CSS112 Computer Programming

ดร. พีระจักร์ วิฑูรชาติ

Contents

- List,Set,Dict Comprehension
  - Basic
  - List comprehension with List constructor
  - Conditional List Comprehension
  - If else in expression
  - Scope of variable
  - Using function with Comprehensions

List Comprehension

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
    newlist.append(x.upper())

print(newlist)
```

```
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [ i.upper() for i in fruits]

print(newlist)
```

```
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

- โปรแกรมด้านบนสามารถเขียนในรูปโปรแกรมด้านล่างได้
- โปรแกรมภาพล่างเราเรียกการเขียนแบบนี้ว่า List Comprehension.

Excercise

- ให้นักเรียนเขียน list comprehension เพื่อสร้างลิสต์ใหม่ให้ได้คำตอบด้านล่างจากลิสต์ fruits
- ให้ว่า fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

```
['Apple', 'Banana', 'Cherry', 'Kiwi', 'Mango']
```

Basic: Expression

- Call a method
- Call a function
- Arithmetic Operation
- Ternary Operator

Example

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
fruits3 = fruits*3

def SequentialUpper(aStringInput,num):
    n = num if num < len(aStringInput) else len(aStringInput)-1
    return aStringInput[0:n] + aStringInput[n].upper() + aStringInput[n+1:]

newlist = [ SequentialUpper(i,j) for j,i in enumerate(fruits3)]
print(newlist)
print(fruits3)
print(fruits)
```

```
['Apple', 'bAnana', 'chErry', 'kiwI', 'mangO', 'applE', 'bananA', 'cherrY', 'kiwI', 'mangO', 'applE', 'bananA', 'cherrY', 'kiwI', 'mangO']
['apple', 'banana', 'cherry', 'kiwi', 'mango', 'apple', 'banana', 'cherry', 'kiwi', 'mango', 'apple', 'banana', 'cherry', 'kiwi', 'mango']
['apple', 'banana', 'cherry', 'kiwi', 'mango']
```

- Call a function
  - Enumerate คือ Built-in Function
  - มีการใช้ฟังก์ชันที่เรานิยามเองขึ้นมา

Example Ternary Operator

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newfruits = [ "my "+x if x=="banana" else x for x in fruits]
print(fruits)
print(newfruits)
```

```
['apple', 'banana', 'cherry', 'kiwi', 'mango']
['apple', 'my banana', 'cherry', 'kiwi', 'mango']
```

Set Comprehension

```
fruits = {"apple", "banana", "cherry", "kiwi", "mango"}

newfruits = [ "my "+x if x=="banana" else x for x in fruits]
print(fruits)
print(newfruits)
```

```
{'kiwi', 'apple', 'cherry', 'banana', 'mango'}
['kiwi', 'apple', 'cherry', 'my banana', 'mango']
```

- List มี order
- แต่ set ไม่มี
- เมื่อสร้างจาก Set ก็อาจจะได้ Order มั่วๆ

List from Tuple

```
fruits = ("apple", "banana", "cherry", "kiwi", "mango")
newlist = [x for x in fruits if "a" in x]

print(newlist)
```

```
['apple', 'banana', 'mango']
```

- เราสามารถสร้าง list จาก tuple ได้

Tuple from Tuple ไม่ได้

```
fruits = ("apple", "banana", "cherry", "kiwi", "mango")
newtuple = (x for x in fruits if "a" in x)

print(newtuple)
```

```
<generator object <genexpr> at 0x150c3ceda9e0>
```

- กลายเป็น generator Expression แทน

Basic: Iterable

- List
- Dict
- Set
- Tuple

List Comprehension with List Constructor

```
numlist = list(range(20))
print(numlist)

numlist2 = list( 2*i for i in range(20))
print(numlist2)

numlist3 = list( i**2 if i%2==0 else i/2 for i in range(20))
print(numlist3)

numlist4= list( i**2  for i in range(20) if i%2==0)
print(numlist4)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[0, 0.5, 4, 1.5, 16, 2.5, 36, 3.5, 64, 4.5, 100, 5.5, 144, 6.5, 196, 7.5, 256, 8.5, 324, 9.5]
[0, 4, 16, 36, 64, 100, 144, 196, 256, 324]
```

- ส่งให้ generator expression ให้ List constructor

Condition List Comprehension

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if x != "apple"]

print(newlist)
```

```
['banana', 'cherry', 'kiwi', 'mango']
```

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = []
for i in fruits:
    if i != "apple":
        newlist.append(i)
print(newlist)
```

```
['banana', 'cherry', 'kiwi', 'mango']
```

- โค้ดด้านล่างกับด้านบนได้ผลเหมือนกัน

Condition “Not In” a list

```
alllist = list(range(20))
print(alllist)

evenlist = [i*2 for i in range(10)]
print(evenlist)
oddlist = [ i for i in alllist if i not in evenlist]
print(oddlist)
```

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

Nested Condition

```
numlist = list(range(20))
print(numlist)


numlist1= list( i  for i in range(20) if i%2==0 and i%3 ==0)
print(numlist1)

numlist2= list( i  for i in range(20) if i%2==0 if i%3 ==0)
print(numlist2)
```

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

Excercise

- ให้นักเรียนเขียน list comprehension เพื่อให้ได้คำตอบด้านล่าง

```
numlist = list(range(20))

print(numlist)
#Your code here

#End your code
print('your even number list',evenlist)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
your even number list [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Dict Comprehension

```
#item price in dollars
old_price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}

dollar_to_Baht = 38
new_price = {item: value*dollar_to_Baht for (item, value) in old_price.items()}

print(old_price.items())
print(new_price)
```

```
dict_items([('milk', 1.02), ('coffee', 2.5), ('bread', 2.5)])
{'milk': 38.76, 'coffee': 95.0, 'bread': 95.0}
```

- วิธีปกติในการสร้าง dict ใหม่จาก dict เก่า

Dict from two lists

```
talads =['milk', 'coffee', 'bread']
prices = [1.02,2.5,2.6]
old_price = {k:v for k,v in zip(talads,prices)}
print(old_price)
```

```
{'milk': 1.02, 'coffee': 2.5, 'bread': 2.6}
```

- Use Zip to zip together and then dict comprehension.

Two lists from a dict

```
taladdict = {'milk':1.2,'tea':2.4,'bread':3.6}
taladitem = [k for k,v in taladdict.items()]
taladval = [v for k,v in taladdict.items()]
print(taladdict)
print(taladitem)
print(taladval)
```

```
{'milk': 1.2, 'tea': 2.4, 'bread': 3.6}
['milk', 'tea', 'bread']
[1.2, 2.4, 3.6]
```

Dict with condition

```
talads =['milk', 'coffee', 'bread']
prices = [1.02,2.5,2.6]
old_expensive_price = {k:v for k,v in zip(talads,prices) if v>1.5}
old_no_e_price = {k:v for k,v in zip(talads,prices)  if not "e" in k }
print(old_expensive_price)
print(old_no_e_price)
```

```
{'coffee': 2.5, 'bread': 2.6}
{'milk': 1.02}
```

Dict Comprehension

```
# dictionary comprehension example
square_dict = {num: num*num for num in range(1, 11)}
print(square_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Dict from constructor

```
taladdict = {'milk':1.2,'tea':2.4,'bread':3.6}
taladitem = [k for k,v in taladdict.items()]
taladval = [v for k,v in taladdict.items()]
print(taladdict)
print(taladitem)
print(taladval)
taladthaibaht =dict((k,v*38) for k,v in zip(taladitem,taladval))
print(taladthaibaht)
```

```
{'milk': 1.2, 'tea': 2.4, 'bread': 3.6}
['milk', 'tea', 'bread']
[1.2, 2.4, 3.6]
{'milk': 45.6, 'tea': 91.2, 'bread': 136.8}
```

เรื่องที่ต้องระมัดระวัง

- การใช้ Function ที่ return None ในฐานะ expression
  - เช่นกรณีนี้มีการพิมพ์ออกมาจริง แต่ว่า list ที่ได้คือ [None, None, None]

```
fruits = ("apple", "banana", "cherry", "kiwi", "mango")
newtuple = [print(x) for x in fruits if "a" in x]

print(newtuple)
```

```
apple
banana
mango
[None, None, None]
```

Exercise

- ลองสร้าง dict จากสองลิสต์เอง
- ลองแยก dict กลับมาเป็นสองลิสต์
- ลองสร้าง dict, list, set จาก constructor
- ลอง set comprehension ด้วยเงื่อนไข
- ลองสร้าง dict ที่ key ต้องไม่อยู่ใน set ต้องห้าม
- ลองสร้าง dict ที่ key ต้องอยู่ใน set เฉพาะถึงจะอนุญาติ