Quick Python

Lecturers :

Boontee Kruatrachue Room no. 913
Kritawan Siriboon Room no. 913

Why Python ?

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

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

   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

https://snakify.org/

Study by Yourself : many options

https://snakify.org/lessons/lists/steps/1/

Theory | Steps | Problems

1. How to read and write in Python

Every program is eventually a data processor, so we should know how to input and output data within it. There exists a function, `print()`, to output data from any Python program. To use it, pass a comma separated list of arguments that you want to print to the `print()` function. Let's see an example. Press "run" and then "next" to see how the program is being executed line by line:

run ▶ step by step ☐

```
1  print(5 + 10)
2  print(3 * 7, (17 - 2) * 8)
3  print(2 ** 16)  # two stars are used for exponentiation
4  print(37 / 3)  # single forward slash is a division
5  print(37 // 3)  # double forward slash is an integer di
6          # it returns only the quotient of the division
7  print(37 % 3)  # percent sign is a modulus operator
8          # it gives the remainder of the left value divi
```

Output:

```
2  21 120
3  65536
4  12.333333333333334
5  12
6  1
7
```

1. Input, print and numbers
2. Conditions: if, then, else
3. Integer and float numbers
4. For loop with range
5. Strings
6. While loop
7. Lists
8. Functions and recursion
9. Two-dimensional lists (arrays)
10. Sets
11. Dictionaries

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

prompt >>> รอรับคำสั่ง

```
>>>3 + 5
8
>>> 20 - 2*2
16
>>>print('Hello')  # string ใน '…' หรือ "…"
Hello
>>>
```

Output

เริ่ม comment จาก #
จนจบบรรทัด

Compiler แปลง HLL code **ทั้งหมด**
เป็น obj code แล้วจึง run
เช่น **c, c++, pascal**

Interactive Mode and Script Mode

Interactive Mode ใช้แบบเครื่องคิดเลข

```
>>> miles = 26.2
>>> miles * 1.61
42.182
```

- Assignment : ไม่แสดงผลให้เห็น
- Expression : interpreter ประมวลผลแล้วแสดงผล

Script Mode

```
miles = 26.2
print(miles * 1.61)
```

- เขียน code ใน script แล้ว run
- Assignment : ไม่แสดงผลให้เห็น
- expression ไม่แสดงผลให้เห็น ต้อง print

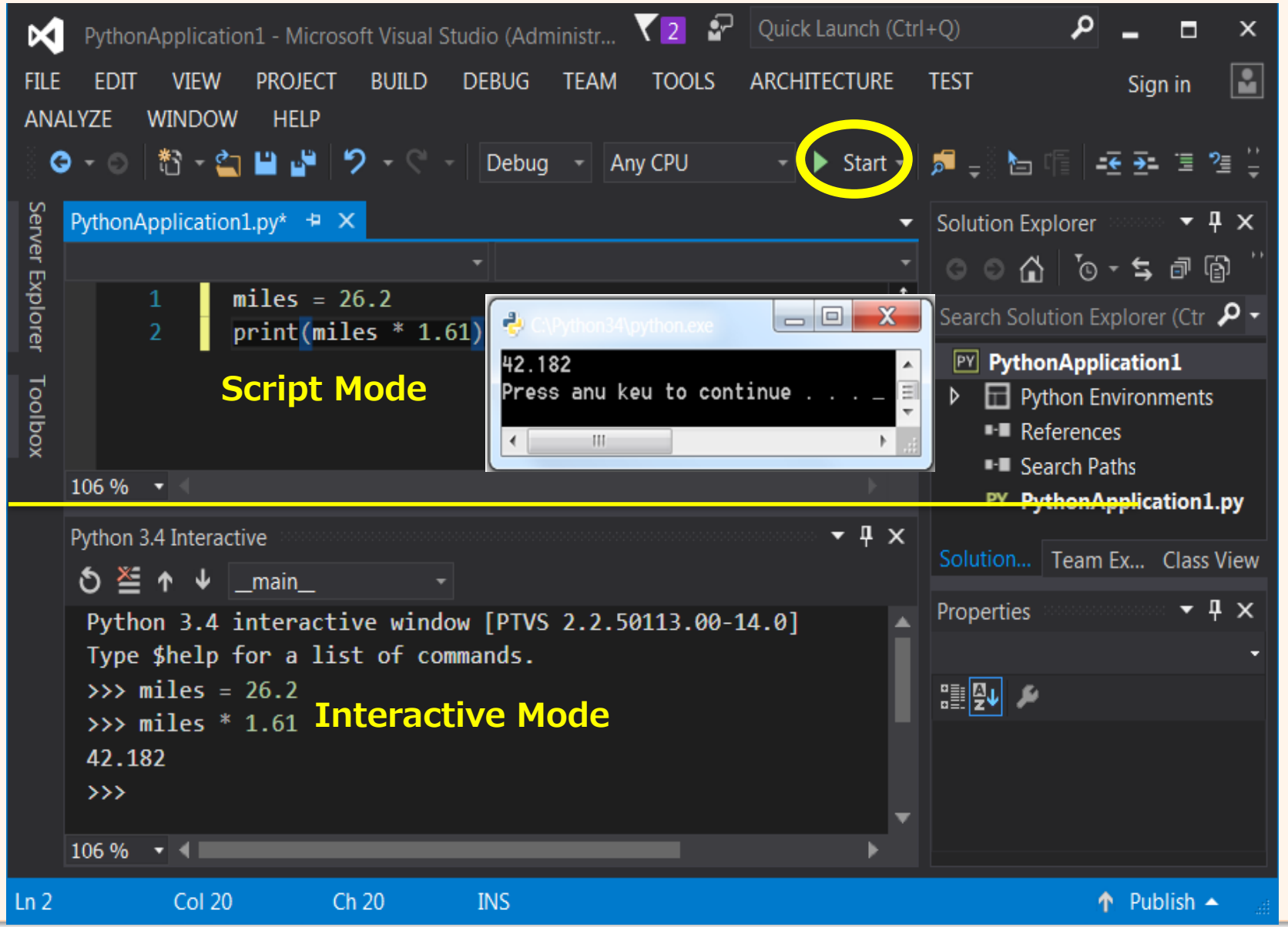
Expression → อะไรก็ตามที่ return ค่า (combination ของ ค่า,variables ,operations)

เช่น 80.2, 5 + x – average(x,y,z), "Hello"

Statement → ส่วนของ code ซึ่ง Python interpreter ประมวลผลได้

ข้อแตกต่าง → expression มีค่า แต่ statement ไม่มีค่า

Microsoft Visual Studio



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 ออกมาใช้

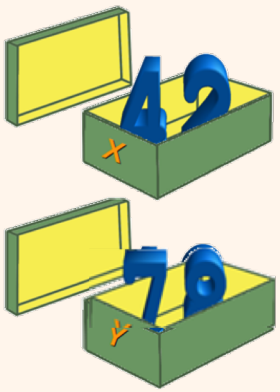
```
x = 5
y = x * 7
```

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;

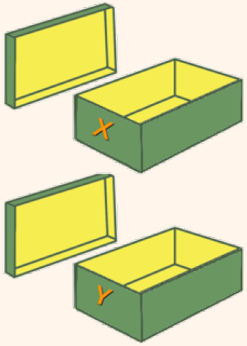
y = 42;

y = 78;

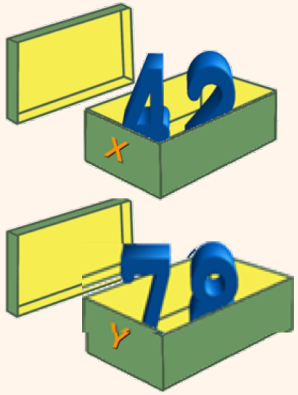
C, C++, Java Variable

- The way variables are implemented in C, C++ or Java.
- Variable names have to be declared in these languages before they can be used.

```
int x;
int y;
```

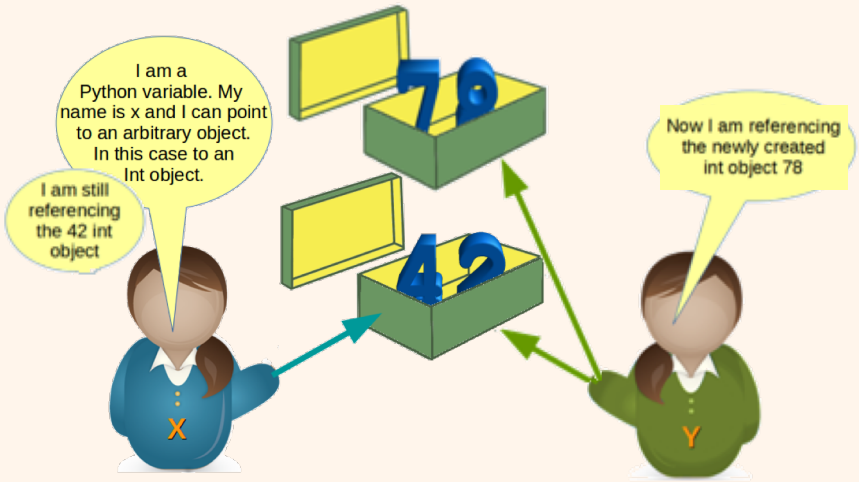


- Such declarations make sure that the program reserves memory for two variables with the names x and y. The variable names stand for the memory location.
- x = 42;
  y = 42;
- y = 78;



Python Variable

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



id Function( )

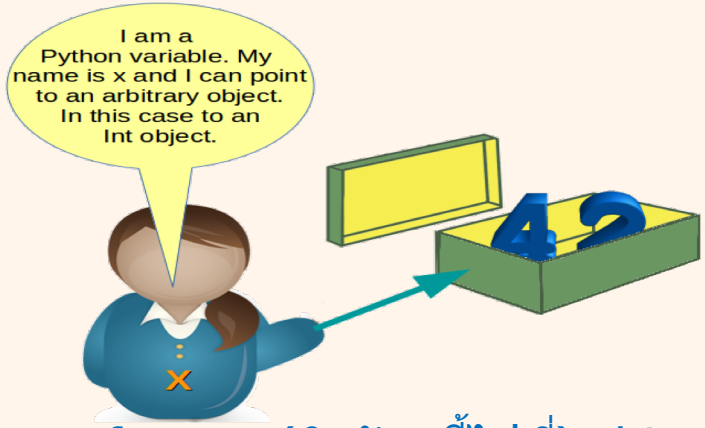
```
>>> x = 42
>>> id(x) 10107136
>>> y = x
>>> id(x), id(y) (10107136, 10107136)
>>> y = 78
>>> id(x), id(y) (10107136, 10108288)
>>>
```

Python Variable

Python variables:

- เป็น dynamic typing model
- ไม่ได้ถูก declared แต่ถูกสร้างโดยการ assign ค่าให้มันครั้งแรก เช่นในตย. assign ค่า 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

Python Variable

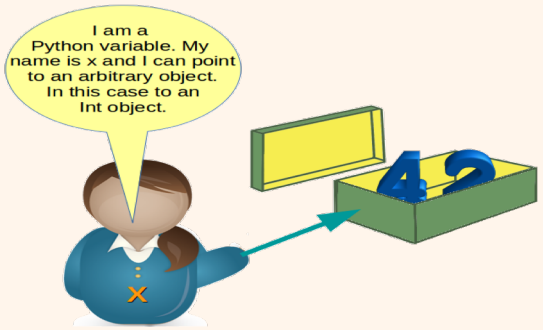
- Python variables follow a dynamic typing model

Identifier ไม่ declared type
แต่
object ที่มัน reference มี type

- Are not declared, and are created by being assigned .

The variable is created the first time when you assign it a value.

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



- have object references as values
- Type information is with the object, not the variable
- Everything in Python is an object

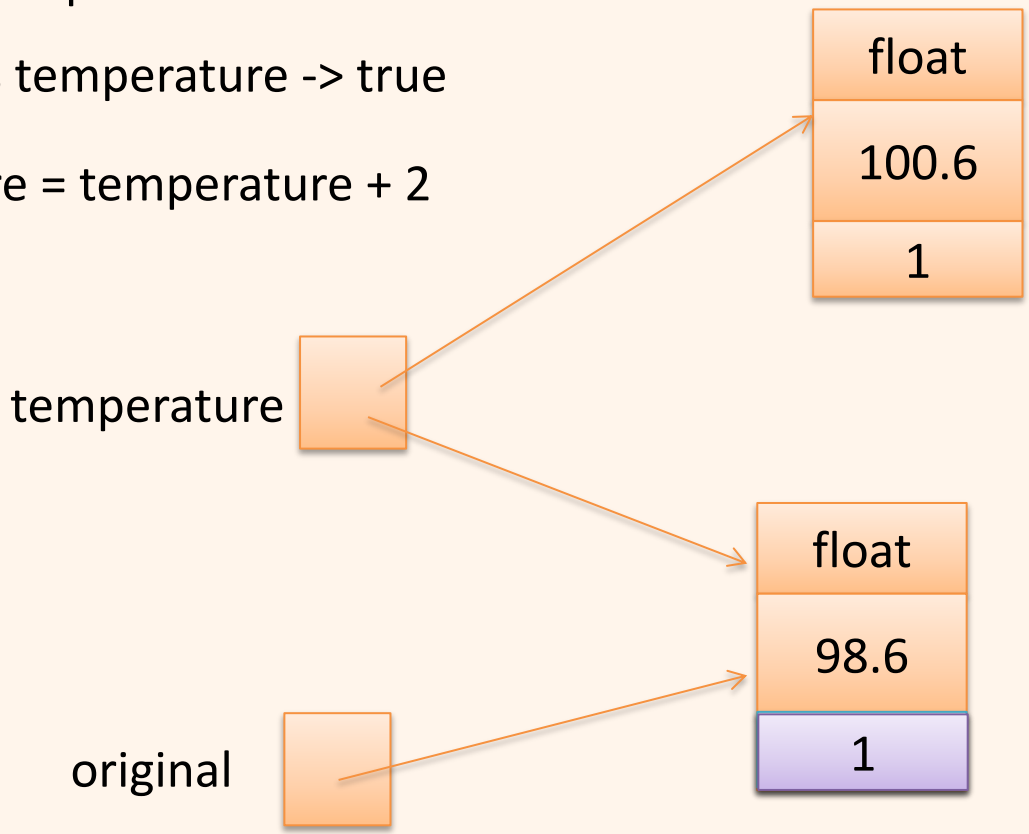
Closer Look

temperature = 98.6

original = temperature

original is temperature -> true

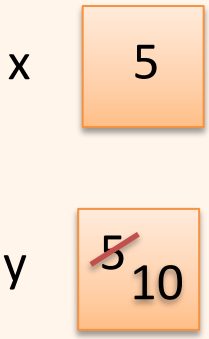
temperature = temperature + 2



C vs Python

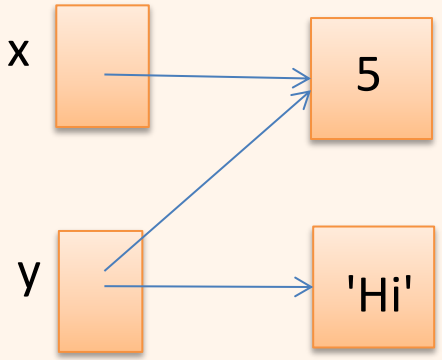
C

```
int x = 5;

int y = x;

y = 10;
```



Python

```
x = 5;

y = x

y = 'Hi'
```



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:

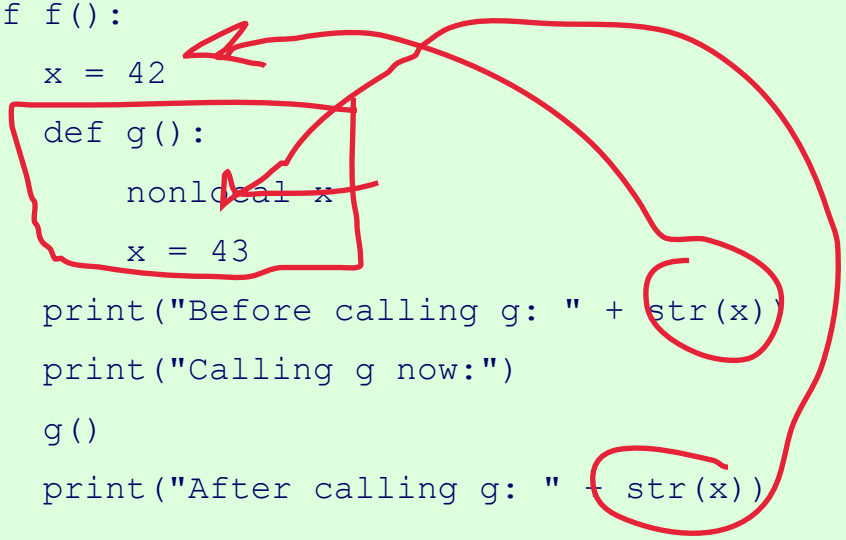
Global Variables in Nested Functions

```
def f():
    x = 42
    def g():
        global x
        x = 43
    print("Before calling g: " + str(x))
    print("Calling g now:")
    g()
    print("After calling g: " + str(x))
f()
print("x in main: " + str(x))
```

```
Before calling g: 42
Calling g now:
After calling g: 42
x in main: 43
```

nonlocal Variables

```
def f():
    x = 42
    def g():
        nonlocal x
        x = 43
    print("Before calling g: " + str(x))
    print("Calling g now:")
    g()
    print("After calling g: " + str(x))


x = 3
f()
print("x in main: " + str(x))
```



```
Before calling g: 42
Calling g now:
After calling g: 43
x in main: 3
```

comment #

```
def f():
    #x = 42
    def g():
        nonlocal x
        x = 43
    print("Before calling g: " + str(x))
    print("Calling g now:")
    g()
    print("After calling g: " + str(x))

x = 3
f()
print("x in main: " + str(x))
```

```
File "example3.py", line 4
    nonlocal x
SyntaxError: no binding for nonlocal 'x' found
```

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)
```

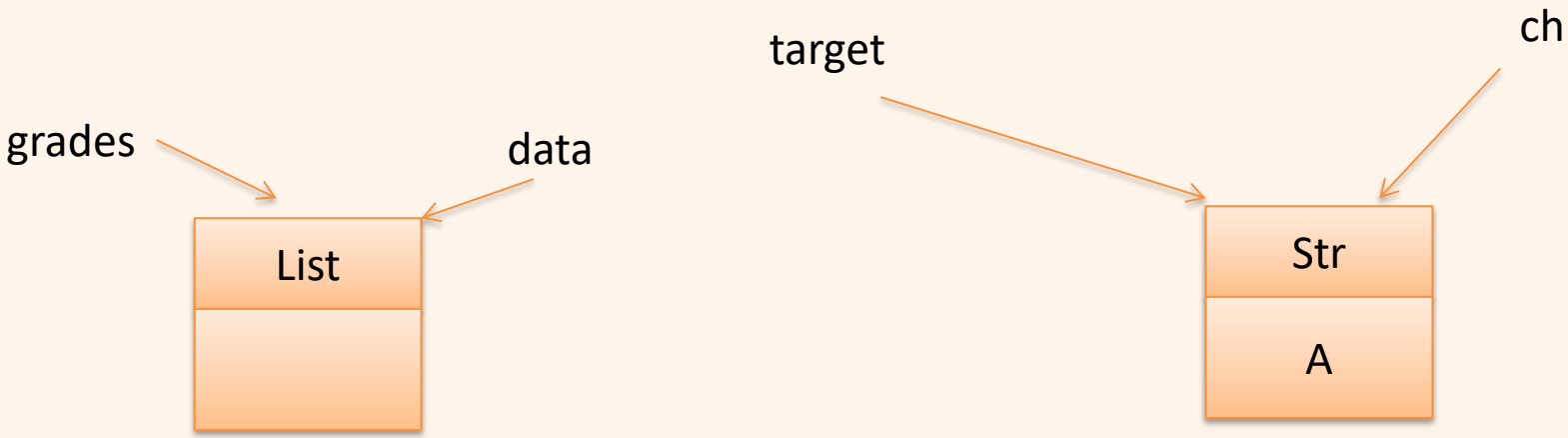
Parameter Passing

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 )
```

```
data = grades
target = 'A'
```



reassigning a new value to parameter, setting data = [ ], breaks the alias.

Functions are First Class Objects

- Can be assigned to a variable
- Can be passed as a parameter
- Can be returned from a function
- Functions are treated like any other variable in Python, the **def** statement simply assigns a function to a variable
- programming languages terminology, ***first-class objects*** : instances of a type that can be assigned to an identifier, passed as a parameter, or returned by a function.

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'
```

Functons as Parameters

funcasparam.py

```
def foo(f, a) :
    return f(a)

def bar(x) :
    return x * x
```

```
>>> from   funcasparam import *
>>> foo(bar, 3)
9
```

Note that the function foo takes two parameters and applies the first as a function (function can be assigned to a variable and can be passed as a parameter) with the second as its parameter

return

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

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

output

```
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
```

Built-in Types

Python :

Object Oriented Programming (OOP)

class เป็นพื้นฐานของทุก data type

Commonly-used built-in class (type) :

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

Complex

```
7 + 3j
```

j / J imaginary part

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 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]
```

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

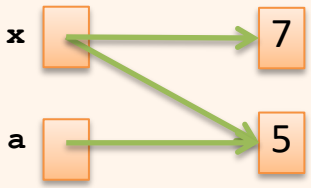
Passing Immutable Variable

output

```
def f(x):
    print('id(x)', id(x))
    x += 2
    print('id(x)', id(x))
```

```
id(a) 499765776
id(x) 499765776
id(x) 499765808
5
```

```
a = 5
print('id(a)', id(a))
f(a)
print(a)
```



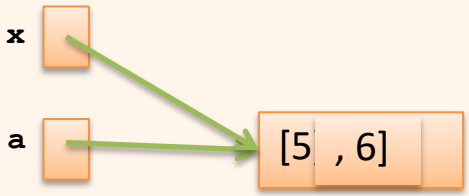
Passing Mutable variable

output

```
def f(x):
    print('id(x)', id(x), x)
    x.append(6)
    print('id(x)', id(x), x)
```

```
id(a) 37745568 [5]
id(x) 37745568 [5]
id(x) 37745568 [5, 6]
id(a) 37745568 [5, 6]
```

```
a = [5]
print('id(a)', id(a), a)
f(a)
print('id(a)', id(a), a)
```



Default Argument

Default argument :
ให้ = ค่านี้ เมื่อไม่มีการ pass ค่ามา

ค่า default จะถูกสร้างขึ้น**ครั้งเดียว**
ณ function definition ใน scope ที่ define functon
ต้อง**ระวัง** เมื่อเป็น mutable type

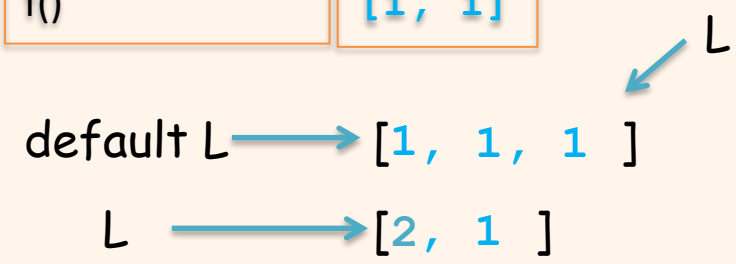
```
def f( L = [] ):
    print(L)
    L.append(1)

f()
f()
f([2])
f()
```

output

```
[]
[1]
[2]
[1, 1]
```

ถ้า f() เป็น constructor ของ stack
จึง init empty stack เพียงครั้งแรกเท่านั้น ทางแก้ →

```
def f(L = None):
    if L is None:
        L = []
    else :
        pass

    L.append(1)

f()
f()
```



if

```
if test expression:
    statement(s)
```

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

```
>>> 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
```

: ต้องมีต่อไปเป็น body block ของบรรทัดนี้

condition ไม่ต้องอยู่ใน ( )

- 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

```
max = (a > b) ? a : b;
```

C

```
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))
```

```
import random
n = 20
to_be_guessed = int(n * random.random()) + 1
guess = 0
while guess != to_be_guessed:
    guess = int(input("New number: "))
    if guess > 0:
        if guess > to_be_guessed:
            print("Number too large")
        elif guess < to_be_guessed:
            print("Number too small")
    else:
        print("Sorry that you're giving up!")
        break
else:
    print("Congratulation. You made it!")
```

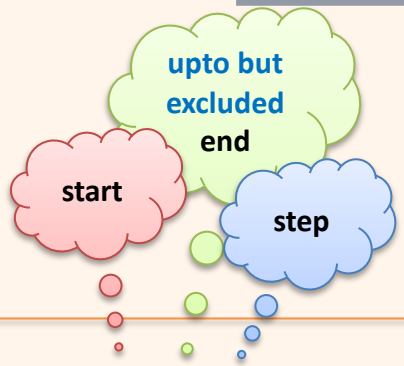
random() of random module that is imported

```
$ python3 number_game.py
New number: 12
Number too small
New number: 15
Number too small
New number: 18
Number too large
New number: 17
Congratulation. You made it!
$
```

If a loop is left by break, the else part is not executed.

range class

`range (a, b, s)` return sequence ของตัวเลข `a, a + 1s, a + 2s, a + 3s, . . ., before b`



range type :

เป็น immutable sequence ของ numbers นิยมสำหรับ loop : for

```
print(list(range(0, 9, 3)))
print(list(range(5)))
print(list(range(1, 5)))
print(list(range(-10, -50, -20)))
print(list(range(1, 0)))
```

```
[ 0, 3, 6 ]
[ 0, 1, 2, 3, 4 ]
[ 1, 2, 3, 4 ]
[ -10, -30 ]
[]
```

argument n ตัวเดียว ตั้งแต่ 0 ไป n ตัว step 1

default step = 1

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 = 'abcdefghi'
```

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

s[0],s[1],…,s[8]

```
abcdefghi
```

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

```
beh
```

for, list


```
for val in sequence:
    Body of for
```

```
list = [2, 1, 3, 4]
for ele in list:
   print(ele) # แต่ละ iteration ตัวแปร ele = ค่าแต่ละค่าใน list
```

```
2
1
3
4
```

```
for i in range(len(list)):

   print(list[i])
```

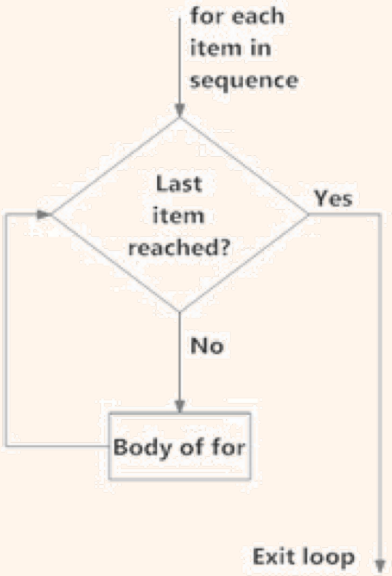


Fig: operation of for loop

```
# 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`

for : collection-controlled loop

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

loop บน slice copy ของ words, ถ้าเปลี่ยนเป็น words เฉยๆ จะ infinite loop

```
>>> for w in words[:]:  # Loop over a slice copy of the entire
list.
...     if len(w) > 6:
...         words.insert(0, w)
...
>>> words
['defenestrate', 'cat', 'window', 'defenestrate']
```

slicing format

start : excluding end : step

```
>>> s = '0123456789'
>>> s[1:3]
'12'
>>> s[2:9:2]
'2468'
```

default start = 0 (ไม่ใส่ หมายถึง ตัวแรก)

default excluding end = len(string) (ไม่ใส่ หมายถึง ความยาวของ string)

defualt step = 1 (ไม่ใส่ หมายถึง 1)

With `for w in words:`, the example would attempt to create an infinite list, inserting defenestrate over and over again.

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

continue
ออกจาก iteration นั้นทันที ไปทำ iteration ถัดไป

break
ออกจาก loop ใกล้สุดที่ล้อมมัน ไม่ทำ else ของ loop ด้วย

else ของ loop
ทำเมื่อทำจนหมด for หรือ while condition เป็น false

```
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
```

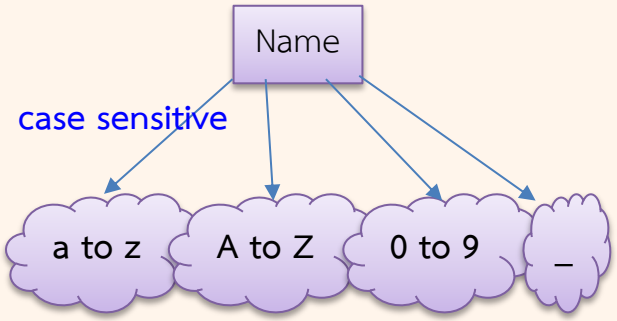
To loop over two or more sequences at the same time, the entries can be paired with the zip() function

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print('What is your {0}?  It is {1}.'.format(q, a))
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

looping through dictionaries, the key and corresponding value can be retrieved at the same time using the items() method

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
...     print(k, v)
...
gallahad the pure
robin the brave
```

Variable (Name, Identifier)



_var1	if	keyword
myVar	elif	keyword
num	9i	ขึ้นต้นด้วย 0--9

ไม่ขึ้นต้นด้วยตัวเลข

ไม่เป็น keywords

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

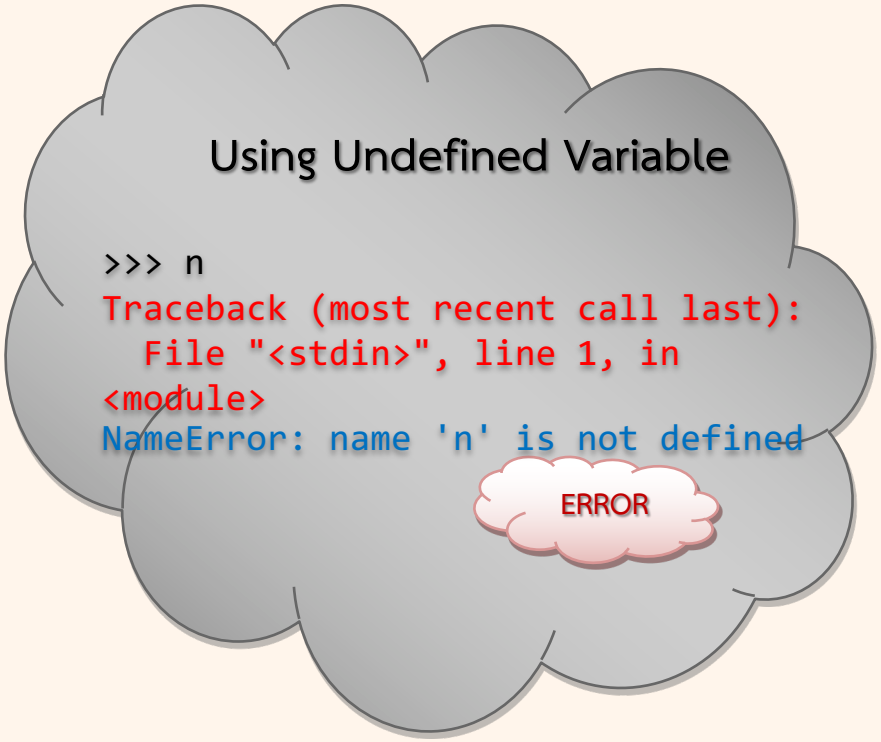
List [0, 1] เก็บของตามลำดับ

ใช้ index access ของที่เก็บ

ไล่จาก หน้า -> หลัง ตัวแรกเริ่มจาก index 0, 1, ...

ไล่จาก หลัง -> หน้า ตัวแรกเริ่มจาก index -1, -2, ...

Undefined Name

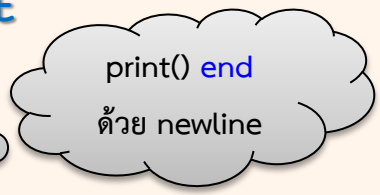


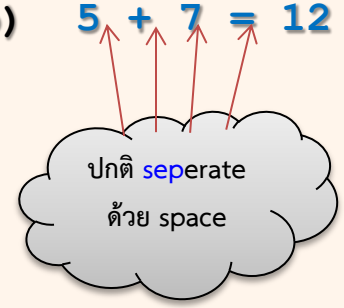
print()

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

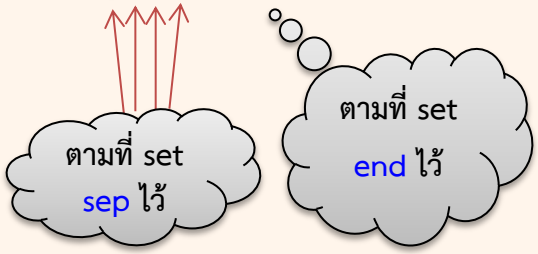
output

```
5
5 + 7 = 12
```

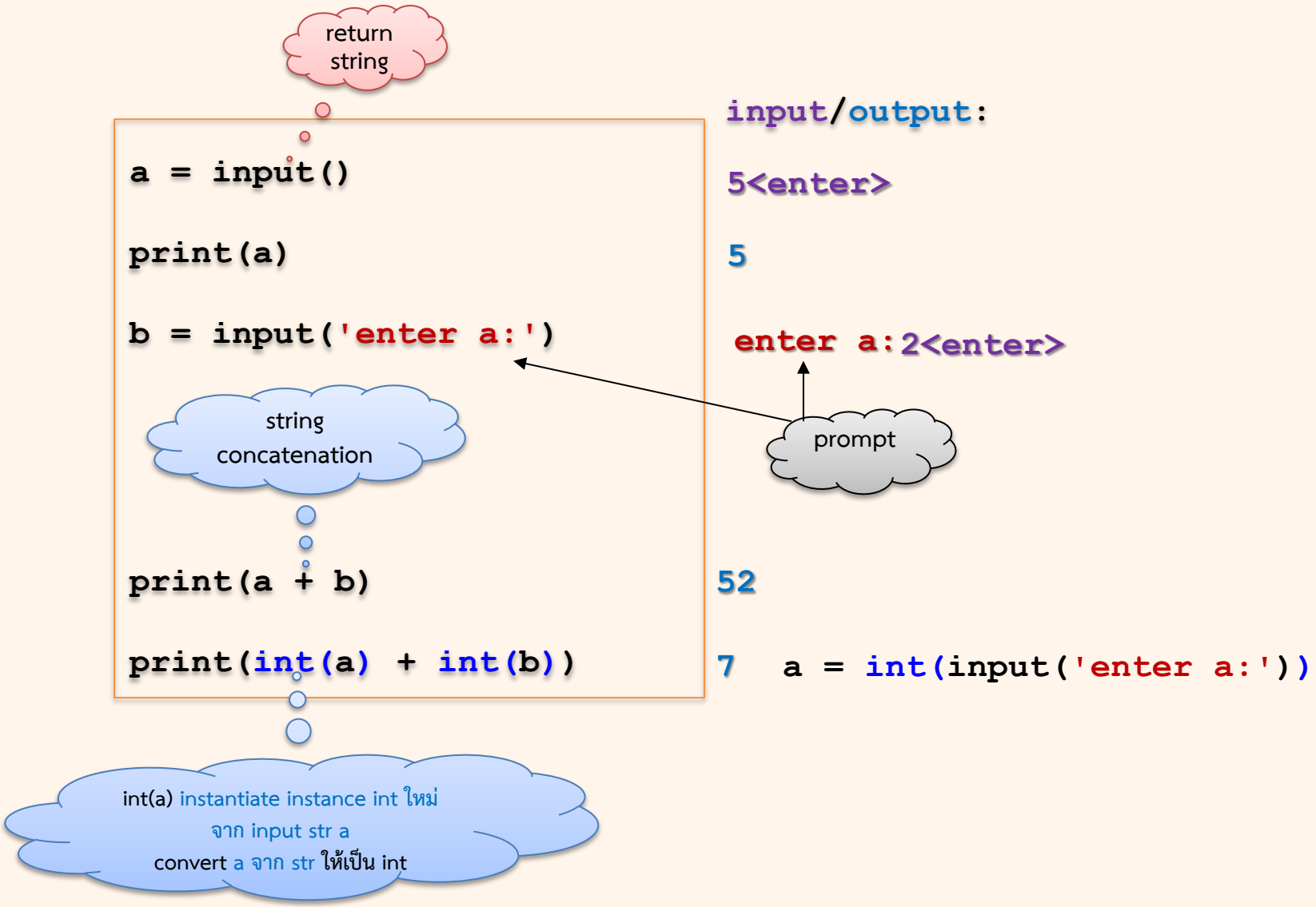





```
>>> print(a, '+', b, "=", a+b, sep = '',end = '***\n')
5+7=12***
```



input(), int()


return string
a = input()
print(a)
b = input('enter a:')
string concatenation
print(a + b)
print(int(a) + int(b))
int(a) instantiate instance int ใหม่ จาก input str a convert a จาก str ให้เป็น int
input/output:
5<enter>
5
enter a:2<enter>
prompt
52
7
a = int(input('enter a:'))

```
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, 2, 1]
```

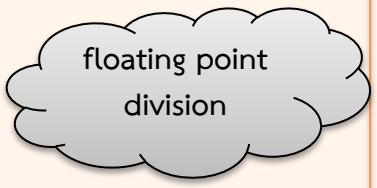
lst = map(int, raw_input().split())

Python2
raw_input() reads a whole line from the input (stopping at the \n) as a string.
.split()creates a list of strings by splitting the input into words.
map(int, ...) creates integers from those words.
Python3
raw_input has been renamed to input
map returns an iterator rather than a list
so a couple of changes need to be made:

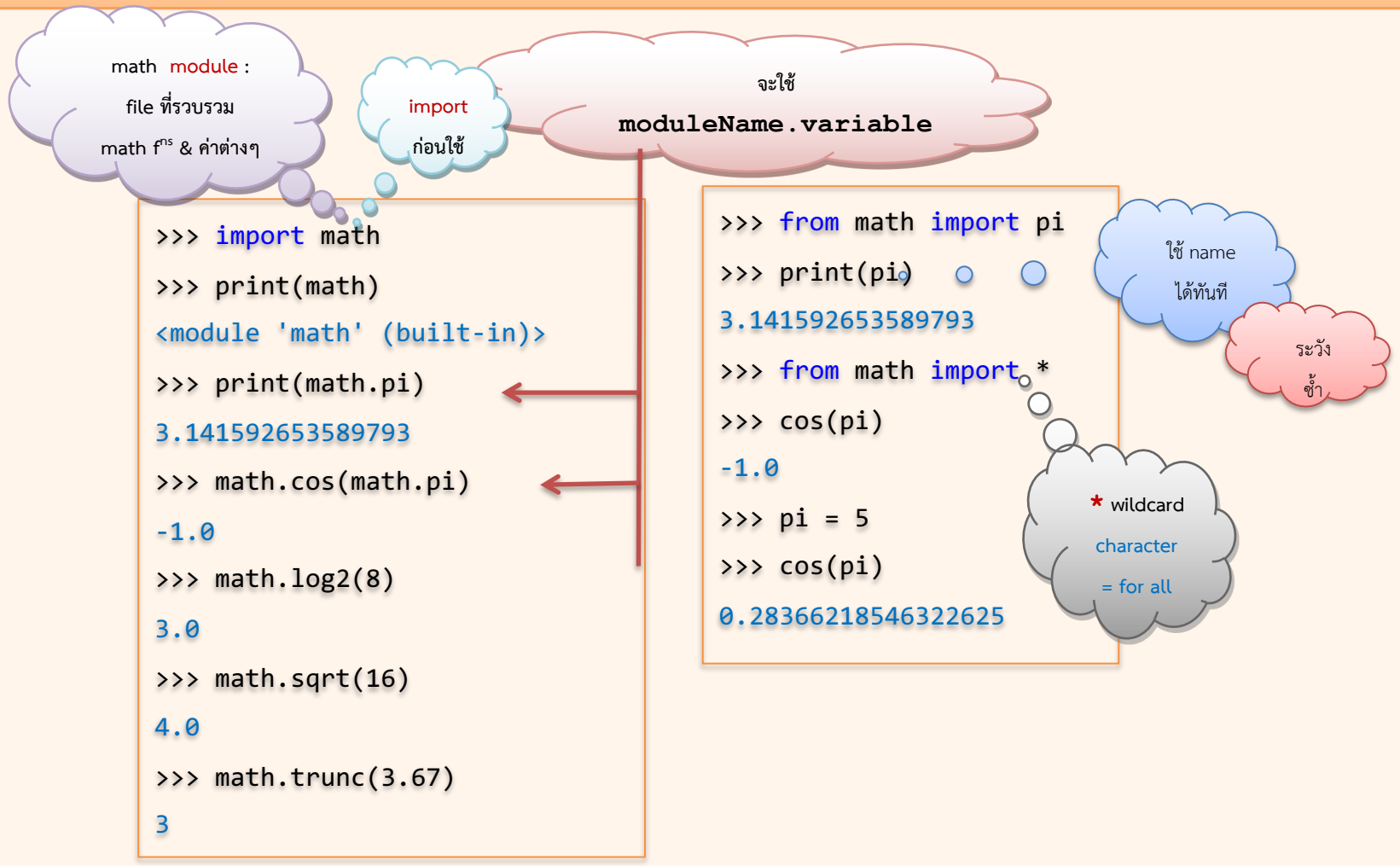
lst = list(map(int, input().split()))

Arithmetic Operators

```
>>> (20 - 2*2)/4
4.0
>>> 5/2
2.5
>>> 5//2
2
>>> 5%2
1
>>> 2**3
8
>>> 3.5 - 2
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

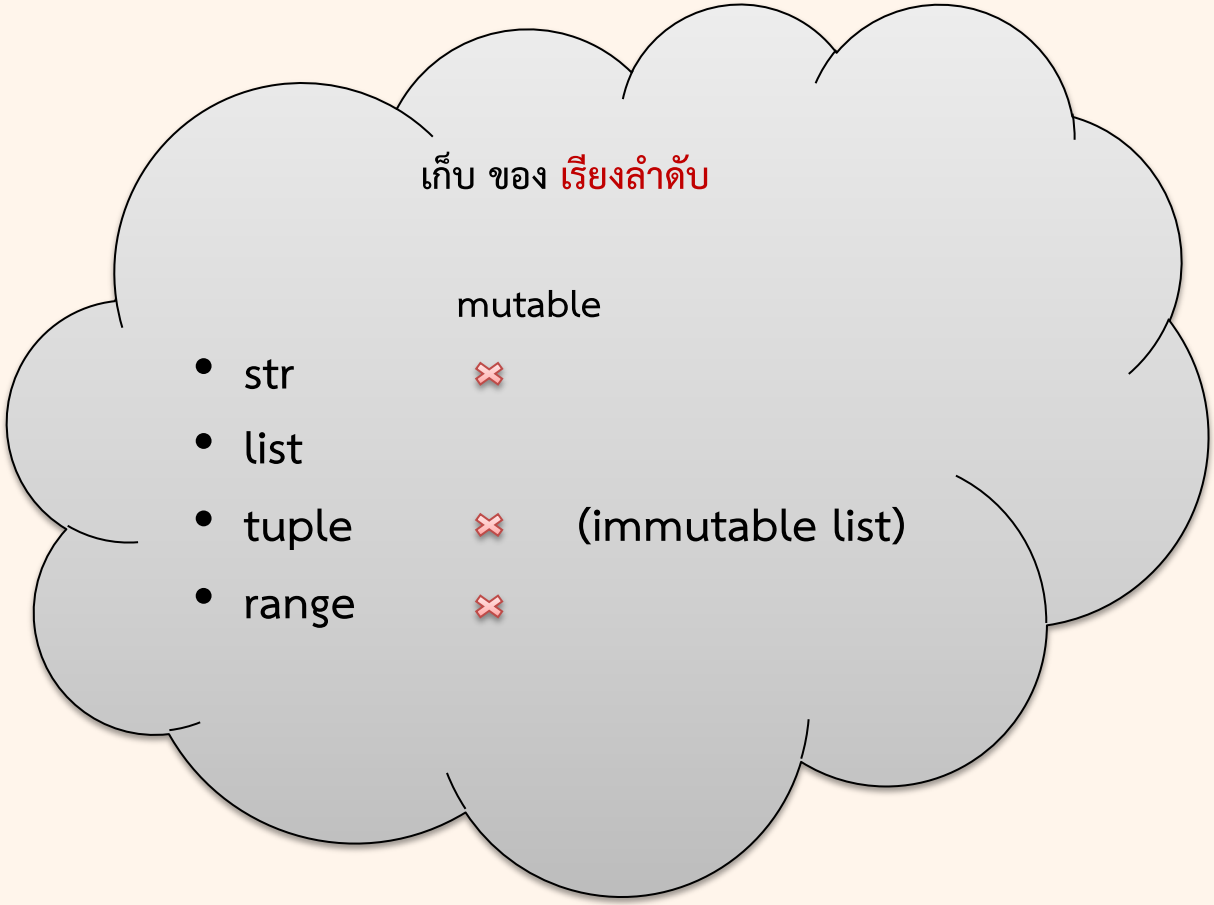
Arithmetic Operator Precedence

lowest precedence

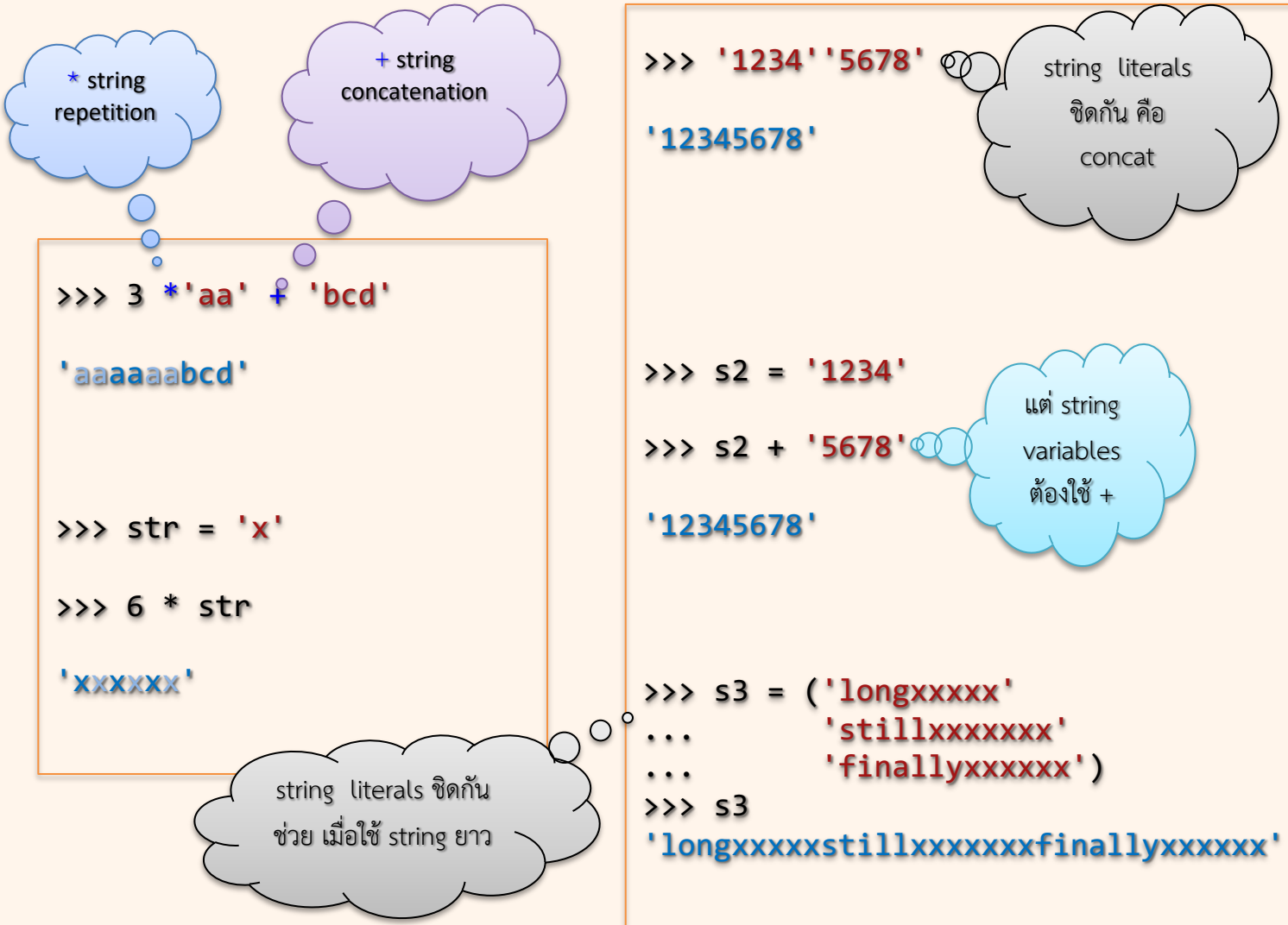
highest precedence

	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	
abs(-5)	5	absolute
int(5.2)	5	int conversion
float(5)	5.00	float conversion
divmod(5,3)	(1,2)	divmod pair
pow(2, 3)	8	
2 ** 3	8	

Sequence classes

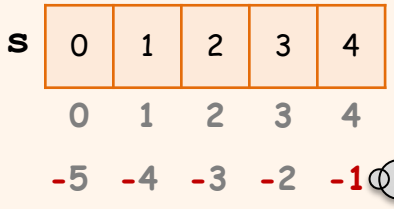


string Repetition & Concatenation



string Indexing (subscript), len()

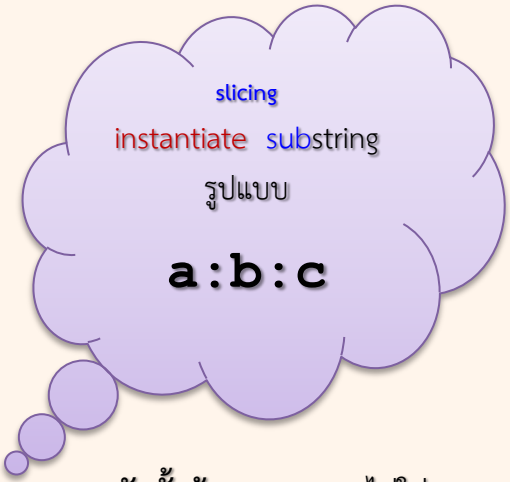
```
>>> s = '01234'
>>> s[0]
'0'

>>> 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


>>> len(s)
5
```



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

len() returns length

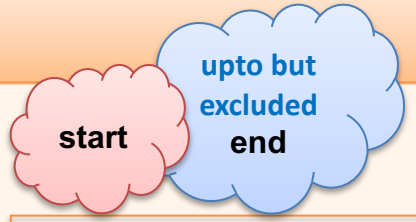
string slicing



a คือ index **ตัวตั้งต้น** (ไม่ใส่ หมายถึง ตัวแรก)

b " **จบที่ไม่รวม b** (ไม่ใส่ หมายถึง ความยาวของ string)

c บอกว่า **step ถัดไปกี่ตัว** (ไม่ใส่ หมายถึง 1)

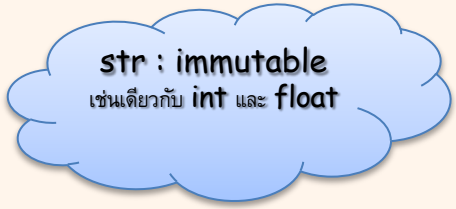



```
>>> s = '0123456789'
>>> s[1:3]
'12'
>>> s[2:9:2]
'2468'
>>> s[:3]
'012'
>>> s[2:]
'23456789'
>>> s[-7:8]
'34567'
>>> 'Hello' + s[-7:8]
'Hello34567'
```

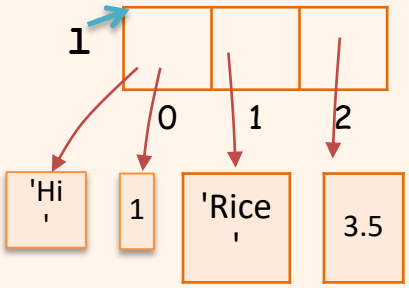
step

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]
     0      1      2
    -3     -2     -1
```

ใช้ index ในการ access

ไล่จาก หน้า -> หลัง ตัวแรกเริ่มจาก index 0, 1, ...

ไล่จาก หลัง -> หน้า ตัวสุดท้ายเริ่มจาก index -1, -2, ...

```
l[0] = 'Hi'
```

เป็น mutable type

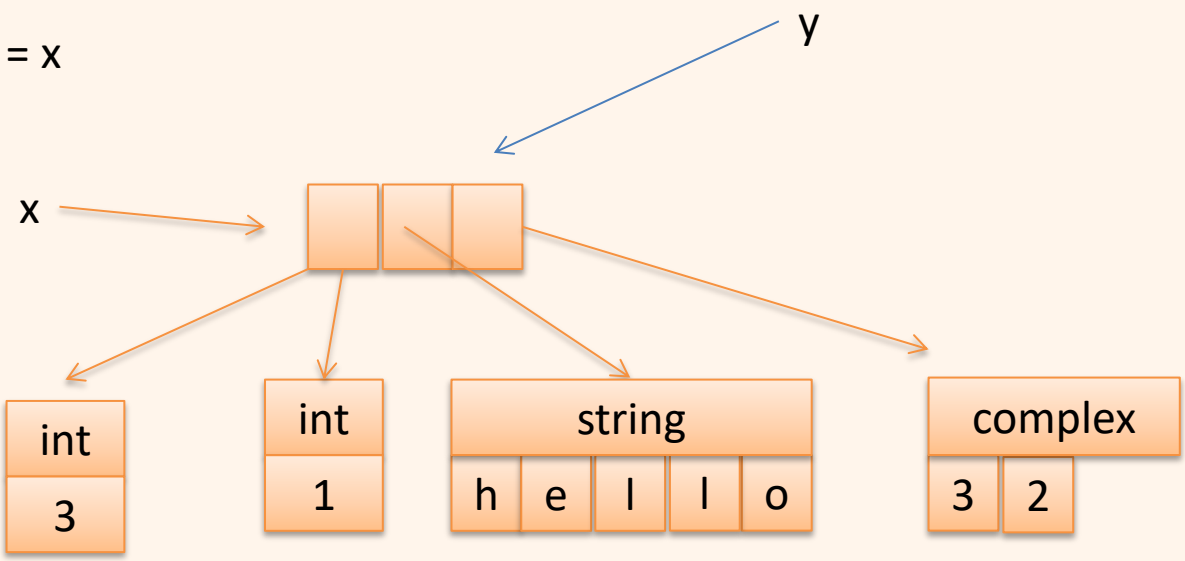
```
print(l)  ['Hi', 'Rice' , 3.5]
```

```
l2 = []  # empty list
```

List : Modifying Content

x = [1,'hello', (3 + 2j)]

y = x



x[0] = 3

x[1][0] = ‘j’

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]

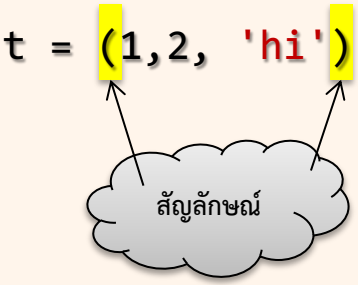
methods	ผลลัพธ์	คำอธิบาย
len(L)	4	จำนวนของใน list
max(L)	7	หา max item, ต้องเป็นไทป์เดียวกัน
min(L)	1	หา min item, ต้องเป็นไทป์เดียวกัน
sum(L)	14	หา sum ของ item, ต้องเป็น number
L.count(3)	2	นับจำนวน 3
L.index(7)	2	หา index ของ 7 ตัวแรก
L.reverse()	[3 , 7 , 3 , 1]	กลับลำดับของของ
L.clear()	[]	ทำให้เป็น 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) เรียงลำดับกัน 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
```

```
(1, 2, 'hi')
1
((1, 2, 'hi'), (1, 2, 3))
[3, 4]
TypeError
'tuple' object does not support item assignment
```



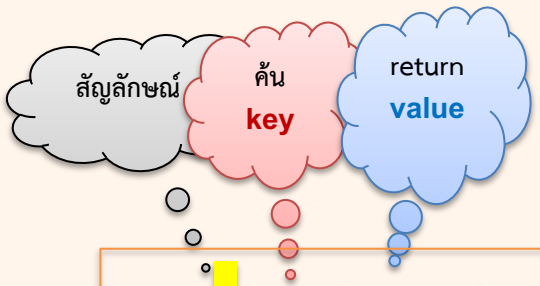
Sequence Operators

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 + 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



dictionary / mapping

map distinct immutable keys กับ values

```
a = {'one': 1, 'two': 2, 'three': 3}
print(a['one'])     # index โดย key
print(a['four'])
```

```
1
KeyError: 'four'
```

```
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'}
```

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

frozenset
immutable

```
print(KMITLskirt)                  {'black', 'blue'}
print('blue' in KMITLskirt)        True
print('blue' not in KMITLskirt)    False
```

```
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'}
```

union
n

difference

ข้อดีของ set : ข้างในใช้ hash table -> optimized checking method

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

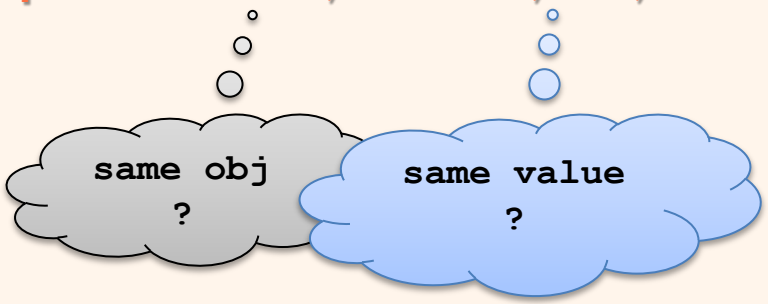
Logical Operators : `and, or, not`

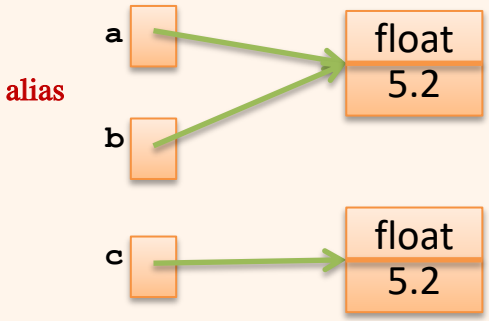
short-circuit :

```
a and b
a or b
```

if a == false ไม่ evaluate b

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





`a is b` returns `True`

`a is c` returns `False`

`a == c` returns `True`

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

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

`'ad' < 'ad'` returns `True`

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

Operator Precedence

lowest precedence

highest precedence

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

Class	Description	Default constructor	conversion	Immutable
bool	Boolean value	bool( ) -> false	bool(0)-> false bool(-1)-> true bool(li) ->true nonempty str,list bool(li) ->false empty str,list	✓
int	integer (arbitrary magnitude)	int( )-> 0	int(−3.9) -> −3. int(137) ->value 137 int(‘7f’ , 16) -> 127. (base 16)	✓
float	floating-point number	float( )-> 0.0.	float(‘ 3.14’)->3.14	✓
list	mutable sequence of objects reference	list( ) ->empty list	list(‘123’)->[‘ 1’ ,’ 2’ ,’ 3’] list(iterable type)	
tuple	immutable sequence of objects	tupple( ) -> empty tuple	tuple(‘123’)->(‘ 1’ ,’ 2’ ,’ 3’) tuple (iterable type)	✓
str	character string	str()->’’, empty str	str(10.3)->’10.3’	✓
set	unordered set of distinct immutable objects	set()->{}, empty set	set(‘1233‘) -> {‘ 1’ ,’ 2’ ,’3’ } set([1,2,3,3]) -> { 1 , 2 ,3 }	
frozenset	immutable form of set class			✓
dict	associative mapping (aka dictionary)	dict()->{}, empty dict	pairs = [('ga' , 'Irish'), ('de' , 'German')] dict(pairs)->{'ga': 'Irish', 'de': 'German'}	

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


```
class myClass:
    pass

def myFun(n):
    pass
```