



# CSS 112 Computer Programming

Lecture III: Operators, Precedence, and Data Type Conversion

Suvil Chomchaiya (Ph.D.)



# Contents

- Python's Operators
- Python's Operator Precedence
- Data Type Conversion in Python



# Python's Operators



# Operators in Python



- ตัวดำเนินการ (Operators) คือกลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ทำงานเหมือนกับพัฟฟ์ชัน
- แตกต่างกันตรงไวยากรณ์หรือความหมายในการใช้งาน



## Slide 4

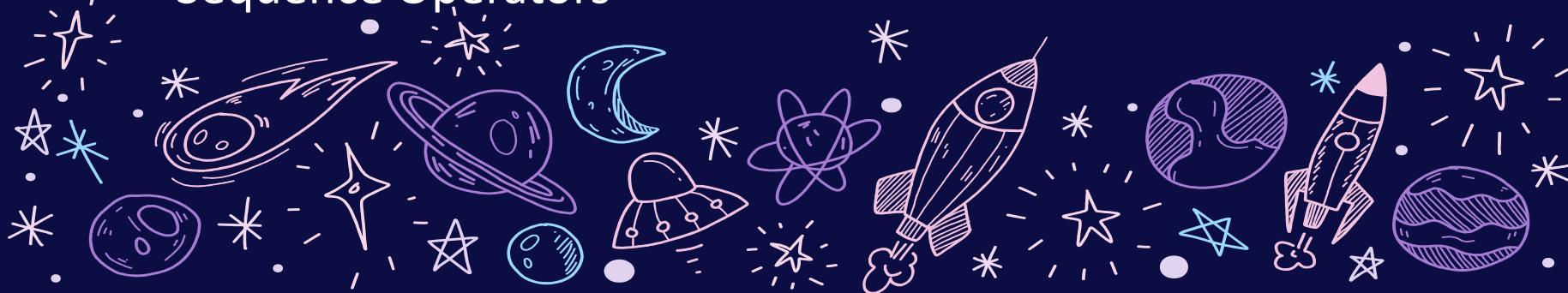
---

**DJ1** Dr. Jard, 8/16/2022

# Operators in Python

## ตัวดำเนินการในภาษา Python

- Assignment operator
- Arithmetic operators
- Comparison operators
- Logical operators
- Bitwise operators
- Sequence Operators



## Slide 5

---

**DJ1** Dr. Jard, 8/16/2022

# Operators in Python: Assignment Operator



- ตัวดำเนินการที่เป็นพื้นฐานที่สุดสำหรับการเขียนโปรแกรมในทุกภาษา เช่น ก็คือ ตัวดำเนินการกำหนดค่า (**Assignment operator**)
- ตัวดำเนินการนี้แสดงโดยใช้เครื่องหมายเท่ากับ (=) ใช้สำหรับกำหนดค่าให้กับตัวแปร
- ตัวอย่าง:

```
a = 3
b = 5.29
c = b
name = 'Mateo'
my_list = [2, 5, 8, 10, 24]
x, y = 10, 20
```

ผลลัพธ์การทำงาน

## Slide 6

---

**DJ1** Dr. Jard, 8/16/2022

# Operators in Python: Arithmetic Operators



- ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators) คือตัวดำเนินการที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ในพื้นฐาน เช่น การบวก การลบ การคูณ และการหาร
- ในภาษา Python ยังมีตัวดำเนินการทางคณิตศาสตร์เพิ่มเติม เช่น การหารเอาเศษ (Modulo) การหารแบบเลขจำนวนเต็ม และการยกกำลัง



**DJ1** Dr. Jard, 8/16/2022

# Operators in Python: Arithmetic Operators



Operator	Name	Example
+	Addition	$a + b$
-	Subtraction	$a - b$
*	Multiplication	$a * b$
/	Division	$a / b$
//	Division and floor	$a // b$
%	Modulo	$a \% b$
**	Power	$a ** b$

## Slide 8

---

**DJ1** Dr. Jard, 8/16/2022

# Operators in Python: Arithmetic Operators



- ตัวดำเนินการทางคณิตศาสตร์ (**Arithmetict operators**) คือตัวดำเนินการที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ในพื้นฐาน เช่น การบวก การลบ การคูณ และการหาร
- ในภาษา Python ยังมีตัวดำเนินการทางคณิตศาสตร์เพิ่มเติม เช่น การหารเอาเศษ (**Modulo**) การหารแบบเลขจำนวนเต็ม และการยกกำลัง



## Slide 9

---

**DJ1** Dr. Jard, 8/16/2022

# Operators in Python: Arithmetic Operators



- ตัวอย่างการใช้ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators)

```
a = 5  
b = 3  
print("a + b = ", a + b)  
print("a - b = ", a - b)  
print("a * b = ", a * b)  
print("a / b = ", a / b)  
print("a // b = ", a // b) # floor number to integer  
print("a % b = ", a % b) # get division remainder  
print("a ** b = ", a ** b) # power
```

a + b =	8
a - b =	2
a * b =	15
a / b =	1.6666666666666667
a // b =	1
a % b =	2
a ** b =	125

ผลลัพธ์การทำงาน



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Comparison Operators



- ตัวดำเนินการเปรียบเทียบ (Comparison operators) คือตัวดำเนินการที่ใช้สำหรับเปรียบเทียบค่าหรือค่าในตัวแปร
- ผลลัพธ์ของการเปรียบเทียบนั้นจะเป็น **True** หากเงื่อนไขเป็นจริง และเป็น **False**





# Operators in Python: Comparison Operators



Operator	Name	Example
<	Less than	<code>a &lt; b</code>
<=	Less than or equal	<code>a &lt;= b</code>
>	Greater than	<code>a &gt; b</code>
>=	Greater than or equal	<code>a &gt;= b</code>
==	Equal	<code>a == b</code>
!=	Not equal	<code>a != b</code>
is	Object identity	<code>a is b</code>
is not	Negated object identity	<code>a is not b</code>



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Comparison Operators



- ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ (Comparison operators)

```
# Constant comparison
print('4 == 4:', 4 == 4)
print('1 < 2:', 1 < 2)
print('3 > 10:', 3 > 10)
print('2 <= 1.5', 2 <= 1.5)
print()
```

```
# Variable comparison
```

```
a = 10
b = 8
print('a != b:', a != b)
print('a - b == 2:', a - b == 2)
print()
```

```
4 == 4 : True
1 < 2: True
3 > 10: False
2 <= 1.5 False

a != b: True
a - b == 2: True
```

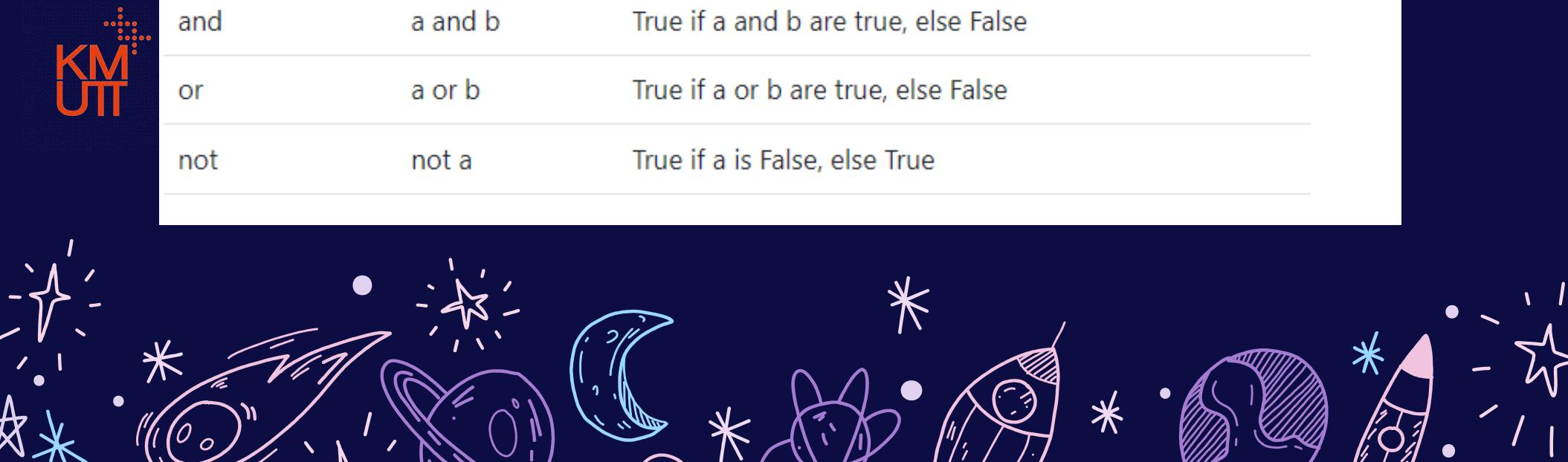
ผลลัพธ์การทำงาน



# Operators in Python: Logical Operators



Operator	Example	Result
and	a and b	True if a and b are true, else False
or	a or b	True if a or b are true, else False
not	not a	True if a is False, else True



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Comparison Operators



- ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ (Comparison operators)

```
# Constant comparison
print('4 == 4:', 4 == 4)
print('1 < 2:', 1 < 2)
print('3 > 10:', 3 > 10)
print('2 <= 1.5', 2 <= 1.5)
print()
```

```
# Variable comparison
```

```
a = 10
b = 8
print('a != b:', a != b)
print('a - b == 2:', a - b == 2)
print()
```

```
4 == 4 : True
```

```
1 < 2: True
```

```
3 > 10: False
```

```
2 <= 1.5 False
```

```
a != b: True
```

```
a - b == 2: True
```

ผลลัพธ์การทำงาน



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Logical Operators



- ตัวดำเนินการตรวจสอบตรรศ (Logical operators) คือตัวดำเนินการที่ใช้สำหรับประเมินค่าทางตรวจสอบตรรศ ซึ่งเป็นค่าที่มีเพียงจริง (True) และเท็จ (False) เท่านั้น
- มักใช้ตัวดำเนินการตรวจสอบตรรศในการเขียน Boolean expression ตั้งแต่หนึ่ง expression ขึ้นไป
- ผลลัพธ์สุดท้ายที่ได้นั้นจะเป็น Boolean



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Logical Operators



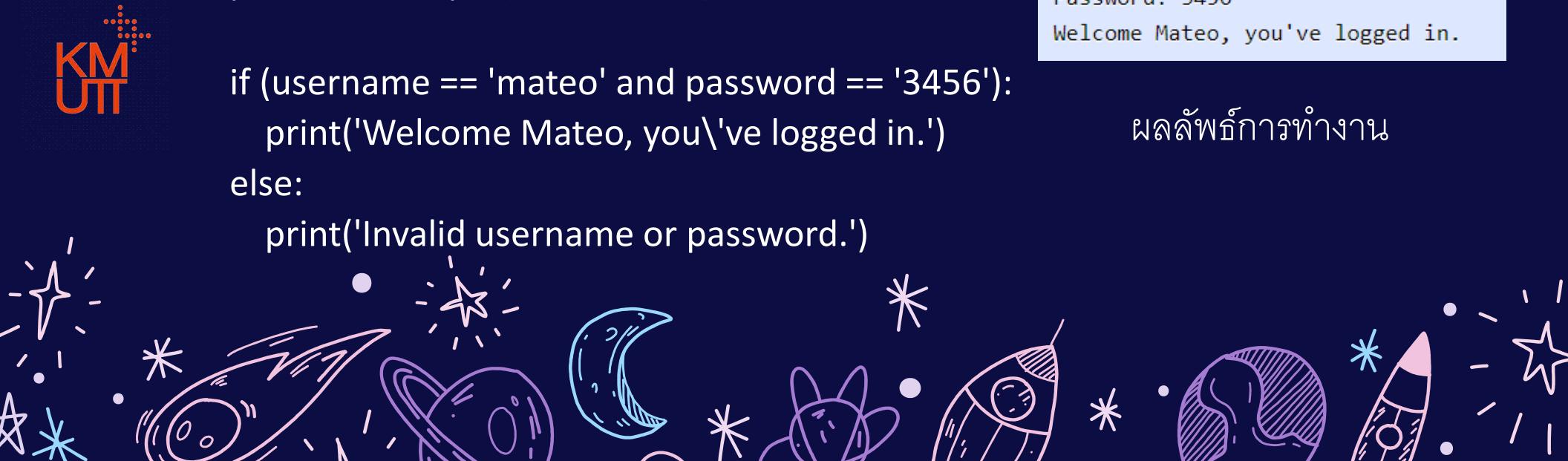
- ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ (Logical operators)

```
print('Log in page')
username = input('Username: ')
password = input('Password: ')
```

```
if (username == 'mateo' and password == '3456'):
    print('Welcome Mateo, you\'ve logged in.')
else:
    print('Invalid username or password.)
```

```
Log in page
Username: mateo
Password: 3456
Welcome Mateo, you've logged in.
```

ผลลัพธ์การทำงาน



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Bitwise Operators



- ตัวดำเนินการระดับบิต (Bitwise operators) เป็นตัวดำเนินการที่ทำงาน **บนระดับบิต** ของข้อมูล หรือ **จัดการข้อมูล** ในระบบเลขฐานสอง
- ตัวดำเนินการระดับบิตมักจะใช้กับการเขียนโปรแกรมระดับต่ำ เช่น การเขียนโปรแกรมเพื่อควบคุมฮาร์ดแวร์
- ตัวดำเนินการระดับบิตใช้จัดการกับบิตของข้อมูลที่เป็นตัวเลข ซึ่งคอมพิวเตอร์จะเก็บค่าเหล่านี้ในหน่วยความจำในรูปแบบของตัวเลขฐานสอง (binary form) ซึ่งประกอบไปด้วยเพียง 1 และ 0 เท่านั้น



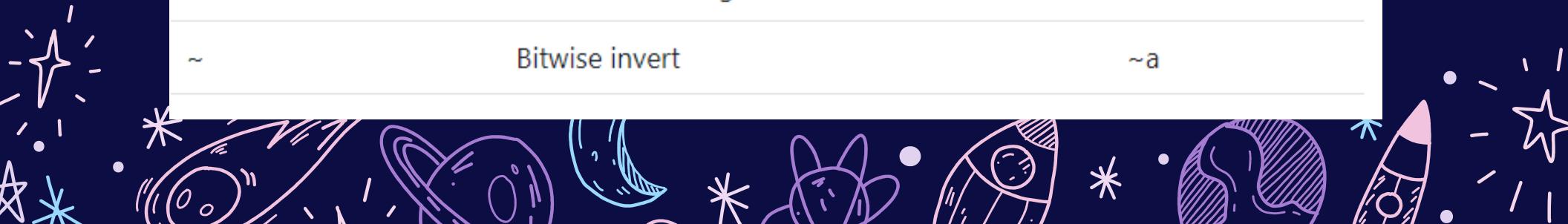
**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Bitwise Operators



Operator	Name	Result
<code>&amp;</code>	Bitwise and	<code>a &amp; b</code>
<code> </code>	Bitwise or	<code>a   b</code>
<code>^</code>	Bitwise exclusive or	<code>a ^ b</code>
<code>&lt;&lt;</code>	Bitwise shifted left	<code>a &lt;&lt; b</code>
<code>&gt;&gt;</code>	Bitwise shifted right	<code>a &gt;&gt; b</code>
<code>~</code>	Bitwise invert	<code>~a</code>



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Bitwise Operators



- ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบ (Bitwise Operators)

```
a = 3 # 00000011
```

```
b = 5 # 00000101
```

```
print('a & b =', a & b)
print('a | b =', a | b)
print('a ^ b =', a ^ b)
print('~a =', ~a)
print('a << 1 =', a << 1)
print('a << 2 =', a << 2)
print('100 >> 1 =', 100 >> 1)
```

```
a & b = 1
a | b = 7
a ^ b = 6
~a = -4
a << 1 = 6
a << 2 = 12
100 >> 1 = 50
```

ผลลัพธ์การทำงาน



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Sequence Operators



- ตัวดำเนินการในการตรวจสอบการเป็นสมาชิก (Sequence Operators หรือ Membership Operators) ในออบเจ็คประเภท List Tuple และ Dictionary



ตัวดำเนินการ `in` ใช้ในการตรวจสอบถ้าหากค่าที่นั้นมีอยู่ในออบเจ็ค ถ้าหากพบจะได้ผลลัพธ์เป็น `True` และหากไม่พบจะได้ผลลัพธ์เป็น `False`

- ตัวดำเนินการ `not in` นั้นจะทำงานตรงกันข้าม หากไม่พบจะได้ผลลัพธ์เป็น `True`

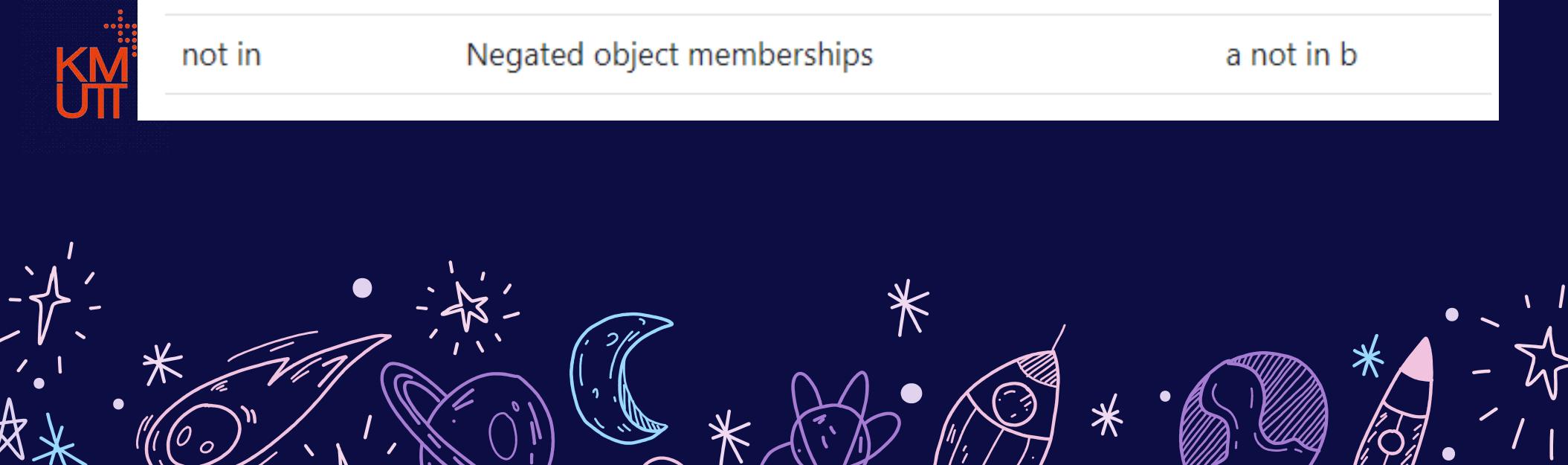




# Operators in Python: Sequence Operators



Operator	Name	Example
in	Object memberships	a in b
not in	Negated object memberships	a not in b



**DJ1**

Dr. Jard, 8/16/2022

# Operators in Python: Sequence Operators



Operator	Name	Example
in	Object memberships	a in b
not in	Negated object memberships	a not in b



**DJ1**

Dr. Jard, 8/16/2022

# Python's Operator Precedence



# Operators Precedence



- ลำดับความสำคัญของตัวดำเนินการใน **Python Operators Precedence** เป็นการจัดลำดับความสำคัญให้กับตัวดำเนินการ
- ในนิพจน์ (**Expression**) หนึ่งๆ อาจมี **operator** มากกว่า 1 ตัวจึงต้องมีการลำดับการคำนวณก่อนหลัง



**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence



Precedence Level	Operator	Explanation
1 (highest)	( )	Parentheses
2	**	Exponentiation
3	-a, +a	Negative, positive argument
4	* , / , // , %, @	Multiplication, division, floor division, modulus, at
5	+, -	Addition, subtraction
6	< , <=, >, >=, ==, !=	Less than, less than or equal, greater, greater or equal, equal, not equal
7	not	Boolean Not
8	and	Boolean And
9	or	Boolean Or

**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence



- ตัวอย่าง

$a=2$

$b=3$

$c=4$

`print(a**b+c)`



- มีค่าเท่ากับ
  - $print (2*2*2+4)$  ผลลัพธ์ = 12



**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence



- $\text{num1} = (10-2*4)$  จะได้เท่ากับ 2 ไม่ใช่ 32 เพราะว่าลำดับการคำนวนครับ  
นี่จะทำ  $2*4$  ก่อน แล้วจึงทำ  $10-8$  จึงได้เท่ากับ 2
- $\text{total} = 10*4*2+21/3$  จะได้เท่ากับ 87 คิดตามขั้นตอน:
  1.  $10 * 4$
  2.  $40 * 2$
  3.  $21 / 3$
  4.  $80+7$



**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence

```
# Precedence of or & and
```

```
meal = "fruit"
```

```
money = 0
```

```
if meal == "fruit" or meal == "sandwich" and money >= 2:
```

```
    print("Lunch being delivered")
```

```
else:
```

```
    print("Can't deliver lunch")
```

Precedence of AND is higher than OR!!!!

Lunch being delivered

Output



**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence

# Precedence of or & and

```
meal = "fruit"
```

```
money = 0
```

```
if (meal == "fruit" or meal == "sandwich") and money >= 2:
```

```
    print("Lunch being delivered")
```

```
else:
```

```
    print("Can't deliver lunch")
```

Can't deliver lunch

Output





# Operators Precedence

```
# Precedence of 'or' & 'and'
```

```
name = "Alex"
```

```
age = 0
```

```
if name == "Alex" or name == "John" and age >= 2 :
```

```
    print("Hello! Welcome.")
```

```
else :
```

```
    print("Good Bye!!")
```

Hello! Welcome.

Output



**DJ1**

Dr. Jard, 8/16/2022

# Operators Precedence



```
# Precedence of 'or' & 'and'
```

```
name = "Alex"
```

```
age = 0
```

**KMUTT**

```
if( name == "Alex" or name == "John" ) and age >= 2 :  
    print("Hello! Welcome.")  
else :  
    print("Good Bye!!")
```

Good Bye !!

Output



# Operators Associativity



- ตัว operator ที่อยู่ในกลุ่มหรือถูกจัดกลุ่มไว้ด้วยกันจะมีลำดับความสำคัญเท่ากัน
- เมื่อ operator สองตัวที่มีลำดับความสำคัญเท่ากันมาอยู่ด้วยกันในนิพจน์ (expression) เดียว การจัดความสัมพันธ์ของ operator หรือ associativity จะเป็นตัวบ่งบอกลำดับของการดำเนินการ
- Associativity คือ ลำดับการดำเนินการของตัว operator ที่มีลำดับความสำคัญเท่ากันและอยู่ในนิพจน์เดียวกัน

เป็นแบบซ้าย  $\rightarrow$  ขวา





# Operators Associativity

# Left-right associativity

# Output: 3

```
print(5 * 2 // 3)
```

# Shows left-right associativity

# Output: 0

```
print(5 * (2 // 3))
```

3  
0

Output



**DJ1**

Dr. Jard, 8/16/2022

# Operators Associativity

# Shows the right-left associativity of \*\*

# Output: 512, Since  $2^{**}(3^{**}2) = 2^{**}9$

```
print(2 ** 3 ** 2)
```

# If 2 needs to be exponented first, need to use ()

# Output: 64

```
print((2 ** 3) ** 2)
```

512  
64

Output





# Data Type Conversion in Python



# Data Type Conversion: การตรวจสอบประเภทข้อมูล ของตัวแปร

- Python สามารถเช็คประเภทข้อมูลของตัวแปรได้โดยการใช้ฟังก์ชัน `type()` และ Python ก็จะคืนผลลัพธ์ว่าเป็นตัวแปรประเภทใด



**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: การตรวจสอบประเภทข้อมูลของตัวแปร

```
a = 'www.kmutt.ac.th'  
b = 2020  
c = True  
print('Type of a : ')  
print(type(a))  
print('Type of b : ')  
print(type(b))  
print('Type of c : ')  
print(type(c))
```

```
Type of a :  
<class 'str'>  
Type of b :  
<class 'int'>  
Type of c :  
<class 'bool'>
```

Output

**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion



- ถึงแม้ Python จะเป็นภาษาที่สามารถกำหนดชนิดของข้อมูลได้ตั้งแต่ตอนที่สร้างตัวแปร แล้วแต่บางครั้งอาจต้องการแปลงชนิดของ ตัวแปร เพื่อนำไปใช้งานต่อในรูปแบบอื่นๆ
- ในภาษา Python สามารถใช้ฟังก์ชันแบบ **built-in** ที่มีอยู่สำหรับแปลงประเภทข้อมูล
- ฟังก์ชันเหล่านี้จะมีชื่อที่เหมือนกับประเภทของต้นเอง (ซึ่งเป็นชื่อของคลาส)





# Data Type Conversion



- ประเภทของการแปลงข้อมูลใน Python

## 1. การแปลงโดยนัย (Implicit Conversion)

- Python จะแปลงประเภทข้อมูลหนึ่งเป็นประเภทข้อมูลอื่นโดยอัตโนมัติ
- กระบวนการนี้ไม่ต้องการการมีส่วนร่วมของผู้ใช้

## 2. การแปลงอย่างชัดเจน (Explicit Conversion)

- ผู้ใช้แปลงชนิดข้อมูลของอ็อปเจ็กต์เป็นชนิดข้อมูลที่จะนำไปใช้งานต่อ
- ใช้ฟังก์ชันที่กำหนดไว้ล่วงหน้า



**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: Implicit Conversion

```
x = 10
```

```
print("x is of type:",type(x))
```

```
y = 10.6
```

```
print("y is of type:",type(y))
```

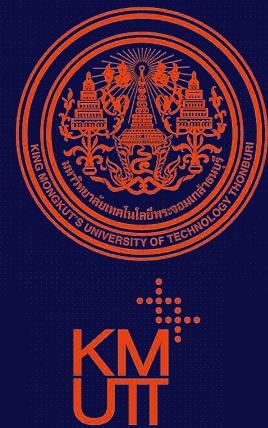
```
z = x + y
```

```
print(z)
```

```
print("z is of type:",type(z))
```

```
x is of type: <class 'int'>
y is of type: <class 'float'>
20.6
z is of type: <class 'float'>
```

Output





# Data Type Conversion: Explicit Conversion



Function	Description
<code>int(x [,base])</code>	แปลงออบเจ็ค x จากฐานที่กำหนด base ให้เป็น Integer
<code>long(x [,base] )</code>	แปลงออบเจ็ค x จากฐานที่กำหนด base ให้เป็น Long
<code>float(x)</code>	แปลงออบเจ็ค x ให้เป็น Floating point number
<code>complex(real [,im])</code>	สร้างตัวเลขจำนวนเชิงซ้อนจากค่า real และค่า imagine
<code>str(x)</code>	แปลงออบเจ็ค x ให้เป็น String
<code>repr(x)</code>	แปลงออบเจ็ค x ให้เป็น String expression
<code>eval(str)</code>	ประเมินค่าของ String
<code>tuple(s)</code>	แปลง Sequence ให้เป็น Tuple
<code>list(s)</code>	แปลง Sequence ให้เป็น List

**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: Explicit Conversion



Function	Description
set(s)	แปลง Sequence ให้เป็น Tuple
dict(d)	แปลงออบเจ็คให้เป็น Dictionary
frozenset(s)	แปลงออบเจ็คให้เป็น Frozen set
chr(x)	แปลงค่าของ Integer ให้เป็น Unicode Char
ord(x)	แปลง Charterer ให้เป็นค่า Integer
hex(x)	แปลง Integer ให้เป็น Hex string
oct(x)	แปลง Integer ให้เป็น Oct string

**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: int() float()



```
# Python code to demonstrate Type conversion  
# using int(), float()
```

```
# initializing string  
s = "10010"  
  
# printing string converting to int base 2  
c = int(s,2)  
print ("After converting to integer base 2 : ", end="")  
print (c)
```

```
# printing string converting to float  
e = float(s)  
print ("After converting to float : ", end="")  
print (e)
```

```
After converting to integer base 2 : 18  
After converting to float : 10010.0
```

Output



**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: `ord()`, `hex()`, `oct()`

```
# initializing integer
s = '4'

# printing character converting to integer
c = ord(s)
print ("After converting character to integer : ",end="")
print (c)

# printing integer converting to hexadecimal string
c = hex(56)
print ("After converting 56 to hexadecimal string : ",end="")
print (c)

# printing integer converting to octal string
c = oct(56)
print ("After converting 56 to octal string : ",end="")
print (c)
```

```
After converting character to integer : 52
After converting 56 to hexadecimal string : 0x38
After converting 56 to octal string : 0o70
```

Output



**DJ1**

Dr. Jard, 8/16/2022

# Data Type Conversion: tuple(), set(), list()

```
# initializing string
s = 'geeks'

# printing string converting to tuple
c = tuple(s)
print ("After converting string to tuple : ",end="")
print (c)
```

```
# printing string converting to set
c = set(s)
print ("After converting string to set : ",end="")
print (c)
```

```
# printing string converting to list
c = list(s)
print ("After converting string to list : ",end="")
print (c)
```

## Output:

```
After converting string to tuple : ('g', 'e', 'e', 'k', 's')
After converting string to set : {'k', 'e', 's', 'g'}
After converting string to list : ['g', 'e', 'e', 'k', 's']
```

## Output



**DJ1**

Dr. Jard, 8/16/2022

# Conclusion

- ลำดับความสำคัญของตัวดำเนินการใน **Python Operators Precedence** เป็นการจัดลำดับความสำคัญให้กับตัวดำเนินการในนิพจน์ (**Expression**)
- **Associativity** คือการเรียงลำดับการดำเนินการจากซ้ายไปขวาของ operator ที่มีลำดับความสำคัญเท่าๆ กัน แต่อยู่ในนิพจน์ (**expression**) เดียวกัน



**DJ1**

Dr. Jard, 8/16/2022

# Thanks!

Any questions?

