Data Structure

SHUTCHON PREMCHAISAWATT

เนื้อหา

1. โครงสร้างข้อมูลพื้นฐาน (Basic Data Structure)

2. โครงสร้างข้อมูลแบบประกอบ (Composite Data Structure) โครงสร้างข้อมูลพื้นฐานหลายๆตัวประกอบกัน)

โครงสร้างข้อมูลพื้นฐาน

1.1 ข้อมูลแบบตัวเลข (Numeric)

1.2 ข้อมูลแบบตัวอักษร (String)

ข้อมูลแบบตัวเลข (Numeric)

- **Integer** จำนวนเต็ม (4, 7, -2)
- **Floating Point Number** จำนวนจริง (0.1, 99.99)
- **Boolean** จำนวนฐานสอง (0 1, TRUE FALSE)
- **Complex Number** (1+2i)

```
In [1]:  1 a = 1

In [2]:  1 b = 2

In [3]:  1 type(a)

Out[3]: int

In [4]:  1 type(b)

Out[4]: int
```

Interger

```
In [5]:  1 c = 0.5

In [6]:  1 d = 3/4

In [7]:  1 type(c)

Out[7]: float

In [8]:  1 type(d)

Out[8]: float
```

Floating-Point Number

```
In [9]:  1 t = True

In [10]: 1 f = False

In [11]: 1 type(t)

Out[11]: bool

In [12]: 1 type(f)

Out[12]: bool
```

Boolean

```
In [13]:
1 z = 1 + 2j

In [14]:
1 z.real

Out[14]: 1.0

In [15]:
1 z.imag

Out[15]: 2.0

In [16]:
1 type(z)

Out[16]: complex
```

Complex Number

โครงสร้างข้อมูลพื้นฐาน

~~1.1 ข้อมูลแบบตัวเลข (Numeric)~~ เชือด

1.2 ข้อมูลแบบตัวอักษร (String)

String

String คือชุดของอักษรที่ต่อกัน เช่น “Python” การสร้างไม่ซ้อน แต่การใช้งานจะมีคำสั่งพิเศษที่เกี่ยวข้องมากมาย

*คำสั่ง type() ใช้ดูตัวแปรว่าเป็นตัวแปรอะไร

```
In [17]:  1 char1 = 'Hello Wold'

In [18]:  1 char2 = '0123456789'

In [19]:  1 type(char1)

Out[19]: str

In [20]:  1 type(char2)

Out[20]: str
```

การสร้าง String

```
In [26]:  1 char3 = '0123456789'

In [27]:  1 char3[3:]
Out[27]: '3456789'

In [28]:  1 char3[:7]
Out[28]: '0123456'

In [29]:  1 char3[3:7]
```

การเลือกเฉพาะบางส่วน ด้วย [:]

```
In [30]:  1 char4 = 'abc'

In [31]:  1 char4.replace('a', 'A')

Out[31]: 'Abc'

In [32]:  1 char4 = 'abc'

In [33]:  1 char4.replace('bc', 'DE')

Out[33]: 'aDE'
```

การแทนที่ String ด้วย replace()

หาความยาวของ String ด้วย len()

```
In [34]:  1 char5 = '12345'

In [35]:  1 len(char5)

Out[35]: 5
```

```
In [36]:  1 char6 = 'Python'

In [37]:  1 'P' in char6

Out[37]: True

In [38]:  1 'p' in char6

Out[38]: False
```

ตรวจว่ามีอักษรนี้ใน String หรือไม่ด้วย in

```
In [39]:  1 char7 = 'one-two-three'

In [40]:  1 char7.split('-')

Out[40]: ['one', 'two', 'three']

In [41]:  1 char8 = 'I love coding'

In [42]:  1 char8.split(' ')

Out[42]: ['I', 'love', 'coding']
```

การแยก String ด้วย split()

```
In [43]:
1 char9 = 'ab'
2 char10 = 'cd'
```

```
In [44]:
1 char9 + char10
```

```
Out[44]: 'abcd'
```

การต่อ String (Concatenation)

```
In [45]:
1 char11 = 'A'
2 char12 = 'B'
3 char13 = 'C'
```

```
In [46]:
1 char11 + char12 + char13
```

```
Out[46]: 'ABC'
```

การต่อ String (Concatenation)

โครงสร้างข้อมูลพื้นฐาน

~~1.1 ข้อมูลแบบตัวเลข (Numeric)~~ เชือด

~~1.2 ข้อมูลแบบตัวอักษร (String)~~ เชือด

เนื้อหา

~~1. โครงสร้างข้อมูลพื้นฐาน (Basic Data Structure)~~ เชือด

2. โครงสร้างข้อมูลแบบประกอบ (Composite Data Structure) โครงสร้างข้อมูลพื้นฐานหลายๆตัวประกอบกัน)

โครงสร้างข้อมูลแบบประกอบ

1. List
2. Tuple
3. Dictionary
4. Set

List

โครงสร้างข้อมูลแบบ List หรือรายการ เป็นการจัดเก็บข้อมูลแบบเรียงลำดับ เหมือนกับการเขียนรายการสิ่งของลงในกระดาษ โดยแต่ละรายการจะมีลำดับที่แน่นอน เช่น รายการที่ 1, 2, 3 ไปเรื่อยๆ และสามารถเพิ่ม ลบ หรือแก้ไขรายการได้

ยกตัวอย่างเช่น ถ้าเราอยากเก็บรายชื่อผลไม้ เราก็สามารถใช้ List ในการจัดเก็บได้ ดังนี้

1.แอปเปิ้ล 2.ส้ม 3.กล้วย 4.มะม่วง

ผลไม้ = [แอปเปิ้ล, ส้ม, กล้วย, มะม่วง]

ถ้าอยากเข้าถึงแอปเปิ้ล ต้องเขียนตัวแปร ตามด้วยลำดับ ** แต่อย่าลืมลำดับใน Python เริ่มจาก 0 **

ผลไม้[ลำดับ]

ถ้าอยากเข้าถึงมะม่วง จะต้องเขียนว่าอะไร?

ภาพรวมของ List

“CRUD SORT LEN IN”

```
In [47]:
1 list1 = [1, 2, 3, 4, 5]
2 list2 = ['a', 'b', 'c', 'd']
3 list3 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [48]:
1 type(list1)
```

```
Out[48]: list
```

```
In [49]:
1 type(list2)
```

```
Out[49]: list
```

การสร้าง List

```
In [51]:  1 list4 = [1, 2, 3, 'a', 'b', 'c']

In [52]:  1 list4[0]
Out[52]: 1

In [53]:  1 list4[-1]
Out[53]: 'c'

In [54]:  1 list4[-2]
Out[54]: 'b'
```

การเข้าถึงค่าใน List

```
In [55]:
1 list5 = [1, 2, 3, 'a', 'b', 'c']

In [56]:
1 list5[3:]

Out[56]: ['a', 'b', 'c']

In [57]:
1 list5[:5]

Out[57]: [1, 2, 3, 'a', 'b']

In [58]:
1 list5[3:5]

Out[58]: ['a', 'b']
```

การเข้าถึงค่าใน List

```
In [60]:  1 list6[0] = 0

In [61]:  1 list6
Out[61]: [0, 2, 3, 'a', 'b', 'c']

In [62]:  1 list6[-1] = 'x'

In [63]:  1 list6
Out[63]: [0, 2, 3, 'a', 'b', 'x']
```

การเปลี่ยนแปลงค่าใน List

```
In [64]:  1 list7 = [1, 2, 3, 'a', 'b', 'c']

In [65]:  1 list7.append('d')

In [66]:  1 list7

Out[66]: [1, 2, 3, 'a', 'b', 'c', 'd']

In [67]:  1 list7.append('e')

In [68]:  1 list7
```

การเอาค่ามาต่อใน List ด้วย append()

```
In [69]:
1 list8 = [1, 2, 3]
2 list9 = ['a', 'b', 'c']

In [70]:
1 list8.extend(list9)

In [71]:
1 list8

Out[71]: [1, 2, 3, 'a', 'b', 'c']
```

การเอา List มาต่อกันด้วย extend()

```
In [72]:  1  list10 = [1, 2, 3, 'a', 'b', 'c']

In [73]:  1  list10.insert(1, 'one')

In [74]:  1  list10

Out[74]: [1, 'one', 2, 3, 'a', 'b', 'c']
```

การแทรกข้อมูลใน List ด้วย insert()

```
In [75]:  1 list11 = [1, 2, 3, 'a', 'b', 'c']

In [76]:  1 del list11[2]

In [77]:  1 list11

Out[77]: [1, 2, 'a', 'b', 'c']
```

การลบข้อมูลใน List ด้วยคีย์เวิร์ด del

```
In [78]:  1 list11 = [1, 2, 3, 'a', 'b', 'c']

In [79]:  1 del list11[-2]

In [80]:  1 list11

Out[80]: [1, 2, 3, 'a', 'c']
```

การลบข้อมูลใน List ด้วยคีย์เวิร์ด del

```
In [81]:  1 list12 = [1, 2, 3, 'a', 'b', 'c']

In [82]:  1 list12.remove(1)

In [83]:  1 list12

Out[83]: [2, 3, 'a', 'b', 'c']

In [84]:  1 list12.remove('a')

In [85]:  1 list12
```

ลบข้อมูลใน List ด้วย remove()

```
In [86]:  1 list13 = [1, 2, 3, 'a', 'b', 'c']

In [87]:  1 list13.clear()

In [88]:  1 list13

Out[88]: []
```

ล้างข้อมูลใน List ทั้งหมดด้วย clear()

```
In [89]:  1 list14 = [1, 5, 4, 2, 3]

In [90]:  1 list14.sort()

In [91]:  1 list14

Out[91]: [1, 2, 3, 4, 5]
```

การเรียงลำดับข้อมูลใน List ด้วย sort()

```
In [92]:  1  list14 = [1, 5, 4, 2, 3]

In [93]:  1  list14.sort(reverse = True)

In [94]:  1  list14

Out[94]: [5, 4, 3, 2, 1]
```

การเรียงลำดับข้อมูลจากมากมาหาน้อยใน List ด้วย sort()

```
In [95]:  1  list14 = [1, 5, 4, 2, 3]

In [96]:  1  sorted_list14 = sorted(list14)

In [97]:  1  sorted_list14

Out[97]: [1, 2, 3, 4, 5]
```

การเรียงลำดับข้อมูลใน List ด้วย sorted()

```
In [98]:  1 list14 = [1, 5, 4, 2, 3]

In [99]:  1 sorted_list14 = sorted(list14, reverse = True)

In [100]: 1 sorted_list14

Out[100]: [5, 4, 3, 2, 1]
```

การเรียงลำดับข้อมูลจากมากมาหาน้อยใน List ด้วย sorted()

```
In [101]:  1 list15 = [1, 2, 3, 'a', 'b', 'c']

In [102]:  1 len(list15)

Out[102]: 6

In [103]:  1 list16 = ['a', 'b', 'c', 'd', 'e']

In [104]:  1 len(list16)

Out[104]: 5
```

หาขนาดความยาวตัวอักษรด้วย len()

```
In [105]:  1 list17 = [1, 2, 3, 'a', 'b', 'c']

In [106]:  1 'b' in list17

Out[106]: True

In [107]:  1 4 in list17

Out[107]: False
```

ตรวจว่ามี list นี้ในอีก list หรือไม่ด้วยคีย์เวิร์ด in

โครงสร้างข้อมูลแบบประกอบ

1. ~~List~~ **เชือด กว่าจะจบ**
2. Tuple
3. Dictionary
4. Set

Tuple

tuple เป็นโครงสร้างข้อมูลแบบหนึ่งในภาษาโปรแกรมมิ่ง เปรียบเสมือนกล่องใส่ของที่สามารถเก็บข้อมูลหลายๆ อย่างไว้ด้วยกัน โดยมีลักษณะสำคัญดังนี้

1. ข้อมูลในกล่องจะถูกเรียงลำดับเป็นชุด เช่น (แอปเปิ้ล, ส้ม, กล้วย) หมายถึงมีผลไม้ 3 ชนิดเรียงกันอยู่ในกล่อง
2. ข้อมูลในกล่องอาจเป็นข้อมูลชนิดเดียวกันหรือต่างชนิดก็ได้ เช่น (1, 2, 3) เป็นตัวเลขทั้งหมด ส่วน ("แดง", 10, True) มีทั้งข้อความ ตัวเลข และค่าความจริง
3. *เมื่อสร้าง tuple แล้วจะไม่สามารถเปลี่ยนแปลงข้อมูลภายในได้ เหมือนกล่องที่ปิดผนึกไว้เรียบร้อยแล้ว
4. สามารถเข้าถึงข้อมูลในกล่องได้โดยอ้างอิงลำดับที่ของข้อมูลนั้นเหมือน List เช่น (1,2,3) ถ้าต้องการเลข 2 ก็เรียก tuple[1] โดยเริ่มนับจาก 0

ตัวอย่างการใช้งาน tuple เช่น เก็บข้อมูลวันเดือนปีเกิด เก็บพิกัดตำแหน่งบนแผนที่ หรือเก็บชุดคำตอบจากฟังก์ชันที่ return หลายค่า เป็นต้น

โดยสรุป tuple คือโครงสร้างข้อมูลเหมือนกล่องที่เก็บข้อมูลได้หลายอย่างแบบเรียงลำดับ อ่านข้อมูลได้แต่ไม่สามารถแก้ไข ซึ่งเหมาะสำหรับกรณีที่ต้องเก็บข้อมูลที่เกี่ยวข้องกันเป็นชุดๆ และไม่ต้องการให้มีการเปลี่ยนแปลงข้อมูลดังกล่าว

ภาพรวมของ Tuple

“CR LEN IN”

```
In [108]:
1 tuple1 = (1, 2, 3, 4, 5)
2 tuple2 = ('a', 'b', 'c', 'd')
3 tuple3 = (1, 2, 3, 'a', 'b', 'c')

In [109]:
1 type(tuple1)

Out[109]: tuple

In [110]:
1 type(tuple2)

Out[110]: tuple

In [111]:
1 type(tuple3)

Out[111]: tuple
```

การสร้าง Tuple

```
In [112]:  1 tuple4 = (1, 2, 3, 'a', 'b', 'c')

In [113]:  1 tuple4[0]

Out[113]: 1

In [114]:  1 tuple4[-1]

Out[114]: 'c'

In [115]:  1 tuple4[-2]

Out[115]: 'b'
```

การเข้าถึง Tuple

```
In [116]:  1 tuple5 = (1, 2, 3, 'a', 'b', 'c')

In [117]:  1 tuple5[3:]
Out[117]: ('a', 'b', 'c')

In [118]:  1 tuple5[:5]
Out[118]: (1, 2, 3, 'a', 'b')

In [119]:  1 tuple5[3:5]
Out[119]: ('a', 'b')
```

การเข้าถึง Tuple

```
In [120]:  1 tuple6 = (1, 2, 3, 'a', 'b', 'c')

In [121]:  1 len(tuple6)

Out[121]: 6

In [122]:  1 tuple7 = ('a', 'b', 'c', 'd', 'e')

In [123]:  1 len(tuple7)

Out[123]: 5
```

การหาจำนวนสมาชิกของ Tuple ด้วย len()

```
In [124]:
1 tuple8 = (1, 2, 3, 'a', 'b', 'c')

In [125]:
1 'b' in tuple8

Out[125]: True

In [126]:
1 4 in tuple8

Out[126]: False
```

การตรวจว่ามีค่านี้ใน Tuple หรือไม่ ด้วย in

โครงสร้างข้อมูลแบบประกอบ

1. ~~List~~ **เชือด กว่าจะจบ**
2. ~~Tuple~~ **เชือด**
3. Dictionary
4. Set

Dictionary

โครงสร้างข้อมูลแบบ dictionary เปรียบเสมือนสมุดรายชื่อที่เราใช้อยู่ในชีวิตประจำวัน ซึ่งภายในสมุดรายชื่อนั้นจะประกอบไปด้วยชื่อของคนๆนั้น และข้อมูลอื่นๆที่เกี่ยวข้องกับคนๆนั้น เช่น เบอร์โทรศัพท์ ที่อยู่ อีเมล เป็นต้น

ในทางเดียวกันนั้น dictionary ในการเขียนโปรแกรมก็ทำหน้าที่คล้ายๆกัน โดยจะประกอบไปด้วย key และ value โดยที่ key ก็คือชื่อหรือคำที่ใช้อ้างอิง ส่วน value ก็คือข้อมูลที่เกี่ยวข้องกับ key นั้นๆ

ตัวอย่างเช่น ถ้าเราต้องการจัดเก็บข้อมูลเกี่ยวกับนักเรียน เราอาจจะตั้ง key เป็นชื่อนักเรียน และ value ก็จะเป็นข้อมูลอื่นๆของนักเรียนคนนั้น เช่น

{ "สมชาย": {"เกรดเฉลี่ย": 3.5, "อายุ": 17, "ชั้นปี": 3}, "สมหญิง": {"เกรดเฉลี่ย": 3.2, "อายุ": 16, "ชั้นปี": 2} }

จากตัวอย่างนี้ เราจะเห็นว่า key คือ "สมชาย" และ "สมหญิง" ส่วน value ก็คือข้อมูลที่อยู่ในวงเล็บมุมเหลี่ยมที่เกี่ยวข้องกับนักเรียนคนนั้นๆ

สิ่งสำคัญของ dictionary คือเราสามารถเข้าถึงข้อมูลได้อย่างรวดเร็ว เนื่องจากเราใช้ key ในการค้นหาข้อมูลนั้นๆ โดยไม่จำเป็นต้องเรียงลำดับข้อมูลทั้งหมด ซึ่งช่วยประหยัดเวลาและทรัพยากรในการประมวลผลได้มาก

ภาพรวมของ Dictionary

“CRUD LEN IN”

```
In [127]:
1 dict1 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
2 dict2 = {'fullname' : 'John Doe', 'hobby' : ['coding', 'study']}
```

```
In [128]:
1 type(dict1)

Out[128]: dict
```

```
In [129]:
1 type(dict2)

Out[129]: dict
```

การสร้าง Dictionary

```
In [130]:  1 dict3 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [131]:  1 dict3['firstname']

Out[131]: 'John'

In [132]:  1 dict3['age']

Out[132]: 32
```

การเข้าถึง Dictionary

```
In [133]:  1  dict4 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [134]:  1  dict4['firstname'] = 'Mario'

In [135]:  1  dict4

Out[135]: {'firstname': 'Mario', 'lastname': 'Doe', 'age': 32}
```

การเปลี่ยนค่าใน Dictionary

```
In [136]:
1 dict5 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [137]:
1 dict5['blood_group'] = 'O'

In [138]:
1 dict5

Out[138]: {'firstname': 'John', 'lastname': 'Doe', 'age': 32, 'blood_group': 'O'}
```

การเพิ่มค่าใน Dictionary

```
In [142]:
1 dict7 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [143]:
1 del dict7['lastname']

In [144]:
1 dict7

Out[144]: {'firstname': 'John', 'age': 32}
```

การลบข้อมูลใน Dictionary ด้วย del

```
In [145]:
1 dict8 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [146]:
1 dict8.clear()

In [147]:
1 dict8

Out[147]: {}
```

การล้างข้อมูลใน Dictionary ด้วย clear()

```
In [148]:  1 dict9 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [149]:  1 len(dict9)

Out[149]: 3

In [150]:  1 len(dict9.keys())

Out[150]: 3

In [151]:  1 len(dict9.values())

Out[151]: 3
```

การหาจำนวนสมาชิกใน Dictionary ด้วย len()

```
In [152]:
1 dict10 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [153]:
1 'firstname' in dict10

Out[153]: True

In [154]:
1 'firstname' in dict10.keys()

Out[154]: True

In [155]:
1 'John' in dict10.values()

Out[155]: True
```

การตรวจว่ามีค่านี้ใน Dictionary หรือไม่ ด้วย in

```
In [156]:  1 dict10 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}

In [157]:  1 'weight' in dict10
Out[157]: False

In [158]:  1 'weight' in dict10.keys()
Out[158]: False

In [159]:  1 70 in dict10.values()
Out[159]: False
```

การตรวจว่ามีค่านี้ใน Dictionary หรือไม่ ด้วย in

โครงสร้างข้อมูลแบบประกอบ

~~1. List~~ **เชือด กว่าจะจบ**

~~2. Tuple~~ **เชือด**

~~3. Dictionary~~ **เชือด**

4. Set

Set

โครงสร้างข้อมูลแบบ set เปรียบเสมือนกล่องที่ใช้สำหรับใส่ข้อมูลต่างๆ โดยที่ข้อมูลแต่ละชิ้นจะไม่ซ้ำกันและไม่มีการจัดลำดับ

จินตนาการว่าคุณมีกล่องกระดาษ และใส่ลูกบอลลงไปในกล่องนั้น แต่ละลูกบอลมีสีและขนาดที่แตกต่างกัน เมื่อคุณใส่ลูกบอลลงไปในกล่อง ถ้ามีลูกบอลสีหรือขนาดเดิมอยู่แล้ว กล่องจะไม่รับลูกบอลเม็ดใหม่นั้นเข้าไป เพราะภายในกล่องจะไม่มีลูกบอลที่ซ้ำกันสองเม็ด

นั่นคือหลักการทำงานของ set ข้อมูลในนั้นจะไม่ซ้ำกัน และไม่มีการจัดเรียงลำดับ เช่น ถ้าคุณมี set ของตัวเลขดังนี้: {1, 2, 3, 4, 5} และคุณพยายามเพิ่มตัวเลข 3 เข้าไปอีก set ก็จะยังคงมีสมาชิกเพียง 5 ตัว คือ 1, 2, 3, 4, 5 เท่านั้น

คุณสมบัติพิเศษของ set คือการค้นหาข้อมูลทำได้รวดเร็วมาก เนื่องจากมีกระบวนการตรวจสอบว่าข้อมูลนั้นมีในสมาชิกหรือไม่โดยใช้เวลาน้อยมาก

ดังนั้น set จึงเหมาะที่จะใช้สำหรับการจัดเก็บข้อมูลที่ไม่ซ้ำกัน และต้องการความรวดเร็วในการค้นหาข้อมูล เช่น รายการคำพิเศษในพจนานุกรม เป็นต้น

ภาพรวมของ Set

“CRUD LEN IN”

```
In [160]:
1 set1 = {1, 2, 3, 4, 5}
2 set2 = {1, 2, 3, 'a', 'b', 'c'}

In [161]:
1 type(set1)

Out[161]: set

In [162]:
1 type(set2)

Out[162]: set
```

การสร้าง Set

```
In [163]:
1 set3 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [164]:
1 for x in set3:
2     print(x)
```

```
d
b
c
e
a
```

การเข้าถึงข้อมูลใน Set

```
In [168]:
1 set5 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [169]:
1 set5.update({1, 2})
```

```
In [170]:
1 set5
```

```
Out[170]: {1, 2, 'a', 'b', 'c', 'd', 'e'}
```

การเพิ่มค่าใน Set ด้วย update()

```
In [171]:  1  set6 = {'a', 'b', 'c', 'd', 'e'}

In [172]:  1  set6.remove('a')

In [173]:  1  set6

Out[173]: {'b', 'c', 'd', 'e'}

In [174]:  1  set6.remove('c')

In [175]:  1  set6
```

การลบค่าใน Set ด้วย remove

```
In [176]:  1  set7 = {'a', 'b', 'c', 'd', 'e'}

In [177]:  1  set7.clear()

In [178]:  1  set7

Out[178]: set()
```

การล้างค่าใน Set ด้วย clear()

```
In [179]:
1 set8 = {1, 2, 3, 4, 5}
```

```
In [180]:
1 len(set8)
```

```
Out[180]: 5
```

การหาจำนวนสมาชิกใน Set ด้วย len()

```
In [181]:
1 set9 = {1, 2, 3, 4, 5}

In [182]:
1 1 in set9

Out[182]: True

In [183]:
1 'a' in set9

Out[183]: False
```

การตรวจว่ามีค่านี้ใน Set หรือไม่

```
In [184]:
1 setA = {1, 2, 3, 4, 5}
2 setB = {3, 4, 5, 6, 7}
```

```
In [185]:
1 setA | setB
```

```
Out[185]: {1, 2, 3, 4, 5, 6, 7}
```

การ Union ใน Set

```
[186]:
1  setA = {1, 2, 3, 4, 5}
2  setB = {3, 4, 5, 6, 7}
```

```
[187]:
1  setA & setB
```

```
Out[187]: {3, 4, 5}
```

การ Intersection ใน Set

```
In [188]:
1 setA = {1, 2, 3, 4, 5}
2 setB = {3, 4, 5, 6, 7}

In [189]:
1 setA - setB

Out[189]: {1, 2}

In [190]:
1 setB - setA

Out[190]: {6, 7}
```

การหาผลต่างใน Set

```
In [191]:
1 setA = {1, 2, 3, 4, 5}
2 setB = {3, 4, 5, 6, 7}
```

```
In [192]:
1 setA ^ setB
```

```
Out[192]: {1, 2, 6, 7}
```

การ Symmetric Difference ใน Set

โครงสร้างข้อมูลแบบประกอบ

~~1. List~~ **เชือด กว่าจะจบ**

~~2. Tuple~~ **เชือด**

~~3. Dictionary~~ **เชือด**

~~4. Set~~ **เชือด**

เนื้อหา

~~1. โครงสร้างข้อมูลพื้นฐาน (Basic Data Structure)~~ เชือด

~~2. โครงสร้างข้อมูลแบบประกอบ (Composite Data Structure) โครงสร้างข้อมูลพื้นฐานหลายๆตัวประกอบกัน)~~ เชือด

	List	Tuple	Dictionary	Set
Create	listA=[] listB=[1, 2, 3]	tupleA=() tupleB=(1, 2, 3)	dictA={} dictB={'key':value}	setA=set() setB={1, 2, 3}
Read	listB[start] listB[start:end]	tupleB[start] tupleB[start:end]	dictB['key']	for b in setB: print(b)

สรุปการสร้างและการเข้าถึง

		List	Tuple	Dictionary	Set
Update	Replace	✓	✗	✓	✗
	append	✓		✗	✗
	insert	✓		✗	✗
	extend	✓		✗	✗
	add	✗		✗	✓
	update	✗		✓	✓

สรุปคำสั่งที่สามารถใช้ได้

		List	Tuple	Dictionary	Set
Delete	del	✓	✗	✓	✗
	remove	✓		✗	✓
	clear	✓		✓	✓
len		`len(listB)`	`len(tupleB)`	`len(dictB)`	`len(setB)`
in		`'a' in listB`	`'a' in tupleB`	`'keyA' in dictB`	`'a' in setB`

สรุปคำสั่งที่สามารถใช้ได้

สรุปคำสั่งใน Set

Set
Union (|)
Intersection (&)
Difference (-)
Symmetric Difference (^)