါလိမ့်မည်။

```
a_list = []
for number in range(1,6):
    if number % 2 == 1:
        a_list.append(number)
a_list
```

```
[1, 3, 5]
```

Nested loops ဥပမာ - for loop နှစ်ခုပါဝင်သည်။ (သမာရိုးကျနည်း)

```
rows = range(1,4)       # [1, 2, 3]
cols = range(1,3)       #[1, 2]
for row in rows:
    for col in cols:
        print(row, col)
```

```
1 1
1 2
2 1
2 2
```

```
3 1
3 2
```

(row, col) ပုံစံမျိုးဖြင့် tuple များ ပါသည့် list တစ်ခုတည်ဆောက်ရန်(Pythonic နည်း)

```
rows = range(1,4)
cols = range(1,3)
cells = [(row, col) for row in rows for col in cols]
for cell in cells:
    print(cell)
```

```
(1, 1)
(1, 2)
(2, 1)
(2, 2)
(3, 1)
(3, 2)
```

5.18 List Comprehensions and Readability

အောက်တွင် list တစ်ခုပြုလုပ်သည့် နည်း နှစ်မျိုးကို ဖော်ပြထားသည်။ လေ့လာခါစ သူများအတွက် ပထမ ဥပမာမှာ သမာရိုးကျ for loop ဖြင့် ရေးထားသောကြောင့် ဖတ်ရန်လွယ်ကူသည်ဟု ထင်ကြသည်။ Coding အတွေ့အကြုံ အသင့်အတင့် ရလာပြီးနောက်ပိုင်း ဒုတိယ ဥပမာမှာ ဖတ်ရန်လွယ်ကူသည်ဟု လက်ခံလာကြသည်။ ဥပမာ - ပေးထားသည့် string တစ်ခုကို unicode သို့ပြောင်းပြီး list တစ်ခု ပြုလုပ်သည်။

```
# Build a list of Unicode codepoints from a string,
symbols = '$¢£¥€¤'
codes = []                #Empty list တစ်ခုတည်ဆောက်သည်။
for symbol in symbols:  # for loop ကိုသုံး၍ element တစ်ခုချင်းစီကို ပြောင်းသည်။
    codes.append(ord(symbol))  # element တစ်ခုချင်းစီကို list သို့ထည့်သည်။

codes
```

```
[36, 162, 163, 165, 8364, 164]
```

ဥပမာ - ပေးထားသည့် string တစ်ခုကို Unicode သို့ပြောင်းပြီး တစ်ခု ပြုလုပ်သည်ရန်

```
# Build a list of Unicode codepoints from a string, take two.
symbols = '$¢£¥€¤'
codes = [ord(symbol) for symbol in symbols]
codes
```

```
[36, 162, 163, 165, 8364, 164]
```

5.19 List Arguments

Function တစ်ခုထဲသို့ list တစ်ခုကို ထည့်ပေးနိုင်သည်။ Function သည် ထိုထည့်ပေးသည့် list ကို တစ်ခု reference အဖြစ် ယူထားသည်။ Function က list parameter ကို ပြောင်းလဲလိုက်လျှင်(modify လုပ်လျှင်) ထို function ကို ခေါ်သုံးသူ(caller)က ပြောင်းလဲသွားမှုကို မြင်ရလိမ့်မည်။ ဥပမာ- ဝင်လာသည့် list ၏ index နံပါတ် 0 (first element) ကို ဖြုတ်ပေးရန် function တစ်ခု တည်ဆောက်သည်။

```
def delete_head(t):
    del t[0]                    # index နံပါတ် 0 ကို delete လုပ်သည်။

letters = ['a', 'b', 'c']           # letters list
delete_head(letters)                #function ကို ခေါ်သုံးသည်။
print(letters)
```

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

Parameter t နှင့် variable letters တို့သည် object တစ်ခုတည်းကို ကိုယ်စားပြုသည်။(same object)။

5.20 Multidimensional Lists and Tuples

List တစ်ခုထဲတွင် တခြားသော list များ အထပ်လိုက် ထားနိုင်သည်။ Multidimensional lists သို့မဟုတ် lists of lists ဟုခေါ်သည်။ အောက်တွင် 2 dimensional list (list of lists) တစ်ခုကို ဥပမာ အဖြစ် ပြထားသည်။

```
#2 dimensional  list (list of lists)
a = [[1,2,3,4], [5,6,7,8]]
a
```

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

List of list များသည် ဇယားများ(tables) နှင့် တခြားသော data structures များတွင် အသုံးဝင်သည့် သဘောတရာ ဖြစ်သည်။ NumPy arrays နှင့် matrices အခန်းတွင် အသေးစိတ် ရှင်းပြ ထားသည်။

```
a[0]
```

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

```
a[1]
```

```
[5, 6, 7, 8]
```

```
a[1][0]
```

```
5
```

```
a[2][1]
```

```
IndexError: list index out of range
```

5.21 append() Method နှင့် + Operator

.append() method သည် မူလ list ကို modify လုပ်သည်။ **+** operator သည် list အသစ်တစ်ခု ပြုလုပ် ပေးသည်။ ထို့ကြောင့် မိမိလုပ်လိုသည့် အလုပ် ကိုက်ညီသည့် append() method သို့မဟုတ် + operator ကို ခွဲခြား အသုံးပြုတတ်ရန် လိုသည်။

```
t1 = [1, 2]
t2 = t1.append(3)
print(t1)
print(t2)          # None ပေးလိမ့်မည်။
```

```
[1, 2, 3]
None
```

```
t3 = t1 + [4]
print(t3)
```

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

ထိုကွဲပြားချက်သည် function နှင့်တွဲ၍ အသုံးပြုသည့်အခါ အလွန် အရေးကြီးသည့် အချက် ဖြစ်လာသည်။ List method အများစုမှာ argument ကို modify လုပ်ပြီး None ပြန်ထုတ်ပေးသည်။ String method များသည် string အသစ်တစ်ခု ထုတ်ပေးပြီး မူလ string ကို သူ့အတိုင်း ချန်ထားသည်။

5.22 Errors From List

```
x = [0,1,2,3,4,5,6,7,8,9]
x[10]
#  Index နံပါတ် 10 မရှိသောကြောင့် list index out of range Error ပေးလိမ့်မည်။
```

```
IndexError: list index out of range
```

5.23 Lists and Tuples

- Python တွင် tuple နှင့် list ဟူ၍ sequence structures နှစ်မျိုးရှိသည်။ Tuple နှင့် list ထဲတွင် ဘာမျှ မရှိသည့် empty tuple နှင့် empty list တို့ ဖြစ်နိုင်သည်။ Tuple နှင့် list တို့ထဲတွင် ရှိသည့် element များ အားလုံးသည် Python object များ ဖြစ်ကြသည်။
- Python တွင် list များသည် one-dimensional သာ ဖြစ်သည်။ List ကို object များအား အစီအစဉ်တကျ နေရာတကျ ထည့်ထားနိုင်သည့် ကွန်တိန်နာ(ordered containers) အဖြစ် မှတ်ယူနိုင်သည်။

- List တစ်ခုကို လေးထောင့်ကွင်း [ ] သို့မဟုတ် list() စသည့်နည်း နှစ်မျိုး ဖြင့် ပြုလုပ်နိုင်သည်။ Element ကို ကော်မာ ခံရေး၍ လေးထောင့်ကွင်း အတွင်း၍ ထည့်ရေးသည်။
- Python list တခြားသော language များနှင့် မတူသည့်အချက်မှာ list ထဲတွင် ရှိသည့် element များ၏ အမျိုးအစား တူညီရန် မလိုအပ်ပေ။
- list နှစ်ခုကို အပေါင်းလက္ခဏာဖြင့်(addition operator (+))ဖြင့် သာမန်နည်းဖြင့် ပေါင်းနိုင် (concatenate လုပ်နိုင်)သည်။

```
[1, 1] + [2, 3, 5] + [8]
```

```
[1, 1, 2, 3, 5, 8]
```

End of List