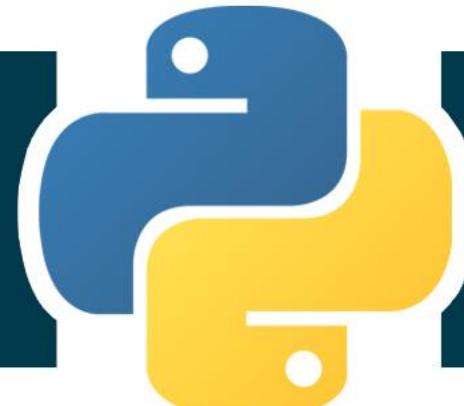


ຕະລຸຍໂຈກຍ

PYTHON



500 ນ້ອ

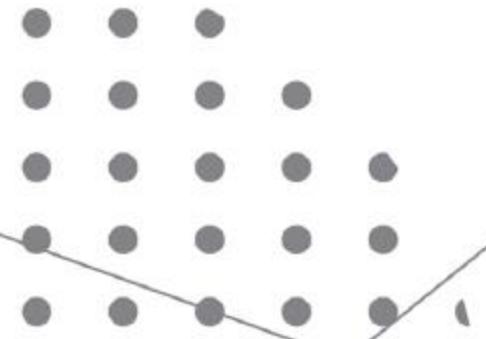
ນຖານ ຜົນປະສາກຄັດ

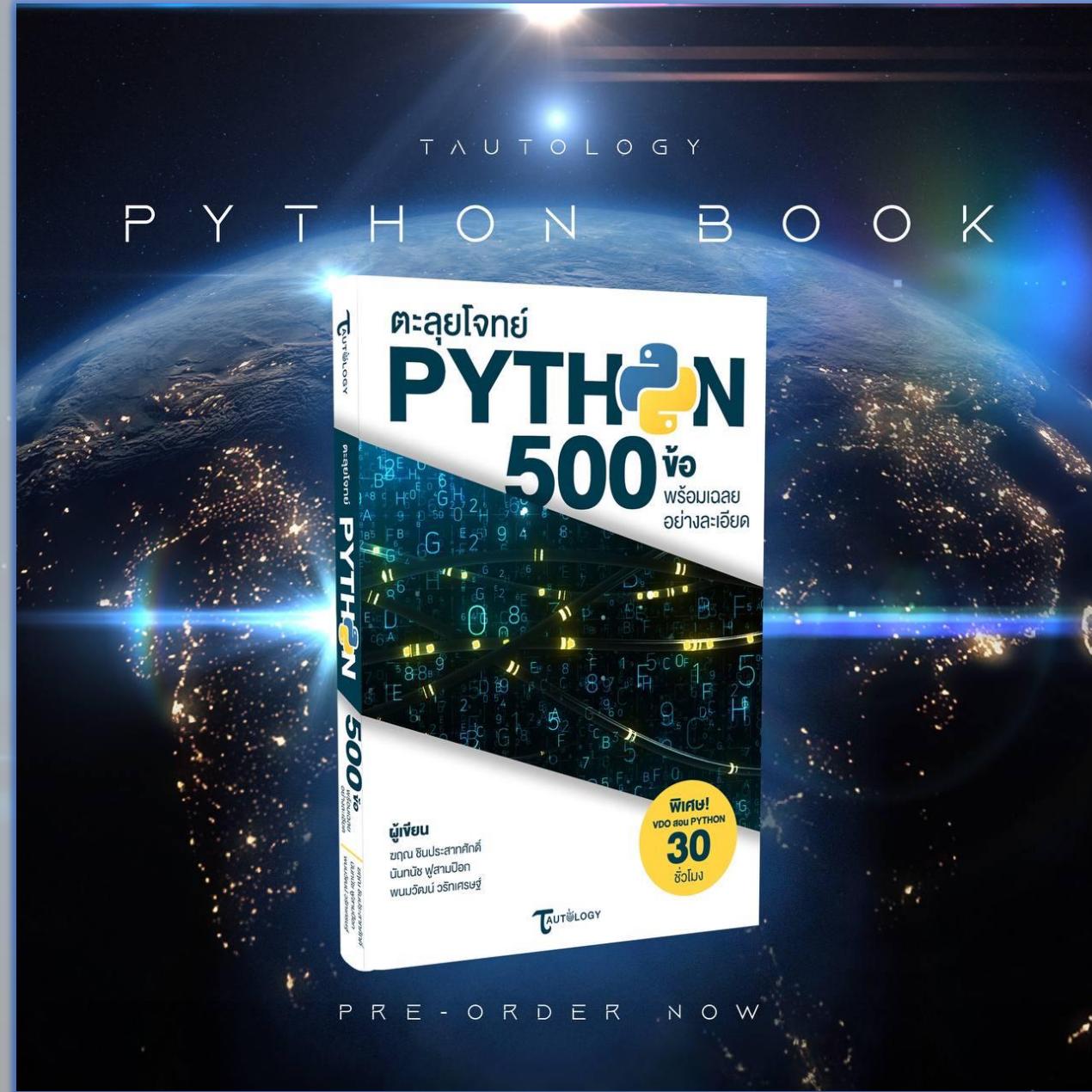


# KRIN CHINPRASATSAK

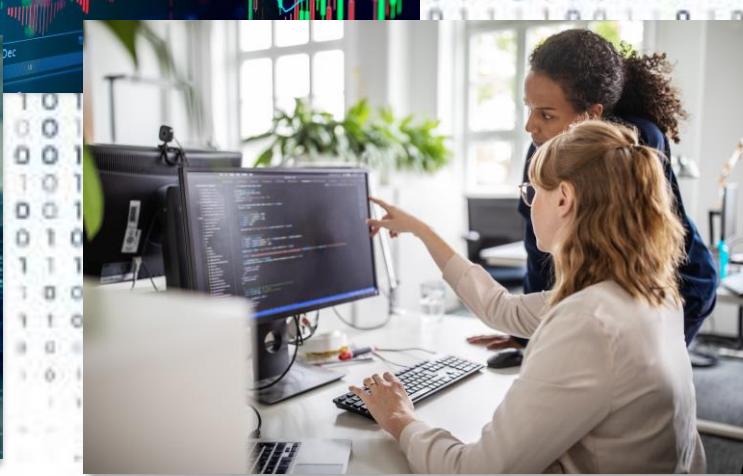
DATASCIENTIST  
INSTRUCTOR  
SPEAKER

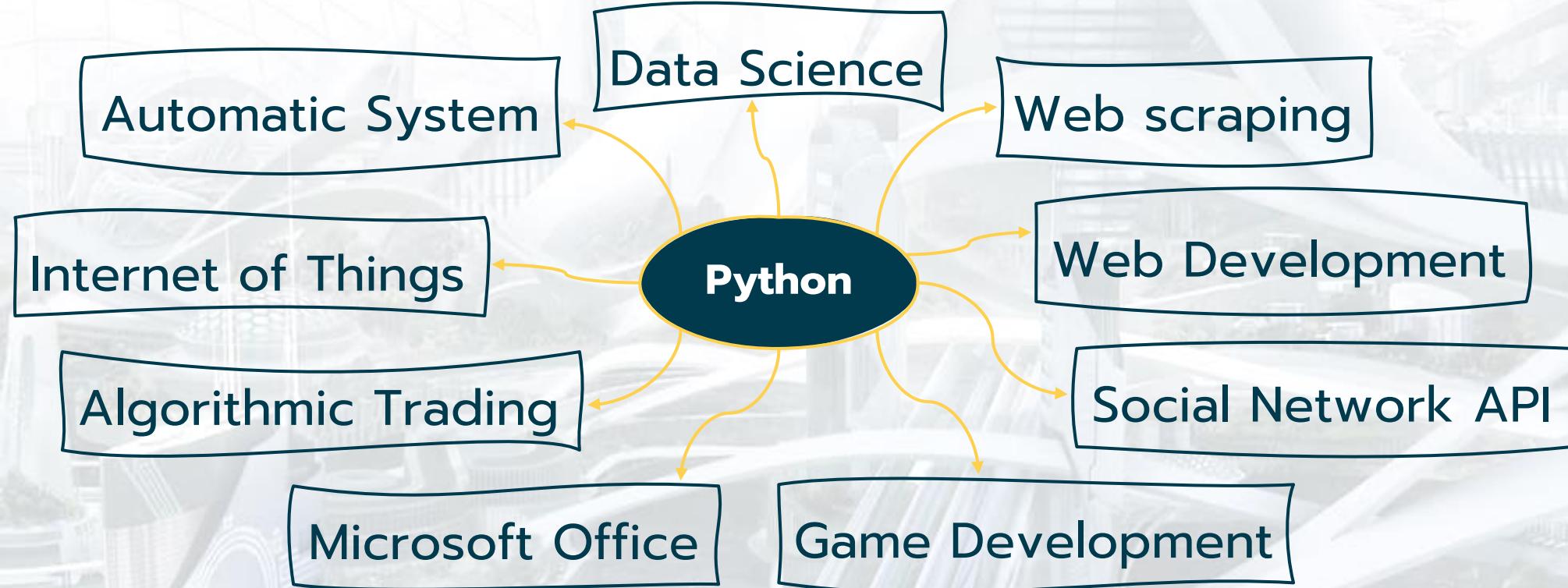
Experienced data scientist, instructor and speaker,  
AI consultant for the public and private sectors,  
CEO & Co-founder of MADEBYAI and QuantMetric





# "Beginner to Expert"





# "All Level"



# Facebook Group



## TAUTOLOGY ตะลุยโจทย์ Python 500 ข้อ

กลุ่มส่วนตัว · สมาชิก 519 คน



+ เข้า群

เกี่ยวกับ

การพูดคุย

ห้อง

สมาชิก

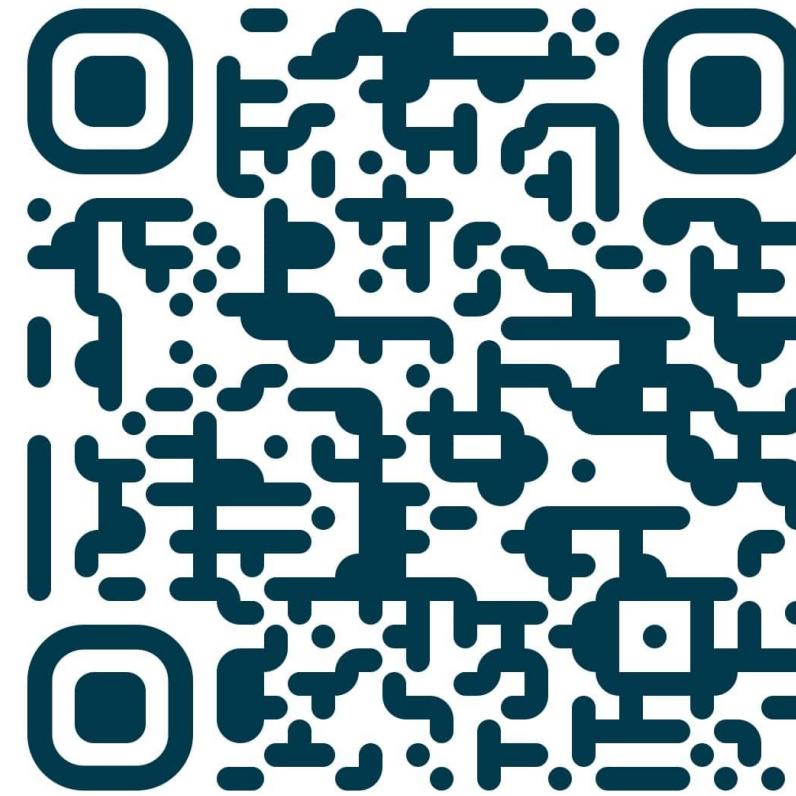
งานกิจกรรม

สื่อ



...

# Facebook Group



# Introduction

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# ประวัติความเป็นมาของ Python

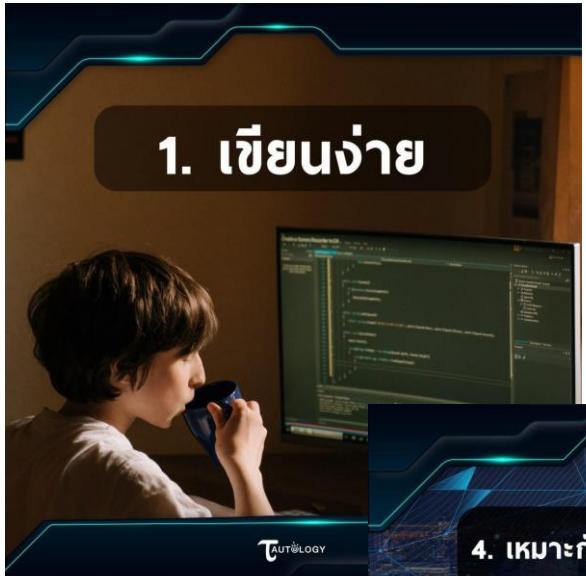


Guido Van Rossum

# ประวัติความเป็นมาของ Python



# ข้อดีของ Python มีอะไรบ้าง ?



Ref: <https://www.facebook.com/tautologyai/posts/2987242824730932>

# Python ดูกำไรใช้กี่เห็นบ้าง ?



# วิธีการติดตั้งใช้งาน Python

colab



# Print Command

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# Print Command

1. Print String
2. Print Numeric
3. Print String + Numeric

# 1. Print String

1.1 Quote

1.2 Double Quote

1.3 Multiple Line

1.4 String with Quotation Marks

1.5 String with Backslash

# 1.1 Quote

```
In [1]: ┌─[1] print('Hello')
```

```
In [2]: ┌─[1] print('I Love Python')
```

I Love Python

```
In [3]: ┌─[1] print('100')
```

100

# 1.2 Double Quote

```
In [4]: ┌─ 1 ─┐ print("Hello")
```

```
Hello
```

```
In [5]: ┌─ 1 ─┐ print("I Love Python")
```

```
I Love Python
```

```
In [6]: ┌─ 1 ─┐ print("100")
```

```
100
```

# 1.3 Multiple Line

In [7]: ►

```
1 print(''  
2 Hello  
3 Hello Python  
4 I Love Python''' )
```

```
Hello  
Hello Python  
I Love Python
```

# 1.3 Multiple Line

```
In [8]: ► 1 print("")  
2 Hello  
3 Hello Python  
4 I Love Python")
```

```
Hello  
Hello Python  
I Love Python
```

# 1.4 String with Quotation Marks

1.4.1 String with Quote

1.4.2 String with Double Quote

## **1.4.1 String with Quote**

```
In [9]: ┏━ 1 print("it's me")
```

```
it's me
```

```
In [10]: ┏━ 1 print('it\'s me')
```

```
it's me
```

## 1.4.2 String with Double Quote

```
In [11]: ➜ 1 print('he said "Python is my life."')
```

```
he said "Python is my life."
```

```
In [12]: ➜ 1 print("he said \"Python is my life.\\".")
```

```
he said "Python is my life.".
```

# 1.5 String with Backslash

```
In [13]: 1 print('/\\')
```

/\

```
In [14]: 1 print('/\\\\\\\\\\\\\\\\')
```

/\\\\\\\\\\\\

# Print Command

- ✓ 1. Print String
- 2. Print Numeric
- 3. Print String + Numeric

## 2. Print Numeric

### 2.1 Numeric

### 2.2 %d %f %s

## 2.1 Numeric

```
In [15]: ┌─[1] print(1)
```

```
1
```

```
In [16]: ┌─[1] print(1.5)
```

```
1.5
```

```
In [17]: ┌─[1] print(3.50)
```

```
3.5
```

## 2.2 %d %f %s

```
In [18]: ► 1 print('%d' %3)
```

3

```
In [19]: ► 1 print('%f' %1)
```

1.000000

```
In [20]: ► 1 print('%.2f' %1)
```

1.00

## 2.2 %d %f %s

```
In [21]: █ 1 print('%s' %7)
```

7

```
In [22]: █ 1 print('%s' %7.0)
```

7.0

## 2.2 %d %f %s

```
In [23]: ► 1 print('%d %.3f %s' %(2, 5, 7.23))
```

```
2 5.000 7.23
```

```
In [24]: ► 1 print('%.2f %s %d' %(3, 3.0, 3))
```

```
3.00 3.0 3
```

# Print Command

- ✓ 1. Print String
- ✓ 2. Print Numeric
- 3. Print String + Numeric

# 3. Print String + Numeric

3.1 Comma

3.2 String Concatenation

3.3 %d %f %s

## 3.1 Comma

```
In [25]: ► 1 print('I Love You', 3000)
```

I Love You 3000

```
In [26]: ► 1 print('5 + 7 =', 5 + 7)
```

5 + 7 = 12

## 3.2 String Concatenation

```
In [27]: ► 1 print('I Love You ' + str(3000))
```

I Love You 3000

```
In [28]: ► 1 print('5 + 7 = ' + str(5 + 7))
```

5 + 7 = 12

## 3.3 %d %f %s

```
In [29]: 1 print('I Love You %d' %3000)
```

```
I Love You 3000
```

```
In [30]: 1 print('5 + 7 = %d' %(5+7))
```

```
5 + 7 = 12
```

# Print Command

- ✓ 1. Print String
- ✓ 2. Print Numeric
- ✓ 3. Print String + Numeric

# Variable

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# Variable

1. Create & Assign
2. Create Multiple Variable
3. Identifier

# 1. Create & Assign

```
In [1]: ┌ 1 | firstname = 'John'
```

```
In [2]: ┌ 1 | lastname = 'Doe'
```

```
In [3]: ┌ 1 | age_full = 'I'm 32 years old.'
```

```
In [4]: ┌ 1 | print('My firstname is', firstname + '.')
          2 | print('My lastname is', lastname + '.')
          3 | print(age_full)
```

My firstname is John.

My lastname is Doe.

I'm 32 years old.

# 1. Create & Assign

```
In [5]: ┌ 1 | a = 5
```

```
In [6]: ┌ 1 | b = 7
```

```
In [7]: ┌ 1 | print('a =', a)
          2 | print('b =', b)
          3 | print('a + b =', a + b)
```

```
a = 5
b = 7
a + b = 12
```

# Variable

- ✓ 1. Create & Assign
- 2. Create Multiple Variable
- 3. Identifier

## 2. Create Multiple Variable

```
In [8]: ➜ 1 c = d = e = 5
```

```
In [9]: ➜ 1 print('c =', c)
          2 print('d =', d)
          3 print('e =', e)
```

```
c = 5
d = 5
e = 5
```

## 2. Create Multiple Variable

```
In [10]: ► 1 f = 6; g = 7; h = 8;
```

```
In [11]: ► 1 print('f =', f)
          2 print('g =', g)
          3 print('h =', h)
```

f = 6

g = 7

h = 8

# Variable

- ✓ 1. Create & Assign
- ✓ 2. Create Multiple Variable
- 3. Identifier

# 3. Identifier

3.1 Character & Underscore First

3.2 Case Sensitive

3.3 No Special Character

3.4 Reversed Word

# 3.1 Character & Underscore First

```
In [12]: ► 1 a1 = 5
```

```
In [13]: ► 1 print('a1 =', a1)  
          a1 = 5
```

```
In [14]: ► 1 _a = 7
```

```
In [15]: ► 1 print('_a =', _a)  
          _a = 7
```

# 3.1 Character & Underscore First

In [16]: ► 1 | 1a = 5

```
File "<ipython-input-16-ce2af4c14e7e>", line 1
  1a = 5
  ^
SyntaxError: invalid syntax
```

In [17]: ► 1 | 3x = 7

```
File "<ipython-input-17-c7219bffa8d3>", line 1
  3x = 7
  ^
SyntaxError: invalid syntax
```

## 3.2 Case sensitive

```
In [18]: ► 1 a = 5  
          2 b = 7
```

```
In [19]: ► 1 A = 10  
          2 B = 14
```

```
In [20]: ► 1 print('a =', a)  
          2 print('b =', b)  
          3 print('A =', A)  
          4 print('B =', B)
```

```
a = 5  
b = 7  
A = 10  
B = 14
```

# Special Character

”

{ } [ ] ! @ # \$ % ^  
& \* ( ) - = / | \ . < >

”

## 3.3 No Special Character

```
In [21]: ➜ 1 c$d = 7
          File "<ipython-input-21-81e2e827bf95>", line 1
            c$d = 7
            ^
SyntaxError: invalid syntax
```

## 3.3 No Special Character

In [22]: ➜ 1 c.d = 8

```
-----  
AttributeError                                     Traceback (most recent call last)  
<ipython-input-22-56b67979db83> in <module>  
----> 1 c.d = 8  
  
AttributeError: 'int' object has no attribute 'd'
```

## 3.3 No Special Character

```
In [23]: ➜ 1 c d = 9
          File "<ipython-input-23-757563a21641>", line 1
            c d = 9
            ^
SyntaxError: invalid syntax
```

# Reserved Word

”

and, assert, break, class, continue, def,  
del, elif, else, except, exec, finally, for, from,  
global, if, import, in, is, lambda, not, or,  
pass, print, raise, return, try, while, yield

”

## 3.4 Reserved Word

```
In [24]: ┌ 1 | if = 5
          └
          File "<ipython-input-24-1a9bb92cc795>", line 1
              if = 5
                  ^
SyntaxError: invalid syntax
```

## 3.4 Reserved Word

```
In [25]: 1 for = 10
          File "<ipython-input-25-597287a2d762>", line 1
              for = 10
                  ^
SyntaxError: invalid syntax
```

## 3.4 Reserved Word

```
In [26]: 1 del = 15
          File "<ipython-input-26-f23a0aa76da8>", line 1
            del = 15
            ^
SyntaxError: invalid syntax
```

# Variable

- ✓ 1. Create & Assign
- ✓ 2. Create Multiple Variable
- ✓ 3. Identifier

# Data Structure

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# Data Structure

1. Basic Data Structure
2. Composite Data Structure

# 1. Basic Data Structure

1.1 Numeric

1.2 String

# 1.1 Numeric

1.1.1 Integer

1.1.2 Floating-Point Number

1.1.3 Boolean

1.1.4 Complex Number

# 1.1.1 Integer

```
In [1]: █ 1 a = 1
```

```
In [2]: █ 1 b = 2
```

```
In [3]: █ 1 type(a)
```

Out[3]: int

```
In [4]: █ 1 type(b)
```

Out[4]: int

## 1.1.2 Floating-Point Number

```
In [5]: ► 1 c = 0.5
```

```
In [6]: ► 1 d = 3/4
```

```
In [7]: ► 1 type(c)
```

Out[7]: float

```
In [8]: ► 1 type(d)
```

Out[8]: float

## 1.1.3 Boolean

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

```
In [10]: 1 f = False
```

```
In [11]: 1 type(t)
```

Out[11]: bool

```
In [12]: 1 type(f)
```

Out[12]: bool

# 1.1.4 Complex Number

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

```
In [14]: 1 z.real
```

Out[14]: 1.0

```
In [15]: 1 z.imag
```

Out[15]: 2.0

```
In [16]: 1 type(z)
```

Out[16]: complex

# 1. Basic Data Type

1.1 Numeric 

1.2 String

# 1.2 String

1.2.1 Create String

1.2.2 Read String

1.2.3 String + replace

1.2.4 String + len

1.2.5 String + in

1.2.6 String + split

1.2.7 String + Concatenation

## 1.2.1 Create String

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

```
In [18]: ► 1 char2 = '0123456789'
```

```
In [19]: ► 1 type(char1)
```

Out[19]: str

```
In [20]: ► 1 type(char2)
```

Out[20]: str

## 1.2.2 Read String

```
In [21]: 1 char3 = '0123456789'
```

```
In [22]: 1 char3[0]
```

```
Out[22]: '0'
```

```
In [23]: 1 char3[5]
```

```
Out[23]: '5'
```

```
In [24]: 1 char3[-1]
```

```
Out[24]: '9'
```

```
In [25]: 1 char3[-2]
```

```
Out[25]: '8'
```

## 1.2.2 Read String

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

```
In [27]: 1 char3[3:]
```

```
Out[27]: '3456789'
```

```
In [28]: 1 char3[:7]
```

```
Out[28]: '0123456'
```

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

```
Out[29]: '3456'
```

## 1.2.3 String + replace

```
In [30]: ➜ 1 | char4 = 'abc'
```

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

Out[31]: 'Abc'

```
In [32]: ➜ 1 | char4 = 'abc'
```

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

Out[33]: 'aDE'

## 1.2.4 String + len

```
In [34]: ➜ 1 char5 = '12345'
```

```
In [35]: ➜ 1 len(char5)
```

```
Out[35]: 5
```

## 1.2.5 String + in

```
In [36]: ➜ 1 char6 = 'Python'
```

```
In [37]: ➜ 1 'P' in char6
```

Out[37]: True

```
In [38]: ➜ 1 'p' in char6
```

Out[38]: False

## 1.2.6 String + split

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

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

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

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

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

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

## 1.2.7 String + Concatenation

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

```
In [44]: ► 1 char9 + char10
```

Out[44]: 'abcd'

## 1.2.7 String + Concatenation

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

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

Out[46]: 'ABC'

# 1. Basic Data Structure

1.1 Numeric 

1.2 String 

# Data Structure

- ✓ 1. Basic Data Structure
- 2. Composite Data Structure

# 2. Composite Data Structure

2.1 List

2.2 Tuple

2.3 Dictionary

2.4 Set

# List

“CRUD + sort + len + in”

# 2.1 List

2.1.1 Create List

2.1.2 Read List

2.1.3 Update List

2.1.4 Delete List

2.1.5 List + sort

2.1.6 List + len

2.1.7 List + in

## 2.1.1 Create List

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

```
In [48]: ► 1 type(list1)
```

Out[48]: list

```
In [49]: ► 1 type(list2)
```

Out[49]: list

```
In [50]: ► 1 type(list3)
```

Out[50]: list

## 2.1.2 Read List

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

```
In [52]: █ 1 list4[0]
```

Out[52]: 1

```
In [53]: █ 1 list4[-1]
```

Out[53]: 'c'

```
In [54]: █ 1 list4[-2]
```

Out[54]: 'b'

## 2.1.2 Read List

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

```
In [56]: ┏━ 1 ┡ list5[3:]
```

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

```
In [57]: ┏━ 1 ┡ list5[:5]
```

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

```
In [58]: ┏━ 1 ┡ list5[3:5]
```

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

## 2.1.3 Update List : Replace

```
In [59]: █ 1 list6 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [60]: █ 1 list6[0] = 0
```

```
In [61]: █ 1 list6
```

```
Out[61]: [0, 2, 3, 'a', 'b', 'c']
```

```
In [62]: █ 1 list6[-1] = 'x'
```

```
In [63]: █ 1 list6
```

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

## 2.1.3 Update List : append

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

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

```
In [66]: 1 list7
```

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

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

```
In [68]: 1 list7
```

```
Out[68]: [1, 2, 3, 'a', 'b', 'c', 'd', 'e']
```

## 2.1.3 Update List : extend

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

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

```
In [71]: ► 1 list8
```

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

## 2.1.3 Update List : insert

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

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

```
In [74]: ► 1 list10
```

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

## 2.1.4 Delete List : del

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

```
In [76]: 1 del list11[2]
```

```
In [77]: 1 list11
```

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

## 2.1.4 Delete List : del

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

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

```
In [80]: 1 list11
```

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

## 2.1.4 Delete List : remove

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

```
In [82]: █ 1 list12.remove(1)
```

```
In [83]: █ 1 list12
```

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

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

```
In [85]: █ 1 list12
```

```
Out[85]: [2, 3, 'b', 'c']
```

## 2.1.4 Delete List : clear

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

```
In [87]: 1 list13.clear()
```

```
In [88]: 1 list13
```

```
Out[88]: []
```

## 2.1.5 List + sort

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

```
In [90]: 1 list14.sort()
```

```
In [91]: 1 list14
```

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

## 2.1.5 List + sort

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

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

```
In [94]: ┏━ 1 list14
```

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

## 2.1.5 List + sort

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

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

```
In [97]: ► 1 sorted_list14
```

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

## 2.1.5 List + sort

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

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

```
In [100]: ┏━ 1 sorted_list14
```

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

## 2.1.7 List + len

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

```
In [102]: 1 len(list15)
```

Out[102]: 6

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

```
In [104]: 1 len(list16)
```

Out[104]: 5

## 2.1.5 List + in

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

```
In [106]: █ 1 'b' in list17
```

Out[106]: True

```
In [107]: █ 1 4 in list17
```

Out[107]: False

# 2. Composite Data Structure

2.1 List 

2.2 Tuple

2.3 Dictionary

2.4 Set

# Tuple

“ CR~~X~~ + len + in ”

# 2.2 Tuple

2.2.1 Create Tuple

2.2.2 Read Tuple

2.2.3 Tuple + len

2.2.4 Tuple + in

## 2.2.1 Create Tuple

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

```
In [109]: ┏━ 1 type(tuple1)
```

Out[109]: tuple

```
In [110]: ┏━ 1 type(tuple2)
```

Out[110]: tuple

```
In [111]: ┏━ 1 type(tuple3)
```

Out[111]: tuple

## 2.2.2 Read Tuple

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

```
In [113]: 1 tuple4[0]
```

Out[113]: 1

```
In [114]: 1 tuple4[-1]
```

Out[114]: 'c'

```
In [115]: 1 tuple4[-2]
```

Out[115]: 'b'

## 2.2.2 Read Tuple

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

```
In [117]: 1 tuple5[3:]
```

```
Out[117]: ('a', 'b', 'c')
```

```
In [118]: 1 tuple5[:5]
```

```
Out[118]: (1, 2, 3, 'a', 'b')
```

```
In [119]: 1 tuple5[3:5]
```

```
Out[119]: ('a', 'b')
```

## 2.2.3 Tuple + len

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

```
In [121]: 1 len(tuple6)
```

Out[121]: 6

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

```
In [123]: 1 len(tuple7)
```

Out[123]: 5

## 2.2.4 Tuple + in

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

```
In [125]: 1 'b' in tuple8
```

Out[125]: True

```
In [126]: 1 4 in tuple8
```

Out[126]: False

# 2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary

2.4 Set

# Dictionary

“ CRUD + len + in ”

# 2.3 Dictionary

2.1.1 Create Dictionary

2.1.2 Read Dictionary

2.1.3 Update Dictionary

2.1.4 Delete Dictionary

2.1.5 Dictionary + len

2.1.6 Dictionary + in

## 2.3.1 Create Dictionary

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

```
In [128]: ➜ 1 type(dict1)
```

Out[128]: dict

```
In [129]: ➜ 1 type(dict2)
```

Out[129]: dict

## 2.3.2 Read Dictionary

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

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

Out[131]: 'John'

```
In [132]: ➜ 1 dict3['age']
```

Out[132]: 32

## 2.3.3 Update Dictionary : Replace

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

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

```
In [135]: ➜ 1 dict4
```

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

## 2.3.3 Update Dictionary : Add

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

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

```
In [138]: ➜ 1 dict5
```

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

## 2.3.3 Update Dictionary : update

```
In [139]: 1 dict6 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [140]: 1 dict6.update({'blood_group' : 'O', 'height' : 180})
```

```
In [141]: 1 dict6
```

```
Out[141]: {'firstname': 'John',
            'lastname': 'Doe',
            'age': 32,
            'blood_group': 'O',
            'height': 180}
```

## 2.3.4 Delete Dictionary : del

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

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

```
In [144]: ➜ 1 dict7
```

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

## 2.3.4 Delete Dictionary : clear

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

```
In [146]: ➜ 1 dict8.clear()
```

```
In [147]: ➜ 1 dict8
```

```
Out[147]: {}
```

## 2.3.5 Dictionary + len

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

```
In [149]: 1 len(dict9)
```

```
Out[149]: 3
```

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

```
Out[150]: 3
```

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

```
Out[151]: 3
```

## 2.3.6 Dictionary + in

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

```
In [153]: 1 'firstname' in dict10
```

Out[153]: True

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

Out[154]: True

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

Out[155]: True

## 2.3.6 Dictionary + in

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

```
In [157]: 1 'weight' in dict10
```

Out[157]: False

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

Out[158]: False

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

Out[159]: False

## 2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary 

2.4 Set

# Set

“ CRUD + len + in ”

# Set

”

- + union
- + intersection
- + difference
- + Symmetric Difference

”

# 2.4 Set

2.4.1 Create Set

2.4.2 Read Set

2.4.3 Update Set

2.4.4 Delete Set

2.4.5 Set + len

2.4.6 Set + in

## 2.4 Set

2.4.7 Set + Union

2.4.8 Set + Intersection

2.4.9 Set + Difference

2.4.10 Set + Symmetric Difference

## 2.4.1 Create Set

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

```
In [161]: 1 type(set1)
```

Out[161]: set

```
In [162]: 1 type(set2)
```

Out[162]: set

## 2.4.2 Read Set

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

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

```
d  
b  
c  
e  
a
```

## 2.4.3 Update Set : add

```
In [165]: ┏━ 1 ┡ set4 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [166]: ┏━ 1 ┡ set4.add(1)
```

```
In [167]: ┏━ 1 ┡ set4
```

```
Out[167]: {1, 'a', 'b', 'c', 'd', 'e'}
```

## 2.4.3 Update Set : update

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

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

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

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

## 2.4.4 Delete Set : remove

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

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

```
In [173]: 1 set6
```

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

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

```
In [175]: 1 set6
```

```
Out[175]: {'b', 'd', 'e'}
```

## 2.4.4 Delete Set : clear

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

```
In [177]: 1 set7.clear()
```

```
In [178]: 1 set7
```

```
Out[178]: set()
```

## 2.4.5 Set + len

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

```
In [180]: ┏━━━ 1 ┛ len(set8)
```

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

## 2.4.6 Set + in

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

```
In [182]: 1 1 in set9
```

Out[182]: True

```
In [183]: 1 'a' in set9
```

Out[183]: False

## 2.4.7 Set + Union

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

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

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

## 2.4.8 Set + Intersection

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

```
In [187]: ➜ 1 setA & setB
```

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

## 2.4.9 Set + Difference

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

```
In [189]: ➜ 1 setA - setB  
  
Out[189]: {1, 2}
```

```
In [190]: ➜ 1 setB - setA  
  
Out[190]: {6, 7}
```

## 2.4.10 Set + Symmetric Difference

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

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

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

# 2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary 

2.4 Set 

# Data Structure

- ✓ 1. Basic Data Structure
- ✓ 2. Composite Data Structure

# Conclusion

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

# Conclusion

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

# Conclusion

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

# Conclusion

## Set

Union (|)

Intersection (&)

Difference (-)

Symmetric Difference  
(^)

# Operator

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# Operator

$$y = x^2 + 2x + 1$$

- $y, x, 1$  គឺជាពួកដំបូនការ (Operand)
- $=, +$  គឺជាពួកដំបូនការ (Operator)
- $y = x^2 + 2x + 1$  កំណែមុទ្ធបាន គឺជានិធនី (expression)

# Operator

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Membership Operators
5. Operator Precedence

# 1. Arithmetic Operators

1.1 บวก (+)

1.2 ลบ (-)

1.3 คูณ (\*)

1.4 หาร (/)

1.5 ยกกำลัง (\*\*)

1.6 เศษจាតหาร (%)

1.7 ผลหารปัดเศษลง (//)

# 1.1 ԱԿՑ (+)

```
In [1]: █ 1 | a = 5  
          2 | b = 10  
          3 | c = 7
```

```
In [2]: █ 1 | a + b
```

Out[2]: 15

```
In [3]: █ 1 | print(b + c)
```

17

# 1.2 au (-)

```
In [4]: ➜ 1 a = 5  
2 b = 10  
3 c = 7
```

```
In [5]: ➜ 1 a - b
```

Out[5]: -5

```
In [6]: ➜ 1 b - c
```

Out[6]: 3

# 1.3 គុណ (\*)

```
In [7]: █ 1 | a = 5  
          2 | b = 10  
          3 | c = 7
```

```
In [8]: █ 1 | a * b
```

Out[8]: 50

```
In [9]: █ 1 | b * c
```

Out[9]: 70

# 1.4 អាស (/)

```
In [10]: █ 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [11]: █ 1 a/b
```

Out[11]: 0.5

```
In [12]: █ 1 b/c
```

Out[12]: 1.4285714285714286

# 1.5 ยกกำลัง (\*\*)

```
In [13]: ► 1 | a = 2  
          2 | b = 3  
          3 | c = 4
```

```
In [14]: ► 1 | a**b
```

Out[14]: 8

```
In [15]: ► 1 | b**c
```

Out[15]: 81

# 1.6 เศษหาร (%)

```
In [16]: 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [17]: 1 a%b
```

Out[17]: 5

```
In [18]: 1 b%c
```

Out[18]: 3

# 1.7 မာဟပ်ဒေသလွန် (//)

```
In [19]: █ 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [20]: █ 1 a//b
```

Out[20]: 0

```
In [21]: █ 1 b//c
```

Out[21]: 1

# Operator

- ✓ 1. Arithmetic Operators
- 2. Comparison Operators
- 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

## 2. Comparison Operators

2.1 ตรวจสอบความเท่ากัน (==)

2.2 ตรวจสอบความไม่เท่ากัน (!=)

2.3 ตรวจสอบความมากกว่า (>)

2.4 ตรวจสอบความน้อยกว่า (<)

2.5 ตรวจสอบความมากกว่าหรือเท่ากับ (>=)

2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ (<=)

## 2.1 ตรวจสอบความเท่ากัน (==)

```
In [22]: 1 a = 5  
          2 b = 'name'
```

```
In [23]: 1 a == 5
```

Out[23]: True

```
In [24]: 1 a == 7
```

Out[24]: False

```
In [25]: 1 b == 'name'
```

Out[25]: True

```
In [26]: 1 b == 'fullname'
```

Out[26]: False

## 2.2 ตรวจสอบความไม่เท่ากัน ( $\neq$ )

```
In [27]: █ 1 a = 5  
          2 b = 'name'
```

```
In [28]: █ 1 a != 5
```

Out[28]: False

```
In [29]: █ 1 a != 7
```

Out[29]: True

```
In [30]: █ 1 b != 'name'
```

Out[30]: False

```
In [31]: █ 1 b != 'fullname'
```

Out[31]: True

## 2.3 តម្រូវសែបគារមាត្រកំរែ (>)

```
In [32]: █ 1 a = 5
```

```
In [33]: █ 1 a > 4
```

Out[33]: True

```
In [34]: █ 1 a > 5
```

Out[34]: False

```
In [35]: █ 1 a > 6
```

Out[35]: False

## 2.4 ตรวจสอบความน้อยกว่า (<)

```
In [36]: █ 1 a = 5
```

```
In [37]: █ 1 a < 4
```

Out[37]: False

```
In [38]: █ 1 a < 5
```

Out[38]: False

```
In [39]: █ 1 a < 6
```

Out[39]: True

## 2.5 ตรวจสอบความมากกว่าหรือเท่ากับ ( $\geq$ )

```
In [40]: █ 1 a = 5
```

```
In [41]: █ 1 a >= 4
```

Out[41]: True

```
In [42]: █ 1 a >= 5
```

Out[42]: True

```
In [43]: █ 1 a >= 6
```

Out[43]: False

## 2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ ( $\leq$ )

```
In [44]: 1 | a = 5
```

```
In [45]: 1 | a <= 4
```

Out[45]: False

```
In [46]: 1 | a <= 5
```

Out[46]: True

```
In [47]: 1 | a <= 6
```

Out[47]: True

# Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

# 3. Logical Operators

3.1 and

3.2 or

3.3 not

# 3.1 and

<b>p</b>	<b>q</b>	<b>p and q</b>
True	True	True
True	False	False
False	True	False
False	False	False

# 3.1 and

```
In [48]: 1 t = True  
          2 f = False
```

```
In [49]: 1 t and f
```

Out[49]: False

## 3.2 or

<b>p</b>	<b>q</b>	<b>p or q</b>
True	True	True
True	False	True
False	True	True
False	False	False

## 3.2 or

```
In [50]: 1 t = True  
          2 f = False
```

```
In [51]: 1 t or f
```

Out[51]: True

## 3.3 not

<b>p</b>	<b>not p</b>
True	False
False	True

## 3.3 not

```
In [52]: ┏━ 1 | t = True  
         2 | f = False
```

```
In [53]: ┏━ 1 | not t
```

Out[53]: False

```
In [54]: ┏━ 1 | not f
```

Out[54]: True

# Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

# 4. Comparison Operators

4.1 in

4.2 not in

# 4.1 in

```
In [55]: ➜ 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [56]: ➜ 1 1 in listA
```

Out[56]: True

```
In [57]: ➜ 1 1 in tupleA
```

Out[57]: False

# 4.1 in

```
In [58]: ➜ 1 dictA = {'name' : 'John', 'age' : 32}  
        2 setA = {'Sushi', 'Salmon'}
```

```
In [59]: ➜ 1 'name' in dictA
```

Out[59]: True

```
In [60]: ➜ 1 'Egg' in setA
```

Out[60]: False

## 4.2 not in

```
In [61]: █ 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [62]: █ 1 1 not in listA
```

Out[62]: False

```
In [63]: █ 1 1 not in tupleA
```

Out[63]: True

## 4.2 not in

```
In [64]: 1 dictA = {'name' : 'John', 'age' : 32}  
         2 setA = {'Sushi', 'Salmon'}
```

```
In [65]: 1 'name' not in dictA
```

Out[65]: False

```
In [66]: 1 'Egg' not in setA
```

Out[66]: True

# Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- ✓ 4. Membership Operators
- 5. Operator Precedence

# 5. Operators Precedence

( )

\*\*

\* , / , % , //

+ , -

<= , < , > , >=

== , !=

=

# Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- ✓ 4. Membership Operators
- ✓ 5. Operator Precedence

# Flowchart

ຕະລຸຍໂຈກຍ Python 500 ຫ້ອ

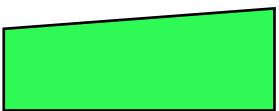
# Flowchart



เริ่ม, จบ



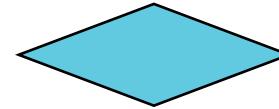
การประมวลผล  
การคำนวณ



รับอินพุต



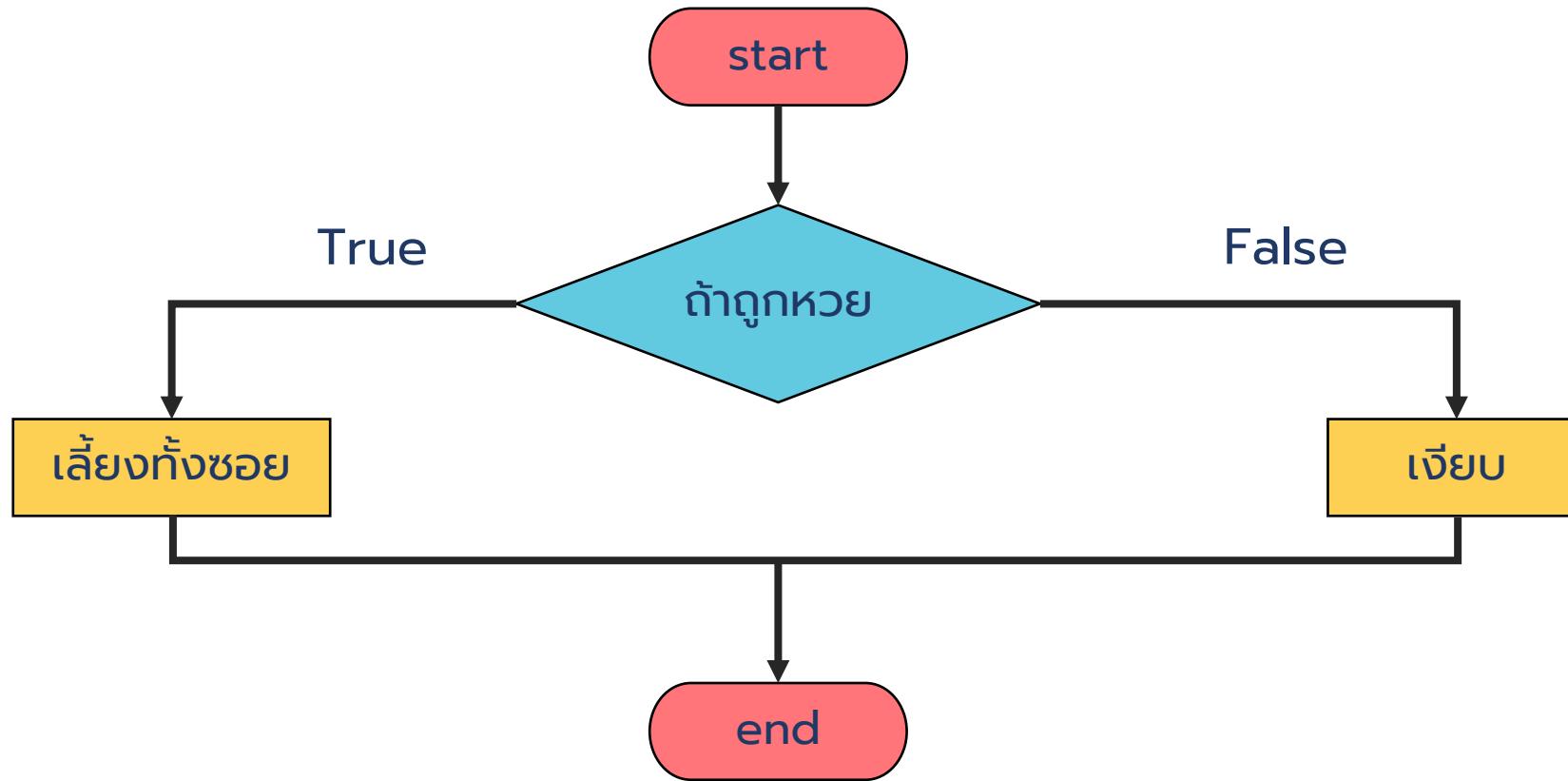
พิเศษ



เงื่อนไข  
(if-else)

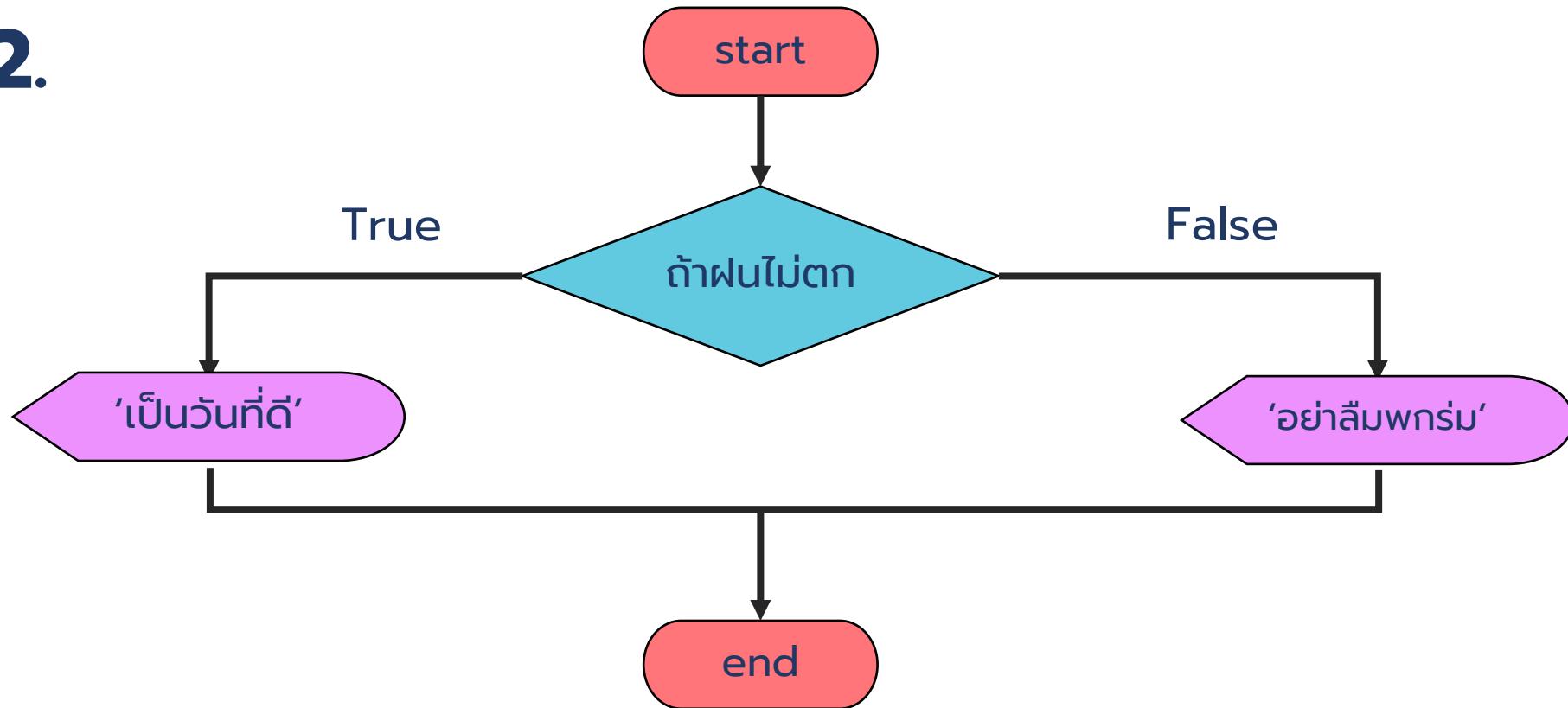
# Flowchart

Ex 1.



# Flowchart

Ex 2.



# If-Else

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# If-Else

1. If
2. If-Else
3. If-Elif
4. If-Elif-Else

# 1. If

In [1]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

Please insert your name: John

Hi John

Nice to meet you

# 1. If

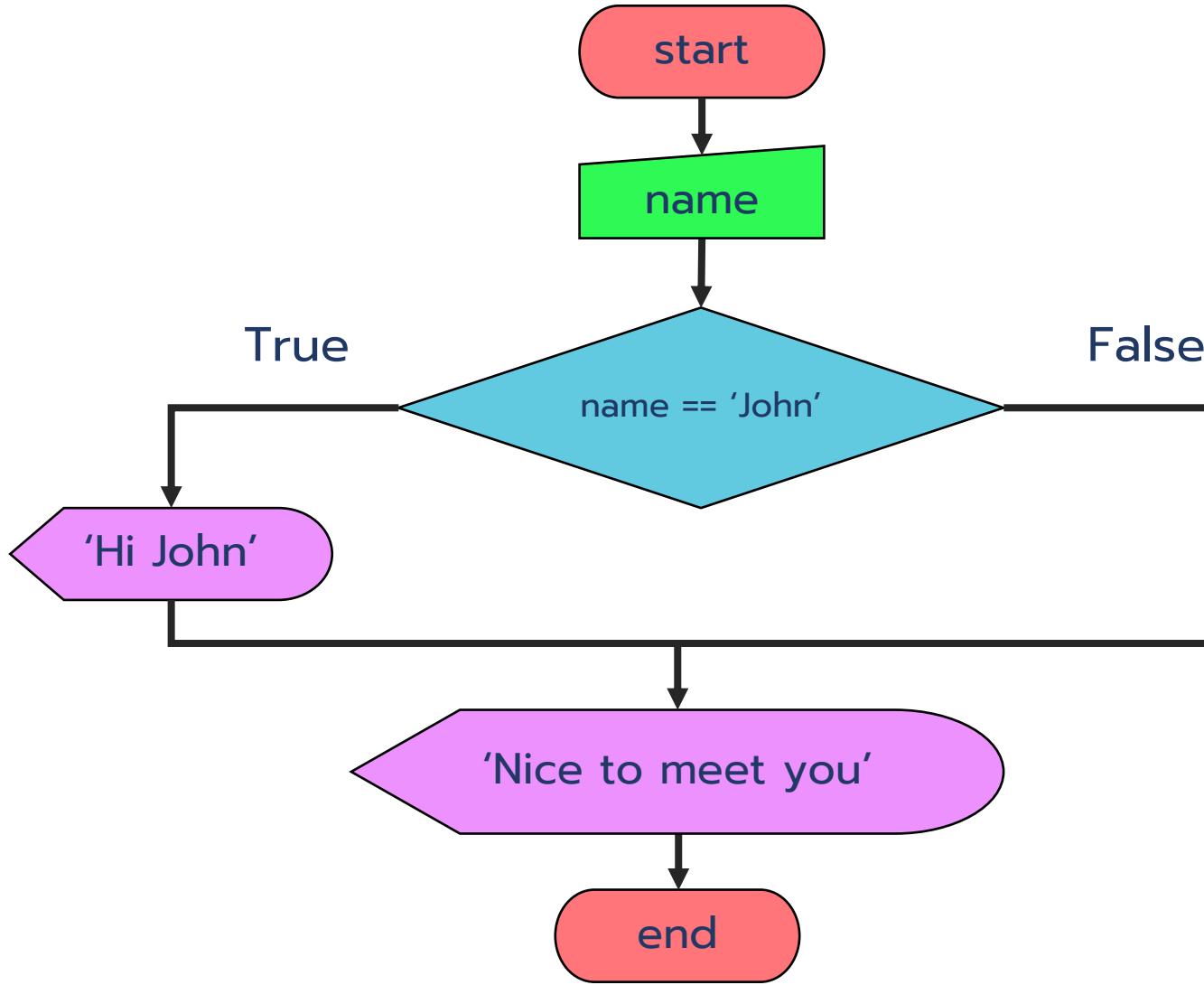
In [2]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

Please insert your name: Jane

Nice to meet you

# 1. If



# If-Else

- ✓ 1. If
- 2. If-Else
- 3. If-Elif
- 4. If-Elif-Else

## 2. If-Else

```
In [3]: ► 1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

```
Please insert score: 50
Pass
```

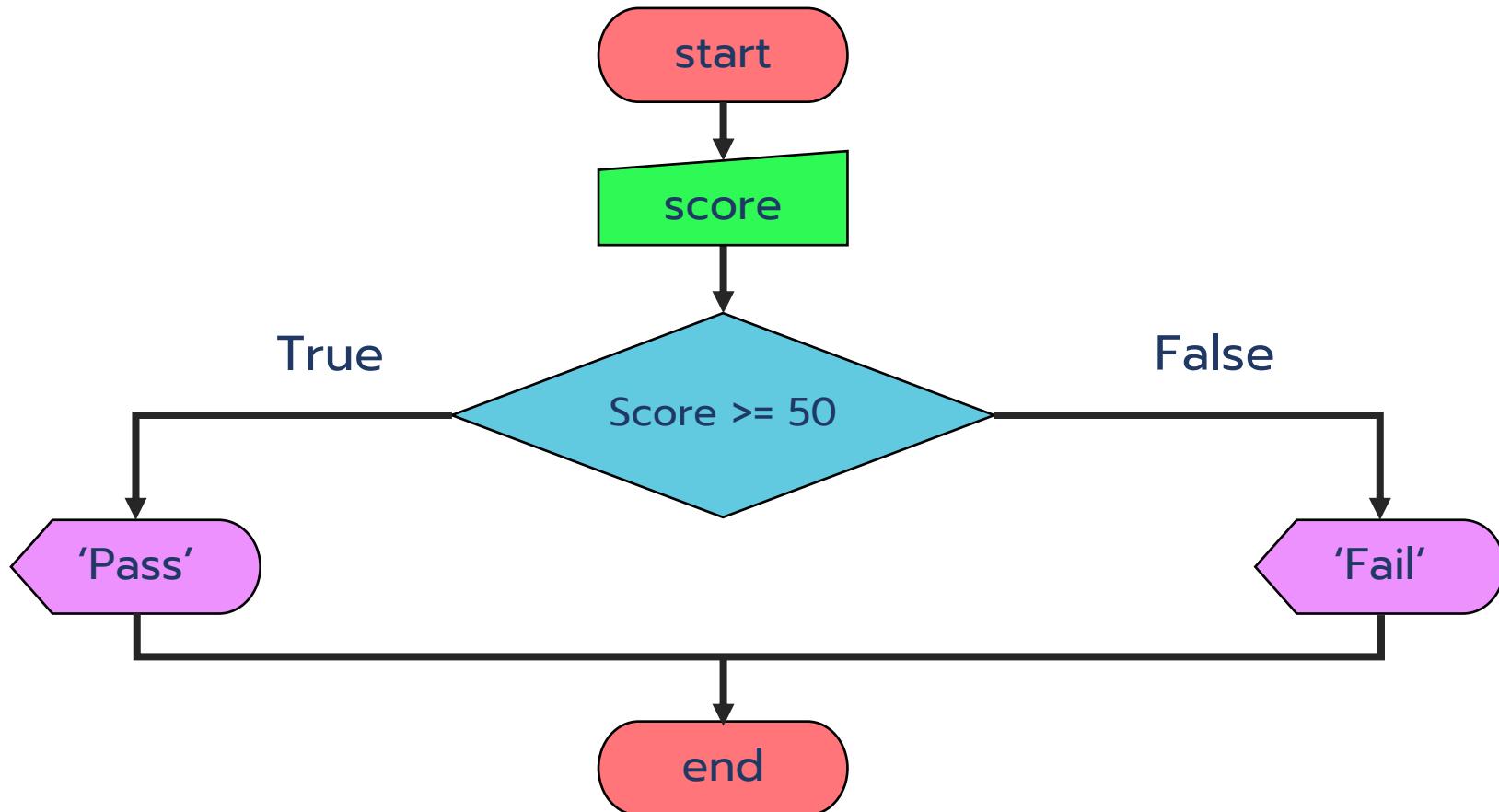
## 2. If-Else

In [4]: ➞

```
1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

```
Please insert score: 49
Fail
```

## 2. If-Else



# If-Else

- ✓ 1. If
- ✓ 2. If-Else
- 3. If-Elif
- 4. If-Elif-Else

### 3. If-Elif

In [5]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Alan Turing

Oh God !

Glad to see you

### 3. If-Elif

In [6]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Isac Newton

OMG !

Glad to see you

### 3. If-Elif

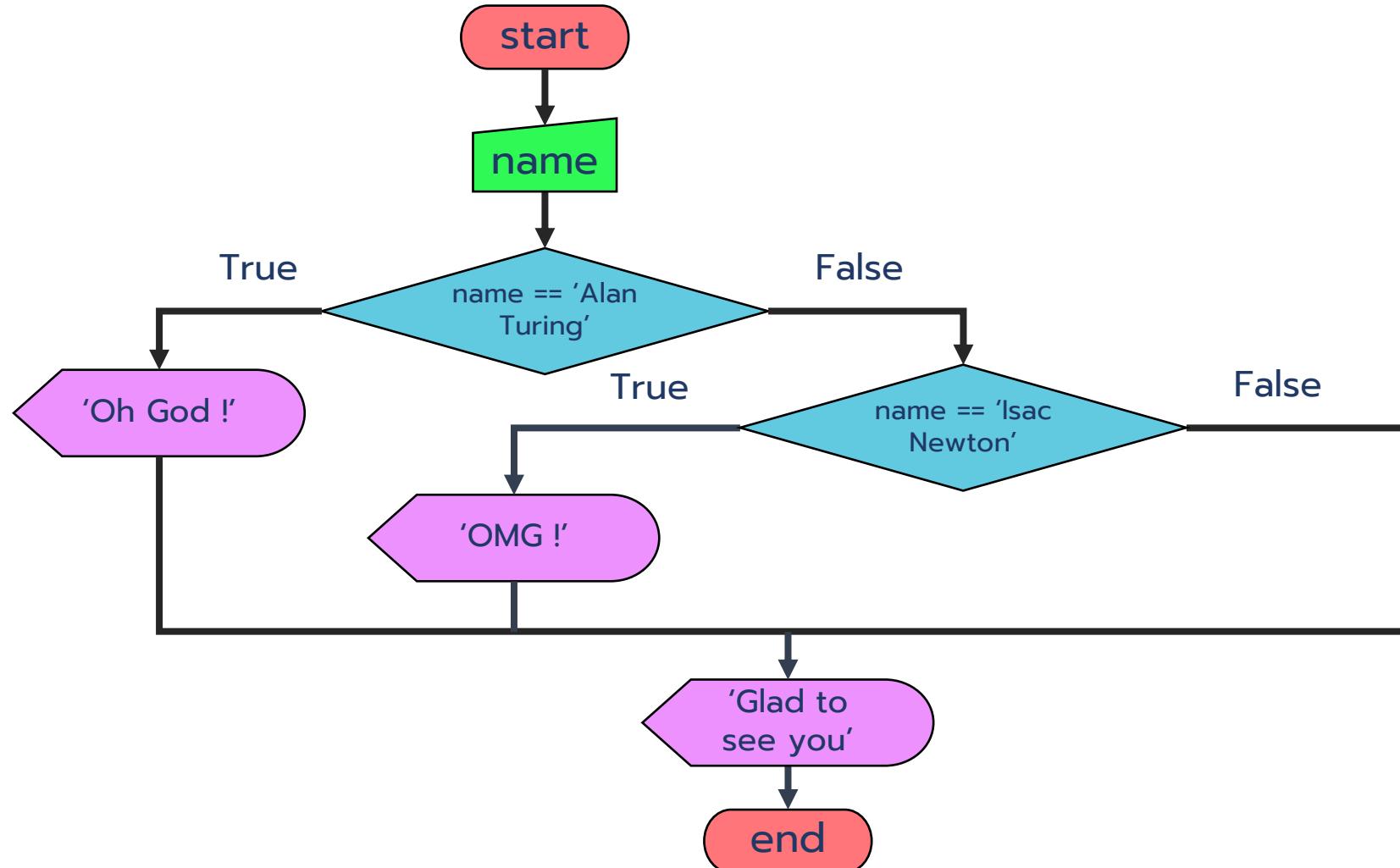
In [7]: ➞

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Yoda

Glad to see you

# 3. If-Elif



# If-Else

- ✓ 1. If
- ✓ 2. If-Else
- ✓ 3. If-Elif
- 4. If-Elif-Else

# 4. If-Elif-Else

In [8]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Gift

I miss you

# 4. If-Elif-Else

In [9]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Cherry

I love you

# 4. If-Elif-Else

In [10]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Wine

I need you

# 4. If-Elif-Else

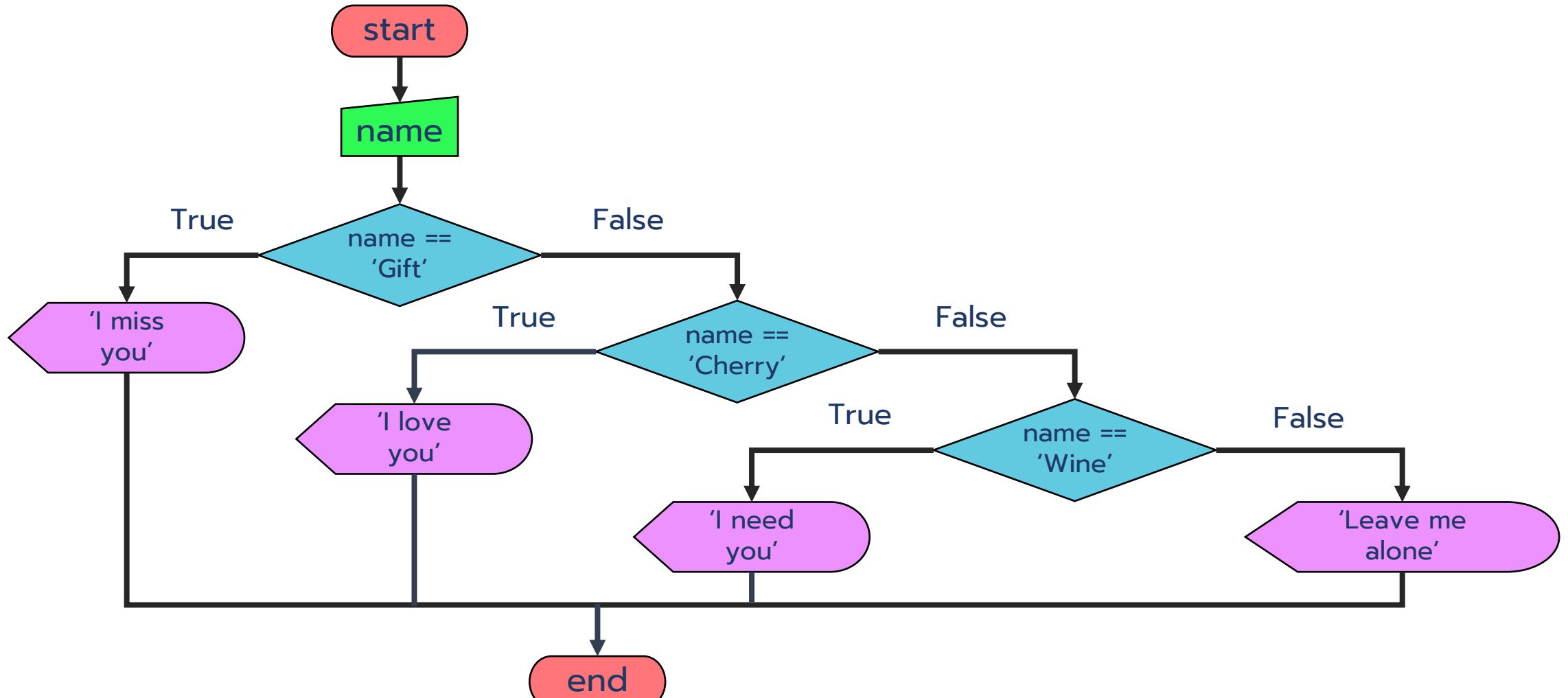
In [11]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Longan

Leave me alone

# 4. If-Elif-Else



# If-Else

- ✓ 1. If
- ✓ 2. If-Else
- ✓ 3. If-Elif
- ✓ 4. If-Elif-Else

# For & While Loop

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# For & While Loop

1. For
2. While
3. Loop & Statement

# 1. For

1.1 Standard For

1.2 For + String

1.3 For + List

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

# 1.1 Standard For

1.1.1 for i in range (end)

1.1.2 for i in range (start, end)

1.1.3 for i in range (start, end, step)

# 1.1.1 for i in range (end)

```
In [1]: ┌ 1 for i in range(5):  
  2     print(i)
```

```
0  
1  
2  
3  
4
```

## 1.1.2 for i in range (start, end)

```
In [2]: ┏ 1 for i in range(5, 11):  
 ┗ 2     print(i)
```

```
5  
6  
7  
8  
9  
10
```

## 1.1.3 for i in range (start, end, step)

```
In [3]: ┌ 1 for i in range(10, 20, 2):  
  2     print(i)
```

```
10  
12  
14  
16  
18
```

# 1.2 For + String

In [4]: ➤

```
1 char = 'Python'  
2 n = len(char)  
3 for i in range(n):  
4     print(char[i])
```

P  
y  
t  
h  
o  
n

# 1.2 For + String

In [5]: ➔

```
1 char = 'Python'  
2 for i in range(len(char)):  
3     print(char[i])
```

P  
y  
t  
h  
o  
n

# 1.2 For + String

In [6]: ➞

```
1 char = 'Python'  
2 for c in char:  
3     print(c)
```

P  
y  
t  
h  
o  
n

# 1.2 For + String

In [7]: ➜

```
1 char = 'Python'  
2 for i,c in enumerate(char):  
3     print(i, c)
```

```
0 P  
1 y  
2 t  
3 h  
4 o  
5 n
```

# 1.3 For + List

```
In [8]: ┏━
1 listx = ['a', 'b', 'c', 'd', 'e']
2 n = len(listx)
3 for i in range(n):
4     print(listx[i])
```

```
a
b
c
d
e
```

# 1.3 For + List

```
In [9]: ┏━ 1 listx = ['a', 'b', 'c', 'd', 'e']  
      2 for i in range(len(listx)):  
      3     print(listx[i])
```

```
a  
b  
c  
d  
e
```

# 1.3 For + List

```
In [10]: ┏━
 1 | listx = ['a', 'b', 'c', 'd', 'e']
 2 | for x in listx:
 3 |     print(x)
```

```
a
b
c
d
e
```

# 1.3 For + List

```
In [11]: ➜ 1 listx = ['a', 'b', 'c', 'd', 'e']
          2 for i, x in enumerate(listx):
          3     print(i, x)
```

```
0 a
1 b
2 c
3 d
4 e
```

# 1.4 For + Tuple

```
In [12]: ► 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 n = len(tuplex)
          3 for i in range(n):
          4     print(tuplex[i])
```

```
a
b
c
d
e
```

# 1.4 For + Tuple

```
In [13]: ➜ 1 tuples = ('a', 'b', 'c', 'd', 'e')
          2 for i in range(len(tuples)):
          3     print(tuples[i])
```

```
a
b
c
d
e
```

# 1.4 For + Tuple

```
In [14]: ► 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for x in tuplex:
          3     print(x)
```

```
a
b
c
d
e
```

# 1.4 For + Tuple

```
In [15]: ➜ 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for i, x in enumerate(tuplex):
          3     print(i, x)
```

```
0 a
1 b
2 c
3 d
4 e
```

# 1.5 For + Dictionary

```
In [16]: ► 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
          2 for key in dictx:
          3     print(key)

firstname
lastname
age
```

# 1.5 For + Dictionary

```
In [17]: ➜ 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
      2 for key in dictx.keys():
      3     print(key)

firstname
lastname
age
```

# 1.5 For + Dictionary

```
In [18]: ► 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}  
2 for value in dictx.values():  
3     print(value)
```

John  
Doe  
32

# 1.6 For + Set

```
In [19]: ► 1 setx = {1, 2, 3, 'a', 'b', 'c'}  
      2 for x in setx:  
      3     print(x)
```

```
b  
1  
2  
3  
a  
c
```

# For & While Loop

- ✓ 1. For
- 2. While
- 3. Loop & Statement

# 2. While

In [20]:

```
1 i = 1
2 while i <= 5:
3     print(i)
4     i = i + 1
```

```
1
2
3
4
5
```

# 2. While

In [21]: ►

```
1 i = 5
2 while i <= 15:
3     print(i)
4     i = i + 2
```

```
5
7
9
11
13
15
```

# 2. While

In [22]: ►

```
1 i = 0
2 while True:
3     print(i)
4     if i == 5:
5         break
6     i = i + 1
```

```
0
1
2
3
4
5
```

# For & While Loop

- ✓ 1. For
- ✓ 2. While
- 3. Loop & Statement

# 1. Loop & Statement

3.1 break

3.2 continue

3.3 pass

# 3.1 break

```
In [23]: ➜ 1 for i in range(100000):  
    2     print(i)  
    3     if i == 5:  
    4         break
```

```
0  
1  
2  
3  
4  
5
```

## 3.2 continue

In [24]:

```
1 for i in range(10):
2     if i%2 == 0:
3         continue
4         print('hello')
5     else:
6         print(i)
```

```
1
3
5
7
9
```

# 3.3 pass

```
In [25]: ➜ 1 for i in range(10):
2     if i%2 == 0:
3         pass
4         print('hello')
5     else:
6         print(i)
```

```
hello
1
hello
3
hello
5
hello
7
hello
9
```

# 3.3 pass

In [26]: ➜

```
1 v_list = [120, 70, 90]
2 for v in v_list:
3     if v >= 120:
4         pass
5     else:
6         print('500 baht finepage x')
```

500 baht finepage x  
500 baht finepage x

# For & While Loop

- ✓ 1. For
- ✓ 2. While
- ✓ 3. Loop & Statement

# Function

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# ทำไมต้องใช้ Function

**Ex.** **list1 = [1, 3, 4, 7, 11]**  
**list2 = [4, 3, 2, 1, 0]**

- ➔ อยากรอตรวจสอบของ list1
- ➔ อยากรอตรวจสอบของ list2

# ทำไมต้องใช้ Function

**Ex.**

```
sum1 = 0
for i in range(len(list1)):
    sum1 = sum1+list1[i]
sum2
for i in range(len(list2)):
    sum2 = sum2+list2[i]
```

# ทำไมต้องใช้ Function

**Ex.**

**list1 = [1, 3, 4, 7, 11]**

⋮

**listN = [7, 5, 12, 4, 0]**

➔ อยากหาผลรวมของ list1

⋮

➔ อยากหาผลรวมของ listN

# ทำไมต้องใช้ Function

ช้าช้อน !

จะเขียนทำไมหลายรอบ !

# ทำไมต้องใช้ Function

**Ex.**

```
def Sum(list):
    sumx = 0
    for i in range(len(list)):
        sumx = sumx + list[i]
    return sumx
```

# ทำไมต้องใช้ Function

**Ex.**

**Sum1 = Sum(list1)**

**sum2 = Sum(list2)**

**:**

**sumN = Sum(listN)**

# Function

1. With Parameter
2. Without Parameter

# 1. With Parameter

In [1]: ►

```
1 def sum_list(listA):
2     sumx = 0
3     for i in range(len(listA)):
4         sumx = sumx + listA[i]
5     return sumx
```

In [2]: ►

```
1 list1 = [1, 2, 3, 4, 5]
2 list2 = [6, 7, 8, 9, 10]
```

In [3]: ►

```
1 sum_list(list1)
```

Out[3]: 15

In [4]: ►

```
1 sum_list(list2)
```

Out[4]: 40

# 1. With Parameter

```
In [5]: ┏ 1 def plus(a, b):  
      2     c = a + b  
      3     return c
```

```
In [6]: ┏ 1 plus(1, 3)
```

Out[6]: 4

```
In [7]: ┏ 1 def minus(a, b):  
      2     d = a - b  
      3     return d
```

```
In [8]: ┏ 1 minus(1, 3)
```

Out[8]: -2

# 1. With Parameter

```
In [9]: ► 1 def plus_minus(a, b, c):  
    2     c = a + b + c  
    3     d = a - b - c  
    4     return c, d
```

```
In [10]: ► 1 result1, result2 = plus_minus(1, 3, 5)
```

```
In [11]: ► 1 result1
```

Out[11]: 9

```
In [12]: ► 1 result2
```

Out[12]: -11

# Function

- ✓ 1. With Parameter
- 2. Without Parameter

## 2. Without Parameter

In [13]:

```
1 def tree():
2     print('''
3         ---1---
4         --121--
5         -12321-
6         1234321''')
```

In [14]:

```
1 tree()
```

```
---1---
--121--
-12321-
1234321
```

## 2. Without Parameter

In [15]:

```
1 def PYTHON():
2     print('''
3     --XXXXXX-- XX----XX--XXXXXXXXX--XX----XX----XXXXXX---XX----XX--
4     --XX---XX--XX----XX----XX----XX----XX----XX----XX----XXX----XX--
5     --XX---XX--XX--XX----XX----XX----XX----XX----XX----XX----XXXX--XX--
6     --XXXXXX----XXXX----XX----XXXXXXXXX--XX----XX--XX--XX--XX--XX--XX-
7     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXX--'
8     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXX--'
9     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXXX----XX----XX--'''')
```

In [16]:

```
1 PYTHON()
```

```
--XXXXXX-- XX----XX--XXXXXXXXX--XX----XX----XXXXXX---XX----XX--
--XX---XX--XX----XX----XX----XX----XX----XX----XX----XX----XXX----XX--
--XX---XX--XX--XX----XX----XX----XX----XX----XX----XX----XX----XXXX--XX--
--XXXXXX----XXXX----XX----XXXXXXXXX--XX----XX--XX--XX--XX--XX--XX--XX-
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXX--'
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXX--'
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXXX----XX----XX--'
```

# Function

- ✓ 1. With Parameter
- ✓ 2. Without Parameter

# OOP

ຕະລຸຍໂຈກຍ Python 500 ຊອ

# ทำความรู้จัก OOP

1. **OOP เป็นเทคนิคการเขียนโปรแกรมแบบหนึ่ง ที่ม่องทุกอย่างเป็น Object**
2. **การเขียน Code แบบ OOP ทำให้ง่าย & เป็นระเบียบ ในการ พัฒนา/ปรับปรุง**

# OOP

1. Attribute
2. Method

# 1. Attribute

```
In [1]: ┏━ 1 class Warrior:  
      2     def __init__(self, HP, ATK, DEF):  
      3         self.HP = HP  
      4         self.ATK = ATK  
      5         self.DEF = DEF
```

```
In [2]: ┏━ 1 warrior1 = Warrior(100, 45, 25)
```

```
In [3]: ┏━ 1 print("warrior1's HP =", warrior1.HP)  
      2 print("warrior1's ATK =", warrior1.ATK)  
      3 print("warrior1's DEF =", warrior1.DEF)
```

```
warrior1's HP = 100  
warrior1's ATK = 45  
warrior1's DEF = 25
```

# 1. Attribute

```
In [4]: ┏━ 1 | warrior2 = Warrior(100, 35, 30)
```

```
In [5]: ┏━ 1 | print("warrior2's HP =", warrior2.HP)
         2 | print("warrior2's HP =", warrior2.ATK)
         3 | print("warrior2's HP =", warrior2.DEF)
```

```
warrior2's HP = 100
warrior2's HP = 35
warrior2's HP = 30
```

# 1. Attribute

```
In [6]: ┏ 1 class Car:  
      2     def __init__(self, brand, model, color):  
      3         self.brand = brand  
      4         self.model = model  
      5         self.color = color
```

```
In [7]: ┏ 1 car1 = Car('Toyota', 'Camry', 'Black')
```

```
In [8]: ┏ 1 print("car1's brand =", car1.brand)  
      2 print("car1's model =", car1.model)  
      3 print("car1's color =", car1.color)
```

```
car1's brand = Toyota  
car1's model = Camry  
car1's color = Black
```

# 1. Attribute

```
In [9]: ► 1 car2 = Car('Honda', 'Civic', 'White')
```

```
In [10]: ► 1 print("car2's brand =", car2.brand)
          2 print("car2's model =", car2.model)
          3 print("car2's color =", car2.color)
```

```
car2's brand = Honda
car2's model = Civic
car2's color = White
```

# OOP

- ✓ 1. Attribute
- 2. Method

## 2. Method

In [11]: ►

```
1 class Warrior:  
2     def __init__(self, HP, ATK, DEF):  
3         self.HP = HP  
4         self.ATK = ATK  
5         self.DEF = DEF  
6  
7     def training(self):  
8         self.HP = self.HP + 5  
9         self.ATK = self.ATK + 10  
10        self.DEF = self.DEF + 5
```

In [12]: ►

```
1 warrior1 = Warrior(100, 45, 25)  
2 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))  
3 warrior1.training()  
4 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))
```

HP = 100, ATK = 45, DEF = 25

HP = 105, ATK = 55, DEF = 30

## 2. Method

In [13]: ►

```
1 class Car:  
2     def __init__(self, brand, model, color):  
3         self.brand = brand  
4         self.model = model  
5         self.color = color  
6  
7     def set_color(self, color):  
8         self.color = color
```

In [14]: ►

```
1 car1 = Car('Toyota', 'Camry', 'Black')  
2 print('old color = %s' %car1.color)  
3 car1.set_color('Red')  
4 print('new color = %s' %car1.color)
```

old color = Black  
new color = Red

# OOP

- ✓ 1. Attribute
- ✓ 2. Method



**TAUTOLOGY**  
INNOVATION SCHOOL.

**Thank You**