

Python basic recap

Kiatnarong Tongprasert

Why Python ?



1. **Easy & flexible** : Code เล็กกว่าภาษาอื่น
2. **Powerful**.

- Guido van Rossum : early 1990s
- Python 2 : 2000
- major version Python 3 : 2008
- latest version freely available at www.python.org

Very popular as a server-side language. Google (spider, search engine, Google Maps), Netflix and Pinterest use it a lot. Youtube, Quora, Reddit, Dropbox, Yahoo, Battlefield 2, Civilization 4, NASA, AlphaGene – all of them use Python; see the entire list [here](#).

3. **High demand for programmers**. See open job positions on [StackOverflow](#)

Contents

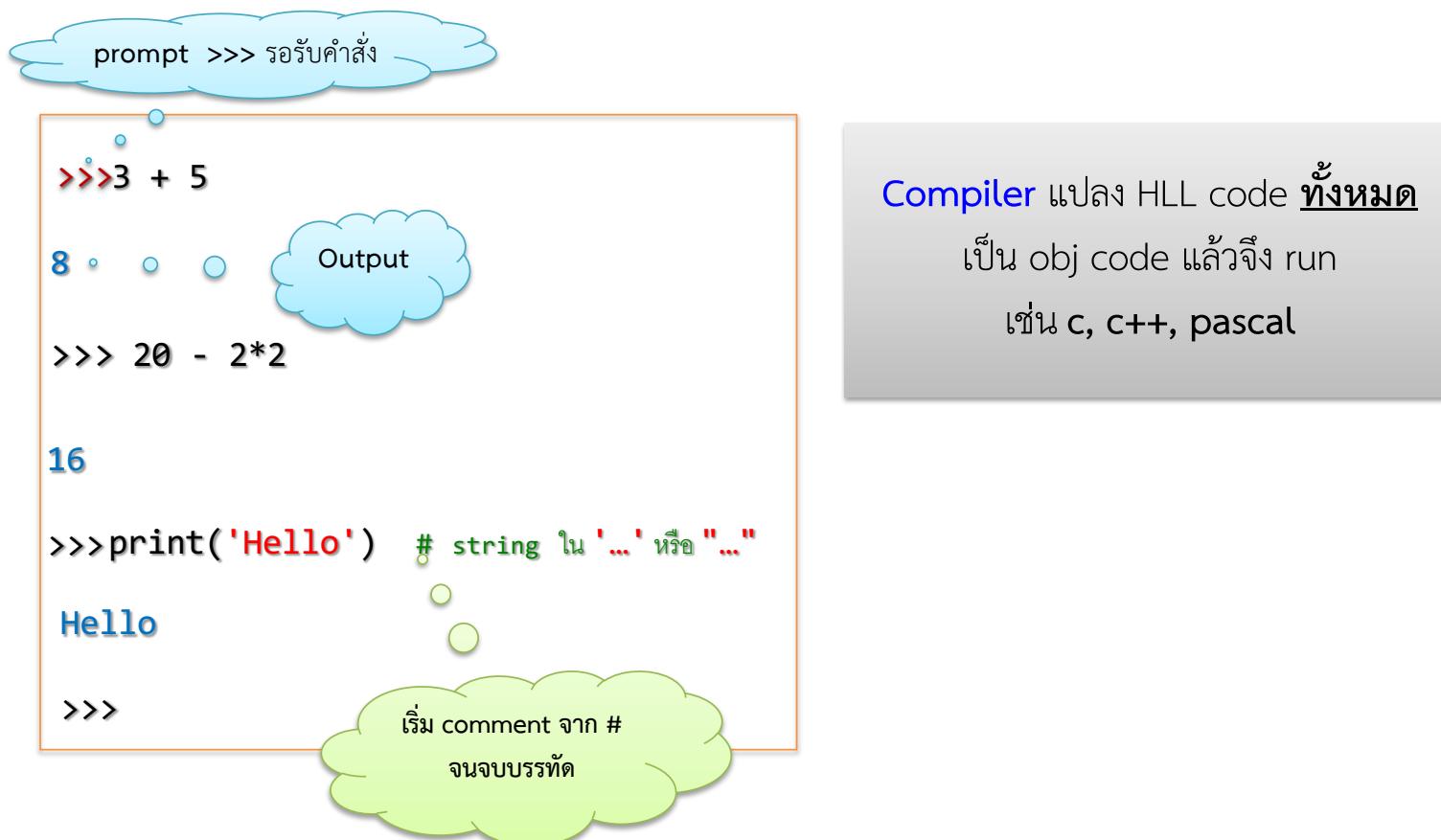
- 1. Web Development
- 2. Games
- 3. Graphics
- 4. Financial
- 5. Science
- 6. Electronic Design Automation
- 7. Software Development
- 8. Education
- 9. Business Software
- 10. Government

Presentation Overview & References

- Interpreter
- Dynamic Typing
- Functions
- Global & Local Variables
- Control Flow Statements
- Data Types
 - <https://interactivepython.org/runestone/static/pythonds/index.html>
 - http://www.python-course.eu/python3_course.php
 - <https://docs.python.org/3/tutorial/index.html>
 - <https://www.python.org/>

Python Interpreter

Interpreter แปลง HLL code เป็น obj code ทีละคำสั่ง และรันคำสั่งนั้น จึงแปลงคำสั่งถัดไป
java, perl, python, shell script, vb script



Variable

Variable คือ ?

- **vary** = แตกต่าง, เปลี่ยนแปลง **variable** : เปลี่ยนค่าได้
- เป็นตัวที่จะเข้าถึง **memory location** ที่ใช้โดย computer program
(เป็น reference ของ memory location)
- เป็น **symbolic name** สำหรับ **physical location**
- เราใช้ variable เก็บข้อมูลใน memory location หรือ ดึงข้อมูล จาก memory location ออกมายัง

$x = 5$

$y = x * 7$

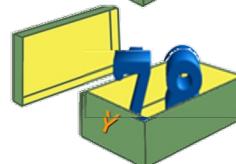
C, C++, Java Variable

- ใน C, C++, Java variable ถูกสร้างโดย declare ว่า variable ชื่อนี้ เป็น type ใด ซึ่งทำให้โปรแกรมสามารถสำรอง memory สำหรับ variable นั้น ตามขนาด type ที่ระบุ

```
int x;
```



```
int y;
```



- variable จะต้องถูก declare ก่อนใช้
- ชื่อ variable ใช้แทนพื้นที่หน่วยความจำ (memory location) ที่สำรองไว้ ดังนั้น $x = 42$; $y = 72$; เป็นการเก็บค่า ลงใน memory locations ดังรูป

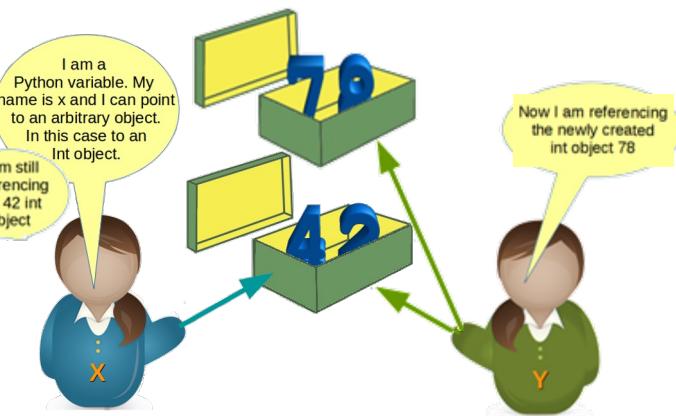
```
x = 42;
```

```
y = 42;
```

```
y = 78;
```

Python Variable

```
>>> x = 42  
>>> y = x  
>>> y = 78
```



`id Function()`

```
>>> x = 42  
>>> id(x) 10107136  
>>> y = x  
>>> id(x), id(y) (10107136, 10107136)  
>>> y = 78  
>>> id(x), id(y) (10107136, 10108288)  
>>>
```

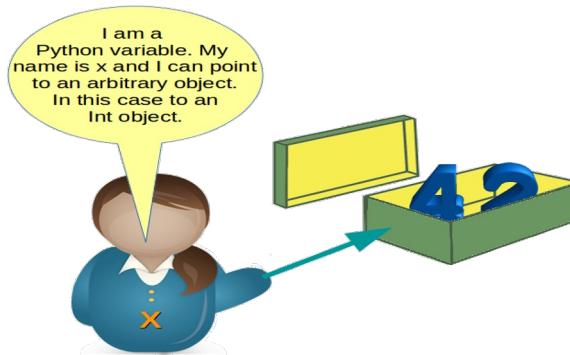
Python Variable

Python variables:

- เป็น **dynamic typing model**
- ไม่ได้ถูก **declared** แต่ถูกสร้างโดยการกำหนดค่าให้มันครั้งแรก เช่นในตบ. กำหนดค่า

42 ให้ variable x

```
>>> x = 42
```



- เรียกว่า ตอนนี้ variable x reference (อิงกับ, ชี้ไปที่) object 42
- ค่าของ variable หมายถึงค่า object ที่มัน references ขณะนั้น
- Type ของข้อมูล อยู่ที่ object ไม่ใช่ variable
- ทุกอย่างใน Python เป็น object → OOP

Identifier ไม่ declared type
แต่ object ที่มัน reference มี type

Closer Look

`temperature = 98.6`

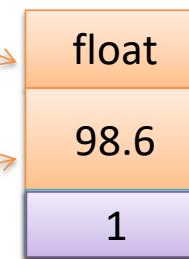
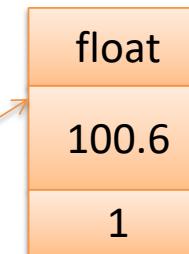
`original = temperature`

`original is temperature -> true`

`temperature = temperature + 2`

`temperature`

`original`



Global and local Variables in Functions

```
def f():
    print(s)
s = "I love Paris in the summer!"
f()
```

I love Paris in the summer!

s defined as string "I love Paris in the summer!", before calling f().
no local variable s in f(), i.e. no assignment to s, the value from the global
variable s will be used.

```
def f():
    s = "I love London!"
    print(s)

s = "I love Paris!"
f()
print(s)
```

I love London!

I love Paris!

```
>>> def f():
...     print(s)
...     s = "I love London!"
...     print(s)
...
...
```

```
>>> s = "I love Paris!"
```

```
>>> f()
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in f
UnboundLocalError: local variable 's' referenced before
assignment
```

```
>>>
```

A variable can't be both local and global inside of a function.

So Python decides that we want a local variable due to the assignment to `s` inside of `f()`, so the first `print` statement before the definition of `s` throws the error message above.

To tell Python, that we want to use the global variable, we have to explicitly state this by using the keyword "global",

```
def f():
    global s
    print(s)
    s = "Only in spring, but London is great as well!"
    print(s)
s = "I am looking for a course in Paris!"
f()
print(s)
```

```
I am looking for a course in Paris!
Only in spring, but London is great as well!
Only in spring, but London is great as well!
```

No local s.

```
def f():
    s = "I am globally not known"
    print(s)
f()
print(s)
```

Local variables of functions can't be accessed from outside, when the function call has finished:

Function Basic

Dynamic typing parameter and return value

functionbasics.py

```
def max(x,y) :  
    if x > y :  
        return x  
    else :  
        return y  
  
def f():  
    return  
  
def ff():  
    i=5
```

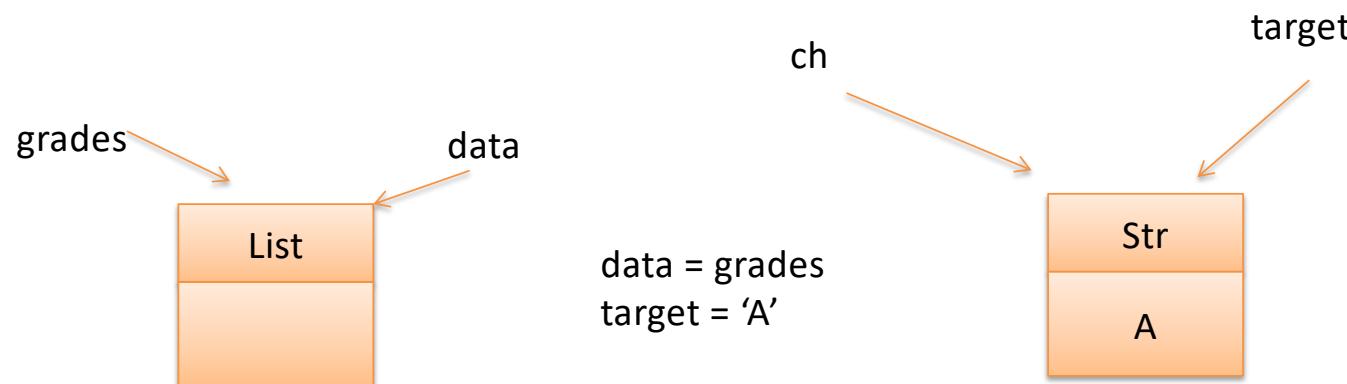
```
>>> import functionbasics  
>>> max(3,5)  
5  
>>> max('hello', 'there')  
'there'  
>>> max('3', 'hello')  
'hello'  
  
>>> print(f(), ff())  
(None, None)
```

Parameter Passing

Parameter passing
follows the semantics of standard *assignment statement*.

```
def count(data, target):
    n = 0
    for item in data:
        if item == target: # a match
            n += 1
    return n
```

```
ch = 'A'
prizes = count(grades, ch )
```



Function names are like any variable

- Functions are objects
- The same reference rules hold for them as for other objects

```
>>> x = 10
>>> x
10
>>> def x () :
...     print 'hello'
>>> x
<function x at 0x619f0>
>>> x()
hello
>>> x = 'blah'
>>> x
'blah'
```

return

output

```
def triangleArea(height, base):
    return 1/2 * height * base
```

```
a = triangleArea(20, 5)
print(a)
```

50.0

```
def addOne(x, y, z):
    return x+1, y+1, z+1
```

```
a, b, c = 5, 10, 15.2
a, b, c = addOne(a, b, c)
print(a, b, c)
```

6 11 16.2

return หลายค่า

Built-in Types

Python :
Object Oriented Programming (OOP)
class เป็นพื้นฐานของทุก data type

immutable type: เปลี่ยน content ไม่ได้
`s = 'Hi'
s[1] = 'A' # error
s = 7 # ok ชี้ object ใหม่ object 7`

mutable type: เปลี่ยน content ได้
`lst = [1,2,3]
print(lst) # => [1,2,3]
lst[0] = 7
print(lst) # => [7,2,3]`

Commonly-used built-in class (type) :

- numbers
 - integral
 - int
 - bool
 - float
 - complex
- sequences (เก็บของเรียงลำดับ)
 - immutable
 - str (string)imaginary part
 - tuple
 - byte
 - mutable
 - list
 - range
 - bytearray
- mappings
 - dict (mutable)
- set
 - set (mutable)
 - frozenset (immutable)
- callable types (~ fn call)
 - class
 - function
 - ...

type ?

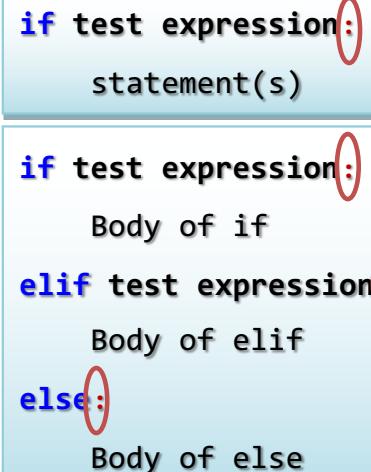
```
>>> type(5)  
<class 'int'>  
  
>>> type(3.5)  
<class 'float'>  
  
>>> type("Hi")  
<class 'str'>  
  
>>> type('Python')  
<class 'str'>  
  
>>> type([1,2,3])  
<class 'list'>
```

Immutable ?

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Table 1.2: Commonly used built-in classes for Python

if



```
>>> x = int(input("Please enter an integer: "))  
Please enter an integer: 42  
>>> if x < 0:  
...     x = 0  
...     print('Negative changed to zero')  
... elif x == 0:  
...     print('Zero')  
... elif x == 1:  
...     print('Single')  
... else:  
...     print('More')  
...  
More
```



- There can be zero or more **elif** parts, and the **else** part is optional.
- The keyword '**elif**' is short for 'else if', and is useful to avoid excessive indentation.
- An **if elif elif** sequence is a substitute for the **switch** or **case** statements found in other languages.

The ternary if

```
if (a > b)
    max=a;
else
    max=b;
```

```
max = a if (a > b) else b
```

Python

The Python version is more readable. It can be read as "max shall be a if a is greater than b else b".

ternary if statement is an expression, can be used within another expression:

```
max = (a if (a > b) else b) * 2.45 - 4
```

while

```
while test expression:  
    Body of while
```

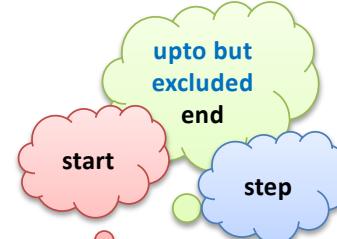
```
#!/usr/bin/env python3  
  
n = 100  
  
s = 0  
counter = 1  
while counter <= n:  
    s = s + counter  
    counter += 1  
  
print("Sum of 1 until %d: %d" % (n,s))
```

range class

range (a, b, s)

return sequence
ของตัวเลข

a, a + 1s, a + 2s, a + 3s, ..., before b



print(list(range(0, 9, 3)))

print(list(range(5)))

argument n ตัวเดียว
ตั้งแต่ 0 ไป n ตัว step 1

default step = 1

print(list(range(1, 5)))

print(list(range(-10, -50, -20)))

print(list(range(1, 0)))

range type :

เป็น immutable sequence ของ numbers นิยมสำหรับ loop : for

[0, 3, 6]

[0, 1, 2, 3, 4]

[1, 2, 3, 4]

[-10, -30]

[]

for, sequence : range()

แต่ละ iteration ตัวแปร i = ค่าแต่ละค่าใน sequence

```
0,1,2,3,4  
for i in range(5):  
    print(i, end = ' ')
```

default end = '\n'

0 1 2 3 4

```
s = 'abcdefghijklm'
```

```
0,1,2,...,8  
9  
for i in range(len(s)):  
    print(s[i], end = '') s[0],s[1],...,s[8]
```

abcdefghijklm

```
1,4,7  
for i in range(1,8,3):  
    print(s[i], end = '')
```

bh

for, list

var ซึ่งເອົາຈາກ
sequence ມາໃນແຕ່ລະ
iteration

for val in sequence:
Body of for

```
list = [2, 1, 3, 4]
for ele in list:
    print(ele) # ແຕ່ລະ iteration ຕັ້ງແປ່ ele = ຄໍາແຕ່ລະຄໍາໃນ list →
for i in range(len(list)):
    print(list[i])
```

2
1
3
4

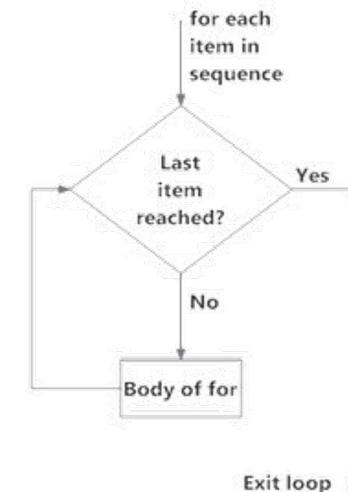


Fig: operation of for loop

Find sum of elements in list

```
list = [2, 1, 3, 4]
sum = 0
for ele in list:
    sum += ele

print("Sum of list elements = ", sum)
```

→ Sum of list elements = 10

range() testing

```
>>> for i in range(4):  
...     print(i)  
...
```

output ?

```
range(5, 10)
```

expression output ?

```
range(0, 10, 3)
```

expression output ?

```
range(-10, -100, -30)
```

expression output ?

```
>>> for i in reversed(range(1, 10, 2)):  
...     print(i)  
...
```

output ?

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])
...
```

output ?

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print(i, v)
...
```

output ?

looping through a sequence, retrieved both position index and corresponding value using
enumerate() function

break, else clauses on loops



```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, '=', x, '*', n//x)
            break
    else: # else ของ for
        # loop fell through without finding a factor
        print(n, 'is prime')
```

```
2 is prime
3 is prime
4 = 2 * 2
5 is prime
6 = 2 * 3
7 is prime
8 = 2 * 4
9 = 3 * 3
```

Continue statement

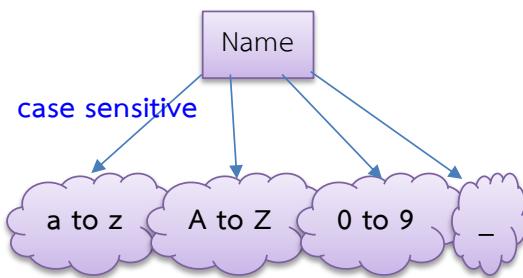
Continue statement, borrowed from C, continues with the next iteration of the loop:

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print("Found an even number", num)
...     continue
...     print("Found a number", num)
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9
```

use the sorted() function which returns a new sorted list while leaving the source unaltered

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange',
'banana']
>>> for f in sorted(set(basket)):
...     print(f)
...
apple
banana
orange
pear
```

Variable (Name, Identifier)



ไม่ขึ้นต้นด้วยตัวเลข
ไม่เป็น keywords

<code>_var1</code>	<code>if</code>	<code>keyword</code>
<code>myVar</code>	<code>elif</code>	<code>keyword</code>
<code>num</code>	<code>9i</code>	ขึ้นต้นด้วย 0-9

Python Keywords

False	and	break	def	else	for	if	is	not	raise	while
None	as	class	del	except	from	import	lamda	or	return	with
True	assert	continue	elif	finally	global	in	nonlocal	pass	try	yield

Multiple Assignments

```
>>> a, b, c = 1, 3.5, 'Hello'  
>>> print(a, b, c)  
1 3.5 Hello
```



```
>>> i = j = k = 'same'  
>>> print(id(i), id(j), id(k))  
37565440 37565440 37565440
```

ลำดับการ evaluate ตามลำดับเลข

exp3, exp4 = exp1, exp2
exp3 = exp1 exp4 = exp2
a = 10 b = 5 10 5

i = 1 x[1] = 2 1 2

```
>>> a = 5  
>>> b = 10  
>>> a, b = b, a  
>>> print(a,b)  
10 5
```

```
>>> x = [7, 3]  
>>> i = 0  
>>> i, x[i] = 1, 2  
>>> print(x)  
[7, 2]
```

-2 -1
0 1
List [0, 1] เก็บของตามลำดับ
ใช้ index access ของที่เก็บ
'เล่าจาก หน้า -> หลัง ตัวแรกเริ่มจาก index 0, 1, ...
'เล่าจาก หลัง -> หน้า ตัวแรกเริ่มจาก index -1, -2, ...

Undefined Name

Using Undefined Variable

```
>>> n
Traceback (most recent call last):
  File "<stdin>", line 1, in
    <module>
NameError: name 'n' is not defined
```

ERROR

print()

```
a = 5  
b = 2  
print(a)  
print(a, '+', b, "=", a+b)
```

output

5 . .
5 + 7 = 12

ปกติ seperate
ด้วย space

print() end
ด้วย newline

```
>>> print(a, '+', b, "=", a+b, sep = ' ', end = '***\n')  
5+7=12***
```

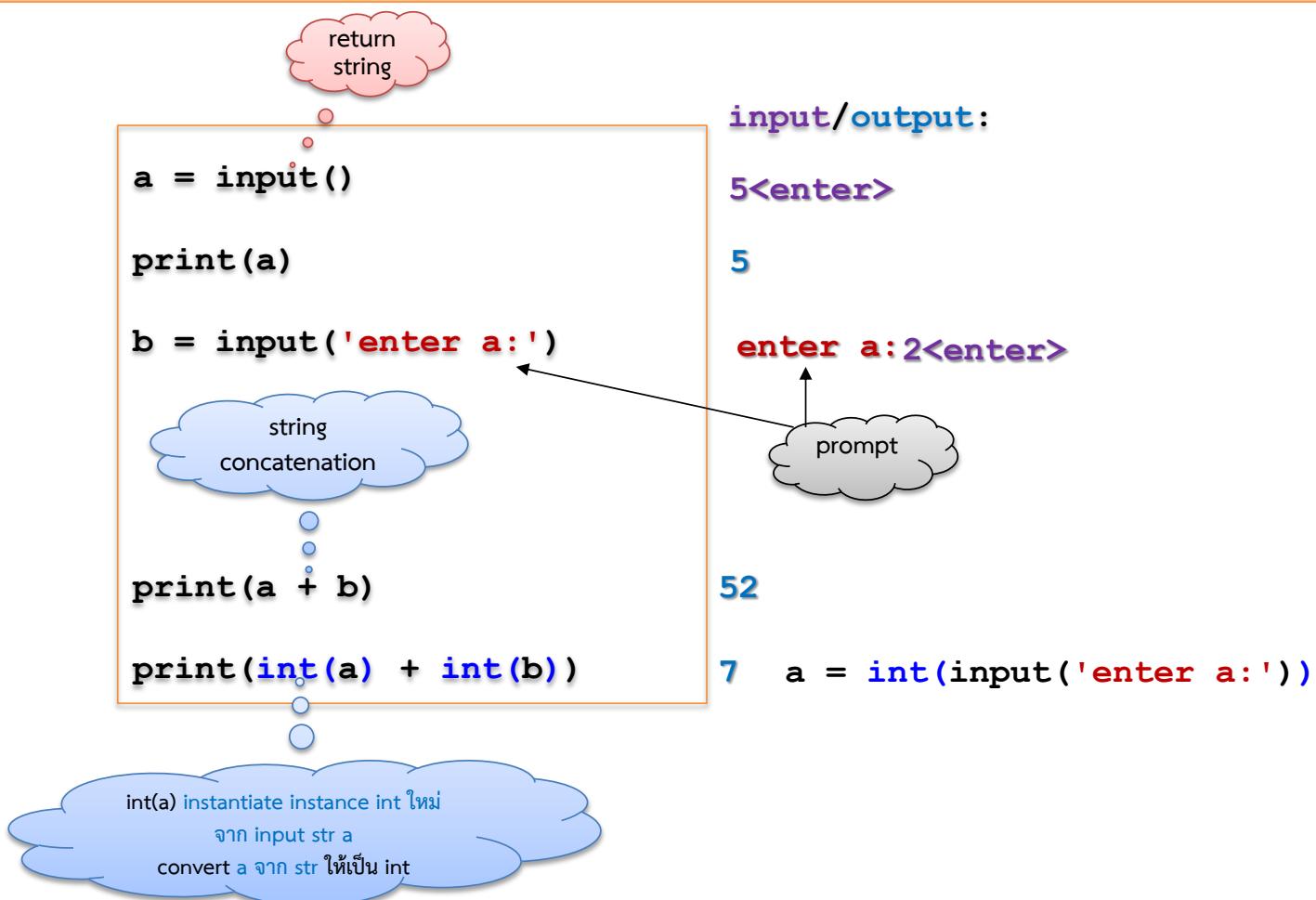
ตามที่ set
sep ไว้

ตามที่ set
end ไว้

สลับที่ได้

null
character

input(), int()



```
for c in input('input:').split():
    print(c, end = ' ')
```

input/output
input:1 2 3 4 5
1 2 3 4 5

```
l = [c for c in input('input:').split()]
print('l = ', l)
```

input/output
input:1 2 3 4 5 2 3 1
l = [1, 2, 3, 4, 5, 2, 3, 1]

Arithmetic Operators

```
>>> (20 - 2*2) / 4 ... (...) ทำก่อน
```

4.0

```
>>> 5/2 ... floating point division
```

2.5

```
>>> 5//2 ... div
```

2

```
>>> 5%2 ... mod
```

1

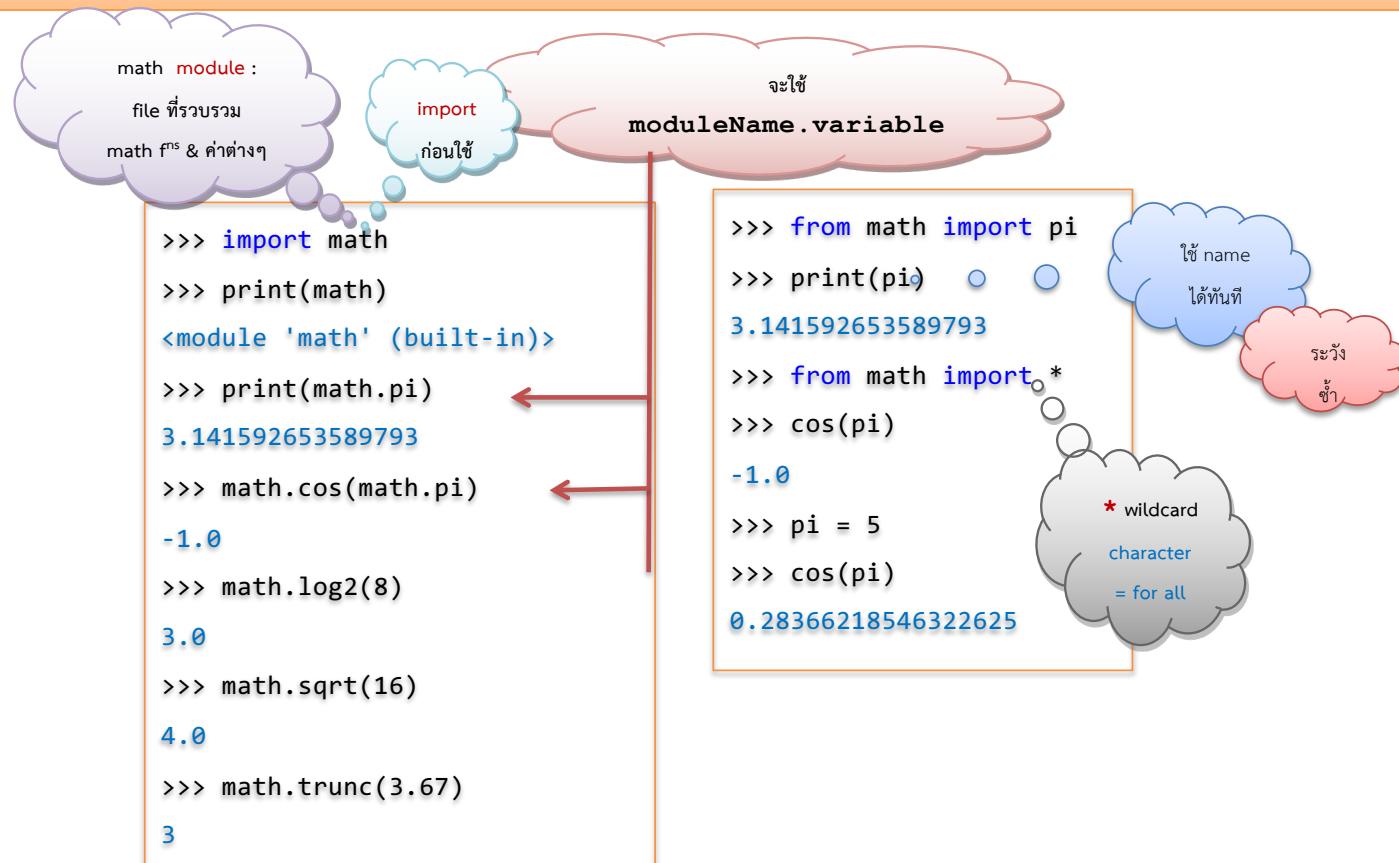
```
>>> 2**3 ... power
```

8

```
>>> 3.5 - 2 ... mix : convert int เป็น float ก่อน
```

1.5

import statement, module



Arithmetic & Bitwise Operators

Arithmetic Operators :

- + addition
- subtraction
- * multiplication
- / true division
- // integer division
- % the modulo operator

Bitwise Operators :

- ~ bitwise complement (prefix unary operator)
- & bitwise and
- | bitwise or
- ^ bitwise exclusive-or
- << shift bits left, filling in with zeros
- >> shift bits right, filling in with sign bit

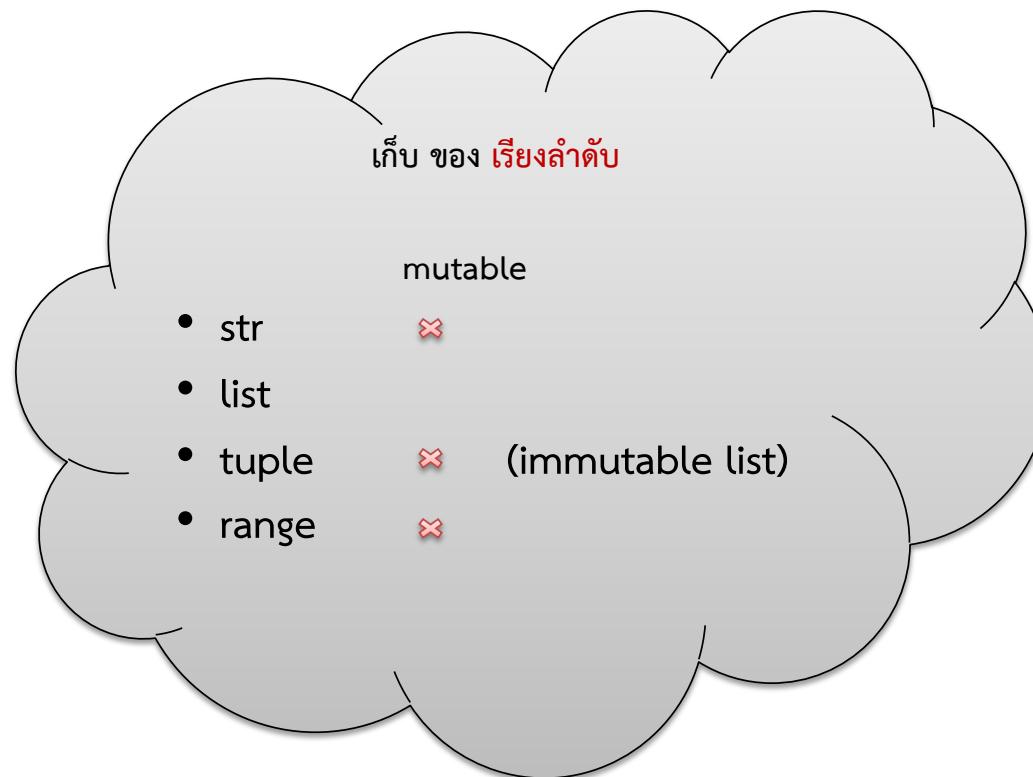
Arithmetic Operator Precedence

	output	
$5 + 3$	8	
$5 - 3$	2	
$5 * 3$	15	
$5 / 3$	1.6666666666666667	
$5 // 3$	1	div
$5 \% 3$	2	mod
-5	-5	
+5	5	
<code>abs(-5)</code>	5	absolute
<code>int(5.2)</code>	5	int conversion
<code>float(5)</code>	5.00	float conversion
<code>divmod(5,3)</code>	(1,2)	divmod pair
<code>pow(2, 3)</code>	8	
<code>2 ** 3</code>	8	

lowest
precedence

highest
precedence

Sequence classes



string Repetition & Concatenation

* string repetition

+ string concatenation

```
>>> 3 *'aa' + 'bcd'  
'aaaaaabacd'  
  
>>> str = 'x'  
  
>>> 6 * str  
'xxxxxx'
```

string literals ชิดกัน
ช่วย เมื่อใช้ string ยาว

```
>>> '1234' '5678'  
'12345678'  
  
>>> s2 = '1234'  
>>> s2 + '5678'  
'12345678'  
  
>>> s3 = ('longxxxxx'  
...  
...  
'stillxxxxxxxx'  
'finallyxxxxx')  
>>> s3  
'longxxxxxstillxxxxxxxxfinallyxxxxx'
```

string literals
ชิดกัน คือ
concat

แต่ string
variables
ต้องใช้ +

string Indexing (subscript), len()

```
>>> s = '01234'  
>>> s[0]  
'0'  
  
>>> s[-1]  
'4'  
>>> s[-2]  
'3'  
  
>>> s[9]  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
IndexError: string index out of range
```

s	0	1	2	3	4
	0	1	2	3	4
	-5	-4	-3	-2	-1

ใช้ index access ของที่เก็บ
ໄลจาก หน้า -> หลัง ตัวแรกเริ่มจาก index 0, 1, ...
ໄลจาก หลัง -> หน้า ตัวแรกเริ่มจาก index -1, -2, ...

```
>>> len(s)
```

5

len()
returns
length

string slicing



The diagram shows several code examples with annotations:

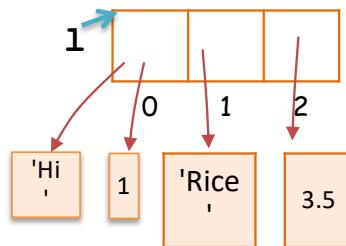
- `>>> s = '0123456789'`
- `>>> s[1:3]` *start* *upto but excluded end*
- `'12'`
- `>>> s[2:9:2]` *step*
- `'2468'`
- `>>> s[:3]`
- `'012'`
- `>>> s[2:]`
- `'23456789'`
- `>>> s[-7:8]`
- `'34567'`
- `>>> 'Hello' + s[-7:8]`
- `'Hello34567'`

str : immutable

```
>>> s = '0123456789'  
>>> s[0] = 'a' • • •  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'str' object does not support item  
assignment
```



list 1



list เก็บ ของ (คละ type ได้) เรียงลำดับกัน

`l = [1, "Rice", 3.5]` ใช้ index ในการ access
 0 1 2 ได้จาก หน้า -> หลัง ตัวแรกเริ่มจาก index 0, 1, ...
 -3 -2 -1 ได้จาก หลัง -> หน้า ตัวสุดท้ายเริ่มจาก index -1, -2, ...

`l[0] = 'Hi'` เป็น mutable type

```
print(1) ['Hi', 'Rice' , 3.5]
```

```
12 = [] # empty list
```

list : Repetition, Concatenation, len(), append(), nested lists

```
>>> li = [1,2]  
>>> lis = [3,4,5]  
>>> 2*li + lis  
[1, 2, 1, 2, 3, 4, 5]  
  
>>> len(li)  
2
```

```
>>> li  
[1, 2]  
>>> li.append(3)  
>>> li  
[1, 2, 3]
```

```
>>> li  
[1, 2, 3]  
>>> li.append([3,4])  
>>> li  
[1, 2, 3, [3, 4]]  
  
>>> li[3]  
[3, 4]  
  
>>> li[3][1]  
4
```



Python List

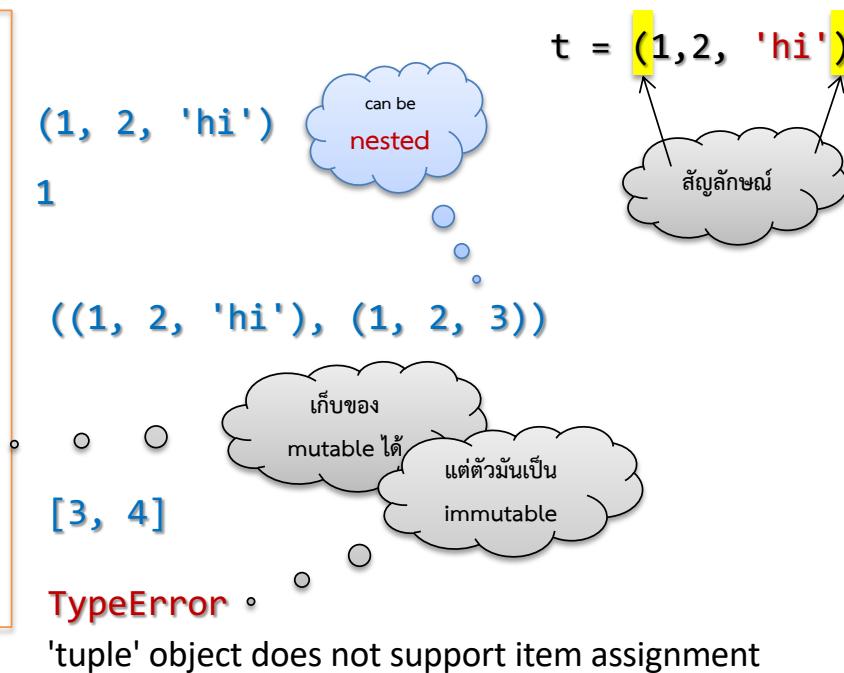
L = [1, 3, 7, 3]

methods	ผลลัพธ์	คำอธิบาย
len(L)	4	จำนวนของใน list
max(L)	7	หา max item, ต้องเป็นไทป์เดียวกัน
min(L)	1	หา min item, ต้องเป็นไทป์เดียวกัน
sum(L)	14	หา sum ของ item, ต้องเป็น number
L.count(3)	2	นับจำนวน 3
L.index(7)	2	หา index ของ 7 ตัวแรก
L.reverse()	[3 , 7 , 3 , 1]	กลับลำดับของของ
L.clear()	[]	ทำให้เป็น empty list
L.append(5)	[1 , 3 , 7 , 3 , 5]	insert object ที่ท้าย list
L.extend([6,7])	[1 , 3 , 7 , 3 , 6, 7]	insert list ที่ท้าย list
del L[1]	[1, 7 , 3]	remove item index 1
L.remove(3)	[1, 7 , 3]	remove item แรกที่มีค่า = 3
L.insert(1, "Hi")	[1 , "Hi", 3 , 7 , 3]	insert new item แทรกที่ index ที่กำหนด
L.pop(0)	[3, 7 , 3]	remove & return item index 0 , ไม่ใส่ index คือตัวขวาสุด

tuple class

tuple เก็บ ของ (คละ type ได้) เรียงลำดับ
กัน เป็น immutable (list : mutable)

```
t = 1, 2, 'hi'  
print(t)  
print(t[0])  
t2 = t, (1, 2, 3)  
print(t2)  
  
t3 = ([1, 2], [3, 4])  
print(t3[1])  
t[0] = 5
```



Sequence Operators

Sequence Operators : (str, tuple, list , and range)

s[j]

element at index *j*

s[start:stop]

slice including indices [start,stop)

s[start:stop:step]

slice including indices start, start + step, start + 2 step, . . . ,
up to but not equalling or stop

s + t

concatenation of sequences

k s

s + s + ... (k times)

val in s

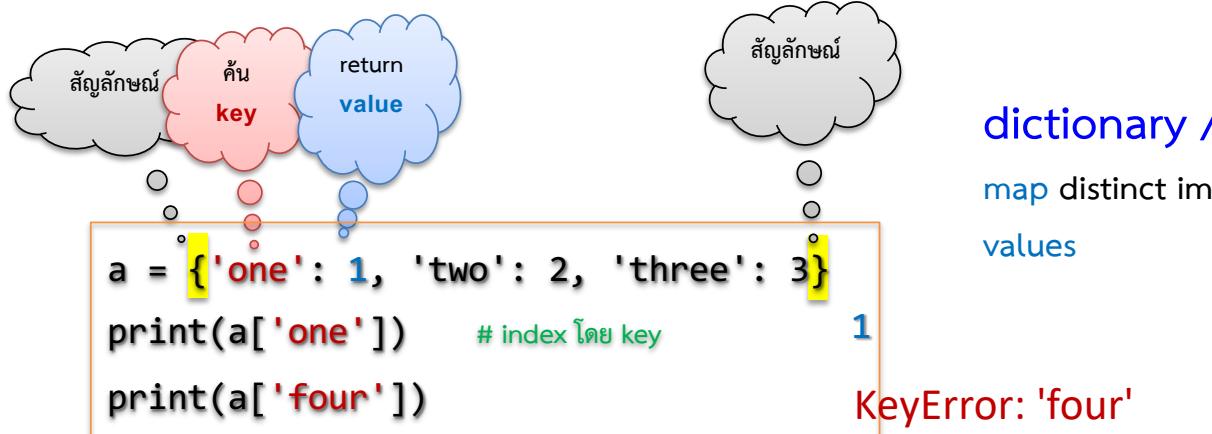
containment check

val not in s

non-containment check

```
c = '{'  
c in ['(', '{', '[']           returns True  
  
c in '({['                   returns True  
  
3 in range(0, 10)            returns True  
3 in range(0, 10, 2)          returns False  
  
s = '0123456789'  
print(s[0:5:3])                03  
  
list = list(range(0,10))  
print(list[0:10:2])            [0, 2, 4, 6, 8]
```

dict class



mutable

dictionary / mapping

map distinct immutable **keys** กับ
values

```
b = dict(one=1, two=2, three=3)
c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
d = dict([('two', 2), ('one', 1), ('three', 3)])
e = dict({'three': 3, 'one': 1, 'two': 2})
```

```
print(a == b == c == d == e)
print(a is b)
```

same obj
?

True

False

set class

```
KMITLskirt = {'blue', 'black'}
```

```
print(KMITLskirt)  
print('blue' in KMITLskirt)  
print('blue' not in KMITLskirt)
```

```
{'black', 'blue'}  
True  
False
```

set เก็บของ คละ type ได้ ไม่ซ้ำกัน ไม่มีลำดับ
ของเป็น mutable หรือ immutable ก็ได้
แต่ set เป็น mutable



```
a = set('abc')  
b = set('ade')  
  
print(a)      {'a', 'c', 'b'}  
print(b)      {'a', 'e', 'd'}  
  
print(a | b)  {'a', 'c', 'e', 'b', 'd'}  
print(a & b)  {'a'}  
print(a - b)  {'c', 'b'}  
⋮
```

A grey cloud-shaped graphic containing the word "difference" in black text.

A grey cloud-shaped graphic containing the word "union" in black text.

ข้อดีของ set : ข้างในใช้ hash table -> optimized checking method

Set and Dictionary Operators

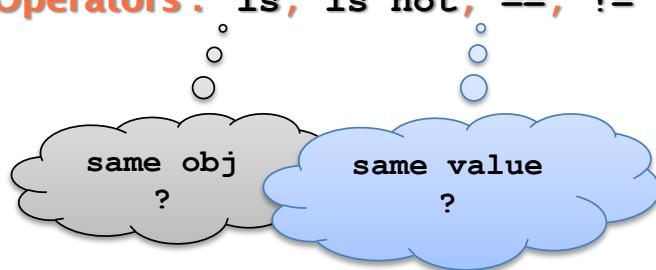
Set and Dictionary Operators :

key in s	containment check
key not in s	non-containment check
s1 == s2	s1 is equivalent to s2
s1 != s2	s1 is not equivalent to s2
s1 <= s2	s1 is subset of s2
s1 < s2	s1 is proper subset of s2
s1 >= s2	s1 is superset of s2
s1 > s2	s1 is proper superset of s2
s1 s2	the union of s1 and s2
s1 & s2	the intersection of s1 and s2
s1 - s2	the set of elements in s1 but not s2
s1 ^ s2	the set of elements in precisely one of s1 or s2

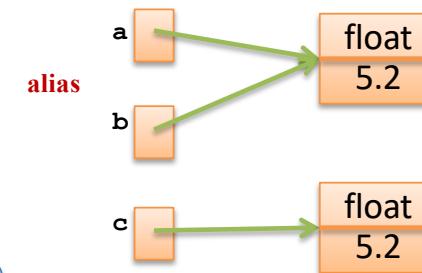
Operators

Logical Operators : and, or, not

Equality Operators : is, is not, ==, !=



Comparison Operators : <, <=, >, >=



`a is b` returns `True`

`a is c` returns `False`

`a == c` returns `True`

`3 <= 5 <= 10` returns `True`

`'ad' < 'ad'` returns `False`

`5 < 'a'` exception raise : `'TypeError'`

Operator Precedence

lowest precedence



highest precedence

Operator	Description
<code>lambda</code>	Lambda expression
<code>if - else</code>	Conditional expression
<code>or</code>	Boolean OR
<code>and</code>	Logical Operators
<code>not x</code>	Boolean AND
<code>in, not in, is, is not, <, <=, >, >=, !=, ==</code>	Comparisons, Membership & Identity test Operators
<code> </code>	Bitwise OR
<code>^</code>	Bitwise Operators
<code>&</code>	Bitwise XOR
<code><<, >></code>	Bitwise AND
<code><<, >></code>	Shifts
<code>+, -</code>	Addition and subtraction
<code>*, @, /, //, %</code>	Arithmetics Operators
<code>+x, -x, ~x</code>	Multiplication, matrix multiplication division, remainder
<code>**</code>	Positive, negative, bitwise NOT
<code>await x</code>	Exponentiation
<code>x[index], x[index:index], x(arguments...), x.attribute</code>	Sequence Operators
<code>(expressions...), [expressions...], {key: value...}, {exp ressions...}</code>	Subscription, slicing, call, attribute reference Binding or tuple display, list display, dictionary display, set display

Data Type Summary

- Lists, Tuples, and Dictionaries can store any type (including other lists, tuples, and dictionaries!)
- Only lists and dictionaries are mutable
- All variables are references

Data Type Summary

- Integers: 2323, 3234L
- Floating Point: 32.3, 3.1E2
- Complex: 3 + 2j, 1j
- String: s = '123'
- Lists: l = [1,2,3]
- Tuples: t = (1,2,3)
- Dictionaries: d = {'hello' : 'there', 2 : 15}

pass

pass คือไม่ทำอะไร
 เช่น รอไว้ก่อน ทำที
 หลัง

```
class myClass:  
    pass  
  
def myFun(n):  
    pass
```