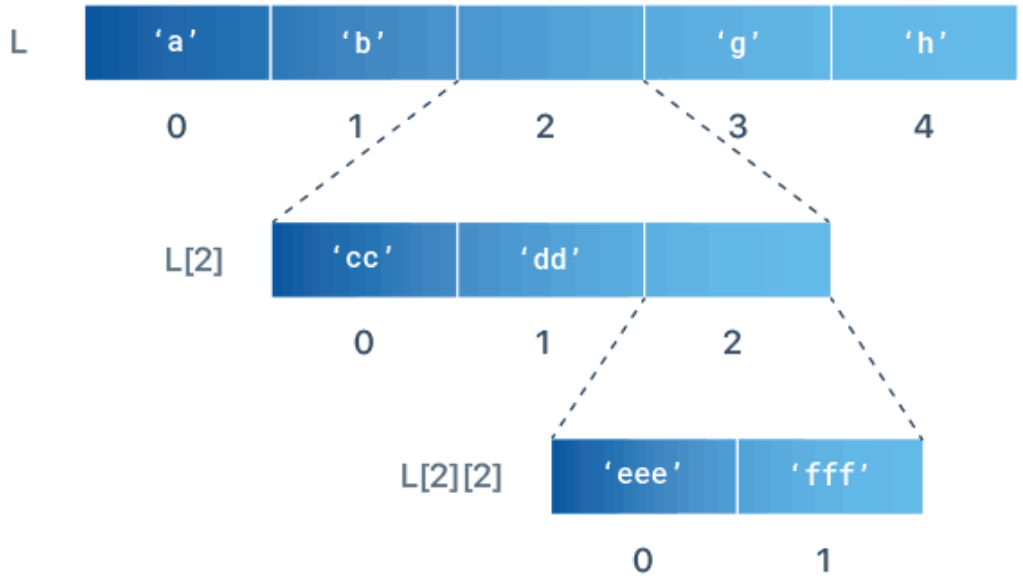
CSS112 Computer Programming

ดร. พีระจักร์ วิฑูรชาติ

Contents

- List of List
- Flatten the list of list
- Comprehension with condition

List of List



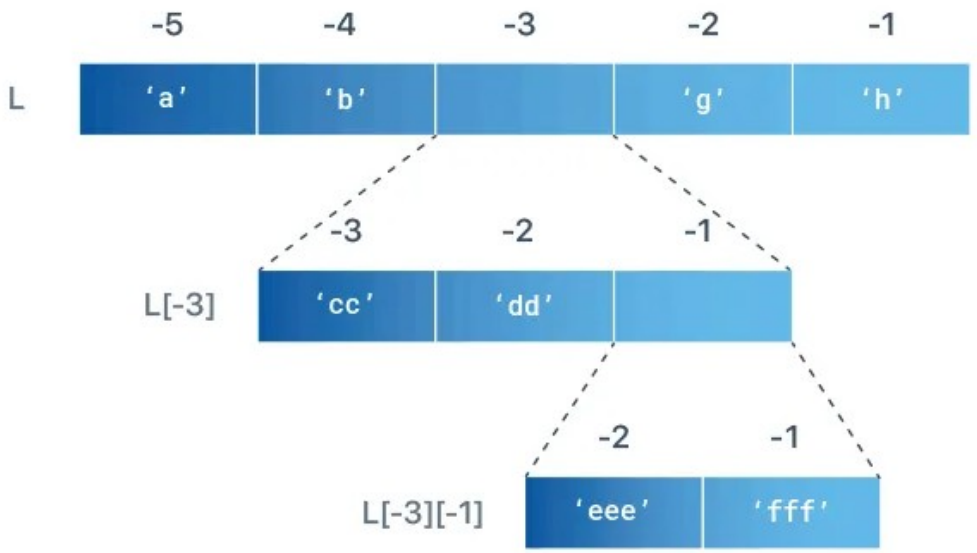

```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']

print(L[2])
# Prints ['cc', 'dd', ['eee', 'fff']]

print(L[2][2])
# Prints ['eee', 'fff']

print(L[2][2][0])
# Prints eee
```

List of List: negative index




```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']

print(L[-3])
# Prints ['cc', 'dd', ['eee', 'fff']]

print(L[-3][-1])
# Prints ['eee', 'fff']

print(L[-3][-1][-2])
# Prints eee
```

ลองดู

```
L = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', 'h']
```

- เรียก L อย่างไรให้พิมพ์ ‘ccc’?
  - Positive index
  - Negative index

Creating List of List

```
In [1]: L = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', 'h']

In [6]: L[1][1][0]
Out[6]: 'ccc'

In [9]: L[-3][-3][-2]
Out[9]: 'ccc'
```

Insert, Delete element

```
In [1]: L = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', 'h']
```

```
In [10]: L[1][1] = 0
```

```
In [11]: print(L)
```

```
['a', ['bb', 0, 'ee', 'ff'], 'g', 'h']
```

```
In [12]: L[1].append('xx')
         print(L)
```

```
['a', ['bb', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
In [15]: L[1].insert(0,'yyyy')
         print(L)
```

```
['a', ['yyyy', 'bb', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
In [17]: del L[1][1]
         print(L)
```

```
['a', ['yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

Remove Element(s)

```
In [20]: L1add = ['yyyy', 0, 'ee', 'ff', 'xx']
         L[1].extend(L1add)
         print(L)

['a', ['yyyy', 0, 'ee', 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']

In [21]: L[1].remove('ee')
         print(L)

['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

- ถ้าคุณต้องการลบออกจากลิสต์โดยใช้ค่า
- จะสังเกตุได้ว่าทุกelementที่มีค่าเป็น 'ee' ของ L[1] ถูกลบออกไป
- ยกตัวอย่างการใช้ list.extend ในกรณี list of list

หาความยาวของลิสต์ในลิสต์

Find Nested List Length

You can use the built-in len() function to find how many items a nested sublist has.

```
L = ['a', ['bb', 'cc'], 'd']

print(len(L))
# Prints 3

print(len(L[1]))
# Prints 2
```

Iterate through list of list

Iterate through a Nested List

To iterate over the items of a nested list, use simple for loop.

```
L = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
for list in L:
    for number in list:
        print(number, end=' ')
# Prints 1 2 3 4 5 6 7 8 9
```

ระมัดระวัง

```
print(L)
```

```
['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
for kL in L:
    for i in kL:
        if i=='yyyy':
            i = 'kkkk'
```

```
print(L)
```

```
['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

- เพราะว่า iterator กำลังทำงานบน immutable object

ถ้า iterator กำลังทำงานบน mutable

```
In [85]: L = ['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], [['g']], 'h']

In [86]: for kL in L:
             for i in kL:
                 if i==['g']:
                     i+=['gy']

In [87]: print(L)
['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], [['g', 'gy']], 'h']
```

- สังเกตุ [‘g’] กลายเป็น [‘g’,’gy’] เพราะ iterator เป็น mutable

อีกวิธี

```
L = ['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
for kL in L:
    for i in kL:
        if i=='yyyy':
            i = 'kkkk'
```

```
print(L)
```

```
['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

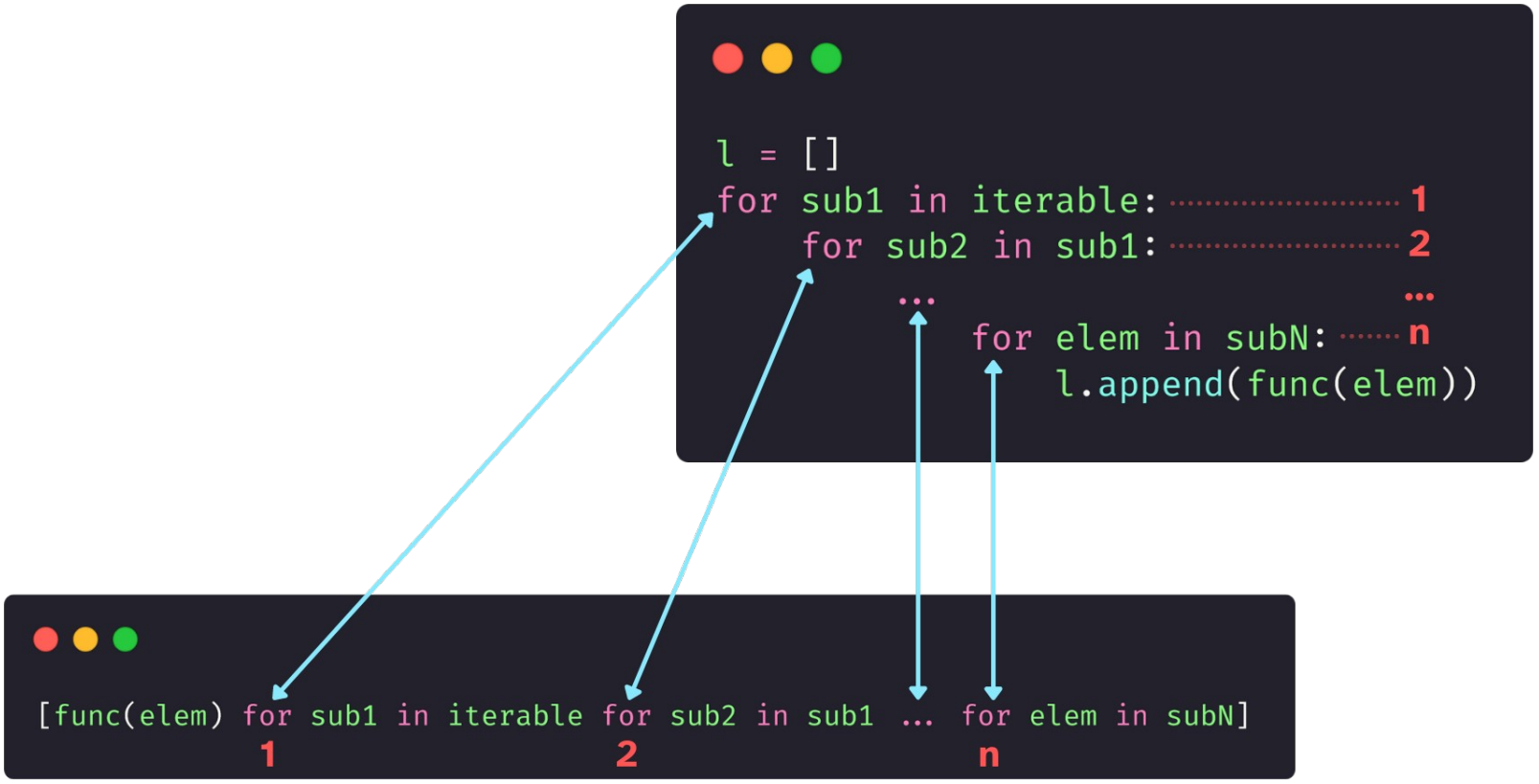
```
for kL in L:
    for i in range(len(kL)):
        if kL[i] == 'yyyy':
            kL[i] = 'kkkk'
```

```
print(L)
```

```
['a', ['kkkk', 0, 'ff', 'xx', 'kkkk', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

- เราต้องการเปลี่ยน ‘yyyy’ ทุกค่าเป็น ‘kkkk’

Nested List Comprehension



- ลูปนอก ลูปใน ค่าที่ต้องการ append

ตัวอย่าง

```
>>> lists = [[1, 2, 3], [4, 5, 6, 7], [8, 9]]
>>> [elem for sublist in lists for elem in sublist]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Sublist = [1,2,3] ใน iter ที่1
- Sublist = [4,5,6,7] ใน iter ที่1

Example

```
l = []
for i in range(3):
    for j in range(2):
        l.append((i,j))

print(l)
```

```
[(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

```
ll = [(i,j) for i in range(3) for j in range(2)]
print(ll)
```

```
[(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

ลองดู

- จงสร้างผลคูณคาร์เทเซียนของ ลิสต์สองลิสต์ดังต่อไปนี้ โดยใช้ nested list comprehension
  - fruits = ['apple','banana','carrot']
  - num = list(range(3))
  - ตอบ [('apple', 0), ('apple', 1), ('apple', 2), ('banana', 0), ('banana', 1), ('banana', 2), ('carrot', 0), ('carrot', 1), ('carrot', 2)]

ตัวอย่างซับซ้อน

```
L = ['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
for kL in L:
    for i in kL:
        if i=='yyyy':
            i = 'kkkk'
```

```
print(L)
```

```
['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
for kL in L:
    for i in range(len(kL)):
        if kL[i] == 'yyyy':
            kL[i] = 'kkkk'
```

```
print(L)
```

```
['a', ['kkkk', 0, 'ff', 'xx', 'kkkk', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

เขียนด้วย loop ก่อน

```
print(L)
```

```
['yyyy', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
lst = []
for kL in L:
    if isinstance(kL,list):
        kLl = []
        for i in range(len(kL)):
            if kL[i] == 'yyyy':
                kLl.append('kkkk')
            else:
                kLl.append(kL[i])
        lst.append(kLl)
    else:
        if kL == 'yyyy':
            lst.append('kkkk')
        else:
            lst.append(kL)
print(lst)
```

```
['kkkk', ['kkkk', 0, 'ff', 'xx', 'kkkk', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

- โปรดสังเกตุ ‘yyyy’ มีสองระดับ

Nested list comprehension

```
print(L)
```

```
['yyyy', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

```
Lcomp=[ [ 'kkkk' if kL[i] == 'yyyy' else kL[i] for i in range(len(kL))] \
       if isinstance(kL,list) else 'kkkk' if kL == 'yyyy' else kL  for kL in L]
```

```
print(Lcomp)
```

```
['kkkk', ['kkkk', 0, 'ff', 'xx', 'kkkk', 0, 'ee', 'ff', 'xx'], 'g', 'h']
```

- วิธีนี้ก็ได้

Nested List and Recursive Fn

```
L = ['a', ['yyyy', 0, 'ff', 'xx', 'yyyy', 0, 'ee', 'ff', 'xx'], [['g'],['yyyy'],[[[[[[['yyyy']]]]]]], 'yyyy']
```

```
def recurs_replace(L):
    if isinstance(L,str):
        if(L == 'yyyy'):
            return 'kkkk'
        else:
            return L
    if isinstance(L,int):
        return L
    else:
        return [recurs_replace(i) for i in L]


Lfn = recurs_replace(L)
print(Lfn)
```

```
['a', ['kkkk', 0, 'ff', 'xx', 'kkkk', 0, 'ee', 'ff', 'xx'], [['g'], ['kkkk'], [[[[[[['kkkk']]]]]]], 'kkkk']
```

- If there are unknown depth of nested list, recursive function can be handy

การสร้างnested listแบบเงื่อนไข

```
NUMNUM = [[1,2,3],[3,4],[12,6,9]]
evenNum = []
for i in NUMNUM:
    for j in i:
        if not j%2:
            evenNum.append(j)

evenNum2 = [j for i in NUMNUM for j in i if not j%2 ]
print(evenNum,evenNum2)
```

```
[2, 4, 12, 6] [2, 4, 12, 6]
```

```
evenNum = []
for i in NUMNUM:
    s = []
    for j in i:
        if not j%2:
            s.append(j)
    evenNum.append(s)

evenNum2 = [ [j for j in i if not j%2] for i in NUMNUM ]
print(evenNum,evenNum2)
```

วิธีคิด

- For I in NUMNUM return a list อย่างแน่นอน
- ข้างในลิสต์ทำลิสต์แบบมีเงื่อนไขตามปกติ

ลองดู

- L = [['yyyy'], ['yyyy', 'a1', 'ff', 'xx', 'yyyy', 'a2', 'ee', 'ff', 'xx'], ['ghhh', 'h']]
- สร้างลิสต์ใหม่จาก comprehension โดยเลือกเฉพาะstringที่มีlengthมากกว่า3โดยยังคง list ใน list ไว้
- คำตอบคือ
  - Lgreater3 = [['yyyy'], ['yyyy', 'yyyy'], ['ghhh']]

เฉลย

```
L = [['yyyy'], ['yyyy', 'a1', 'ff', 'xx', 'yyyy', 'a2', 'ee', 'ff', 'xx'], ['ghhh', 'h']]

Llen = [[j for j in i if len(j)>3] for i in L ]
print(Llen)
```

ลองใช้ recursive functionดู

- ทำข้อเดิม ลองใช้ recursive function

เฉลย

```
def recurs_len3(L):
    if isinstance(L,str):
        if len(L)>3:
            return L
    else:
        return [j for j in [recurs_len3(i) for i in L] if j is not None ]
LE = [['yyyy'], ['yyyy', 'a1', 'ff', 'xx', 'yyyy', 'a2', 'ee', 'ff', 'xx'], ['ghhh', 'h']]
Lrecurse3 = recurs_len3(LE)
print(Lrecurse3 )
```

```
[['yyyy'], ['yyyy', 'yyyy'], ['ghhh']]
```

ลองดู

- L = [['yyyy'], ['yyyy', 'a1', 'ff', 'xx', 'yyyy', 'a2', 'ee', 'ff', 'xx'], ['ghhh', 'h']]
- สร้างลิสต์ใหม่จาก comprehension โดยเลือกเฉพาะstringที่มีlengthมากกว่า3โดยยังคง list ใน list ไว้
- เงื่อนไขเพิ่มเติม ทำเฉพาะลิสต์ในลิสต์ที่มี 'yyyy' อยู่เท่านั้น
- คำตอบคือ
  - Lgreater3 = [['yyyy'], ['yyyy', 'yyyy']]