



**Week 2 :**

# **Introduction to Python**

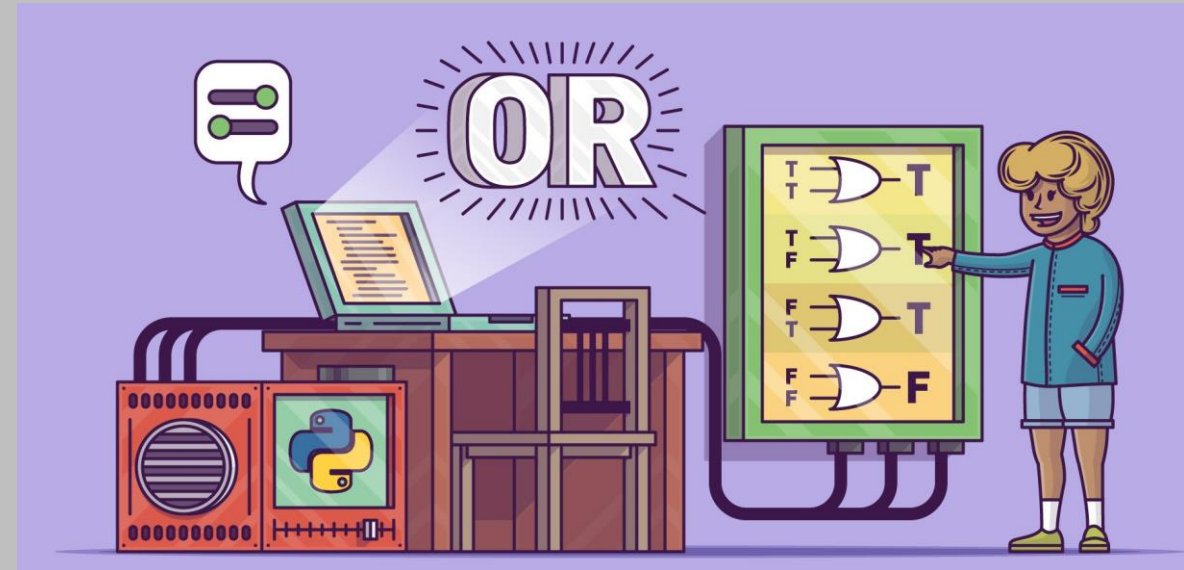
**Natakorn Pramayan, Ph.D.**



# Content

- **Booleans**
- **Operator**
- **Collections (Arrays)**
- **Dictionaries**

# Python Booleans



Booleans ในภาษา Python หมายถึงค่า 1 ใน 2 ค่า คือ True หรือ False  
สามารถใช้สำหรับการเปรียบเทียบค่าหรือตรวจสอบเงื่อนไขว่าเป็นจริงหรือเท็จได้

### ตัวอย่าง

3	<code>print(69 == 79)</code>	→	False
4	<code>print(35 &gt; 12)</code>	→	True
5	<code>print(69 == 79)</code>	→	False
6	<code>print(99 == 99)</code>	→	True
7			

ผลลัพธ์

## แบบฝึกหัดที่ 2.1

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมรับค่าตัวเลข 2 จำนวน แล้วนำมาเปรียบเทียบว่ามีค่าเท่ากันหรือไม่

ตัวอย่าง

มีค่าไม่เท่ากัน

```
Input First Number  
15  
Input Second Number  
20  
15 = 20 : False
```

มีค่าเท่ากัน

```
Input First Number  
10  
Input Second Number  
10  
10 = 10 : True
```

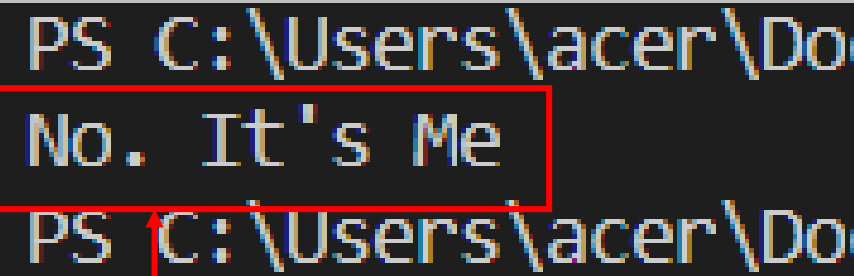


การนำ Boolean ไปใช้ร่วมกับคำสั่ง if...else loop ที่ทำงานเมื่อเงื่อนไขเป็นจริง  
Boolean จึงมีความสำคัญมากในการเขียนโปรแกรม

### ตัวอย่าง

```
3 Me = 24
4 myFriend = 30
5 if Me > myFriend :
6 | print("I am older than you !!")
7 else :
8 | print("No. It's Me")
```

การใช้คำสั่ง if ในการตั้งเงื่อนไขการแสดงความ



```
PS C:\Users\acer\Do
No. It's Me
PS C:\Users\acer\Do
```

ผลลัพธ์ที่ปรากฏบนจอ

# การหาค่าความจริงโดยใช้คำสั่ง bool

คำสั่ง bool( ) สามารถใช้ในการตรวจสอบค่าความเป็นจริงของตัวแปรใด ๆ

ในภาษา Python ได้ ซึ่งจะ return ค่ากลับมาเป็น Boolean ( True หรือ False )

## ตัวอย่าง

```
3 print(bool("Python"))  
4 print(bool(1991))
```

```
True  
True
```

การใช้คำสั่ง bool() ในการตรวจสอบค่าความเป็นจริง

Return ค่ากลับมาเป็น true

การใช้คำสั่ง bool( ) ตรวจสอบค่าใด ๆ ส่วนใหญ่จะ return ค่าเป็น True ถ้าค่านั้น ๆ  
มีข้อมูลอยู่

### ตัวอย่าง

```
3  #ตรวจสอบค่าอักษร (String) ให้ค่าเป็น True ยกเว้นว่าอักษรนั้นว่างเปล่า (empty)
4  print(bool("Hello"))
5  print(bool(""))
```

True  
False

```
7  #ตรวจสอบค่าตัวเลข (Number) ให้ค่าเป็น True ยกเว้นตัวเลขเป็น 0
8  print(bool(15))
9  print(bool(0))
```

True  
False



# **Python**

# **Operators**

Operators หรือตัวดำเนินการ ใช้สำหรับการดำเนินการกับค่าหรือตัวแปร ในภาษา Python แบ่งตัวดำเนินการออกเป็น 7 กลุ่ม

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Identity Operators
6. Membership Operators
7. Bitwise Operators

# Arithmetic Operators ใช้กับค่าที่เป็นตัวเลข เพื่อดำเนินการทางคณิตศาสตร์

Operators	Name	Example
+	บวก	$x + y$
-	ลบ	$x - y$
*	คูณ	$x * y$
/	หาร (มีทศนิยม)	$x / y$
%	Modulus	$x \% y$
**	ยกกำลัง	$x ** y$
//	หาร (ไม่มีทศนิยม)	$x // y$

# Assignment Operators เป็นการมอบค่า (Assign) ให้กับตัวแปร

Operators	Example
=	x = 5
+=	x +=3   x = x + 3
-=	x -=3   x = x - 3
*=	x *=3   x = x * 3
/=	x /=3   x = x / 3
%=	x %=3   x = x % 3
//=	x //=3   x = x // 3
**=	x **=3   x = x ** 3
&=	x &=3   x = x & 3
=	x  =3   x = x   3
^=	x ^=3   x = x ^ 3
>>=	x >>=3   x = x >> 3
<<=	x <<=3   x = x << 3

## Comparison Operators ใช้สำหรับเปรียบเทียบค่า 2 ค่า

Operators	Name	Example
==	เท่ากัน	$x == y$
!=	ไม่เท่ากัน	$x != y$
>	มากกว่า	$x > y$
<	น้อยกว่า	$x < y$
>=	มากกว่าหรือเท่ากับ	$x >= y$
<=	น้อยกว่าหรือเท่ากับ	$x <= y$

ข้อสังเกต  $x = 5$  เป็นการ Assign ค่า 5 ให้กับ  $x$  ส่วน  $x == 5$  เป็นการเปรียบเทียบว่า  $x$  มีค่าเท่ากับ 5 หรือไม่

## Logical Operators ใช้สำหรับรวมคำสั่งแบบมีเงื่อนไข

Operators	Description	Example
and	ให้ค่า True เมื่อคำสั่ง (statement) ทั้งคู่เป็นจริง	$x < 5$ and $x < 10$
or	ให้ค่าเป็น True ถ้าคำสั่งใด คำสั่งหนึ่งเป็นจริง	$x < 5$ or $x < 4$
not	ให้ค่าตรงกันข้าม เช่น ผลลัพธ์เป็นจริง จะให้ค่าเป็นเท็จ	not( $x < 5$ and $x < 10$ )

**Identity Operators** ใช้สำหรับเปรียบเทียบวัตถุ (Object) เป็นการเปรียบเทียบว่าเป็น object เดียวกันหรือไม่

Operators	Description	Example
is	ให้ค่า True เมื่อตัวแปรเป็น object เดียวกัน	x is y
is not	ให้ค่า True เมื่อตัวแปร ไม่ใช่ object เดียวกัน	x is not y

## ตัวอย่าง

```
3 x = ["ComED", "KKU"] # x เป็น object
4 y = ["ComED", "KKU"] # y เป็น object
5 z = x # สร้าง object z โดยสืบทอดคุณสมบัติ object x
6
7 print(x is z)
8 print(x is y)
```

True  
False

`print(x is z)` # return True เพราะ z และ x เป็น object เดียวกัน

`print(x is y)` # return False เพราะ x และ y ไม่ใช่ object เดียวกัน แม้จะมีค่าเหมือนกัน



## Membership Operators ใช้สำหรับตรวจสอบค่าที่อยู่วัตถุ

Operators	Description	Example
in	ให้ค่า True ถ้าค่าที่ระบุมีอยู่ใน object	x in y
not in	ให้ค่า True ถ้าค่าที่ระบุไม่มีอยู่ใน object	x not in y

## ตัวอย่าง

```
3 x = ["ComED", "KKU"]  
4 print("KKU" in x)
```

```
True
```

return True เพราะค่าที่ระบุ "KKU" มีอยู่ใน object x

## Bitwise Operators ใช้สำหรับเปรียบเทียบค่าของตัวเลข (Binary)

Operators	Name	Description
&	AND	ให้ค่า bit เป็น 1 ถ้า bit ทั้งคู่เป็น 1
	OR	ให้ค่า bit เป็น 1 ถ้า bit ใด bit หนึ่งเป็น 1
^	XOR	ให้ค่า bit เป็น 1 ถ้า bit ใด bit หนึ่งเป็น 1 และ อีก bit เป็น 0
~	NOT	กลับ bit ทั้งหมด เช่น จาก 0 เป็น 1
<<	Zero left shift	เติม 0 ด้านขวา และตัด bit ที่เป็น 0 ด้านซ้าย
>>	Signed right shift	เติม 0 ด้านซ้าย และตัด bit ด้านขวา

# ตัวอย่าง

```
3   a = 60           # 60 = 0011 1100
4   b = 13           # 13 = 0000 1101
5   c = 0
6
7   c = a & b        # 12 = 0000 1100
8   print(c)
9
10  c = a | b        # 61 = 0011 1101
11  print(c)
12
13  c = a ^ b        # 49 = 0011 0001
14  print(c)
15
16  c = ~a           # -61 = 1100 0011
17  print(c)
18
19  c = a << 2        # 240 = 1111 0000
20  print(c)
21
22  c = a >> 2        # 15 = 0000 1111
23  print(c)
24
```

## แบบฝึกหัดที่ 2.2

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมแปลงจำนวนวันที่รับเข้ามาให้แสดงในหน่วย ชั่วโมง นาที และวินาที ตามลำดับ

ตัวอย่าง

```
Day Converter Program
Input number of Days --> 2
2 Days --> Hour 48 Hours
2 Days --> Minutes 2880 Minutes
2 Days --> Seconds 172800 Seconds
```



# **Python**

# **Collections (Arrays)**

**Collection** เป็นประเภทข้อมูลที่เกิดจากการรวมกลุ่มของข้อมูลที่มีความสัมพันธ์กัน อาจเป็นข้อมูลประเภท (Type) เดียวกัน หรือข้อมูลคนละประเภท ซึ่งในภาษา Python แบ่ง Collection ออกเป็น 4 ประเภท ได้แก่

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

## 1. List

เป็นข้อมูลที่มีการเรียงลำดับ สามารถเพิ่ม ลบ แก้ไขข้อมูลได้ มีข้อมูลซ้ำกันได้ ในภาษา Python สามารถสร้าง List ได้ด้วยการใช้เครื่องหมาย [ ]

### ตัวอย่าง

```
3  thislist = ["Com", "ED", "KKU",99]
4  print(thislist)
```

```
[ 'Com', 'ED', 'KKU', 99]
```

← ผลลัพธ์



## การเข้าถึงข้อมูลใน List

### ตัวอย่าง

ตำแหน่ง    ตำแหน่ง    ตำแหน่ง  
          0            1            2



```
4  thislist = ["Com", "ED", "KKU"]  
5  print(thislist[1])
```

ED

ผลลัพธ์แสดงค่าตำแหน่งที่ 1

หมายเหตุ การเรียงลำดับ (index) ในทางคอมพิวเตอร์เริ่มต้นจาก 0

## ตัวอย่าง

```
4 thislist = ["Com", "ED", "KKU"]  
5 print(thislist[-1])
```

```
PS C:\Users\an...> python3  
KKU  
PS C:\Users\an...
```

← ผลลัพธ์แสดงค่าตำแหน่งที่ -1

การระบุ index เป็นค่าติดลบ (Negative Indexing) เป็นการเข้าถึงข้อมูลจากด้านหลังของ Collection  
thislist[-1] หมายถึง ข้อมูลสุดท้าย , thislist[-2] หมายถึง ข้อมูลรองสุดท้าย

การระบุ **Range of Indexes** เป็นการระบุจุดเริ่มต้นและจุดสิ้นสุดของข้อมูลที่จะเข้าถึงได้

### ตัวอย่าง

```
4 thislist = ["Com", "ED", "KKU", 99]  
5 print(thislist[1:3])  
6
```

เป็นการระบุจุดเริ่มต้นที่ index 1 (นับ)  
และสิ้นสุดที่ index 3 (ไม่นับ)

```
['ED', 'KKU']
```

แสดงผลออกทางหน้าจอเป็นค่า index 1 และ 2

หากไม่ระบุค่าเริ่มต้น Range จะเริ่มจากข้อมูลตัวแรก

### ตัวอย่าง

```
4  thislist = ["Com", "ED", "KKU",99]
5  print(thislist[:3])
6
```

```
PS C:\Users\user\Documents> python3
['Com', 'ED', 'KKU']
PS C:\Users\user\Documents>
```

หากไม่ระบุค่าสิ้นสุด Range เข้าถึงข้อมูลจนถึงข้อมูลตัวสุดท้าย

ตัวอย่าง

```
3  
4  thislist = ["Com", "ED", "KKU",99]  
5  print(thislist[1:])
```

```
PS C:\Users\acer\Doc  
[ 'ED', 'KKU', 99]  
PS C:\Users\acer\Doc
```

## การเปลี่ยนแปลงข้อมูลใน List

สามารถทำได้โดยการระบุ index ของข้อมูลที่ต้องการเปลี่ยนแปลง

### ตัวอย่าง

```
3  
4 thislist = ["Com", "ED", "KKU",99]  
5 thislist[0] = "ComED"  
6 print(thislist)  
7
```

← เปลี่ยนแปลงค่า index 0 จาก Com เป็น ComED

```
PS C:\Users\user\Documents> python3  
['ComED', 'ED', 'KKU', 99]  
PS C:\Users\user\Documents>
```

## การเพิ่มข้อมูลใน List

ใช้คำสั่ง `append( )` เพื่อเพิ่มข้อมูลไปที่ท้ายของ list

### ตัวอย่าง

```
4  thislist = ["Com", "ED", "KKU",99]
5  thislist.append("Hello")
6  print(thislist)
```

ใช้คำสั่ง `append()` เพิ่ม Hello

```
PS C:\Users\acer\Documents\PS Codes>
['Com', 'ED', 'KKU', 99, 'Hello']
PS C:\Users\acer\Documents\PS Codes>
```

แสดงผลออกทางหน้าจอ

ใช้คำสั่ง insert( ) เพื่อเพิ่มข้อมูลไปยังตำแหน่งที่กำหนด

### ตัวอย่าง

```
3
4  thislist = ["Com", "ED", "KKU",99]
5  thislist.insert(1,"Hello")
6  print(thislist)
7
```

ใช้คำสั่ง insert() เพิ่ม Hello  
ไปยัง index 1

```
[ 'Com', 'Hello', 'ED', 'KKU', 99]
```



## การลบข้อมูลใน List

ใช้คำสั่ง `remove( )` เพื่อลบข้อมูลที่ระบุ

ตัวอย่าง

```
3  
4  thislist = ["Com", "ED", "KKU",99]  
5  thislist.remove("Com")  
6  print(thislist)  
7
```

```
PS C:\Users\acer\Doc  
['ED', 'KKU', 99]  
PS C:\Users\acer\Doc
```

ใช้คำสั่ง pop( ) เพื่อลบข้อมูลที่ระบุ Index หรือข้อมูลสุดท้าย (หากไม่ระบุ Index)

### ตัวอย่าง

```
4  thislist = ["Com", "ED", "KKU",99]
5  thislist.pop()
6  print(thislist)
```

```
['Com', 'ED', 'KKU']
```

```
D:\C:\Users\user\Documents
```

ใช้คำสั่ง `del( )` เพื่อลบข้อมูลที่ระบุ Index หรือลบ list ทั้งหมด

### ตัวอย่าง

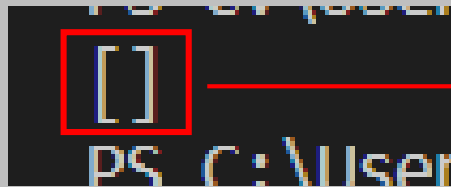
```
4  thislist = ["Com", "ED", "KKU",99]
5  del thislist[3] # ["Com" , "ED" , "KKU"]
6  del thislist # ลบ list ทั้งหมด
```

ใช้คำสั่ง `clear( )` เพื่อให้ list เป็นค่าว่าง (ลบข้อมูลทั้งหมดใน list)

### ตัวอย่าง

```
4  thislist = ["Com", "ED", "KKU",99]
5  thislist.clear()
6  print(thislist)
```

```
PS C:\User
```



ผลลัพธ์แสดงเป็นค่าว่าง

## 2. Tuple

เป็น Collection ที่มีการเรียงลำดับข้อมูล และไม่สามารถแก้ไขข้อมูลได้ สร้างโดยใช้เครื่องหมาย ()

### ตัวอย่าง

```
2  
3  thistuple = ("Com", "ED", "KKU")  
4  print(thistuple)
```

```
Python 3.8.5 Shell  
( 'Com', 'ED', 'KKU')  
Python 3.8.5 Shell
```

## การเข้าถึงข้อมูลใน Tuple

สามารถเข้าถึงข้อมูลใน Tuple ได้ด้วยการระบุ index ในเครื่องหมาย [ ]

### ตัวอย่าง

```
3  thistuple = ("Com", "ED", "KKU")
4  print(thistuple[1])
5  thistuple = ("Com", "ED", "KKU")
6  print(thistuple[-1])
7  thistuple = ("Com", "ED", "KKU")
8  print(thistuple[0:1])
```

index ที่ต้องการเข้าถึงข้อมูล

```
PS C:\Users\ad...
ED
KKU
('Com', )
PS C:\Users\ad...
```

ผลลัพธ์แสดงออกทางหน้าจอ

## การเปลี่ยนแปลงข้อมูลใน Tuple

เมื่อ Tuple ถูกสร้างขึ้น จะไม่สามารถเปลี่ยนแปลงข้อมูลใน Tuple ได้ แต่สามารถใช้การแปลง Tuple ให้เป็น List เพื่อเปลี่ยนแปลงข้อมูล และแปลงกลับเป็น Tuple อีกครั้งได้

### ตัวอย่าง

```
3 x = ("Com", "ED", "KKU") # ("Com" , "ED" , "KKU")
4 y = list(x) # ["Com" , "ED" , "KKU"]
5 y[0] = "ComED" # ["ComED" , "Com" , "ED" , "KKU"]
6 x = tuple(y) # ("ComED" , "Com" , "ED" , "KKU")
```

## การลบข้อมูลใน Tuple

เมื่อ Tuple ถูกสร้างขึ้น จะไม่ลบข้อมูลใน Tuple ได้ แต่สามารถใช้คำสั่ง `del( )` ในการลบ Tuple ทั้งหมดได้

### ตัวอย่าง

```
2
3   x = ("Com", "ED", "KKU") # ("Com" , "ED" , "KKU")
4   del x # deleted completely
5   print(x) # Name Error : 'x' is not defined
6
```

หมายเหตุ : เมื่อ Tuple ถูกสร้างขึ้น จะไม่สามารถเพิ่มข้อมูลใน Tuple ได้



### 3. Set

เป็น Collection ที่ไม่มีการจัดเรียงข้อมูลและไม่มี index สร้างโดยใช้เครื่องหมาย { }  
มีข้อมูลซ้ำกันไม่ได้

#### ตัวอย่าง

```
3 thisset = {"Com", "ED", "KKU", "KKU"}  
4 print(thisset)
```

ข้อมูลที่ซ้ำกัน

```
{'KKU', 'ED', 'Com'}
```

ผลลัพธ์ไม่แสดงข้อมูลที่ซ้ำกัน

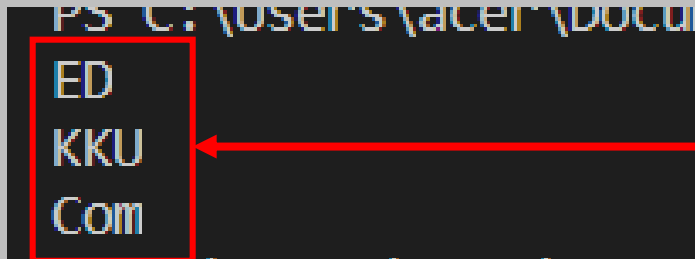
เนื่องจาก Set เป็น Collection ที่ไม่มีการจัดเรียงข้อมูล ดังนั้น ข้อมูลที่แสดงออกทางหน้าจอจากการใช้คำสั่ง print( ) จะถูกสลับตำแหน่งไปเรื่อย ๆ

## การเข้าถึงข้อมูลใน Set

ไม่สามารถเข้าถึงข้อมูลใน Set ได้ เนื่องจาก Set ไม่มี Index ตำแหน่งของข้อมูลจะถูกสลับตลอดเวลา แต่สามารถใช้การ Loop ในการแสดงข้อมูลใน Set ได้

### ตัวอย่าง

```
3 thisset = {"Com", "ED", "KKU"}
4 for x in thisset: # ใช้ตัวแปร x ในการวน loop (การทำงานซ้ำ ๆ)
5     print(x) # ข้อมูลใน Set จะถูกเก็บไว้ในตัวแปร x และแสดงออกมาทางหน้าจอทีละชุด
```



```
PS C:\Users\acer\Documents>
ED
KKU
Com
```

หน้าจอแสดงผลออกมาทีละชุด

\*เมื่อ Set ถูกสร้างขึ้น จะไม่สามารถเปลี่ยนแปลงข้อมูลใน Set ได้

## การเพิ่มข้อมูลใน Set

ใช้คำสั่ง `add( )` ในการเพิ่มข้อมูลใน Set หรือใช้คำสั่ง `update( )` เพื่อเพิ่มข้อมูลที่ละ

หลายตัว

### ตัวอย่าง

```
3 thisset = {"Com", "ED", "KKU"}
4 thisset.add("Hello")
5 print(thisset)
6 thisset.update(["I", "Am", "The Flash"])
7 print(thisset)
```

ผลลัพธ์จากการใช้คำสั่ง `add( )`

`{'Com', 'ED', 'KKU', 'Hello'}`

`{'The Flash', 'KKU', 'Com', 'Am', 'ED', 'Hello', 'I'}`

ผลลัพธ์จากการใช้คำสั่ง `update( )`

## การลบข้อมูลใน Set

ใช้คำสั่ง `remove( )` หรือ `discard( )` ในการลบข้อมูลใน Set

ใช้ `del( )` ในการลบ Set ทั้งหมด

ใช้ `clear( )` ในการทำให้ Set เป็นค่าว่าง

### ตัวอย่าง

```
3  thisset = {"Com", "ED", "KKU"} # {"Com" , "ED" , "KKU"}
4  thisset.remove("Com") # {"ED" , "KKU"}
5  thisset.discard("Com") # {"ED" , "KKU"}
6  thisset.clear() # {}
7  del thisset # Deleted
```

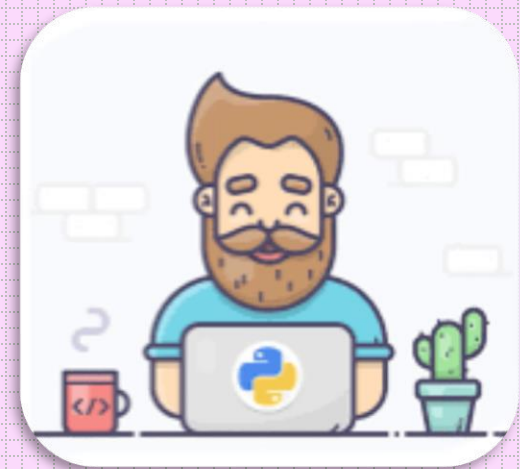
หมายเหตุ หากใช้คำสั่ง `remove( )` ลบข้อมูลที่ไม่มีอยู่ใน Set จะเกิด Error แต่ถ้าใช้ `discard( )` จะไม่แสดง Error

## แบบฝึกหัดที่ 2.3

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมหยิบสินค้าใส่ตะกร้าโดยให้ได้ 5 อย่างจากนั้นแสดงผลทางหน้าจอ

ตัวอย่าง

```
+++++
โปรแกรมหยิบสินค้าใส่ตะกร้า
+++++
หยิบสินค้า ชั้นที่ 1      สบู่
หยิบสินค้า ชั้นที่ 2      ยาสีฟัน
หยิบสินค้า ชั้นที่ 3      ลูกอม
หยิบสินค้า ชั้นที่ 4      น้ำเปล่า
หยิบสินค้า ชั้นที่ 5      มาม่า
สินค้าที่หยิบใส่ตะกร้ามีดังนี้
1. สบู่
2. ยาสีฟัน
3. ลูกอม
4. น้ำเปล่า
5. มาม่า
```

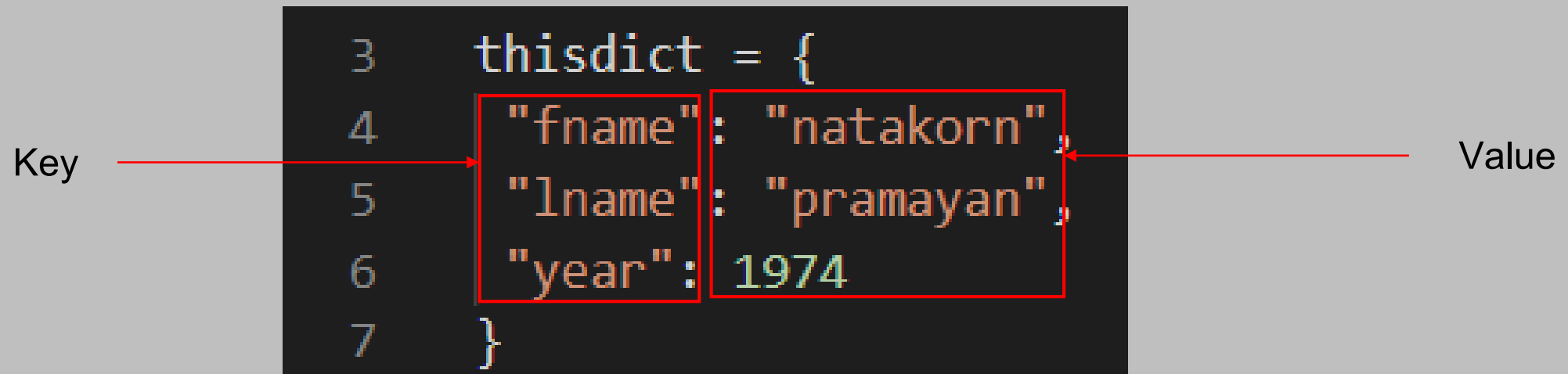


# **Python**

# **Dictionaries**

**Dictionaries** เป็น Collection ที่ไม่มีการเรียงลำดับข้อมูล แต่สามารถเข้าถึงข้อมูลได้ผ่าน Key และสามารถเปลี่ยนแปลงข้อมูลได้ สร้างโดยใช้เครื่องหมาย { } และระบุ Key พร้อม Value

### ตัวอย่าง



```
3  thisdict = {  
4    "fname": "natakorn",  
5    "lname": "pramayan",  
6    "year": 1974  
7  }
```

The diagram illustrates a Python dictionary. A red box highlights the entire dictionary structure. Two red arrows point from the labels 'Key' and 'Value' to the corresponding parts of the first entry: 'fname' is the key and 'natakorn' is the value.

Key	Value
"fname"	"natakorn"
"lname"	"pramayan"
"year"	1974

## การเข้าถึงข้อมูลใน Dictionary

สามารถเข้าถึงข้อมูลใน Dictionary ได้โดยการระบุ Key ภายในเครื่องหมาย [ ]

### ตัวอย่าง

```
3  thisdict = {  
4      "fname": "natakorn",  
5      "lname": "pramayan",  
6      "year": 1974  
7  }  
8  x = thisdict["fname"]  
9  print(x)
```

การเข้าถึงข้อมูลจาก Key "fname"

natakorn

แสดงผลค่า Value ออกทางหน้าจอ



# การเปลี่ยนแปลงข้อมูลใน Dictionary

สามารถเปลี่ยนแปลงข้อมูลใน Dictionary ได้โดยการระบุ Key และค่าที่ต้องการเปลี่ยนแปลง

## ตัวอย่าง

```
3  thisdict = {  
4      "fname": "natakorn",  
5      "lname": "pramayan",  
6      "year": 1974  
7  }  
8  thisdict["year"] = 2020  
9  print(thisdict)
```

ข้อมูลที่มี key เป็นคำว่า year  
ถูกเปลี่ยนจาก 1974 เป็น 2020

```
{'fname': 'natakorn', 'lname': 'pramayan', 'year': 2020}
```

# การเพิ่มข้อมูลใน Dictionary

ทำได้โดยการสร้าง Key ใหม่และ Assign ค่าให้กับ Key นั้น

## ตัวอย่าง

```
2  
3  thisdict = {  
4      "fname": "natakorn",  
5      "lname": "pramayan",  
6      "year": 1974  
7  }  
8  thisdict["nation"] = "Thai"  
9  print(thisdict)
```

สร้าง Key ชื่อว่า "nation"  
Assign ค่า "Thai"

```
{'fname': 'natakorn', 'lname': 'pramayan', 'year': 1974, 'nation': 'Thai'}
```

ผลลัพธ์แสดงออกทางหน้าจอสำหรับค่าที่เพิ่มเข้ามา

## การลบข้อมูลใน Dictionary

สามารถใช้ `pop( )` เพื่อลบข้อมูลที่ระบุ Key ได้ หรือใช้ `popitem( )` เพื่อลบข้อมูลที่ถูกรับเพิ่มล่าสุด หรือใช้ `del( )` ในการลบข้อมูลที่ระบุ Key และใช้ `del( )` ในการลบ Dictionary ทั้งหมดได้เช่นกัน

### ตัวอย่าง

```
3  thisdict = {  
4      "fname": "natakorn",  
5      "lname": "pramayan",  
6      "year": 1974,  
7      "nation": "Thai"  
8  }  
9  thisdict.pop("year") # ลบข้อมูลที่มี key เป็น year  
10 thisdict.popitem() # ลบข้อมูลเพิ่มล่าสุด ลบ nation  
11 del thisdict # ลบตัวแปร thisdict  
12
```

หมายเหตุ : ใน Python ที่ Version เก่ากว่า 3.7 จะเป็นการลบข้อมูลแบบสลับแทน

# อัตราค่าธรรมเนียมผ่านทางบนทางหลวงพิเศษหมายเลข 7

## ช่วงกรุงเทพฯ-ชลบุรี-พัทยา

ลาดกระบัง		รถ 4 ล้อ  / รถ 6 ล้อ  / รถมากกว่า 6 ล้อ 									
25 / 45 / 60	บางบ่อ										
30 / 45 / 70	10 / 15 / 20										
45 / 75 / 110	25 / 45 / 65	บางปะกง									
		15 / 25 / 40									
55 / 90 / 130	35 / 55 / 80	25 / 40 / 55	พนัสนิคม								
		10 / 15 / 20		บ้านบึง							
60 / 100 / 140	40 / 65 / 95	30 / 50 / 70	10 / 20 / 30	10 / 15 / 20							
					บางพระ						
80 / 130 / 190	60 / 100 / 145	50 / 80 / 120	30 / 50 / 75	25 / 40 / 60	20 / 30 / 45						
						หนองขาม					
100 / 160 / 235	80 / 130 / 190	70 / 115 / 165	50 / 85 / 120	45 / 70 / 105	40 / 60 / 90	15 / 30 / 40					
							โป่ง				
105 / 170 / 245	85 / 135 / 195	75 / 120 / 170	55 / 90 / 130	45 / 75 / 110	40 / 65 / 100	20 / 35 / 50	10 / 15 / 20	พัทยา			



## แบบฝึกหัดที่ 2.4

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมคำนวณค่าผ่านทางมอเตอร์เวย์ ดังนี้

1. ลาดกระบัง → บางป่อ
2. ลาดกระบัง → บางประกง
3. ลาดกระบัง → พน์สนิคม
4. ลาดกระบัง → บ้านบึง
5. ลาดกระบัง → บางพระ



## โปรแกรมคำนวณค่าผ่านทางมอเตอร์เวย์

---

รถยนต์ 4 ล้อ                      กด 1

รถยนต์ 6 ล้อ                      กด 2

รถยนต์มากกว่า 6 ล้อ      กด 3

เลือกประเภทยานพาหนะ : 2

ค่าบริการรถยนต์ 6 ล้อ

ลาดกระบัง-->บางบ่อ.....40...บาท

ลาดกระบัง-->บางปะกง..45...บาท

ลาดกระบัง-->พนัสนิคม....75...บาท

ลาดกระบัง-->บ้านฉาง.....90...บาท

ลาดกระบัง-->บางพระ....100..บาท



**Q & A**