

# Tuple – Set – Dict

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# Tuple

- tuple เหมือน list แต่สร้างแล้วเปลี่ยนแปลงไม่ได้
  - list: `x1 = [1, 3, 4]; x2 = [9]`
  - tuple: `t1 = (1, 3, 4); t2 = (9,)`
- มี operations ต่าง ๆ เหมือนลิสต์ (เฉพาะที่ไม่เปลี่ยนค่า)

```
t = (11,22,33)
print(len(t))           # 3
print(t[0],t[-1])       # 11 33
print(t[:2])            # (11,22)
print(33 in t)          # True
print(t.index(33))      # 2

# แก้ไขไม่ได้ แต่สร้างใหม่ได้
t[0] = 99                # ผิด
t = (99,) + t[1:]        # (99,22,33)
```

# Tuple vs. List

- มักใช้ list เก็บข้อมูลความหมายเหมือนกัน ชนิดเดียวกัน แต่แต่ละตัวอาจเปลี่ยนค่า และลิสต์เปลี่ยนขนาดได้
  - ลิสต์ของนักเรียนเรียงตามคะแนนมากไปน้อย
  - ลิสต์ของอุณหภูมิที่วัดได้ทุกชั่วโมง
  - ลิสต์ของลูกค้าที่รอรับบริการ
- มักใช้ tuple เก็บข้อมูลความหมายต่างกัน อาจต่างชนิดกัน ข้อมูลไม่เปลี่ยนแปลง และ 튜เปิ้ลไม่เปลี่ยนขนาด
  - 튜เปิ้ลสองช่องเก็บพิกัด x, y แทนตำแหน่ง
  - 튜เปิ้ลสามช่องเก็บ เลขประจำตัว ชื่อ และ ปีเกิด
- ในทางเทคนิค tuple เล็กกว่า เร็วกว่า (นิดหน่อย)

ตัวอย่างของลิสต์ที่เคยนำเสนอมาก่อนนี้ หลายอันใช้ 튜เปิ้ลจะเหมาะสมกว่า

# ใช้ tuple เหมาะกว่า

```
days_of_week = ("SU", "MO", "TU", "WE", "TH", "FR", "SA")
```

```
[ ("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A") ]
```

```
[ (4, "your"), (4, "kiss"), (2, "is"), (2, "on"), (2, "my"), (4, "list") ]
```

```
words = ["your", "kiss", "is", "on", "my", "list"]  
x = [ (len(s), s) for s in words ]
```

# ใช้ tuple เพื่อป้องกันการแก้ไข

```
[ ["6131001021", "A"], ["6130020221", "B"], ["6130150721", "A"] ]
```

```
[ ("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A") ]
```

```
( ("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A") )
```

```
def print_grades(grades):  
    for name, grade in grades:  
        print(name, "-->" grade)  
    grades[0][1] = "F"  
    grades[0] = (grades[0][0], "F")
```

```
grades = ((grades[0][0], "F"),) + grades[1:]
```

# แบบฝึกหัด: Polynomial

$4x^2 + 3x - 1$  แทนด้วย  $[(4, 2), (3, 1), (-1, 0)]$

```
def add_polynomial( p1, p2 ) :  
    # คำนวณผลบวกของ p1 กับ p2
```

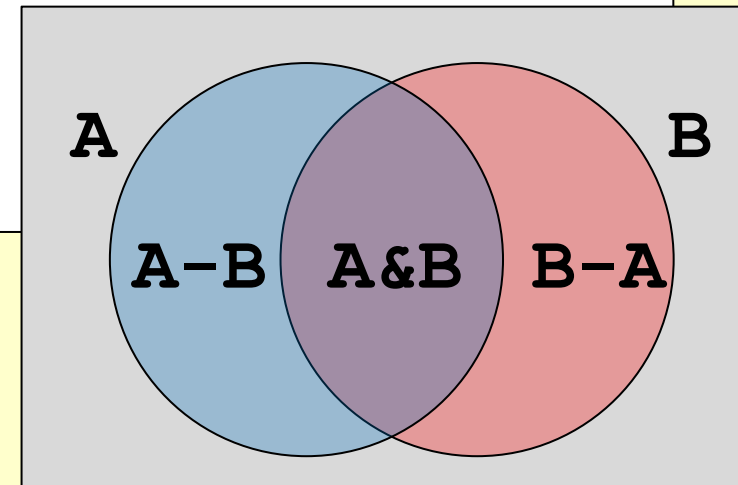
```
def mult_poly(p1, p2) :  
    # คำนวณผลคูณของ p1 กับ p2
```

# Set

- set เป็นที่เก็บข้อมูลที่ไม่ซ้ำกัน และไม่มีลำดับ
- ข้อมูลที่เก็บใน set ต้องเปลี่ยนแปลงไม่ได้
  - เก็บ int, float, bool, str, tuple
  - เก็บ list, dict, set ไม่ได้
- `s = {4, 3, 1, 2}`
- `s = set()` ได้เซตว่าง แต่ `{ }` ได้ dict ว่าง
- ใช้ `len(s)`, `if e in s`, และ `for e in s` ได้

# Set Methods

```
A = {1,2,3,4,5}
B = {3,4,5,6,7}
C = A.union(B)           # {1,2,3,4,5,6,7}
C = A | B                 # {1,2,3,4,5,6,7}
C = A.intersection(B)   # {3,4,5}
C = A & B                 # {3,4,5}
C = A.difference(B)      # {1,2}
C = A - B                # {1,2}
C.add(9)                  # {1,2,9}
C.remove(1)              # {2,9}
print( A <= B, A.issubset(B) ) # True True
for e in A:
    print(e)
C = A ^ B
```





# การสร้างเซตจากข้อมูลในที่เก็บต่าง ๆ

```
s = set(d)
```

เหมือนกับ

```
s = set()
for e in d:
    s.add(e)
```

s = set( [1,2,3,1] )	→ s = {1,2,3}
s = set( (1,2,3,1) )	→ s = {1,2,3}
s = set( "Mono" )	→ s = {"M", "o", "n"}
s = set( {"A":2, "B":2} )	→ s = {"A", "B"}
s = set( {1,2,3} )	→ s = {1,2,3}
s = set( range(1,7,2) )	→ s = {1,3,5}

# การสร้างลิสต์จากข้อมูลในที่เก็บต่าง ๆ

```
x = list(d)
```

เหมือนกับ

```
x = []  
for e in d:  
    x.append(e)
```

<code>x = list( [1,2,3,1] )</code>	<code>→ x = [1,2,3,1]</code>
<code>x = list( (1,2,3,1) )</code>	<code>→ x = [1,2,3,1]</code>
<code>x = list( "Mono" )</code>	<code>→ x = ["M", "o", "n", "o"]</code>
<code>x = list( {"A":2, "B":2} )</code>	<code>→ x = ["A", "B"]</code>
<code>x = list( {1,2,3} )</code>	<code>→ x = [1,2,3]</code>
<code>x = list( range(1,7,2) )</code>	<code>→ x = [1,3,5]</code>

# การสร้างทูเปิลจากข้อมูลในที่เก็บต่าง ๆ

```
t = tuple( [1,2,3,1] )      → t = (1,2,3,1)
t = tuple( (1,2,3,1) )      → t = (1,2,3,1)
t = tuple( "Mono" )          → t = ("M", "o", "n", "o")
t = tuple( {"A":2, "B":2} )  → t = ("A", "B")
t = tuple( {1,2,3} )         → t = (1,2,3)
t = tuple( range(1,7,2) )    → t = (1,3,5)
```

# sort vs. sorted

```
def sorted(x) :  
    out = []  
    for e in x:  
        out.append(e)  
    out.sort()  
    return out
```

`sorted(x)` เป็น built-in function ที่นำข้อมูลใน x มาเรียงลำดับข้อมูล แล้วคืนผลลัพธ์กลับมาเป็นลิสต์ ส่วน x ยังเหมือนเดิม

<code>x = sorted( [1,4,3,2] )</code>	<code>→ x = [1, 2, 3, 4]</code>
<code>x = sorted( {1,4,3,2} )</code>	<code>→ x = [1, 2, 3, 4]</code>
<code>x = sorted( "hello" )</code>	<code>→ x = ["e", "h", "l", "l", "o"]</code>
<code>x = sorted( {22:2, 90:3, 3:23} )</code>	<code>→ x = [3, 22, 90]</code>

```
x = [1, 4, 3, 2]  
x.sort()          # sort ให้กับ list เท่านั้น
```

สังเกตความแตกต่างในการใช้ `x.sort()` กับ `sorted( x )`

*for elem in a\_set*

ข้อมูลถูกหยิบจากเซตออกมาด้วยลำดับที่ไม่แน่นอน

```
S = {11111, 22222, 33333,  
      44444, 55555, 66666}
```

```
for e in S:  
    print(e)
```

```
55555  
11111  
66666  
22222  
33333  
44444
```

# if *e* in *set* ทำงานเร็วกว่า if *e* in *list*

```
import time
def search_all(X):
    b = time.time()
    n = len(X)
    for i in range(n):
        if i in X:      # True
            pass
    for i in range(n):
        if (n+1) in X: # False
            pass
    print(time.time() - b)
```

```
n = 50000
L = []
for i in range(n):
    L.append(i)
S = set()
for i in range(n):
    S.add(i)

search_all(L)
search_all(S)
```

การค้นใน set  
เร็วกว่า list

**มาก**

41.84556722640991

0.0149579048156738

หน่วยเป็นวินาที

## ตัวอย่าง: ฟังก์ชันตรวจสอบข้อมูลซ้ำกันในลิสต์

```
def has_duplicate( x ):  
    for i in range(len(x)-1):  
        for j in range(i+1, len(x)):  
            if x[i] == x[j]:  
                return True  
    return False
```

```
def has_duplicate( x ):  
    d = sorted(x)  
    for i in range(len(x)-1):  
        if d[i] == d[i+1]:  
            return True  
    return False
```

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

## ตัวอย่าง: ฟังก์ชันตรวจสอบข้อมูลซ้ำกันในลิสต์

```
def has_duplicate( x ):  
    s = set( x )  
    return len(s) != len(x)
```

```
def has_duplicate( x ):  
    s = set()  
    for e in x:  
        if e in s:  
            return True  
        s.add(e)  
    return False
```



## ตัวอย่าง: หาสองจำนวนต่างกันที่รวมกันได้ k

```
x = [int(e) for e in input().split()]
d = set(x)
k = int(input())
soln = []
for e in d:
    e1 = k - e
    if e < e1 and e1 in d:
        soln.append((e, e1))
if len(soln)==0:
    print("Not found")
else:
    print(soln)
```

```
1 2 3 4 8 9 -8 -2
11
```

```
[(2, 9), (3, 8)]
```

# ตัวอย่าง: Sieve of Eratosthenes

```
S1 = {
    2, 3, 5, 7,
    11, 13, 17, 19,
    23, 29,
    31, 37,
    41, 43, 47,
}
```

จำนวนเฉพาะ  
ทุกตัวที่มีค่า  
ไม่เกิน N

```
S1 = set(range(2, N+1))
for i in range(2, int(N**0.5) + 1):
    S2 = set(range(2*i, N+1, i))
    S1 = S1 - S2
```

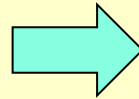
S2

```
{ 4, 6, 8, 10, ..., 50}
{ 6, 9, 12, 15, ..., 48}
{ 8, 12, 16, 20, ..., 48}
{10, 15, 20, 25, ..., 50}
{12, 18, 24, 30, ..., 48}
{14, 21, 28, 35, ..., 49}
```

# แบบฝึกหัด: Winner

ให้เขียนโปรแกรมเพื่อรับผลการแข่งขันฟุตบอล  
จากนั้นให้หาว่าทีมใดบ้างที่ไม่เคยแพ้เลย

5  
Chelsea Liverpool  
ManU Liverpool  
Liverpool ManU  
Chelsea Arsenal  
Everton ManCity



[ 'Chelsea', 'Everton' ]

# List vs. Dict

เก็บข้อมูลที่มีความสัมพันธ์กันได้ด้วย list หรือ dict

```
x = [ "6130102321", "6130238221", "6031022121" ]  
p = [ [10, 9, 10], [9, 10, 8], [5, 8, 7] ]
```

```
x = [ [6130102321, [10, 9, 10]],  
      [6130238221, [ 9, 10, 8]],  
      [6031022121, [ 5, 8, 7]] ]
```

```
d = { "6130102321": [10, 9, 10],  
      "6130238221": [ 9, 10, 8],  
      "6031022121": [ 5, 8, 7] }
```

# List vs. Dict

```
x = [ "6130102321", "6130238221", "6031022121" ]  
p = [ [10, 9, 10], [9, 10, 8], [5, 8, 7] ]
```

```
ID = input()  
if ID in x:  
    print( p[x.index(ID)] )
```



ช้า

# `x.sort()` จะเรียงแต่ในลิสต์ `x`, ข้อมูลใน `p` จะไม่ตรงตาม `x`



```
t = []  
for i in range(len(x)):  
    t.append([x[i], p[i]])  
t.sort()  
for id, sc in t:  
    print(id, sc)
```



ยุ่ง



# List vs. Dict

```
x = [ [6130102321, [10, 9, 10]],  
      [6130238221, [ 9, 10, 8]],  
      [6031022121, [ 5, 8, 7]] ]
```

```
ID = input()  
for sid, scores in x:      # ค้น ID ต้องไล่ค้นเอง  
    if sid == ID:           ข้า, ยุ่ง  
        print( scores )  
        break  
  
x.sort()  # เรียงตาม ID  
for sid, sc in x:           สะดวก  
    print( sc )
```

# List vs. Dict

```
d = { "6130102321": [10, 9, 10],  
      "6130238221": [ 9, 10, 8],  
      "6031022121": [ 5, 8, 7] }
```

```
ID = input()  
if ID in d:  
    print( d[ID] )  เร็ว  
  
# ต้องการเรียงตาม id  
for id in sorted(d):  
    print( id, d[id] )  
  
     สะดวก
```

# แบบฝึกหัด: เรียงประเภทเพลงตามเวลารวม

## Input

9

Shake It Off, Taylor Swift, Pop, 3:39

Rolling In The Deep, Adele, Pop, 3:48

Chandelier, Sia, Pop, 3:36

Roar, Katy Perry, Pop, 3:42

Hotel California, Eagle, Rock, 6:30

We Are the Champions, Queen, Rock, 2:59

Hello Dolly, Louis Armstrong, Jazz, 2:27

Bohemian Rhapsody, Queen, Rock, 5:55

Coward of the County, Kenny Rogers, Country, 4:20

## Output

Rock --> 15:24

Pop --> 14:45

Country --> 4:20

แสดงประเภทเพลง ตามด้วย  
เวลารวมเป็นนาทีและวินาที  
เรียงตามลำดับเวลารวม 3  
อันดับแรกจากมากมาน้อย



# More on Dict

ให้ value ของ dict เป็น list, set หรือ dict ก็ได้

```
{
  'Hello': {'Adele', 'Lionel Richie', 'Prince'},
  'Shake It Off': {'Taylor Swift'},
  'Chandelier': {'Sia'},
  "You've got a Friend": {'Carol King',
                           'James Taylor'},
  'What a Wonderful World': {'Anne Murray',
                              'Louis Armstrong',
                              'Rod Stewart'},
}
```

{str: set}

## ตัวอย่าง: dict

```
birthdate = {  
    "6130192221": (31,12,2000),  
    "6131022521": (28,2,2000),  
    "6230012121": (3,4,2001)  
}
```

{str: tuple}

```
ID = input().strip()  
print("Birth date of", ID, "is", birthdate[ID])  
print("Birth year of", ID, "is", birthdate[ID][2])
```

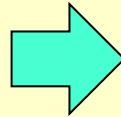
```
series = {  
    "Ranee": {"Roy Marn", "Plerng Boon"},  
    "Urassaya": {"Maya Tawan", "Kleun Cheewit"}  
}
```

{str: set}

```
name = input().strip()  
print(name, "starred in", ", ".join(series[name]))
```

# ตัวอย่าง: ตัวการ์ตูน

```
Ted, bear
Pongo, dog
Fozzie, bear
Winnie-the-Pooh, bear
Nana, dog
Scooby Doo, dog
Garfield, cat
Yogi, bear
Tom, cat
Sylvester, cat
Figaro, cat
Pluto, dog
Baloo, bear
Goofy, dog
Felix, cat
q
```



```
{ 'bear':
  { 'Ted', 'Fozzie'
    'Winnie-the-Pooh',
    'Yogi', 'Baloo' },
  'dog':
    { 'Pongo', 'Nana',
      'Scooby Doo',
      'Pluto', 'Goofy' },
  'cat':
    { 'Garfield', 'Tom',
      'Sylvester',
      'Figaro', 'Felix' }}
```

```
{ str: set }
```

## ตัวอย่าง: ตัวการ์ตูน

```
cartoon = {}  
x = input()  
while x != 'q':  
    name, atype = x.split(", ")  
    if atype not in cartoon:  
        cartoon[atype] = {name}  
    else:  
        cartoon[atype].add(name)  
    x = input()  
print(cartoon)
```

{ str: set }

อย่าเขียน set(name)

# แบบฝึกหัด: แสดงข้อมูลการ์ตูน ตามลำดับที่อ่านเข้ามา

Ted, bear  
Pongo, dog  
Fozzie, bear  
Winnie-the-Pooh, bear  
Nana, dog  
~~Hello Kitty, cat~~  
Scooby Doo, dog  
Garfield, cat  
Yogi, bear  
Tom, cat  
Sylvester, cat  
Pluto, dog  
Goofy, dog  
q

เรียงก่อนหลัง  
ตามทีอ่านเข้ามา

เรียงก่อนหลัง  
ตามทีอ่าน  
เข้ามา

bear: Ted, Fozzie, Winnie-the-Pooh, Yogi  
dog: Pongo, Nana, Scooby Doo, Pluto, Goofy  
cat: ~~Hello Kitty~~, Garfield, Tom, Sylvester

# More on Dict: keys(), values(), items()

```
D = { "Vios": "Toyota", "Fortuner": "Toyota",  
      "Wave": "Honda", "Civic": "Honda" }
```

```
for k in D.keys():  
    print(k)
```

```
for v in D.values():  
    print(v)
```

```
for k,v in D.items():  
    print(k, v)
```

```
Civic  
Fortuner  
Vios  
Wave  
Honda  
Toyota  
Toyota  
Honda  
Civic Honda  
Fortuner  
Toyota  
Vios Toyota  
Wave Honda
```

# ตัวอย่าง: reverse mapping

```
def reverse( d ): # กรณีที่ value ไม่ซ้ำกัน
    r = { }
    for k,v in d.items():
        r[v] = k
    return r
```

```
if v not in r:
    r[v] = {k}
else:
    r[v].add(k)
```

```
def reverse( d ): # กรณีที่ value อาจซ้ำกัน
    r = { }
    for k,v in d.items():
        if v not in r:
            r[v] = set()
        r[v].add(k)
    return r
```

**d**

```
{ "Vios": "Toyota", "Fortuner": "Toyota",
  "Wave": "Honda", "Civic": "Honda" }
```

**r**

```
{ "Toyota": {"Vios", "Fortuner"},
  "Honda": {"Wave", "Civic" }
```

# แบบฝึกหัด: ใครร้องเพลงนี้

11

Input

Hello, Adele

Shake It Off, Taylor Swift

Chandelier, Sia

You've got a Friend, Carol King

Hello, Lionel Richie

What a Wonderful World, Anne Murray

Hello, Prince

What a Wonderful World, Louis Armstrong

You've got a Friend, James Taylor

What a Wonderful World, Rod Stewart

Hello, Sai Wa Si Bor Tim Gun, You've got a Friend

Hello -> Adele, Lionel Richie, Prince

Output

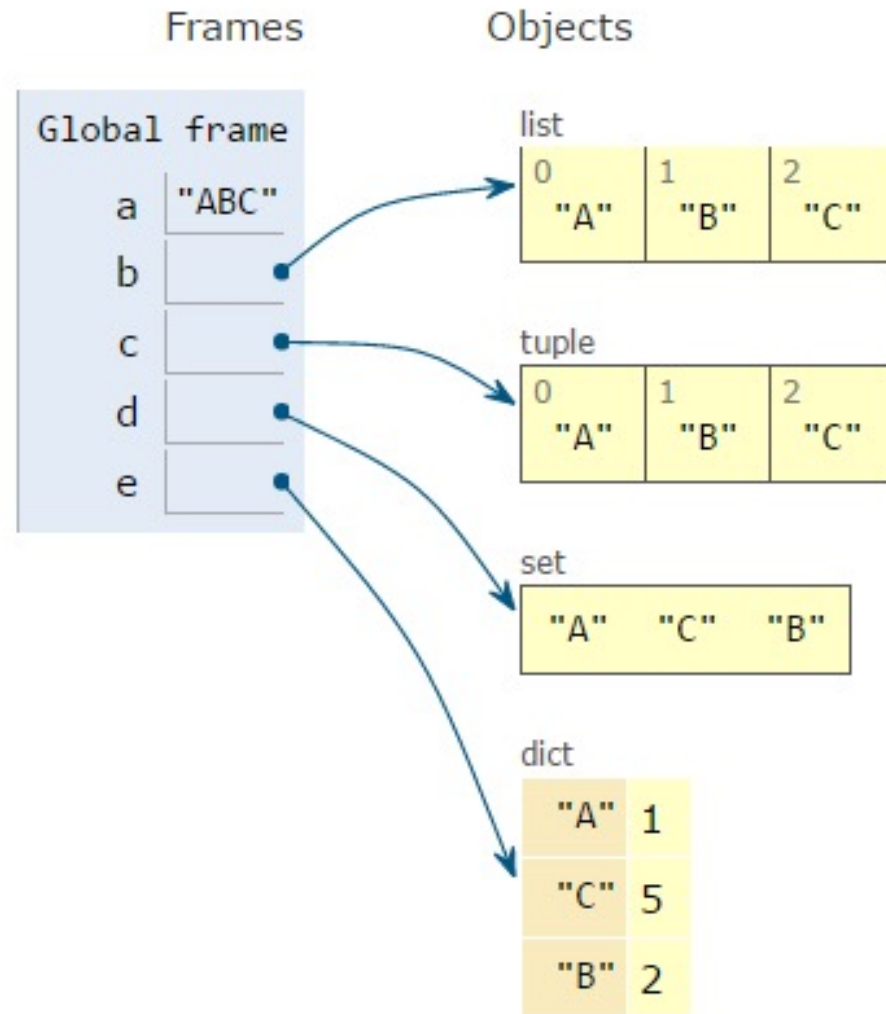
Sai Wa Si Bor Tim Gun -> Not found

You've got a Friend -> Carol King, James Taylor



# สรุป string, list, tuple, set, dict

```
1 a="ABC"  
2 b=["A","B","C"]  
3 c=("A","B","C")  
4 d={"A","B","C"}  
→ 5 e={"A":1,"B":2,"C":5}
```



# สรุปการใช้งาน list, tuple, dict, set

	list	tuple	dict	set
การใช้	<ul style="list-style-type: none"> <li>- ลำดับของข้อมูลมีความหมาย อาจมีการเปลี่ยนแปลง</li> <li>- ข้อมูลในรายการมักมีความหมายเดียวกัน</li> </ul>	<ul style="list-style-type: none"> <li>- ลำดับของข้อมูลมีความหมาย</li> <li>- สร้างแล้วไม่เปลี่ยนแปลง</li> <li>- ข้อมูลใน tuple มักมีความหมายต่างกัน</li> </ul>	<ul style="list-style-type: none"> <li>- เก็บข้อมูลเป็นคู่ๆ key-value โดยใช้ key เข้าถึงข้อมูลเพื่อให้ได้ value มาใช้งาน</li> </ul>	<ul style="list-style-type: none"> <li>- เก็บข้อมูลไม่ซ้ำ ลำดับของข้อมูลไม่มีความหมาย เพื่อตรวจสอบว่ามีข้อมูลหรือไม่ รองรับ set operations</li> </ul>
การเข้าใช้ข้อมูล	ใช้จำนวนเต็มระบุตำแหน่ง <code>d[i]</code>	ใช้จำนวนเต็มระบุ <code>d[i]</code>	ใช้ key เป็นตัวระบุตำแหน่งข้อมูล <code>d[key]</code>	ต้อง <code>for...in...</code> เพื่อแจงข้อมูล
การค้นด้วย <b>in</b>	ค้นจากซ้ายไปขวา	ค้นจากซ้ายไปขวา	มีวิธีค้นที่เร็วมาก	มีวิธีค้นที่เร็วมาก
การสร้าง	<code>x = [1, 2, 3, 4]</code>	<code>t = (1, 2, 3, 4)</code>	<code>d = { "k1": 1, "k2": 2 }</code>	<code>s = {1, 2, 3, 4}</code>
การเพิ่มข้อมูล	<code>x.append(3)</code> <code>x.insert(1, 99)</code>	สร้างแล้วเปลี่ยนแปลงไม่ได้ ต้องสร้างใหม่ <code>t = t + (4, )</code>	<code>d["k1"] = 1</code> <code>d["k2"] = 2</code>	<code>s.add(3)</code>