

# Computer Programming

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

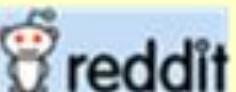
# วัตถุประสงค์

- เข้าใจการทำงานของคำสั่งและโปรแกรม
  - เขียนโปรแกรมให้ทำงานตามขั้นตอนที่ระบุ
  - เขียนโปรแกรมเพื่อแก้ไขปัญหาที่ได้รับ

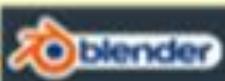
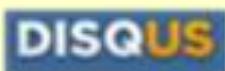
# เรียนการเขียนโปรแกรมเบื้องต้น

- สอนหลักการเขียนโปรแกรมคอมพิวเตอร์โดยใช้ Python เป็นเครื่องมือ
- ไม่ได้สอนภาษา Python อย่างลึกซึ้ง
- ใช้ Python 3 (ไม่ใช่ 2)

# ทำไมใช้ Python



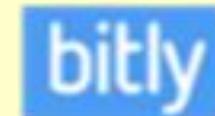
1 | → | one laptop per child



SpamBayes



The Washington Post



# กระบวนการวิธีเรียนวิชานี้

- ดู VDO + ทำแบบฝึกหัดสั้น ๆ นอกเวลา
- ฝึกเขียนโปรแกรมแก้โจทย์ปัญหา  
ในห้องปฏิบัติการคอมพิวเตอร์ + นอกเวลา

# เรียนการเขียนโปรแกรม

ក្នុងមនុស្ស បានចូលរួម ដើម្បី ការបង្កើត (code)

code, code, code, code

# แบบฝึกหัดสั้น ๆ

เขียนคำสั่งตามที่แสดงใน comment

script.py

```
1 # รับจำนวนจริงจาก input เก็บในตัวแปร h  
2 # รับจำนวนจริงจาก input เก็บในตัวแปร w  
3 # h คือความสูงของสามเหลี่ยม  
4 # w คือความยาวฐานของสามเหลี่ยม  
5 # แสดงพื้นที่ของสามเหลี่ยม  
6 #  
7 //
```

Python Shell

In [1]:

Solution

Submit



# แบบฝึกหัดสั้น ๆ

เขียนคำสั่งตามที่แสดงใน comment

script.py

```
1 # รับจำนวนจริงจาก input เก็บในตัวแปร h
2 # รับจำนวนจริงจาก input เก็บในตัวแปร w
3 # h คือความสูงของสามเหลี่ยม
4 # w คือความยาวฐานของสามเหลี่ยม
5 # แสดงพื้นที่ของสามเหลี่ยม
6 #
7 h = int(input())
8 w = int(input())
9 a = h*w/2
10 print(a)
```

h ต้องเป็นจำนวนจริง

IPython Shell

```
Traceback (most recent call last):
  File "script.py", line 8, in <module>
    w = int(input())
ValueError: invalid literal for int() with base
          10: '29.4'
```

In [1]:

Solution

Submit



# แบบฝึกหัดสั้น ๆ

เขียนคำสั่งตามที่แสดงใน comment

script.py

```
1 # รับจำนวนจริงจาก input เก็บในตัวแปร h
2 # รับจำนวนจริงจาก input เก็บในตัวแปร w
3 # h คือความสูงของสามเหลี่ยม
4 # w คือความยาวฐานของสามเหลี่ยม
5 # แสดงพื้นที่ของสามเหลี่ยม
6 #
7 h = float(input())
8 w = float(input())
9 a = h*w/2
10 print(a)
```

Great work!

Solution

Submit

IPython Shell

```
Traceback (most recent call last):
  File "script.py", line 8, in <module>
    w = int(input())
ValueError: invalid literal for int() with base
          10: '29.4'

<script.py> output:
      323.4
```

In [1]:



# สำหรับ 2110101: แก้โจทย์ปัญหาที่ซับซ้อนกว่าในระบบ Grader

MAIN	Submissions				
01_Expr_01	01_Expr_★_Stirling_Factorial [desc]	4	Graded at 05/06/19 21:26:29 score: 100 [ PPPPP ]	<a href="#">Edit</a>	
			<b>View:</b> <a href="#">compiler msg</a> <a href="#">src</a>	<a href="#">submissions</a>	<a href="#">testcases</a>
01_Expr_02	01_Expr_★_Quadratic_Root [desc]	1	Graded at 04/06/19 21:28:14 score: 100 [ PPPPP ]	<a href="#">Edit</a>	
			<b>View:</b> <a href="#">compiler msg</a> <a href="#">src</a>	<a href="#">submissions</a>	<a href="#">testcases</a>
01_Expr_03	01_Expr_★★_An_Expression [desc]	11	Graded at 09/07/19 12:51:27 score: 0 [ x ]	<a href="#">Edit</a>	
			<b>View:</b> <a href="#">compiler msg</a> <a href="#">src</a>	<a href="#">submissions</a>	<a href="#">testcases</a>
01_Expr_04	01_Expr_★★_Body_Surface_Area [desc]	0	-	<a href="#">New</a>	
01_Expr_06	01_Expr_★★★_Duration [desc]	0	-	<a href="#">New</a>	

## Download Software

[www.python.org](http://www.python.org)

[www.thonny.org](http://www.thonny.org)

# Basic I/O, Data Types, Variables, Operators, Expressions, Statements

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# โครงสร้างโปรแกรมและลักษณะการทำงาน

ทำงานทีละคำสั่ง  
จากบนลงล่าง

บางคำสั่งก็เป็น  
แบบเลือกทำ

บางคำสั่งก็  
ทำงานเป็นวงวน

```
import matplotlib.pyplot as plt
import math

n = int(input()) # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

comment

ต้องเริ่มคำสั่งให้อยู่ในแนวเดียวกันกับคำสั่งอื่นในกลุ่ม

# Basic Data Types: String and Number

- String
  - "Hello" 'Hello'
  - "Hello Python" 'Hello Python'
  - "12345" '12345'
  - "" ''
- Number
  - integer
    - 1234, 0
  - floating point
    - -1234.0, 1.5E-15 ←  $1.5 \times 10^{-15}$

# Operator: plus

- ข้อความบวกกัน คือข้อความต่อ กัน
  - "Hello" + " " + "!" ได้ "Hello !"
  - "1" + "1" ได้ "11"
- จำนวนบวกกัน
  - 1 + 1 ได้ 2
  - 1 + 1.0 ได้ 2.0
- ข้อความบวกกับจำนวนไม่ได
  - "1" + 1 ← ผิด

# print( ... ) แสดงผลทางจอภาพ

```
print("Hello")  
print("Python")  
print("Hello" + " " + "Python")  
print("Hello", "Python")  
print("a", "b", "c", 1, 2, 3)  
print("1+1 =", (1+1))
```

program

output
Hello
Python
Hello Python
Hello Python
a b c 1 2 3
1+1 = 2

output

# ตัวแปร (Variables)

- ตัวแปรเป็นที่เก็บข้อมูล
  - ต้องมีชื่อกำกับ
  - เปลี่ยนค่าในที่เก็บได้

`name = "Python"`

นำค่า "Python" เก็บในตัวแปร name

`lang = name`

นำข้อมูลใน name เก็บในตัวแปร lang  
ทำให้ lang กับ name เก็บค่าเหมือนกัน

```
a = 1          # a = 1
b = 2          # a = 1; b = 2
c = a          # a = 1; b = 2; c = 1
d = c + b      # a = 1; b = 2; c = 1; d = 3
d = d + 5      # a = 1; b = 2; c = 1; d = 8
```

หมายความ  $1+2 = a$  ผิด !!!

# ชื่อตัวแปร

- ประกอบด้วยตัวอักษร ตัวเลข หรือเครื่องหมายขีดเส้นใต้
- ตัวอังกฤษใหญ่กับเล็กไม่เหมือนกัน
- ห้ามขีนต้นชื่อด้วยตัวเลข
- ไม่ซ้ำกับคำส่วนของภาษา

```
first_name = "John"  
last_name = "Wick"
```

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

# Basic Data Types

- str ข้อความ
- int จำนวนเต็ม
- float จำนวนจริง (มีจุดทศนิยม)
- bool ค่าจริงกับเท็จเท่านั้น (True กับ False)
- list ชุดข้อมูล เก็บเรียงเป็นลำดับ
  - [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
  - ["SU", "MO", "TU", "WE", "TH", "FR", "SA"]
  - [6230012021, [2110101, 2301107]]
  - [ ]

# ตัวอย่าง: Variables & Data Types

```
first_name = "Ranee"                      # str
last_name = "Campen"                       # str
aka       = "Bella"                         # str
age       = 29                             # int
height    = 1.65                           # float
is_single = True                           # bool
birth_date = [24, 12, 1989]                 # list
tv_series = ["Roy Marn", "Plerng Boon",
             "Bubphe Sanniwat", "Krong Kam"]
```

```
print("Name:", first_name, last_name)
print("Age:", age)
print("Aka:", aka, "or", aka[0:4:1])
d = birth_date[0]
m = birth_date[1]
y = birth_date[2]
print("Born:", str(d) + "/" + str(m) + "/" + str(y))
```

จะได้ศึกษาการจัดการข้อมูล  
ประเภทต่าง ๆ ต่อไป

# Type Conversions

- ข้อมูลส่วนใหญ่แปลงเป็น string ได้ด้วย str
- ข้อมูลบางอย่างก็อาจแปลงเป็น int กับ float ได้

```
s1 = "123"
s2 = " 456"
n = int(s1) + int(s2)      # 579
f = float(s1) + float(s2)   # 579.0
print(s1+s2, n, f)
print(s1+s2 + ", " + str(n) + ", " + str(f))
```

```
a = int(1)                  # a = 1
b = int(1.9)                 # b = 1
c = int(1.9 + 0.5)           # c = 2
d = float(1.1)                # d = 1.1
e = float(1)                  # e = 1.0
f = str("string")             # f = "string"
```

# input() รับสตริงจากแป้นพิมพ์

**input()** รอรับข้อมูลที่ป้อนทางคีย์บอร์ด  
พอกด enter ข้อมูลนั้นจะถูกแปลงเป็นสตริง  
นำมาเก็บในตัวแปร name

```
name = input()  
print(name + " is a very easy language.")  
print("We use " + name + " in our class.")  
print("Hello " + name + ".")
```

program

หลังจากสั่งทำงานแล้วกดคำว่า **Python** ก็จะได้ผลเป็น

```
Python is a very easy language.  
We use Python in our class.  
Hello Python.
```

output

# รับจำนวน คือ รับสตริงแล้วเปลี่ยนเป็นจำนวน

```
x = input()  
d = float(x)  
perimeter = d + d + d + d  
print("Perimeter of square =", perimeter)
```

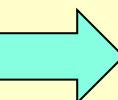
program

หลังจากสั่งทำงานแล้วกด 12 ก็จะได้ผลเป็น

Perimeter of square = 48.0

output

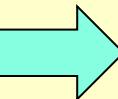
```
x = input()  
d = float(x)  
...
```



d = float(input())

รับจำนวนจริง

```
x = input()  
d = int(x)  
...
```



d = int(input())

รับจำนวนเต็ม

# Basic Arithmetic Operations

+ บวก

- ลบ

\* คูณ

/ หาร

// หารปัดเศษ

% เศษจากการหาร

\*\* ยกกำลัง

= กำหนดค่า

a = 5 + 2 # 7

b = 5 - 2 # 3

c = 5 \* 2 # 10

d = 5 \*\* 2 # 25

e = 5 / 2 # 2.5

f = 5 // 2 # 2

g = 5 % 2 # 1

a,b,c = c,a,b # a = 10

# b = 7

# c = 3

a,b = b,a # สลับค่า a กับ b

# Tips

```
a = float(input())
a = -a          # เปลี่ยนจำนวนลบเป็นบวก จำนวนบวกเป็นลบ
b = int(a)       # ปัดเศษของ a ทิ้ง
c = int(a + 0.5) # ปัดเศษของ a แบบมีขึ้นหรือลง
d = a - int(a)  # ค่าหลังจุดทศนิยมของ a
d = a % 1        # ค่าหลังจุดทศนิยมของ a
e = b % 10       # เลขหลักหน่วยของ b
f = b // 10 % 10 # เลขหลักสิบของ b
g = b // 10**4 % 10 # เลขหลักหมื่นของ b

h = input()        # h เก็บสตริง
h = int(h)         # h เก็บจำนวนเต็ม (ใช้ตัวแปรช้าได้)
```

# แบบฝึกหัด: รับชे�ลเซียส ส่งฟาร์นไฮต์

Input	Output
0	32
100	312

```
celsius = float(input())
```

# Augmented Assignments

```
a = 10
```

```
a = a + 5      # 15
```

```
a = a - 1      # 14
```

```
a = a * 2      # 28
```

```
a = a % 10     # 8
```

```
a = a ** 2     # 64
```

```
a = a // 10    # 6
```

```
a = 10
```

```
a += 5        # 15
```

```
a -= 1        # 14
```

```
a *= 2        # 28
```

```
a %= 10       # 8
```

```
a **= 2       # 64
```

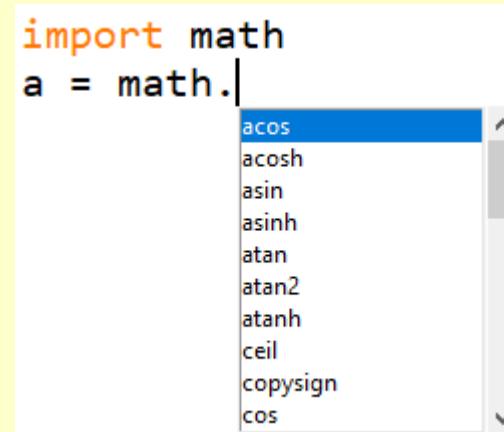
```
a // = 10     # 6
```

# math module

```
import math

degree = float(input())
radian = degree * 3.14159 / 180
radian = degree * math.pi / 180
radian = math.radians(degree)
s = math.sin(radian)
c = math.cos(radian)
g = math.log( 1E100, 10 )      # g = 100.0
```

มีฟังก์ชันทางคณิตศาสตร์  
มากมายให้ใช้ใน math  
module



# Built-in Functions

```
a = abs(-2)                      # a = 2
b = round(2/3, 2)                 # b = 0.67
c = max([4,1,5,3])                # c = 5
d = min([4,1,5,3])                # d = 1
e = sum([4,1,5,3])                # e = 13
f = len([4,1,5,3])                # f = 4
g = str(1234)                     # g = "1234"
h = int("123")                    # h = 123
j = float("-123.4")               # j = -123.4
k = input()
print(a,b,c)
```

จะได้เรียนวิธีการเขียนฟังก์ชันใหม่ ๆ ไว้ใช้เองต่อไป

# ໃຫ້ພຶກ໌ຂັນຂອນ ຈ ກຳນໄດ້

```
x1 = input()  
x2 = int(x1)  
x3 = abs(x2)  
x4 = 1 + x3  
x5 = str(x4)  
print("x5 = " + x5)
```

```
print("x5 = " + str(1 + abs(int(input())))))
```

ເຂົ້າໃຈຢາກ

# Operator Precedence: $2 + 3 * 4 = ?$

- ลำดับการคำนวณ จากก่อนไปหลังเป็นดังนี้
  - ในวงเล็บ
  - ยกกำลัง
  - ติดลบ
  - \* / // %
  - บวก และ ลบ
  - ถ้าพบหารด้วยตัวที่สำคัญเท่ากัน ให้ทำตัวซ้ายไปขวา  
(ยกเว้นยกกำลัง ทำขวามาซ้าย เช่น  $2^{**}3^{**}2 = 2^{**}9 = 512$ )

**PEMDAS:** Parentheses  
Exponential  
Multiply  
Divide  
Add  
Subtract

2 * 3 + 8 / - (2 - 4) - 2**2**3
2 * 3 + 8 / - ( -2 ) - 2**2**3
2 * 3 + 8 / - ( -2 ) - 2**8
2 * 3 + 8 / - ( -2 ) - 256
2 * 3 + 8 / 2 - 256
6 + 8 / 2 - 256
6 + 4.0 - 256
10.0 - 256

-246.0

# ตัวอย่าง: คำนวณพื้นที่และเส้นรอบวงของวงกลม

```
radius = float(input())  
  
area = math.pi * radius**2  
  
circum = 2 * math.pi * radius  
  
print("Area =", round(area, 2))  
  
print("Circumference =", round(circum, 2))
```

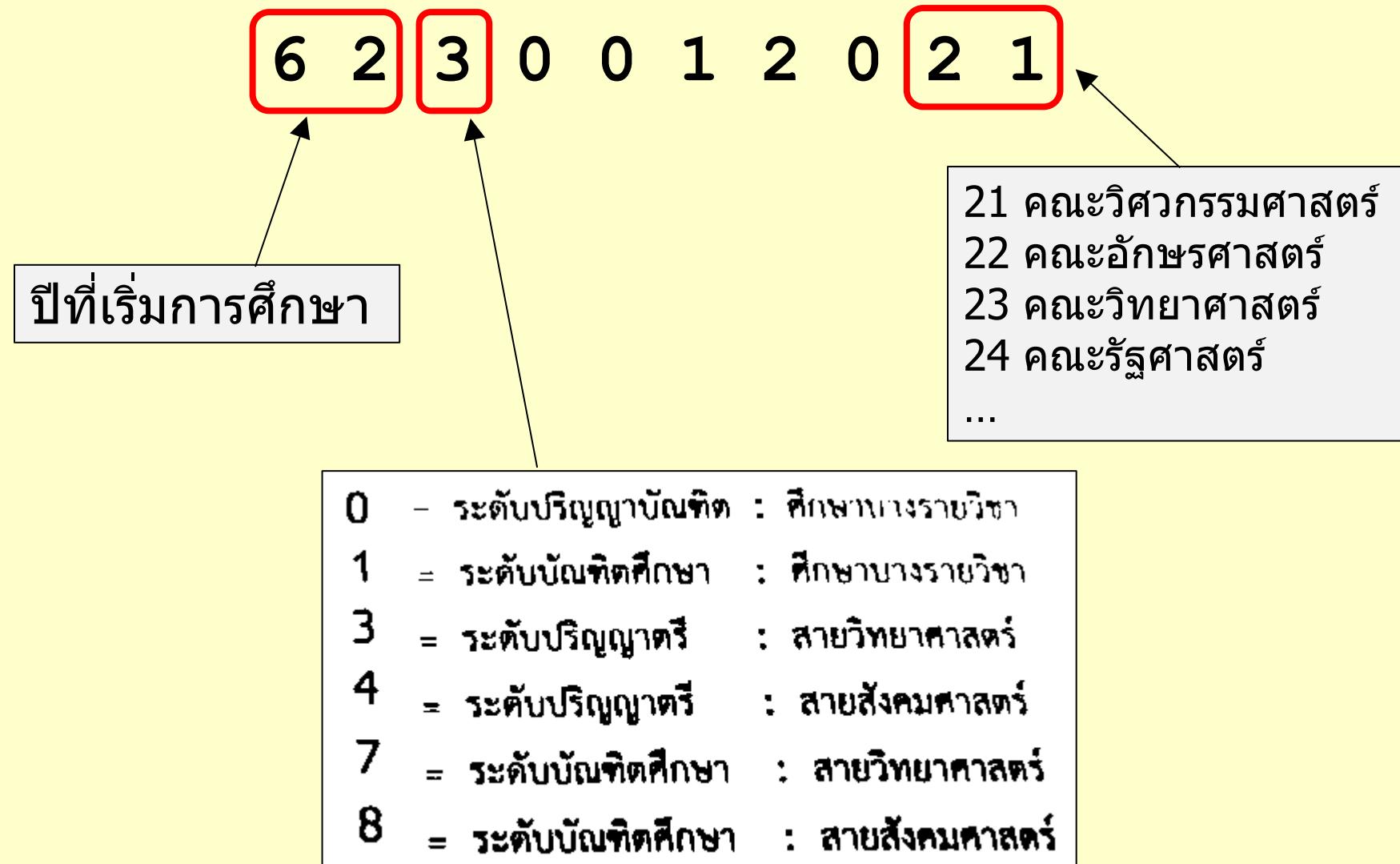
program

หลังจากสั่งทำงานแล้วป้อน 4.5 จะได้ผลเป็น

```
Area = 63.62  
Circumference = 28.27
```

output

# เลขประจำตัวนิสิต



# ตัวอย่าง: รหัสนิสิต → คณะอะไร เข้าปีอะไร ระดับใด

```
stu_id = int(input())
print("Student ID:", stu_id)
fac_code = stu_id % 100          # เลือกสองตัวขวา
year_in = 2500 + stu_id//10**8  # ตัด 8 ตัวขวา
deg_code = stu_id//10**7 % 10
print("Faculty code:", fac_code)
print("Enrollment year:", year_in)
print("Academic degree:", deg_code)
```

program

หลังจากสั่งทำงานแล้วป้อน 6230012021 จะได้ผลเป็น

```
Student ID: 6230012021    output
Faculty code: 21
Enrollment year: 2562
Academic degree: 3
```

output

# แสดงเลขประจำตัวนิสิตที่ลงทะเบียน หลักละบรรทัด

```
# sid  
sid = int(input()) # 6231020121  
d = sid % 10; print(d); sid //= 10 # 623102012  
d = sid % 10; print(d); sid //= 10 # 62310201  
d = sid % 10; print(d); sid //= 10 # 6231020  
d = sid % 10; print(d); sid //= 10 # 623102  
d = sid % 10; print(d); sid //= 10 # 62310  
d = sid % 10; print(d); sid //= 10 # 6231  
d = sid % 10; print(d); sid //= 10 # 623  
d = sid % 10; print(d); sid //= 10 # 62  
d = sid % 10; print(d); sid //= 10 # 6  
d = sid % 10; print(d)
```

เครื่องหมาย ; คั่นคำสั่งที่ต้องการเขียนต่อ กันในบรรทัดเดียว กัน

# แบบฝึกหัด: Stirling Formula

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

```
import math  
n = int(input())
```

# แบบฝึกหัด: : รากของ $ax^2+bx+c = 0$

รับ a b c จากแป้นพิมพ์  
คำนวณและแสดงสองรากจากสูตร

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# Basic String & List Operations

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# เรื่องที่ต้องรู้

- Length, Concatenation, Repetition
- Indexing and slicing
- String splitting

# String and List

- String: ลำดับของอักขระ
  - "ABCDEF"
- List: รายการของข้อมูล
  - [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
  - ["MO", "TU", "WE", "TH", "FR", "SA", "SU"]
- Operations
  - ความยาว    `len("ABC")`    `len([1,2,3])`
  - การต่อ    `"A"+"BC"`        `[1] + [2,3]`
  - การต่อ กัน ซ้ำ ๆ                            `"A" * 3`            `[0,0] * 3`
  - Indexing    `x[k]`
  - Slicing    `x[start : stop : step]`

# String & List: Length, Concatenation, Repetition

```
print(len("123"), len([1,2,3]))  
H = "HBD"  
T = "2U"  
HBD = (H + T)*2 + " " + H*2 + " " + H+T  
print(2*(HBD + " -- "))  
  
x = ([0,1]*2 + [3,4])*3  
print(x)
```

program

3 3

HBD2UHBD2U HBDHBD HBD2U -- HBD2UHBD2U HBDHBD HBD2U --

[0, 1, 0, 1, 3, 4, 0, 1, 0, 1, 3, 4, 0, 1, 0, 1, 3, 4]

output

# String: Indexing

**s = "HELLO World"**

index	0	1	2	3	4	5	6	7	8	9	10
	H	E	L	L	O			W	o	r	d
index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

สตริงขนาด N ตัวอักษร มี index ตั้งแต่ -N ถึง N – 1

s[0] คือ "H"

s[1] คือ "E"

s[10] คือ "d"

s[-1] คือ "d"

s[-2] คือ "l"

s[-11] คือ "H"

**s [ index นอกช่วง ] เจ็บ**

เช่น s[11] s[-12]

# String: Slicing

**s = "HELLO World"**

index	0	1	2	3	4	5	6	7	8	9	10
	H	E	L	L	O		W	o	r	l	d
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

**s[start:stop:step]**

↑ รวมตัวนี้      ↑ ไม่รวมตัวนี้

s[0:5:1]	คือ "HELLO"
s[1:11:2]	คือ "EL ol"
s[10:5:-1]	คือ "dlroW"
s[-1:-11:-2]	คือ "drWOL"

**s[50:99:1]** หรือ **s[-12:-99:-1]** ไม่เจ็บ ได้ ""

# String: Slicing

**s = "HELLO World"**

index	0	1	2	3	4	5	6	7	8	9	10
	H	E	L	L	O			W	o	r	d
index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

**s[start:stop:step]**

ไม่ใส่ start ถือว่า 0

ไม่ใส่ stop ถือว่า len+1

ไม่ใส่ step ถือว่า 1

ถ้า step ติดลบ

ไม่ใส่ start ถือว่า -1

ไม่ใส่ stop ถือว่า -(len+1)

`s[0:11:1]` เหมือน `s[:11:1]` เหมือน `s[0::1]`

เหมือน `s[::1]`     เหมือน `s[::]`

เหมือน `s[:]`     เหมือน `s`

`s[-1:-12:-1]` เหมือน `s[:-12:-1]` เหมือน `s[::-1]`

# ข้อควรระวัง: แก้ไขข้อมูลในสตริงไม่ได้

```
s = "Python"  
s[0] = "J"      # ผิด
```



```
s = "Python"  
s = "J" + s[1:] # ได้ เป็นการสร้างสตริงใหม่
```



# ตัวอย่าง: รหัสนิสิต คณะอะไร เข้าปีอะไร ระดับใด

```
stu_id = input()  
print("Student ID:", stu_id)  
print("Faculty code:", stu_id[-2:])  
print("Enrollment year:", "25" + stu_id[:2])  
print("Academic degree:", stu_id[2])
```

program

หลังจากสั่งทำงานแล้วป้อน 6230012021 จะได้ผลเป็น

```
Student ID: 6230012021  
Faculty code: 21  
Enrollment year: 2562  
Academic degree: 3
```

output

รับเลขประจำตัวเป็นสตริง  
เลือกหลักต่าง ๆ ได้ง่ายกว่ารับเป็นจำนวนเต็ม

# แบบฝึกหัด: เลขประจำตัวบัตรประชาชน

$$n_{12} = \left( 11 - \left( \sum_{k=0}^{11} (13 - k)n_k \right) \% 11 \right) \% 10$$

k	0	1	2	3	4	5	6	7	8	9	10	11	12
	3	7	2	0	8	0	1	4	4	1	1	5	1

$$13-k = 13 \quad 12 \quad 11 \quad 10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2$$

$$13 \times 3 + 12 \times 7 + 11 \times 2 + 10 \times 0 + 9 \times 8 + 8 \times 0 + 7 \times 1 + 6 \times 4 + 5 \times 4 + 4 \times 1 + 3 \times 1 + 2 \times 5 = \textcolor{blue}{285}$$
$$(11 - \textcolor{blue}{285} \% 11) \% 10 = \textcolor{red}{1}$$

เขียนโปรแกรมรับเลขประจำตัวบัตรประชาชนมา 12 หลักทางซ้าย  
แล้วคำนวณหลักขวาสุดด้วยสูตรข้างบน

# List: Indexing & Slicing (เหมือนของ String)

```
#      0   1   2   3   4   5   6   7   8   9   10
# -11-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
program
x = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
print("x =", x)
print(x[0], x[-1], x[:3], x[2:4])
print(x[-3:-1], x[1:-1])
print(x[::-1])
```

```
x = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55] output
0 55 [0, 1, 1] [1, 2]
[21, 34] [1, 1, 2, 3, 5, 8, 13, 21, 34]
[55, 34, 21, 13, 8, 5, 3, 2, 1, 1, 0]
```

## แก้ไขข้อมูลในลิสต์ได้ (แต่แก้ไขในสตริงไม่ได้)

```
x = [0]*7          # [0,0,0,0,0,0,0]
x[1] = 1          # [0,1,0,0,0,0,0]
x[2] = x[0] + x[1] # [0,1,1,0,0,0,0]
x[3] = x[1] + x[2] # [0,1,1,2,0,0,0]
x[4:7] = [3,5,8]   # [0,1,1,2,3,5,8]
```

# ตัวอย่าง: เลขวันเป็นชื่อวัน

```
days_of_week = ["Monday", "Tuesday", "Wednesday",
                 "Thursday", "Friday", "Saturday",
                 "Sunday"]

d = int(input())
print( days_of_week[d - 1] )
```

- |               |
|---------------|
| 1 → Monday    |
| 2 → Tuesday   |
| 3 → Wednesday |
| 4 → Thursday  |
| 5 → Friday    |
| 6 → Saturday  |
| 7 → Sunday    |

# แบบฝึกหัด: รับจำนวนเต็ม และแสดงคำอ่าน

Input                      Output

0                          zero

1                          one

```
n = int(input())
```

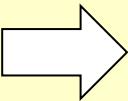
# split แยกสตริงออกเป็นลิสต์

- **x = s.split()**
  - แยกสตริง s ออกเป็นส่วน ๆ ได้เป็นลิสต์ (แยกด้วยช่องว่าง)
  - s = "11 2 33"
  - s.split() ได้ ["11", "2", "33"]
- **x = s.split( sep )**
  - แยกสตริง s ออกเป็นส่วน ๆ ได้เป็นลิสต์ ใช้ sep เป็นตัวแยก
  - s = "11:2: 33"
  - s.split(":") ได้ ["11", "2", " 33"]

```
"a,,,b".split(",") ได้ ["a", "", "", "", "b"]  
"a b".split(" ") ได้ ["a", "", "", "", "b"]  
"a b".split() ได้ ["a", "b"]  
"a,,,b".split(",") ได้ ["a", ",b"]
```

# รูปแบบการรับข้อมูล

```
x = input()  
t = x.split()  
...
```



```
t = input().split()  
...
```

623102021 John Wick

```
t = input().split()  
student_id = t[0]  
first_name = t[1]  
last_name = t[2]
```

623102021,90,87,2.54

```
t = input().split(",")  
student_id = t[0]  
midterm = int(t[1])  
final = int(t[2])  
gpa = float(t[3])
```

# เปลี่ยน วัน/เดือน/ปี ค.ศ. เป็น วัน/เดือน/ปี พ.ศ

```
inp = input()                                program
t = inp.split("/")  # "d/m/y" -> [d,m,y]
y = int(t[2]) + 543
out = t[0] + "/" + t[1] + "/" + str(y)
print( out )
```

หลังจากสั่งทำงานแล้วป้อน 5/12/2019 จะได้ผลเป็น

5/12/2562

output

"5/12/2019" → ["5", "12", "2019"]

# ข้อสังเกต

t = x.split()

ทำไมไม่เขียน →

t = split(x)

- **split** ไม่ใช่ฟังก์ชันทั่วไป
- **split** เป็นฟังก์ชันที่ใช้งานเฉพาะกับสตริง เรียกว่าเป็น *method* ของสตริง
- สตริงมีหลายเมธ็อดให้ใช้
- รูปแบบการใช้งาน :      **สตริง . ชื่อเมธ็อด (...)**

a = x.upper()    # สร้างสตริงใหม่ที่เป็นอังกฤษตัวใหญ่

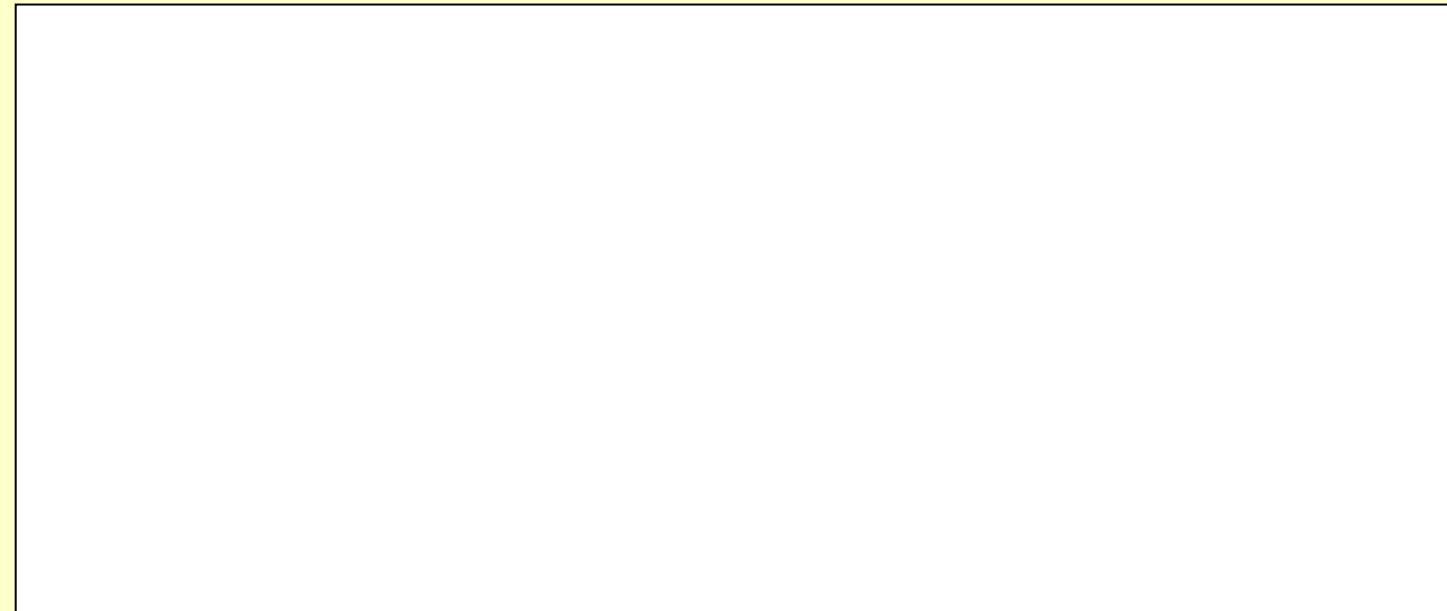
b = x.lower()    # สร้างสตริงใหม่ที่เป็นอังกฤษตัวเล็ก

c = x.strip()    # สร้างสตริงใหม่ที่ลบช่องว่างซ้ายขวาออก

\*\*\* ช่วงแรกนี้ยังไม่ได้ใช้ จะได้ศึกษาในบทถัด ๆ ไป \*\*\*

# แบบฝึกหัด: รับ 20/5/2010 แสดง May 20, 2010

Input	Output
1/1/2019	January 1, 2019
31/12/2020	December 31, 2020

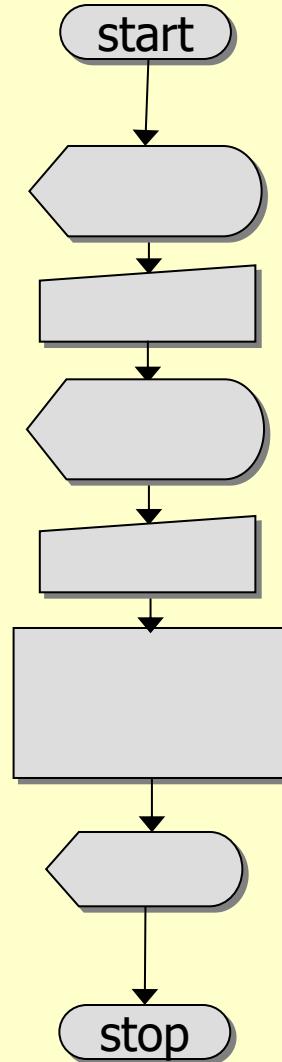


# การทำงานแบบเลือกทำ

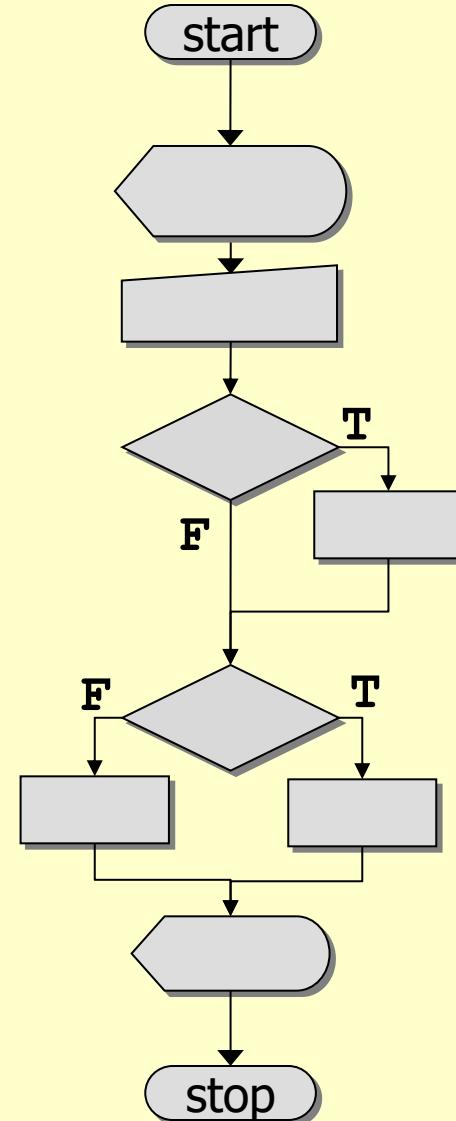
ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# ผังงาน (Flowchart)

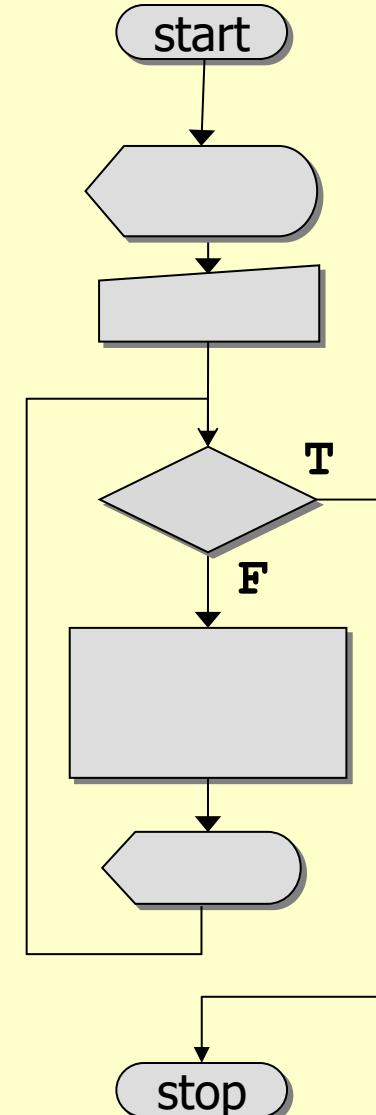
แบบลำดับ



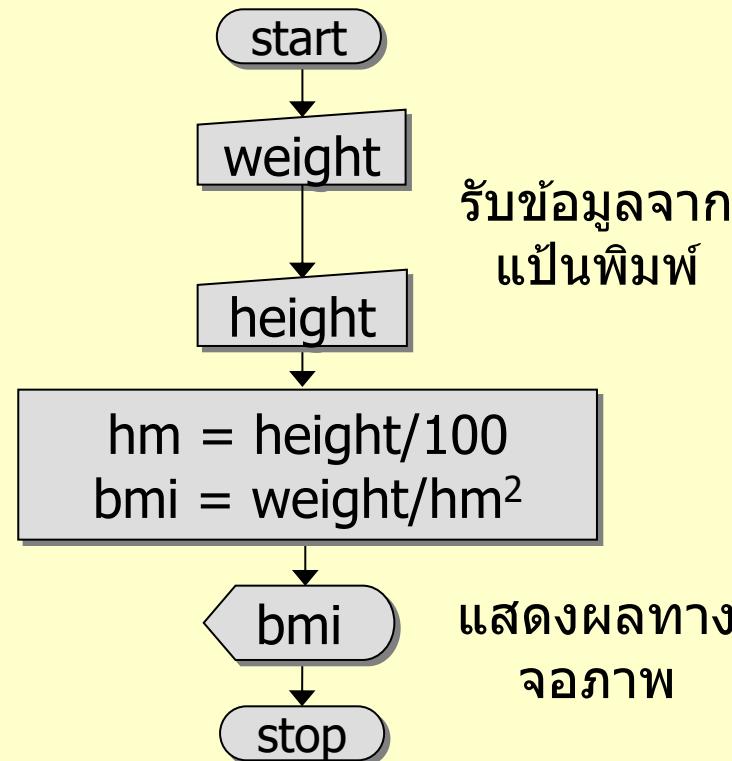
เลือกทำ



วน



# ผังงาน : บรรยายขั้นตอนการทำงาน



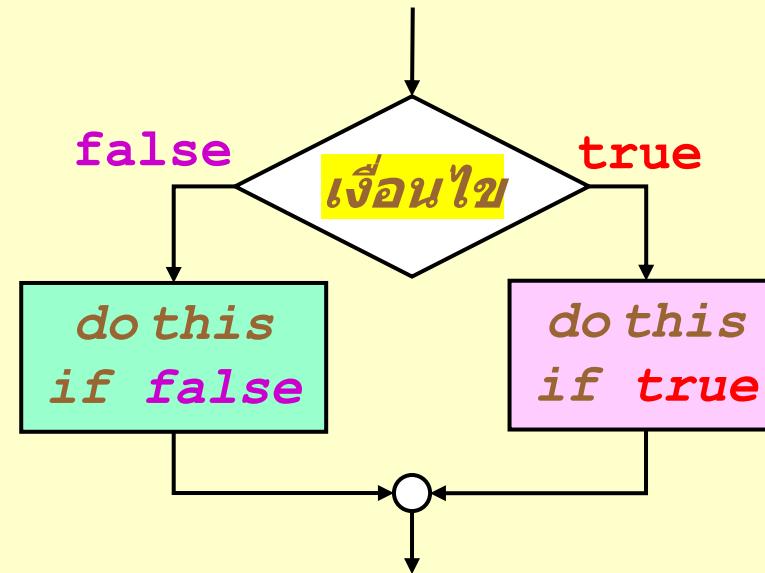
```
weight = float(input())  
  
height = float(input())  
  
hm = height / 100  
bmi = weight / (hm**2)  
  
print( bmi )
```

Flowchart

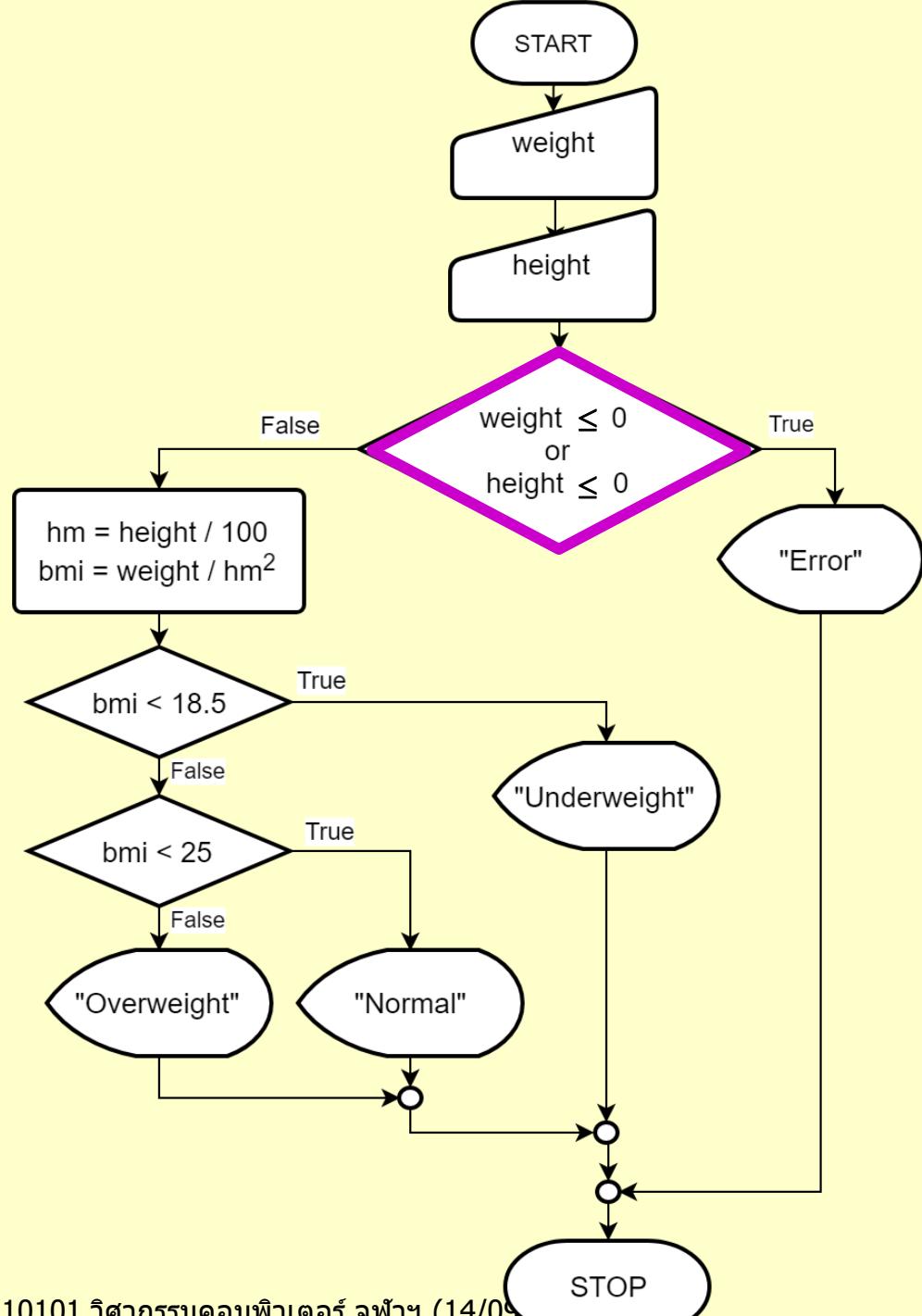


Program

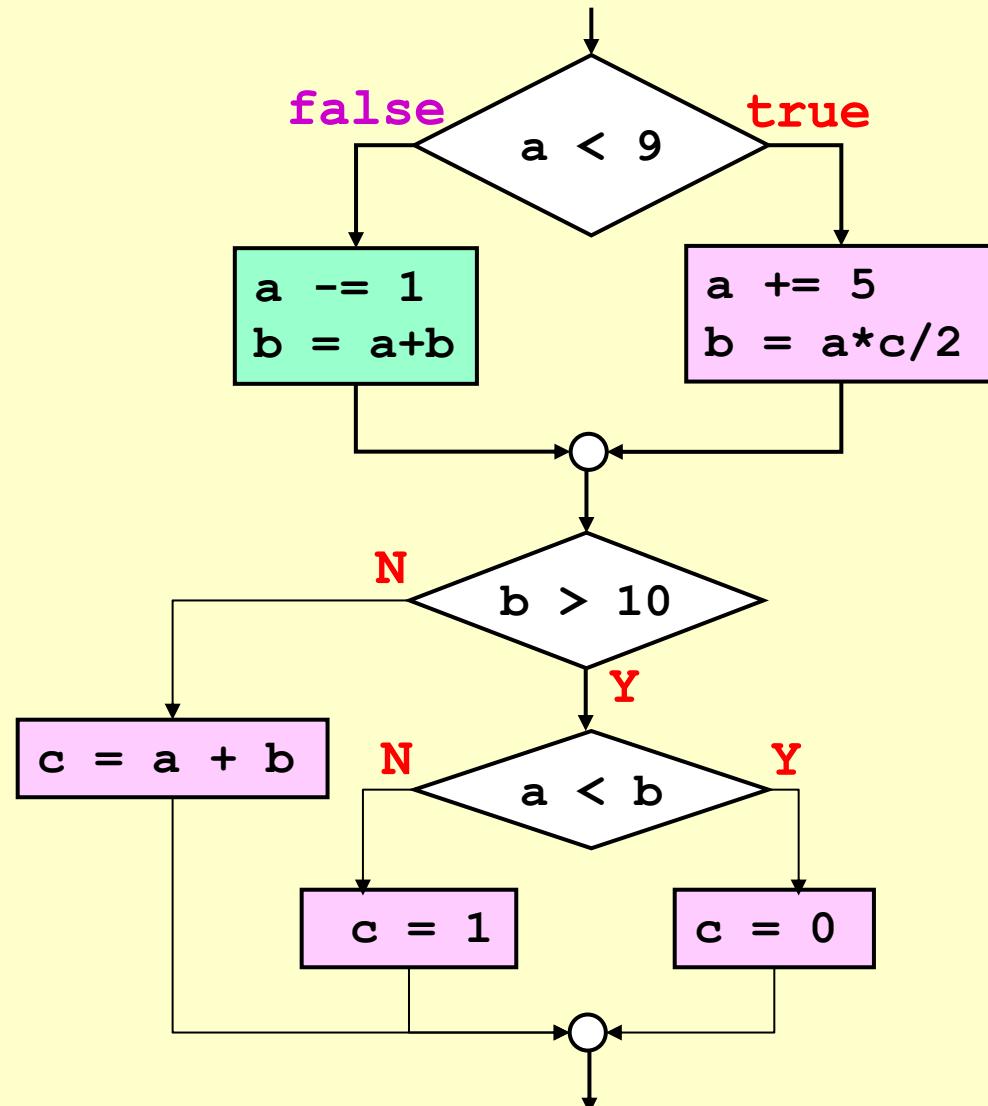
# ผังงานและคำสั่ง : if - else



```
if เงื่อนไข :  
    do this if true  
else :  
    do this if false
```



# การเขียนคำสั่ง if - else



```
if a < 9 :  
    a += 5  
    b = a * c / 2  
else :  
    a -= 1  
    b = a + b
```

ต้องมีเครื่องหมาย :

คำสั่งในกลุ่มเดียวกันต้องเยื้องเข้าไปทางขวา ให้ตรงกัน

```
if a < 9 :  
    a += 5  
    b = a * c / 2  
else :  
    a -= 1  
    b = a + b
```



กลุ่ม if กับ else "ไม่ตรงกันก็ได"

# นิพจน์บูลีน: กำหนดเงื่อนไข

- ได้ผลเป็น True หรือ False
- ใช้เครื่องหมาย `<` `>` `<=` `>=` `!=` `==` เพื่อเปรียบเทียบว่า `<` `>` `≤` `≥` `≠` เท่ากัน
- เข้มการเปรียบเทียบได้ด้วย `and` `or`
- กลับผลการเปรียบเทียบได้ด้วย `not`
- ถ้ามีทั้ง `and` `or` `not` จะทำ `not` ก่อน แล้ว `and` แล้ว `or`

```
if m == 4 or m == 6 or m == 9 or m == 11 :  
    print('ลงท้ายด้วย "ยน"')
```

```
if not(m == 2 or m == 4 or \  
        m == 6 or m == 9 or m == 11) :  
    print('ลงท้ายด้วย "คอม"')
```

```
if m != 2 and m != 4 and \  
    m != 6 and m != 9 and m != 11 :  
    print('ลงท้ายด้วย "คอม"')
```

# ตัวอย่าง

```
if x % 2 == 0 : # ถ้า x เป็นจำนวนคู่
```

```
if 1 <= m <= 12: # ถ้า m มีค่าตั้งแต่ 1 ถึง 12
```

```
if y % 400 == 0: # ถ้า y หารด้วย 400 ลงตัว
```

```
if student_id[-2:] == "21": # ถ้าเป็นนิสิตวิศวฯ
```

```
if tel_no[:2] == "02": # ถ้าเป็นโทรศัพท์ใน กทม
```

# แบบฝึกหัด: หมายเลขอุตสาหกรรมเคลื่อนที่

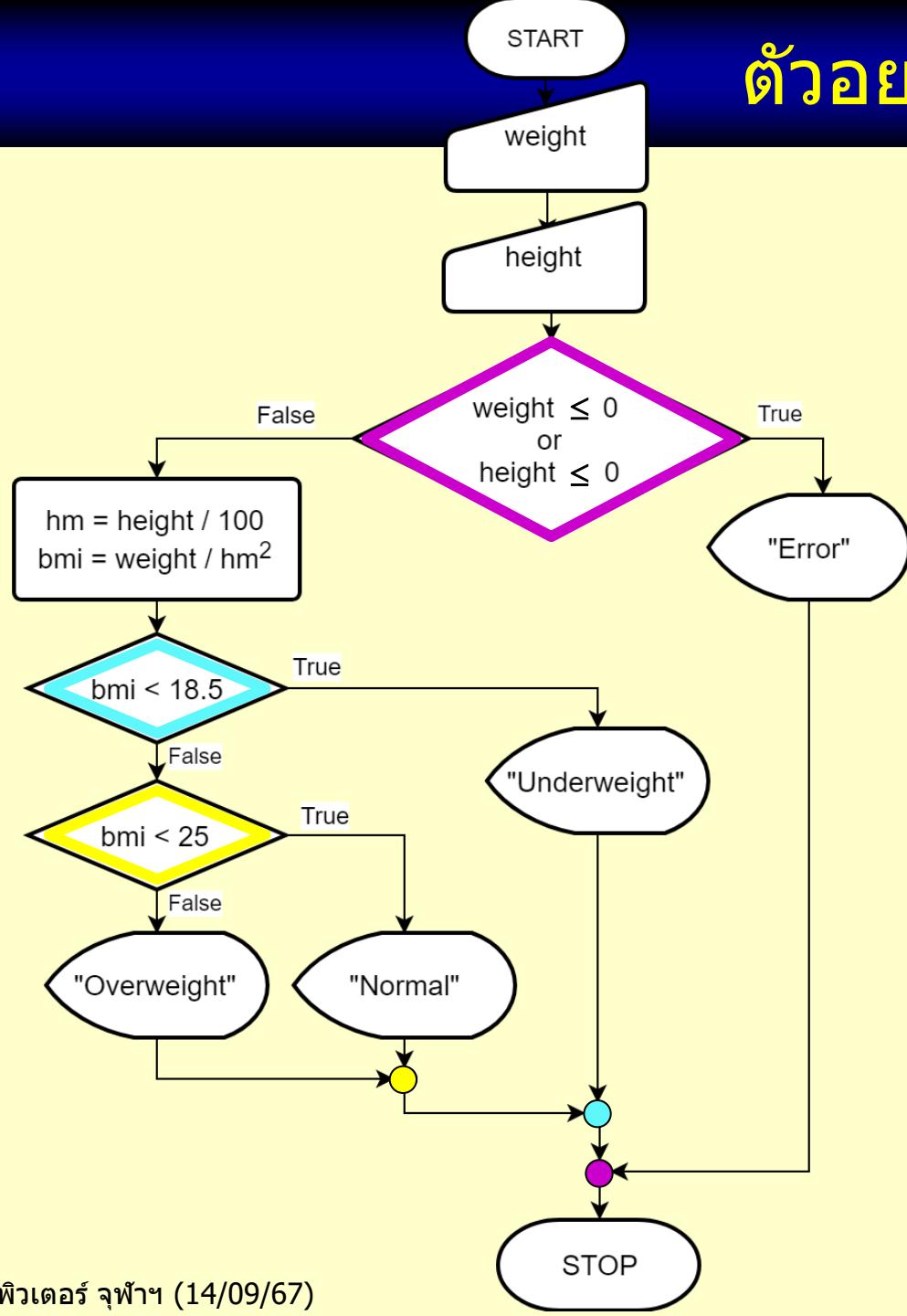
```
tel_no = input()  
  
if  
  
    print("Mobile number")  
else:  
    print("Not a mobile number")
```

tel\_no เป็นหมายเลขอุตสาหกรรมเคลื่อนที่ เมื่อมี

- 10 หลัก และ
- ขึ้นต้นด้วยเลข 06, 08 หรือ 09

(ไม่ต้องตรวจหลักอื่นว่าเป็นตัวเลข  
ถ้าตรวจจะยาวเกินไป ค่อยทำในภายหลัง)

# ตัวอย่าง



```

weight = float(input())

height = float(input())

if weight <= 0 or \
height <= 0 :
    print("Error")
else :
    hm = height / 100
    bmi = weight / hm**2
    if bmi < 18.5 :
        print("Underweight")
    else :
        if bmi < 25 :
            print("Normal")
        else:
            print("Overweight")
  
```

# การเปรียบเทียบลิสต์กับลิสต์

- **เปรียบเทียบที่ละตัวจากซ้ายไปขวา**

[**10**, 2] > [**9**, 9, 9] เป็นจริง

[**10**, 2] > [**10**, 1, 9] เป็นจริง

[**10**, 2] > [**10**] เป็นจริง

[**10**] > [] เป็นจริง

# การเปรียบเทียบสตริงกับสตริง

- อักขระองค์กรตัวเล็กมีค่ามากกว่าตัวใหญ่
- ตัวอักษรมีค่าน้อยไปมากตามลำดับ A ถึง Z  
"A" < "Z" เป็นจริง  
"A" < "B" < "Z" < "a" < "b" < "z" เป็นจริง
- สตริงตัวเลขมีค่าน้อยไปมากตามลำดับ 0 ถึง 9  
"0" < "1" < "2" < "3" < "4" < "9" เป็นจริง
- เปรียบเทียบสตริงจะเปรียบเทียบทีละตัวจากซ้ายไปขวา
  - "**A**BC" < "aA" เป็นจริง
  - "**A**BC" < "**A**CAA" เป็นจริง
  - "**A**BC" < "**A**BCC" เป็นจริง
  - "**1**00" < "**1**9" เป็นจริง

# ตัวอย่าง

```
if "a" <= x <= "z" or \
    "A" <= x <= "Z" :      # ถ้า x เป็นตัวอักษรอังกฤษ
```

```
if "0" <= x <= "9" :      # ถ้า x เป็นตัวเลข
```

```
# ถ้าเกิดหลัง 31 ก.ย. 2540, birth_date = [y,m,d]
if birth_date > [2540,9,31] :
```

# แบบฝึกหัด: เลขใหญ่เล็ก

โปรแกรมรับอักขระสามตัว ตรวจว่า ตัวซ้าย  
ตัวกลาง และตัวขวา เป็น ตัวเลข ตัวอังกฤษใหญ่  
และ ตัวอังกฤษเล็ก ตามลำดับ หรือไม่

Input		Output
9Ab	→	Yes
0xj	→	Yes
123	→	No
1x	→	No

# if a in b : a อยู่ใน b ?

- ถ้า a กับ b เป็นสตริง
  - a in b หวาน่า a เป็นสตริงป่อຍของ b หรือไม่
  - "python" in "I love python" เป็นจริง
  - "python" in "I love Python" เป็นเท็จ
- ถ้า b เป็นลิสต์
  - a in b หวาน่า a เป็นสมาชิกในลิสต์ b หรือไม่
  - 6 in [2, 4, 6, 8, 10] เป็นจริง
  - "Bones" in ["Kirk", "McCoy", "Spock"] เป็นเท็จ
- if not (a in b) : # ถ้า a ไม่อยู่ใน b
- if a not in b : # ถ้า a ไม่อยู่ใน b

# ตัวอย่าง

# ให้ x เป็นสตริงที่เก็บตัวอักษรตัวเดียว

# ทดสอบว่า x เก็บตัวอักษรที่เป็นสระ

```
if x=="a" or x=="e" or x=="i" or \
x=="o" or x=="u" or x=="A" or \
x=="E" or x=="I" or x=="O" or x=="U" :  
    ยานมาก
```

```
if x in "aeiouAEIOU" :
```

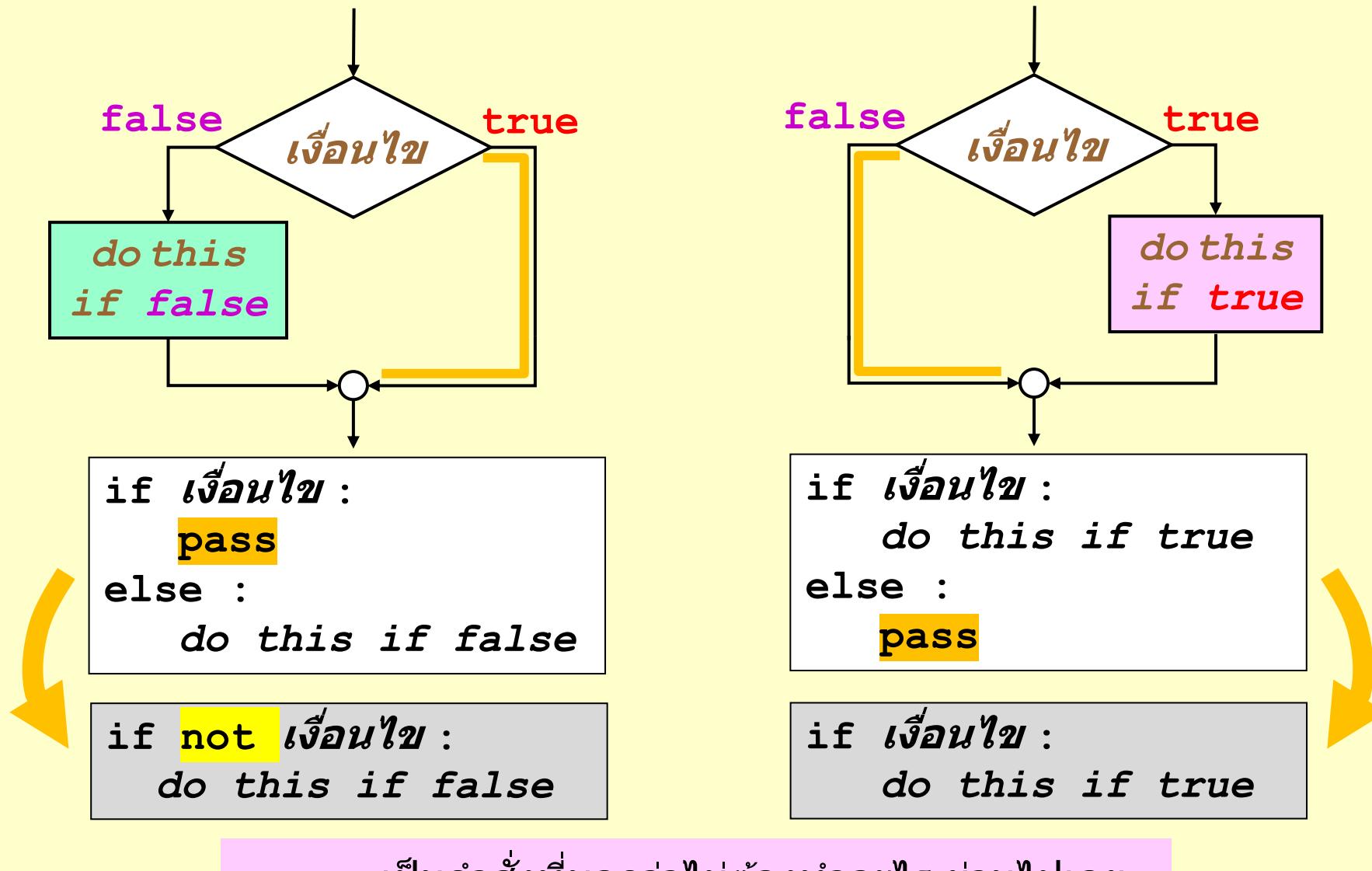
ถ้า x เก็บ "ei"  
ก็เป็นจริง

```
if x in ["a","e","i","o","u",
"A","E","I","O","U"] :  
    แบบนี้ชัวร์สุด
```

# แบบฝึกหัด: ตรวจรหัสที่รับมาว่าถูกต้องหรือไม่ ?

01	สถาบันภาษาไทยสิรินธร	32	คณะทันตแพทยศาสตร์
02	ศูนย์การศึกษาทั่วไป	33	คณะเภสัชศาสตร์
20	บัณฑิตวิทยาลัย	34	คณะนิติศาสตร์
21	คณะวิศวกรรมศาสตร์	35	คณะศิลปกรรมศาสตร์
22	คณะอักษรศาสตร์	36	คณะพยาบาลศาสตร์
23	คณะวิทยาศาสตร์	37	คณะสหเวชศาสตร์
24	คณะรัฐศาสตร์	38	คณะจิตวิทยา
25	คณะสถาปัตยกรรมศาสตร์	39	คณะวิทยาศาสตร์การกีฬា
26	คณะพาณิชยศาสตร์ และการบัญชี	40	สำนักวิชาทรัพยากร การเกษตร
27	คณะครุศาสตร์	51	วิทยาลัยประชารัฐศาสตร์
28	คณะนิเทศศาสตร์	53	วิทยาลัยวิทยาศาสตร์
29	คณะเศรษฐศาสตร์		สาธารณสุข
30	คณะแพทยศาสตร์	55	สถาบันภาษา
31	คณะสัตวแพทยศาสตร์	58	สถาบันบัณฑิตบริหารธุรกิจ ศศินทร์ฯ

# รูปแบบคำสั่งกรณี จริงถึงทำ / เท็จถึงทำ



# ตัวอย่าง: ปีนี้มีกี่วัน

- ให้  $y$  เก็บเลขปี ค.ศ.  
เดือนกุมภาพันธ์มี 29 วัน ก็เมื่อ
  - $y$  หารด้วย 400 ลงตัว หรือ
  - $y$  หารด้วย 4 ลงตัว แต่ต้องหารด้วย 100 ไม่ลงตัว
  - เช่น ปี 2004, 2000, (ปี 2100 ไม่เงื่อนไข)

```
y = int(input())
days_in_year = 365

if (y%400 == 0) or \
(y%4 == 0 and y%100 != 0):
    days_in_year = 366

print(days_in_year)
```

## ตัวอย่าง: ชื่อ 2 แคน 1

```
p1 = float(input())
p2 = float(input())
p3 = float(input())
```

รับข้อมูล

```
min_p = p1
if p2 < min_p:
    min_p = p2
if p3 < min_p:
    min_p = p3
```

หาค่าน้อยสุด

```
total_p = p1 + p2 + p3
print("Total:", total_p)
print("Get on free:", min_p)
print("Pay only:", total_p - min_p)
```

แสดงผล

# แบบฝึกหัด: คณิตศาสตร์

คณิตศาสตร์ 4 คน ตัดคณิตที่ต่ำสุดและสูงสุดออก  
แล้วนำคณิตของสองคนที่เหลือมาหาค่าเฉลี่ย

Input

9.15 9.20 9.30 9.50

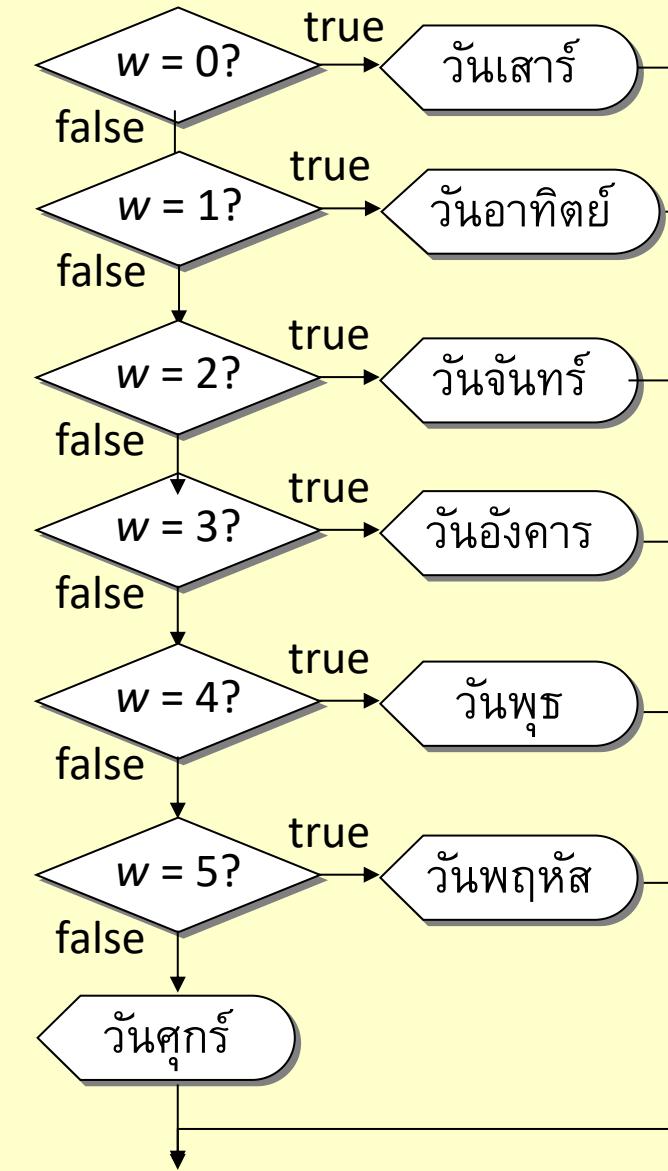
Output

9.25

```
x = input().split()  
s1 = float(x[0])  
s2 = float(x[1])  
s3 = float(x[2])  
s4 = float(x[3])
```

# if – else – if – else - ... if – else

```
if w == 0 :  
    print("เสา")  
else :  
    if w == 1 :  
        print("อาทิตย์")  
    else :  
        if w == 2 :  
            print("จันทร์")  
        else :  
            if w == 3 :  
                print("อังคาร")  
            else :  
                if w == 4 :  
                    print("พุธ")  
                else :  
                    if w == 5 :  
                        print("พฤหัสบดี")  
                    else :  
                        print("ศุกร์")
```



if – else – if – else



if – elif – else

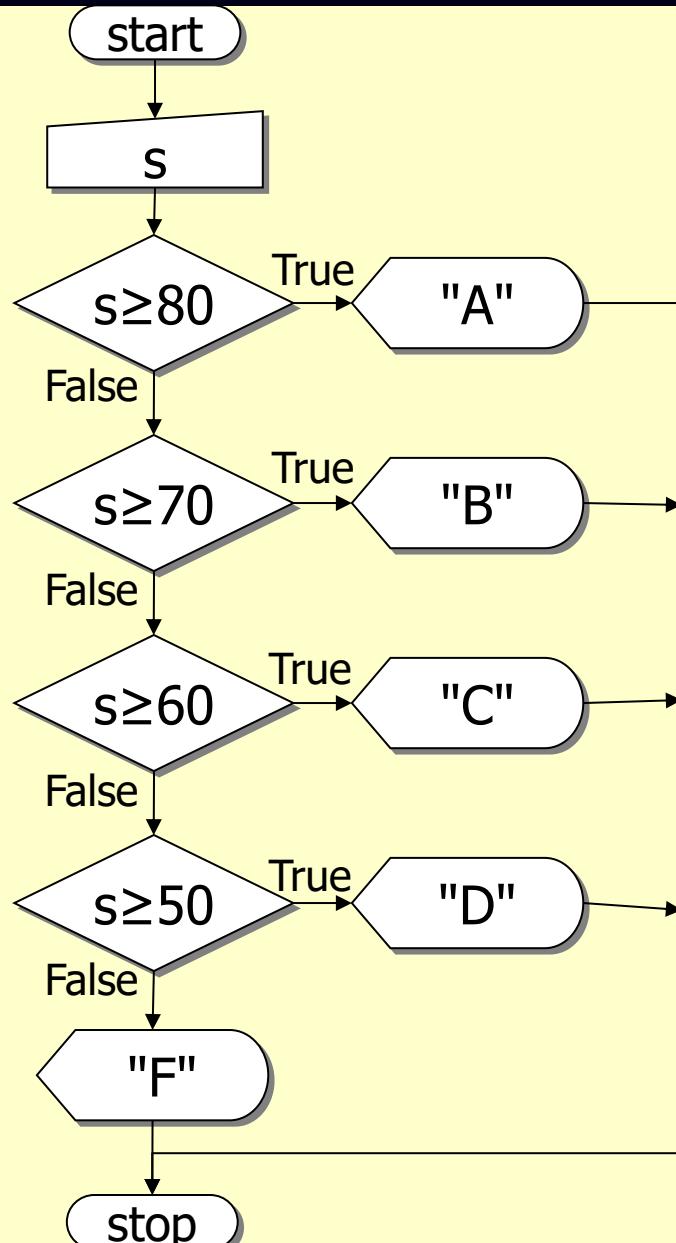
```
if w == 0 :  
    print("เสา")  
else :  
    if w == 1 :  
        print("อาทิตย์")  
    else :  
        if w == 2 :  
            print("จันทร์")  
        else :  
            if w == 3 :  
                print("อังคาร")  
            else :  
                if w == 4 :  
                    print("พุธ")  
                else :  
                    if w == 5 :  
                        print("พฤหัสบดี")  
                    else :  
                        print("ศุกร์")
```



```
if w == 0 :  
    print("เสา")  
  
elif w == 1 :  
    print("อาทิตย์")  
  
elif w == 2 :  
    print("จันทร์")  
  
elif w == 3 :  
    print("อังคาร")  
  
elif w == 4 :  
    print("พุธ")  
  
elif w == 5 :  
    print("พฤหัสบดี")  
else :  
    print("ศุกร์")
```



# แบบฝึกหัด: ตัดเกรด



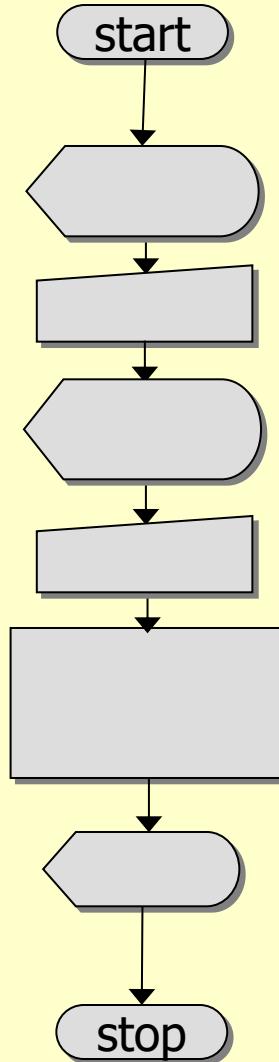
```
s = float(input())
```

# การทำงานแบบง่วน

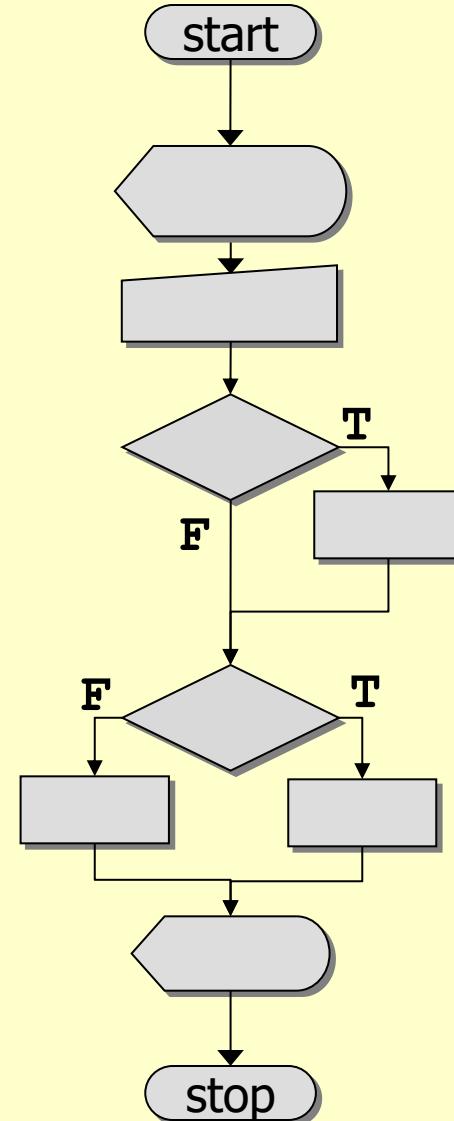
ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# ผังงาน (Flowchart)

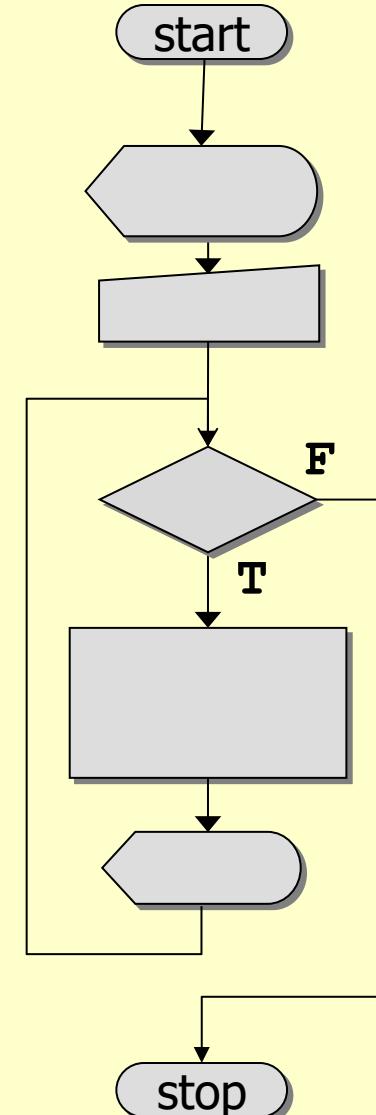
แบบลำดับ



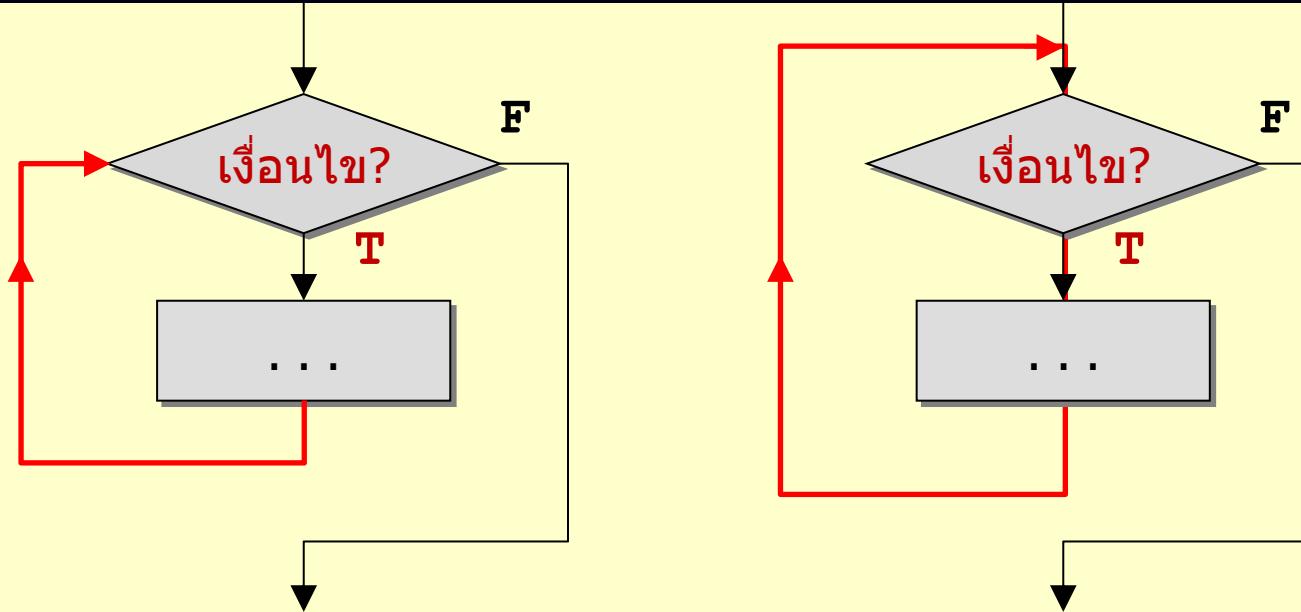
เลือกทำ



วน



# วงวน : while



ต้องมีเครื่องหมาย :

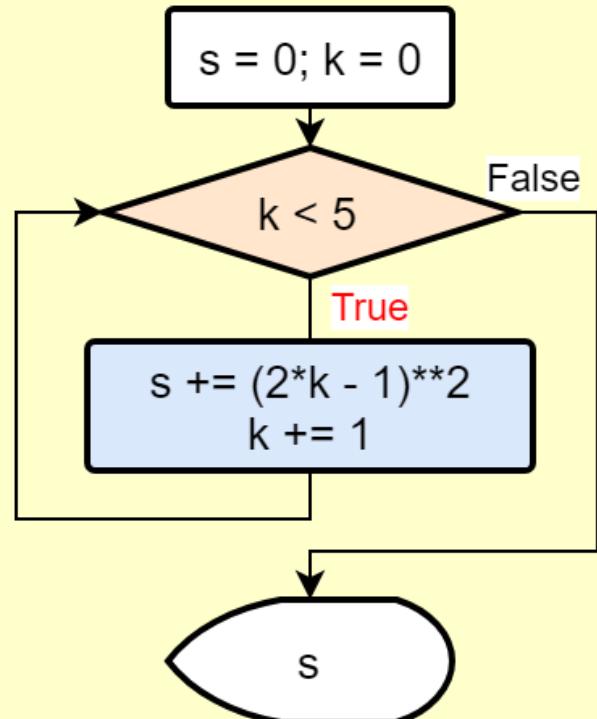
`while เงื่อนไข :`

กลุ่มคำสั่งที่ทำงานเมื่อเงื่อนไขเป็นจริง

คำสั่งในกลุ่มต้องเยื่องเข้าไปทางขวา ให้ตรงกันหมด

# ตัวอย่าง:

$$\sum_{k=0}^4 (2k - 1)^2$$



## ตัวอย่าง:

$$\sum_{k=0}^4 (2k - 1)^2$$

1

```
s = 0
s += (2*0 - 1)**2
s += (2*1 - 1)**2
s += (2*2 - 1)**2
s += (2*3 - 1)**2
s += (2*4 - 1)**2
print(s)
```

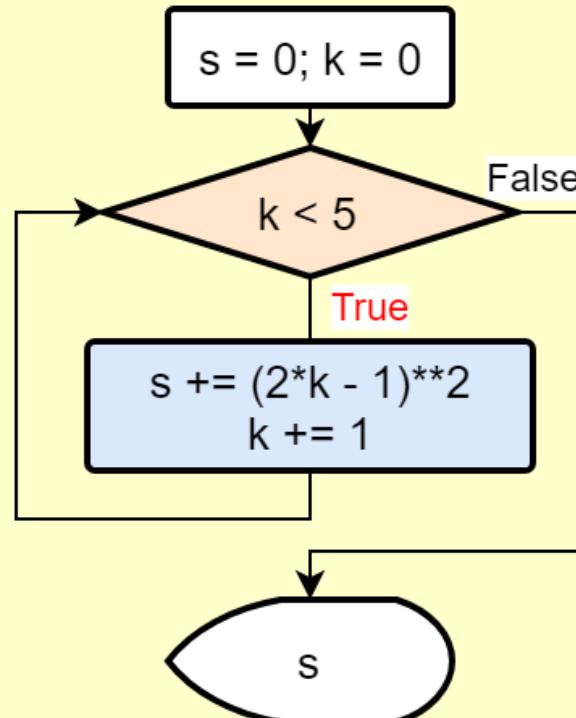
2

```
s = 0; k = 0
s += (2*k - 1)**2; k += 1
s += (2*k - 1)**2;
print(s)
```

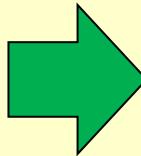
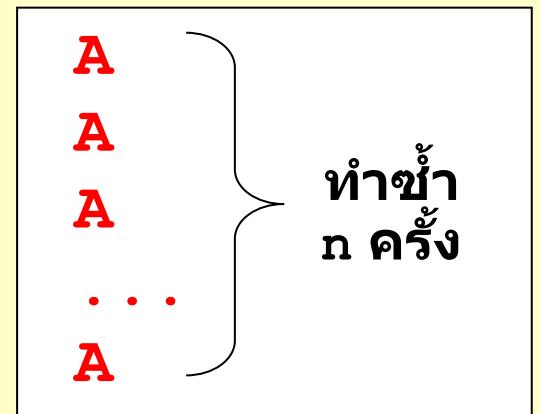
3

```
s = 0; k = 0
while k < 5 :
    s += (2*k - 1)**2
    k += 1
print(s)
```

หมุนทำซ้ำ 5 รอบ k เริ่มที่ 0 แต่ละรอบเพิ่ม 1  
เมื่อ k เป็น 5 เงื่อนไขเป็นเท็จ ออกจากวงวน



# รูปแบบ: วนนเพื่อทำซุ่ดคำสั่งซ้ำ ๆ



```
k = 0
while k < n:
    A
    k += 1
```

```
k = 1
while k <= n:
    A
    k += 1
```

## ตัวอย่าง: หาค่าน้อยสุดจากข้อมูล 5 ตัว

```
min_v = float(input())
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
print("min =", min_v)
```

```
min_v = float(input())
k = 0
while k < 4:
    v = float(input())
    if v < min_v:
        min_v = v
    k += 1
print("min =", min_v)
```

# ตัวอย่าง: หากค่าน้อยสุดจากข้อมูลหลายตัว

4	←	บอกก่อนว่า จะต้องรับกี่ตัว
10.0		
11.2		
15.5		
12.4		

10.0
11.2
15.5
12.4
q ← รับจำนวนที่ q

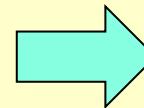
# แบบฝึกหัด: หาค่าเฉลี่ยจากข้อมูลหลายตัว

Input

10.0  
10.0  
11.0  
12.0  
q

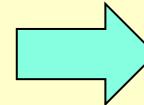
Output

10.75



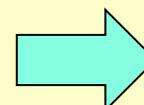
10.5  
q

10.5



q

No Data



# ตัวอย่าง: หารากที่สองด้วยวิธี bisection



# ตัวอย่าง: หารากที่สองด้วยวิธี bisection

- ต้องการหารากที่สองของ  $a$
- ให้  $L = 0, U = a$
- คำตอบอยู่ใน  $[L, U]$
- $x$  = จุดกึ่งกลางของช่วง
- ทำข้างล่างนี้ซ้ำ ถ้า  $x^2 \neq a$ 
  - ถ้า  $x^2 > a$ :  
เปลี่ยนช่วงเป็น  $[L, x]$
  - ถ้า  $x^2 < a$ :  
เปลี่ยนช่วงเป็น  $[x, U]$
  - $x$  = จุดกึ่งกลางของช่วง



$a = 25$		
$L$	$U$	$x$
0	25	12.5
0	12.5	6.25
0	6.25	3.125
3.125	6.25	4.6875
4.6875	6.25	5.46875
4.6875	5.46875	5.078125
4.6875	5.078125	4.882813
4.882813	5.078125	4.980469
4.980469	5.078125	5.029297
4.980469	5.029297	5.004883

$$x = \frac{L + U}{2}$$

# ตัวอย่าง: หารากที่สองด้วยวิธี bisection

- ต้องการหารากที่สองของ  $a$
- ให้  $L = 0, U = a$
- คำตอบอยู่ใน  $[L, U]$
- $x$  = จุดกึ่งกลางของช่วง
- ทำข้างล่างนี้ซ้ำ ถ้า  $x^2 \neq a$ 
  - ถ้า  $x^2 > a$ :  
เปลี่ยนช่วงเป็น  $[L, x]$
  - ถ้า  $x^2 < a$ :  
เปลี่ยนช่วงเป็น  $[x, U]$
  - $x$  = จุดกึ่งกลางของช่วง



```
a = float(input())
L = 0; U = a
x = (L + U)/2
while x**2 != a:
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U)/2
print(x)
```

# ตัวอย่าง: หารากที่สองด้วยวิธี bisection

```
a = float(input())
L = 0; U = a
x = (L + U) / 2
while x**2 != a:
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U) / 2
print(x)
```

```
a = float(input())
L = 0; U = a
x = (L + U) / 2
while ยังไม่ใกล้กัน(x**2, a):
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U) / 2
print(x)
```

$a$  กับ  $b$  ยังไม่ใกล้กันเมื่อ  $|a - b| > \varepsilon \max(a, b)$

ให้  $a$  กับ  $b$  เป็นบวก,  $\varepsilon$  และ  $10^{-9}$   $\text{abs}(a-b) > 1e-9 * \max(a, b)$

## แบบฝึกหัด: หา $\log_{10}a$ ด้วย bisection

เขียนโปรแกรมรับ  $a$  เพื่อหา  $\log_{10} a$  ด้วย bisection  
โดยที่  $1 \leq a \leq 600$

ถ้าต้องการให้ท้าทายขึ้น ก็ให้  $a$  เกิน 600 มาก ๆ ได้

# วงวน for

```
for k in range(start, stop, step) :  
    ...
```

```
for c in a_string :  
    ...
```

```
for e in a_list :  
    ...
```

## แบบที่ 1: for k in range(start, stop, step)

```
for k in range(4,100,2) :
```

...

k = 4, 6, 8, ..., 98

```
for k in range(100, 0,-1) :
```

...

k = 100, 99, 98, ..., 1

```
for k in range(5,100) :
```

...

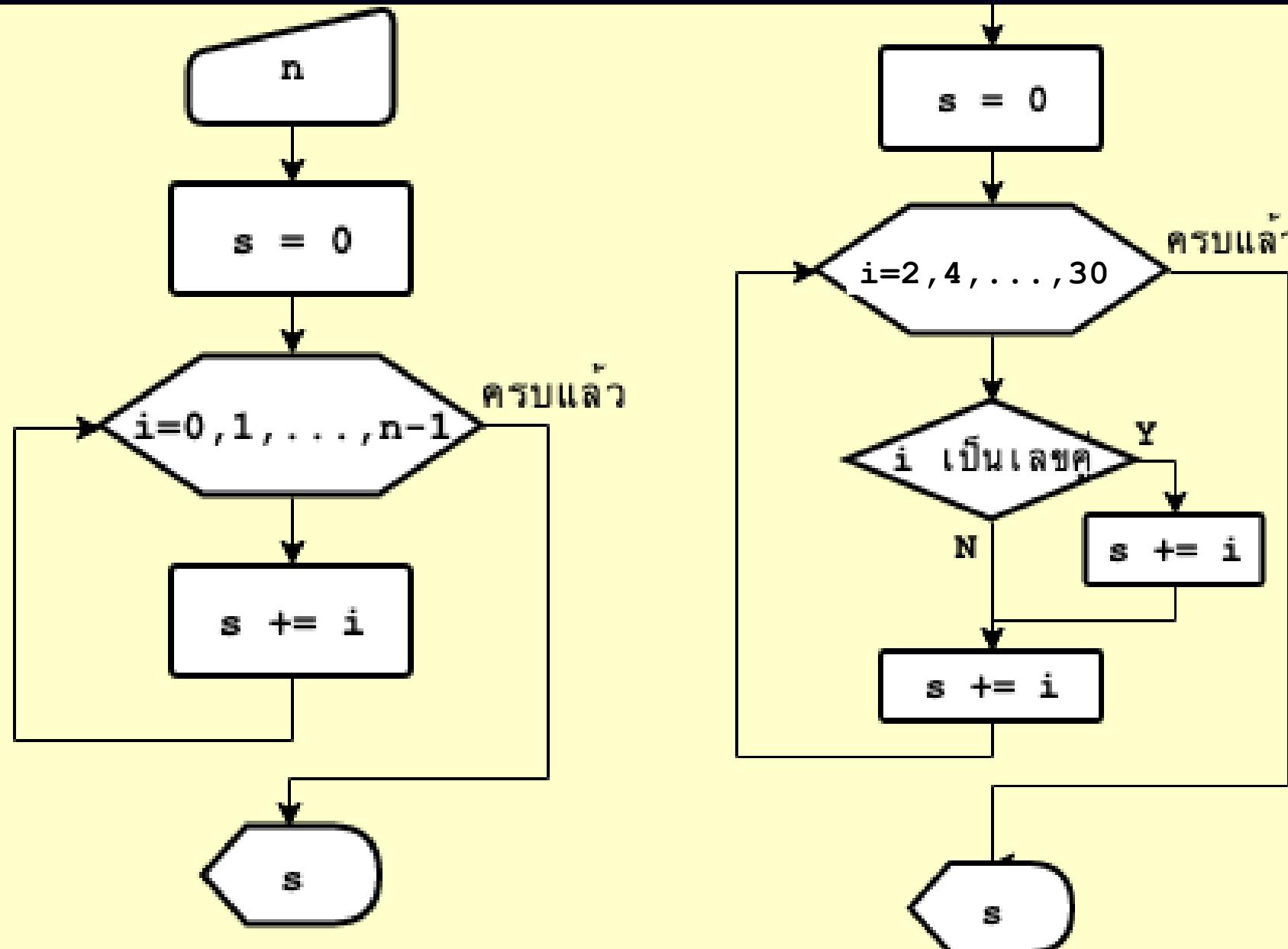
k = 5, 6, 7, ..., 99

```
for k in range(100) :
```

...

k = 0, 1, 2, ..., 99

# แบบที่ 1: for k in range(start, stop, step)



# ตัวอย่าง: $\mu$ และ $\sigma$

```
N = int(input())
x = [0.0]*N
for i in range(N):
    x[i] = float(input())

s = 0
for i in range(N):
    s += x[i]
mean = s / N

s2 = 0
for i in range(N):
    s2 += (x[i]-mean)**2
sd = (s2/n)**0.5

print(mean, sd)
```

input

5
10
11
13
10
12

$$\mu = \frac{\sum_{i=1}^N x_i}{N}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{N}}$$

## ตัวอย่าง: Dot Product $u \cdot v$

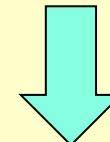
```
x = input().split()
u = [0.0]*len(x)
for i in range(len(x)) :
    u[i] = float(x[i])

x = input().split()
v = [0.0]*len(x)
for i in range(len(x)) :
    v[i] = float(x[i])

dot = 0
for i in range(len(x)) :
    dot += u[i]*v[i]

print( dot )
```

1	2	0	2	1
2	2	1	2	2



12.0
------

## แบบฝึกหัด: ตรวจคำตอบปrynay

```
sol = input() # e.g., ABBBAAABCCBABABDCCDA  
ans = input() # e.g., ABBBAABBCCBAABBDCDDB
```

```
# นับว่า sol กับ ans มีตัวตรงกันกี่ตัว
```

## แบบที่ 2: for each\_character in a\_string

## ตัวอย่าง: นับจำนวนตัวเลขในสตริง

```
s = input()
digit_counts = 0
for ch in s:
    if "0" <= ch <= "9":
        digit_counts += 1
print(digit_counts)
```

แบบนี้ง่ายกว่า

```
s = input()
digit_counts = 0
for i in range(len(s)):
    if "0" <= s[i] <= "9":
        digit_counts += 1
print(digit_counts)
```

# ตัวอย่าง: สร้างสตริงที่ไม่มีเครื่องหมาย ( [ { } ] )

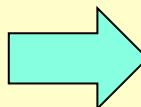
```
s = input()
result = ""
for ch in s:
    if ch not in "([{}])":
        result += ch
print(result)
```

หยิบอุปกรณ์ที่ละตัว  
ถ้าไม่ใช่ ( [ { } ] )  
ก็ต่อเพิ่มให้กับสตริงผลลัพธ์

# แบบฝึกหัด: [ ] กับ ( )

Input

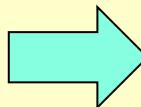
[x + (y - z)]



Output

(x + [y - z])

Programming



Programming

สร้างสตริงใหม่ที่  
แทนวงเล็บ ( ) ด้วย [ ] และ  
แทนวงเล็บ [ ] ด้วย ( )

## แบบที่ 3: for each\_element in a\_list

# ตัวอย่าง: หาค่าเฉลี่ย

```
x = input().split()  
s = 0  
for e in x:  
    s += float(e)  
avg = s/len(x)  
print("Average =", avg)
```

Input

10 20 30 20 10

input().split()

["10", "20", "30", "20", "10"]

Output

Average = 18.0

หยิบมาทีละตัว แปลงเป็น  
float และหาค่าเฉลี่ย

# Tips: ใช้ได้กับสตริงและลิสต์

หยิบทีละตัวจากซ้ายไปขวา

```
for e in x :
```

...

หยิบทีละตัวจากขวามาซ้าย

```
for e in x[::-1] :
```

...

หยิบทีละตัวจากซ้ายไปขวา ไม่เอาตัวสุดท้าย

```
for e in x[:-1] :
```

...

หยิบเฉพาะตัว index คี่จากซ้ายไปขวา

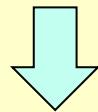
```
for e in x[1::2] :
```

...

# แบบฝึกหัด: นับจำนวน the และ The

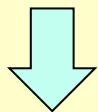
The word "the" is one of the most common words in English.

" ( ) , . '



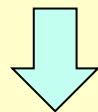
เปลี่ยนเครื่องหมายวรรค  
ตอนด้วยช่องว่าง

The word the is one of the most common words in English



`split()`

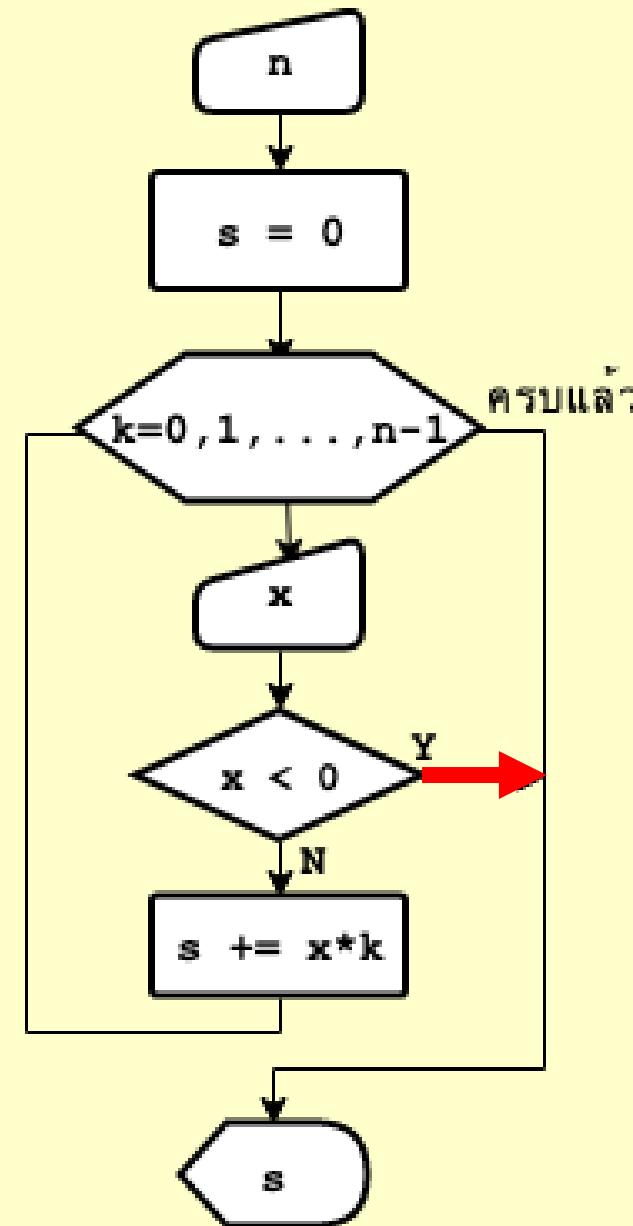
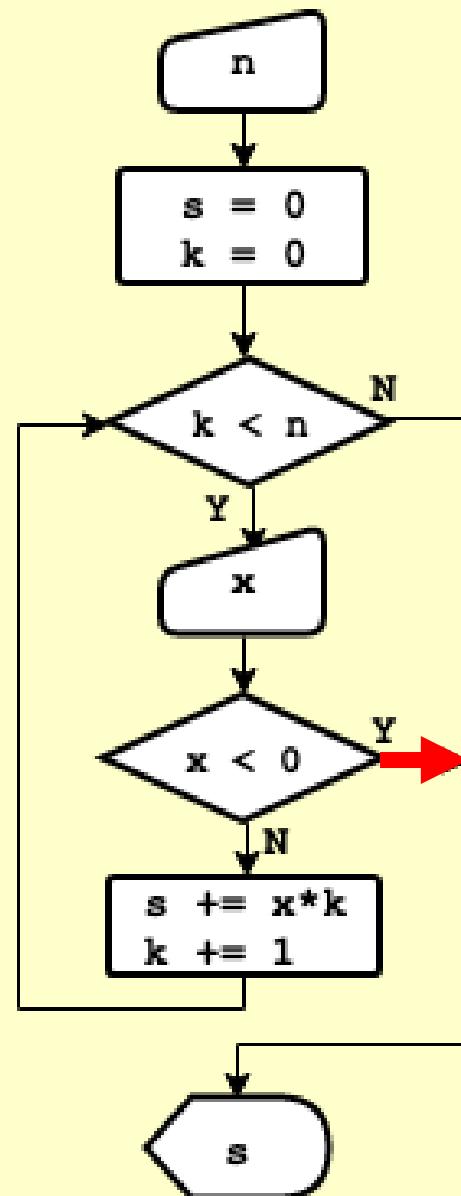
```
["The", "word", "the", "is", "one", "of", "the",
 "most", "common", "words", "in", "English"]
```



นับ the กับ The

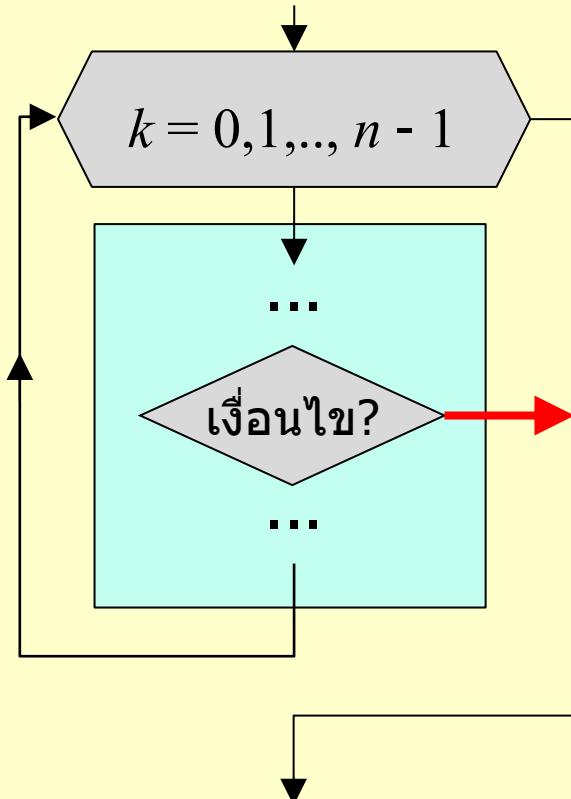
3

# break : คำสั่งเพื่อให้ออกจากวงวน



# break : คำสั่งเพื่อให้ออกจากวน for

```
for k in range(n):  
    ...  
    if เงื่อนไข :  
        break  
    ...
```

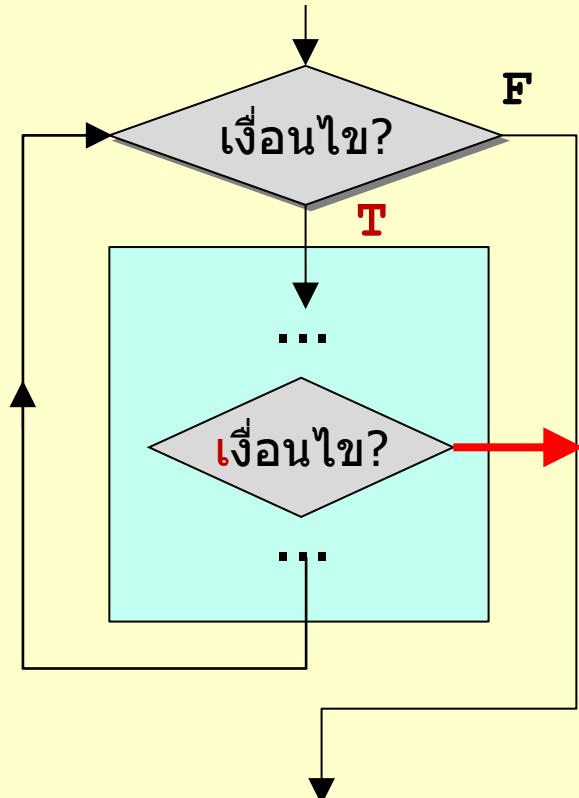


```
n = int(input())  
for k in range(2,n+1):  
    if n%k == 0:  
        break  
    if k == n:  
        print("Prime")  
    else:  
        print("Composite")
```

รับจำนวนเต็ม  $n$   
หา  $k$  ที่หาร  $n$  ลงตัว  $k = 2,3,\dots,n$   
เมื่อออกนอกรอบ  
ถ้า  $k$  เท่ากับ  $n \rightarrow$  เป็นจำนวนเฉพาะ  
ไม่ เช่นนั้น  $\rightarrow$  เป็นจำนวนประกอบ

# break : คำสั่งเพื่อให้ออกจากวงวน while

```
while เงื่อนไข:  
    ...  
    if เงื่อนไข :  
        break  
    ...
```



```
t = input()  
s = 0; n = 0  
while t != "q":  
    s += float(t)  
    n += 1  
    t = input()  
print("Average =", s/n)
```

```
s = 0; n = 0  
while True:  
    t = input()  
    if t == "q":  
        break  
    s += float(t)  
    n += 1  
print("Average =", s/n)
```

10.0  
10.0  
11.0  
12.0  
q

10.0  
10.0  
11.0  
12.0  
q

while True คือวงวนที่หมุนไปเรื่อย ๆ

# แบบฝึกหัด: เกมทายตัวเลข

Guess my number (0 to 99)

You have seven tries

50

**Higher**

75

**Lower**

57

**Higher**

68

**Lower**

62

**Lower**

60

**Lower**

59

**You win**

Guess my number (0 to 99)

You have seven tries

1

**Higher**

2

**Higher**

3

**Higher**

4

**Higher**

5

**Higher**

6

**Higher**

7

**Higher**

**You lose, the number is 9**

# แบบฝึกหัด: เกมทายตัวเลข

```
import random  
  
print("Guess my number (0 to 99)")  
print("You have seven tries")  
n = random.randint(0, 99)    สุ่มเลขระหว่าง 0 ถึง 99  
for k in range(7):  
    m = int(input())    ให้ลองทายอย่างมาก 7 ครั้ง
```

# List Processing

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# List

List คือ รายการของข้อมูล

[ ] ← empty list

[2, 3, 5, 7, 11, 13, 17, 19, 23]

["SU", "MO", "TU", "WE", "TH", "FR", "SA"]

[ ["Ranee", "Campen"],

["Roy Marn", "Plerng Boon",

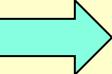
"Buphe Sanniwat", "Krong Kam"] ]

# Basic List Operations

- Length `len([1,2,3])`
- Concatenation `[1] + [2,3] → [1,2,3]`
- Repetition `[0,0] * 3 → [0,0,0,0,0]`
- Indexing `x[k]`
- Slicing `x[start : stop : step]`

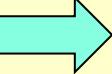
# แบบฝึกหัด: เลขໄດ້ຍ່າງໄປ

0110-02234-9877-0112



5, 6

9 8 7 6 5 4 3 2 1 0



None

```
d = input()
counts = [0] * 10
for c in d:
    if "0" <= c <= "9":
        counts[int(c)] += 1
missing = ""

if missing == "":
    print("None")
else:
    print(missing)
```

ต้องการให้ counts[k]  
เก็บจำนวนที่มีเลข k ปรากฏใน input

หยิบทีละตัวใน input  
ถ้าเป็นเลข ก็ให้เพิ่ม  
จำนวนที่พบรอ 1

นำเลขที่มี counts ของ  
เลขนั้นเป็น 0 มาต่อให้  
สตริง missing

# Basic List Methods

`append, remove, insert, pop, index, sort`

```
d = []
for i in range(5):
    d.append(10*i)      # [0,10,20,30,40] ต่อท้ายลิสต์
d.remove(20)          # [0,10,30,40]
d.insert(2,99)         # [0,10,99,30,40]
d.insert(-1,7)         # [0,10,99,30,7,40]
x = d.pop(1)           # [0,99,30,7,40], x = 10
x = d.pop(-3)          # [0,99,7,40], x = 30
j = d.index(99)        # j = 1
d.sort()               # [0,7,40,99]
b = 40 in d            # b = True
```

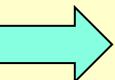
`d.remove(e), d.index(e)` ถ้าหา e ไม่พบใน d จะเกิดข้อผิดพลาด

# ตัวอย่าง: หาเลขเดือนจากชื่อเดือน

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
t = input()
if t in months:
    k = months.index(t)
    th = "th"
    if k == 0: th = "st"
    if k == 1: th = "nd"
    print(t, "is the", str(k+1)+th+" month.")
else:
    print("Invalid month name")
```

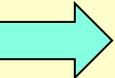
ก่อนใช้ index ต้อง  
มั่นใจว่ามีข้อมูลที่จะค้น

Sep



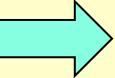
Sep is the 9th month.

Jan



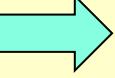
Jan is the 1st month.

Feb



Feb is the 2nd month.

Ok



Invalid month name

# แบบฝึกหัด: ชื่อเล่นอะไร ชื่อจริงอะไร

เขียนโปรแกรม รับชื่อจริงแสดงชื่อเล่น  
รับชื่อเล่นแสดงชื่อจริง โดยไม่ต้องใช้คำสั่ง if

Robert      ⇔    Dick

William     ⇔    Bill

James        ⇔    Jim

John        ⇔    Jack

Margaret    ⇔    Peggy

Edward     ⇔    Ed

Sarah        ⇔    Sally

Andrew     ⇔ [Andy](http://www.kyoto-su.ac.jp/~trobb/nicklist.html)

# รูปแบบการรับข้อมูลเก็บในลิสต์

- ระบุจำนวน ตามด้วยข้อมูลบรรทัดละตัว

4  
10.2  
11.5  
20.0  
22.5

- รับข้อมูลบรรทัดละตัว มีค่าระบุว่าหมด

10.2  
11.5  
20.0  
22.5  
-1

- รับรายการข้อมูลในบรรทัดเดียว

10.2 11.5 20.0 22.5

# แบบที่ 1: ระบุจำนวน ตามด้วยข้อมูลบรรทัดละตัว

input

```
4  
10.2  
11.5  
20.0  
22.5
```

## แบบที่ 2: รับข้อมูลบรรทัดละตัว มีค่าระบุว่าหมด

**input**

```
10.2  
11.5  
20.0  
22.5  
-1
```

## แบบที่ 3: รับรายการข้อมูลในบรรทัดเดียว

**input**

```
10.2 11.5 20.0
```

# ข้อแนะนำ: การต่อท้ายลิสต์

```
data.append( x )
```



เร็ว

```
data = data + [ x ]
```



ช้า

```
data += [ x ]
```



# แบบฝึกหัด: เพิ่มหลัง-เพิ่มน้า

input

เพิ่มหลัง	4	← เพิ่มข้อมูลทุกตัวในบรรทัดนี้
หน้า	1	
หลัง	2	
หน้า	3	
หลัง	4	
หน้า หลัง หน้า หลัง หน้า หลัง	11 12 13 14 15	
หน้า	21	
หลัง	22	
หน้า	23	
หลัง	24	
หน้า	25	
	-1	

output

```
[25, 23, 21, 14, 12, 4, 2, 1, 3, 11, 13, 15, 22, 24]
```

# รูปแบบการหยิบข้อมูลที่ละตัวในลิสต์มาประมวลผล

```
for e in d:
```

```
    ...
```



หยิบข้อมูลที่ละตัว  
ในลิสต์มาใช้งาน

```
for i in range(len(d)):  
    e = d[i] * i
```

```
    ...
```



ถ้าต้องการทั้ง  
ตำแหน่งและข้อมูล

```
for e in d:  
    i = d.index(e)
```

```
    ...
```



อย่าเขียนแบบนี้  
ช้าเปล่า ๆ  
ใช้แบบที่ 2

## ตัวอย่าง: นับจำนวนนิสิตวิศวฯ ในลิสต์

```
student_ids = input().split()  
# ["6131022921", "6240123026", "6130021221", ...]  
c = 0  
for sid in student_ids:  
    if sid[-2:] == "21":  
        c += 1  
print(c)
```

```
student_ids = input().split()  
c = 0  
for i in range(len(student_ids)):  
    if student_ids[i][-2:] == "21":  
        c += 1  
print(c)
```

# ถ้าต้องการเปลี่ยนข้อมูลบางตัวในลิสต์

```
...  
# เปลี่ยนจำนวนติดลบทั้งหมดในลิสต์ d ให้เป็นบวก  
for e in d:  
    if e < 0:  
    e *= -1      # ค่าใน e เปลี่ยน แต่ไม่เปลี่ยนค่าใน d
```

```
...  
# เปลี่ยนจำนวนติดลบทั้งหมดในลิสต์ d ให้เป็นบวก  
for i in range(len(d)):  
    if d[i] < 0:  
    d[i] *= -1 # เปลี่ยนค่าใน d ตามที่ต้องการ
```

# รูปแบบการหยັບຂໍ້ມູນລີ້ທີ່ຕິດກຳນິໃລສຕໍ່ມາປະໜາລົບ

```
for i in range(len(d)-1) :  
    left = d[i]  
    right = d[i+1]  
    ...
```

```
for i in range(1,len(d)) :  
    left = d[i-1]  
    right = d[i]  
    ...
```

# ตัวอย่าง: ลำดับเพิ่มขึ้นหรือไม่

Input

4 6 9 10 14 34 45 99

Output

True

2 3 4 5 5 5 5 5 99

False

```
d = input().split()
is_increasing = True
for i in range(len(d)-1):
    if int(d[i]) >= int(d[i+1]):
        is_increasing = False
        break
print(is_increasing)
```

# แบบฝึกหัด: นับจำนวน "ยอด" (น้อยมาก น้อย)

Input	Output
1 2 3 4 8 2 4 5 9 10 8 9 10 11 15 18 12 20	3
1 2 3 4 5 6 10 11 12 13 45 79 80 81 89 90	0

```
x = input().split()
d = []
for e in x:
    d.append(int(e))
count = 0

print(count)
```

# แบบฝึกหัด: ชุดข้อมูลมีแตกต่างกันกี่ตัว

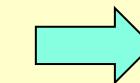
Input

1 2 3 1 2 3 2 3 3 1 1 1 2

Output

อ่านมาเก็บในลิสต์ และ sort

[ 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3 ]



3

```
x = input().split()
d = []
for e in x:
    d.append(int(e))
d.sort()
count = 1
```

วงวนเพิ่ม count ทุกครั้ง ที่พบตัวติดกันต่างกัน

```
print(count)
```

# รูปแบบการค้นข้อมูลในลิสต์

```
[ 'KTB' , 18.9 , 'BBL' , 176.5 , 'SCB' , 132.0 , 'KBANK' , 172.0 ]
```

```
[ 'KTB' , 'BBL' , 'SCB' , 'KBANK' ]
```

```
[ 18.9 , 176.5 , 132.0 , 172.0 ]
```

```
[ [ 'KTB' , 18.9 ] , [ 'BBL' , 176.5 ] , [ 'SCB' , 132.0 ] , [ 'KBANK' , 172.0 ] ]
```

# รูปแบบการค้นข้อมูลในลิสต์: กรณีลิสต์แบบง่าย

กรณีเป็นลิสต์ที่เก็บข้อมูลพื้นฐาน (ไม่ใช่ลิสต์ซ้อนลิสต์)

```
# x เป็นลิสต์ อยากรู้ว่า c ใน x ไหม ?
# x = [3,1,4,5,2,4,7], c = 5
if c in x:
    ...
    ...
```

```
# x เป็นลิสต์ อยากรู้ว่า c เก็บใน index ใดใน x
if c in x:
    j = x.index(c)
```

# รูปแบบการค้นข้อมูลในลิสต์: กรณีเก็บหลายลิสต์

เช่น มีลิสต์ sid เก็บเลขประจำตัว และลิสต์ gpa เก็บเกรด  
เกรดของ sid[k] เก็บอยู่ที่ gpa[k]

```
sid = [6131001021, 6130020221, ...]  
gpa = [3.8, 3.7, ...]
```

ต้องการค้น ID ที่สนใจว่า ได้เกรดเท่าไร ?

1

```
if ID in sid:  
    j = sid.index(ID)  
    print("ID =", ID, gpa[j])  
else:  
    print("Not found")
```

# รูปแบบการค้นข้อมูลในลิสต์: การณีลิสต์ช้อนลิสต์

เข่น ลิสต์ที่เก็บ [ID, gpa]

data = [[6131001021, 3.8], [6130020221, 3.7], ...]

ต้องการค้น ID ที่สนใจว่า ได้เกรดเท่าไร ?

2

```
gpa = -1
for e in data : # e เป็นลิสต์ที่มี 2 ช่อง
    if e[0] == ID:
        gpa = e[1]
        break
if gpa == -1:
    print("Not found")
else:
    print("ID =", ID, gpa)
```

# รูปแบบการค้นข้อมูลในลิสต์: กรณีลิสต์ช้อนลิสต์

เช่น ลิสต์ที่เก็บ [ID, gpa]

data = [[6131001021, 3.8], [6130020221, 3.7], ...]

ต้องการค้น ID ที่สนใจว่า ได้เกรดเท่าไร ?

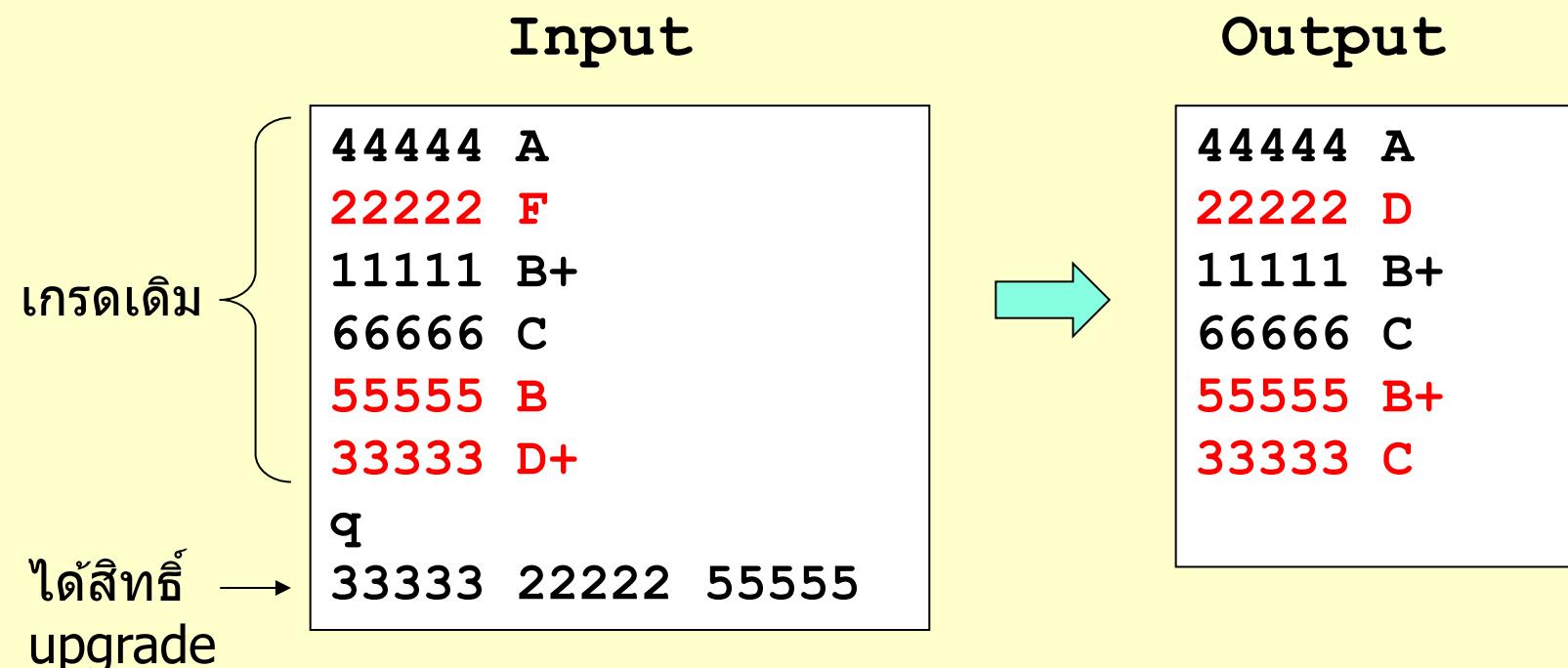
3

```
gpa = -1
for [sid,g] in data :
    if sid == ID:
        gpa = g
        break
if gpa == -1:
    print("Not found")
else:
    print("ID =", ID, gpa)
```

easyกว่าแบบก่อนนี้

# แบบฝึกหัด: upgrade

เขียนโปรแกรมปรับเกรดเพิ่มให้นักเรียนที่ได้รับสิทธิ์คนละหนึ่งประจุ



# การเรียงลำดับข้อมูล

[90, 22, 44, 70, 20, 51, 12, 39, 67, 56, 88, 31, 72, 77, 53]

**d.sort()**

[12, 20, 22, 31, 39, 44, 51, 53, 56, 67, 70, 72, 77, 88, 90]

# การเรียงลำดับข้อมูล

[10, 10, 11, 12, 11, 13, 11, 11, 12, 10, 13, 10, 10, 13, 13]

**d.sort()**

[10, 10, 10, 10, 10, 11, 11, 11, 11, 12, 12, 13, 13, 13, 13]

## การเรียงลำดับข้อมูล: กรณีง่าย

```
x = [3,1,4,5,2,4,7]
x.sort() # x = [1,2,3,4,4,5,7]
```

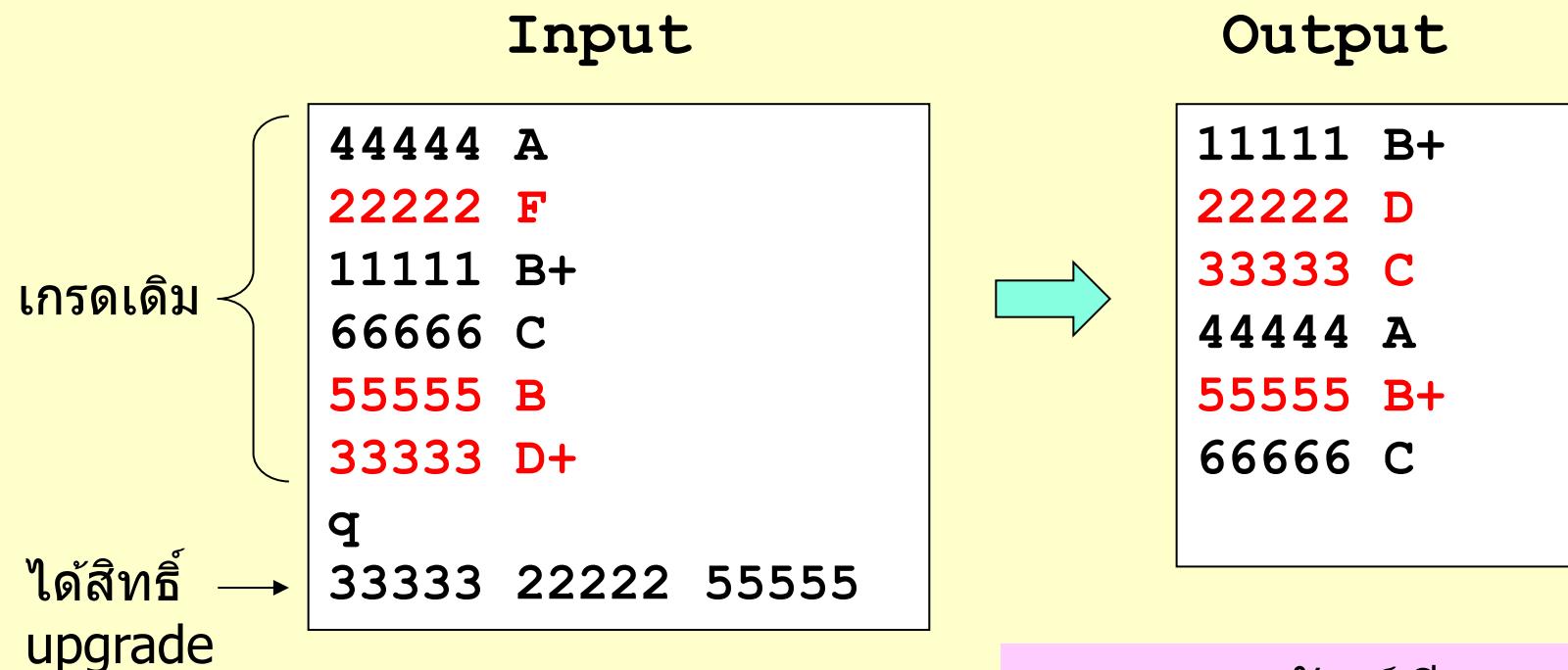
```
x = ["Tom", "Ann", "Don"]
x.sort() # x = ["Ann", "Don", "Tom"]
```

```
x = [[{"A",9}, {"C",1}, {"A",1}]]
x.sort() # x = [{"A",1}, {"A",9}, {"C",1}]
```

เรียงตามตัวแรกก่อน ถ้าตัวแรกเท่ากัน ให้เรียงตามตัวที่สอง

# แบบฝึกหัด: upgrade

เขียนโปรแกรมปรับเกรดเพิ่มให้นักเรียนที่ได้รับสิทธิ์คนละหนึ่งประจุ



แสดงผลลัพธ์เรียงตามเลข  
ประจำตัวจากน้อยไปมาก

# การเรียงลำดับข้อมูล: กรณีซับซ้อน

เรียงตามตัวแรกก่อน ถ้าตัวแรกเท่ากัน ให้เรียงตามตัวที่สอง

```
x = [[ "A", 9], [ "C", 1], [ "A", 1]]  
x.sort() # x = [[ "A", 1], [ "A", 9], [ "C", 1]]
```

เรียงตามตัวที่สองก่อน ถ้าเท่ากัน ให้เรียงตามตัวแรก

```
x = [[ "A", 9], [ "C", 1], [ "A", 1]]  
t = []  
for [a1,a2] in x:  
    t.append([a2,a1])         สร้างลิสต์ใหม่ เก็บสลับตำแหน่ง  
# t = [[9,"A"], [1,"C"], [1,"A"]]  
t.sort()  
# t = [[1,"A"], [1,"C"], [9,"A"]]  
for i in range(len(t)):   สลับตำแหน่งกลับ ไปเก็บในลิสต์เดิม  
    x[i][0],x[i][1] = t[i][1],t[i][0]  
# x = [[ "A", 1], [ "C", 1], [ "A", 9]]
```

# ตัวอย่าง: เรียงลำดับสตริงตามความยาวสตริง

[ "ABC" , "ABCD" , "XYZ" , "OK" ]

เรียงแล้วได้

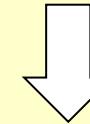
[ "OK" , "ABC" , "XYZ" , "ABCD" ]

```
x = ["ABC", "ABCD", "XYZ", "OK"]
t = []
for e in x:
    t.append([len(e),e])     สร้างลิสต์ใหม่ เก็บความยาวด้วย
# t = [[3,"ABC"], [4,"ABCD"], [3,"XYZ"], [2,"OK"]]
t.sort()
# t = [[2,"OK"], [3,"ABC"], [3,"XYZ"], [4,"ABCD"]]
for i in range(len(t)):
    x[i] = t[i][1]
# x = ["OK", "ABC", "XYZ", "ABCD"]
```

# แบบฝึกหัด: จุดได้แกล์จุดกำเนิดเป็นอันดับสาม

```
n = int(input())
points = []
for k in range(n):    อ่านพิกัดต่าง ๆ เก็บใส่ลิสต์
    d = input().split()
    x = float(d[0])
    y = float(d[1])
```

5
2.0 2.0
2.0 3.0
2.0 2.0
2.0 5.0
-1 -5.0

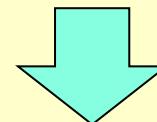


#2: (2.0, 3.0)

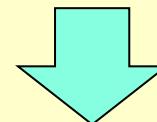
กำหนดให้พิกัดที่ได้รับทั้งหมดมีระยะถึงจุดกำเนิดไม่เท่ากันเลย

# List $\leftrightarrow$ String ด้วย split & join

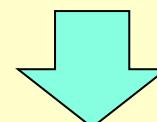
```
"ABC      DEF  GHI    JK".split()
```



```
[ "ABC", "DEF", "GHI", "JK" ]
```



```
", ".join( [ "ABC", "DEF", "GHI", "JK" ] )
```



```
"ABC, DEF, GHI, JK"
```

# split แยกสตริงออกเป็นลิสต์

- **x = s.split()**
  - แยกสตริง s ออกเป็นส่วน ๆ ได้เป็นลิสต์ (แยกด้วยช่องว่าง)
  - s = "11 2 33"
  - s.split() ได้ ["11", "2", "33"]
- **x = s.split( sep )**
  - แยกสตริง s ออกเป็นส่วน ๆ ได้เป็นลิสต์ ใช้ sep เป็นตัวแยก
  - s = "11:2: 33"
  - s.split(":") ได้ ["11", "2", " 33"]

"a,,,b".split(",")	ได้ ["a", "", "", "", "b"]
"a b".split(" ")	ได้ ["a", "", "", "b"]
"a b".split()	ได้ ["a", "b"]
"a,,,b".split(",")	ได้ ["a", ",b"]

# join นำสตริงในลิสต์มาต่อกัน

- `s = sep.join( x )`
  - `sep` เป็น string
  - `x` เป็น list of strings
  - `sep.join(x)` ได้ผลเป็น string ที่สร้างจากการนำ string แต่ละตัวใน `x` มาต่อกัน โดยมี `sep` เป็นตัวคั่น

```
x = ["A", "BC", "DEF", "GH"]
s1 = "".join(x) # "A|BC|DEF|GH"
s2 = ",".join(x) # "A,BC,DEF,GH"
s3 = ">".join(x) # "A><BC><DEF><GH"
```

## ตัวอย่าง:

```
x = [-1, 0, 3, 300, -2, 39, 50, 12, 400, -100]
# เลือกและแสดงข้อมูลใน x เฉพาะที่มีค่า 0 ถึง 255
y = []
for e in x:
    if 0 <= e <= 255:
        y.append( str(e) )
# y = ["0", "3", "39", "50", "12"]
print( " -> ".join(t) )
```

```
0 -> 3 -> 39 -> 50 -> 12
```

# แบบฝึกหัด: Collatz Problem

```
n = int(input())
while n != 1:
    if n%2 == 0:
        n /= 2
    else:
        n = 3*n + 1
```

อยากรู้ว่า n เปลี่ยนแปลง  
อย่างไรจนกลายเป็น 1

input

10

output

10->5->16->8->4->2->1



## ข้อควรระวัง: การแก้ไขลิสต์ภายในวงวน

```
x = [1,1,2,3,1]
for e in x:
    if e == 1:
        x.remove(e)
print(x)
```

รอบที่ 1	x = [1,1,2,3,1]	ได้ e = 1, x.remove(1)
รอบที่ 2	x = [1,2,3,1]	ได้ e = 2
รอบที่ 3	x = [1,2,3,1]	ได้ e = 3
รอบที่ 4	x = [1,2,3,1]	ได้ e = 1, x.remove(1)
	x = [2,3,1]	ผิด

# ข้อควรระวัง: การแก้ไขลิสต์ภายในวงวน

```
x = [1,1,2,3,1]
for i in range(len(x)):
    if x[i] == 1:
        x.pop(i)
```

5

```
x = [1,1,2,3,1]
for i in [0,1,2,3,4]:
    if x[i] == 1:
        x.pop(i)
```

i = 0, x = [1,1,2,3,1]	x.pop(0)
i = 1, x = [1,2,3,1]	
i = 2, x = [1,2,3,1]	
i = 3, x = [1,2,3,1]	x.pop(3)
i = 4, x = [1,2,3]	เงื่อนไขมี x[4]

# ข้อควรระวัง: การแก้ไขลิสต์ภายในวงวน

```
x = [1,1,2,3,1]
i = 0
while i < len(x):
    if x[i] == 1:
        x.pop(i)
    i += 1
print(x)
```

```
x = [1,1,2,3,1]
i = 0
while i < len(x):
    if x[i] == 1:
        x.pop(i)
    else:
        i += 1
print(x)
```

ถูก

i = 0, x = [1,1,2,3,1]	x.pop(0)
i = 1, x = [1,2,3,1]	
i = 2, x = [1,2,3,1]	
i = 3, x = [1,2,3,1]	x.pop(3)
x = [1,2,3]	ผิด

# ข้อควรระวัง: การแก้ไขลิสต์ภายในวงวน

```
x = [1,3]
for e in x:
    if e == 1:
        x.insert(0,9)
```

รอบที่ 1 x = [1,3]	ได้ e = 1, x.insert(0,9)
รอบที่ 2 x = [9,1,3]	ได้ e = 1, x.insert(0,9)
รอบที่ 3 x = [9,9,1,3]	ได้ e = 1, x.insert(0,9)
รอบที่ 4 x = [9,9,9,1,3]	ได้ e = 1, x.insert(0,9)
... ทำงานไม่สิ้นสุด	

สรุป: ออย่าแก้ไขลิสต์ภายในวงวนเลย

# อยากรักษาลิสต์ภายนอก → สร้างลิสต์ใหม่

```
x = [1,1,2,3,1]
for e in x:
    if e == 1:
        x.remove(e)
print(x)
```

```
x = [1,1,2,3,1]
y = []
for e in x:
    if e != 1:
        y.append(e)
x[:] = y
```

`x[:]` คือที่เก็บในลิสต์ `x` ตั้งแต่ช้ายถึงขวาสุด

`x[:] = y` คือ copy ข้อมูลใน `y` ไปเก็บในลิสต์ `x` (`x` และ `y` เป็นคนละลิสต์)

`x = y` ทำให้ตัวแปร `x` ไปเป็นตัวเดียวกับลิสต์ `y`

```
x = [1,3]
for e in x:
    if e == 1:
        x.insert(0,9)
```

```
x = [1,3]
y[:] = x
for e in x:
    if e == 1:
        y.insert(0,9)
```

# การเขียนฟังก์ชัน

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# ฟังก์ชัน

- เคยใช้ built-in function มาหลายฟังก์ชันแล้ว
  - print("Hello"), input(), len(x), round(2/3, 2), abs(-2), ...
- เขียนฟังก์ชันเพื่อแบ่งหน้าที่การทำงานเป็นส่วน ๆ
- จะใช้ฟังก์ชันใด ต้องรู้
  - ชื่อฟังก์ชัน, ข้อมูลที่ต้องส่งให้ฟังก์ชัน, ผลที่ได้จากฟังก์ชัน
- ตัวอย่าง: `print( x )`
  - x เป็นข้อมูลแบบใด ๆ ก็ได้
  - ไม่มีผลที่คืนกลับจากฟังก์ชัน (เราจึงไม่เขียน `a = print(3/5)`)
- ตัวอย่าง: `round(x,d)`
  - x เป็นจำนวน, d เป็นจำนวนหลักหลังจุดทศนิยม
  - ผลที่ได้คือจำนวนจริง x ที่มีจำนวนหลังจุดตามที่กำหนด เช่น `round(2/3,2)` ได้ 0.67 จึงมักเขียน `a = round(2/3, 2)`

หรือ `print( round(2/3, 2) )`

# ฟังก์ชันแรก: hello

```
def hello(name):  
    print("Hello", name)  
  
BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]  
for e in BLACKPINK:  
    hello( e )
```

```
Hello Jisoo  
Hello Jennie  
Hello Rosé  
Hello Lisa
```

ค่าของ **e** ถูกส่งไปให้ตัวแปร **name**  
ของฟังก์ชัน **hello**  
แล้วก็ย้ายการทำงานไปที่ **hello**  
ทำคำสั่งในฟังก์ชันเสร็จ ก็กลับมาทำต่อ

# ฟังก์ชันหนึ่งเรียกอีกฟังก์ชันก็ได้

```
def hello(name):
    print("Hello", name)

def hello_all(names):
    for e in names:
        hello(e)

BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]
hello_all( BLACKPINK )
hello_all( ["Joe", "John", "Kong"] )
```

Hello Jisoo  
Hello Jennie  
Hello Rosé  
Hello Lisa  
Hello Joe  
Hello John  
Hello Kong

# แบบฝึกหัด: แสดงอะไร ?

```
def print1(a):
    for e in a:
        print(e)

def print_all(A):
    for e in A:
        print1(e)

print_all( ["A", ["AB", "C"] ] )
print_all( ["ABC"] )
```

# แบบฝึกหัด: เครื่องบวกเลขฐานสอง

**bin(x):**

Return a binary string prefixed with "0b" constructed from an integer **x**

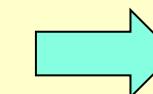
**int(x, base):**

Return an integer constructed from a string **x** in the specified **base** (default is 10).

จงเขียนโปรแกรมรับจำนวนเต็มในรูปฐานสอง 2 จำนวน เพื่อหาผลบวกและแสดงในรูปฐานสอง โดยใช้ built-in functions **bin** กับ **int** ข้างบนนี้

**Input**

100101 11001



**Output**

111110

# องค์ประกอบของฟังก์ชัน

ชื่อฟังก์ชัน

รายการของพารามิเตอร์

```
def dot( u, v ) :  
    p = 0  
    for i in range(len(u)) :  
        p += u[i]*v[i]  
    return p
```

การเรียกใช้ฟังก์ชัน

```
a = [1,2,3]  
b = [1,3,6]  
d1 = dot(a, b)  
d2 = dot(a, [1,3,2])  
print( 2*dot([1,2,3],b) )
```

จากการทำงานของฟังก์ชัน กลับไปทำงานต่อที่ผู้เรียก  
ฟังก์ชัน และยังคืนผลจากฟังก์ชันให้ผู้เรียกได้ด้วย

ต้องเย็บคำสั่งทั้งหลายในฟังก์ชันไปทางขวาให้ตรงกัน

# ตัวแปรภายในฟังก์ชันได้เป็นของฟังก์ชันนั้น

- พารามิเตอร์และตัวแปรในฟังก์ชันได
  - ชื่อซ้ำกับของฟังก์ชันอื่นได
  - เรียกใช้ตัวแปร ที่อยู่ในฟังก์ชันอื่นไม่ได

```
def read_data():
    d = []
    for i in range(int(input())):
        d.append(int(input()))
    return d
#-----
def get_mean(d):
    return sum(d)/len(d)
#-----
def get_median(d):
    d.sort()
    n = len(d)
    return (d[(n-1)//2] + d[n//2])/2
```

# พารามิเตอร์รับข้อมูลจากผู้เรียกฟังก์ชัน

```
def distance(x1, y1, x2, y2):  
    dx = x1 - x2  
    dy = y1 - y2  
    d = (dx**2 + dy**2)**0.5  
    return d
```

- คำสั่งในฟังก์ชันเริ่มทำงาน เมื่อมีคำสั่งอื่นเรียกใช้ (ไม่เรียกไม่ทำ)
- คำสั่งที่เรียก ต้องส่งข้อมูลให้กับพารามิเตอร์ของฟังก์ชัน
  - d1 = distance(10.5, 12.0, 30.0, 50.5)
  - x = 10.5; y = 11.2
  - d2 = distance(x, y, 30.0, 50.5)
- เมื่อคำสั่งในฟังก์ชันเริ่มทำงาน ถือได้เลยว่าพารามิเตอร์มีค่าแล้ว

```
def distance(x1, y1, x2, y2):  
    x1 = float(input()) # แปลก !!!
```

# return: คืนผลลัพธ์ค่าก็ได้

```
# roots of ax**2 + b*x + c = 0
def roots(a, b, c):
    t = (b**2 - 4*a*c)**0.5
    r1 = (-b + t) / (2*a)
    r2 = (-b - t) / (2*a)
    return [r1, r2]
```



```
x1, x2 = roots(6, -6, -36)
print(x1, x2)
```

```
def get_odds( x ):
    odds = []
    for e in x:
        if e%2 == 1:
            odds.append(e)
    return odds
```

หรือใช้การคืนลิสต์ก็ได้

# return : คืนการทำงานกลับสู่ผู้เรียก

- ใช้คำสั่ง **return** เฉย ๆ
- หรือไม่ก็ เมื่อทำงานถึงคำสั่งล่างสุดของฟังก์ชัน ก็คือการคืนการทำงาน

```
def hello(name):  
    print("Hello", name)  
    return
```

```
def hello(name):  
    print("Hello", name)
```

```
def hello(name):  
    if name == "":  
        return  
    print("Hello", name)
```

## ไม่คืนผลทาง return แต่แก้ไขลิสต์ที่ผู้เรียกส่งมา

```
def abs_all( data ):
    d = []
    for e in data:
        d.append(abs(e))
    return d

x = [1,-2,3,-4,-5]
x = abs_all(x)
print( x )
```

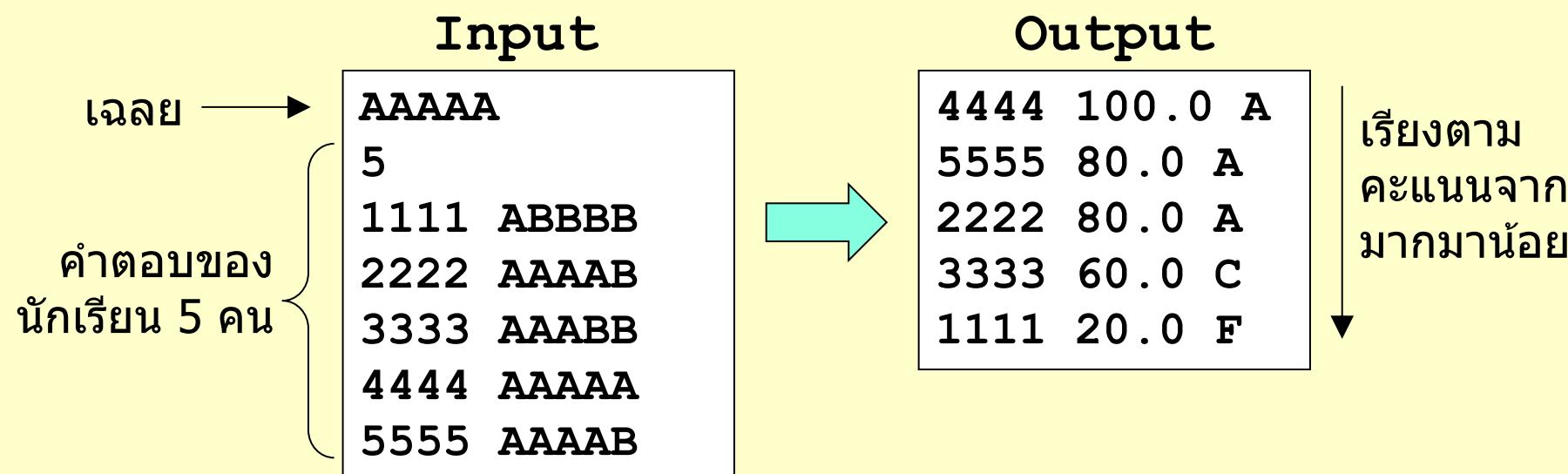
```
def abs_all( data ):
    for k in range(len(data)):
        data[k] = abs(data[k])

x = [1,-2,3,-4,-5]
abs_all(x) # ข้อมูลในลิสต์ x เปลี่ยน
print( x )
```

# แบบฝึกหัด: ตรวจ + ให้เกรด + เรียง + แสดงผล

เขียนโปรแกรมอ่านเฉลยและคำตอบของนักเรียน (หลายคน)  
จากนั้นตรวจให้คะแนนคำตอบและให้เกรดทุกคน และก็นำผลที่ได้  
ไปเรียงลำดับ ปิดท้ายด้วยการแสดงผลลัพธ์ (ดูตัวอย่าง)

มีฟังก์ชันเขียนมาให้แล้วจำนวนหนึ่ง ให้เขียนโปรแกรมที่เรียกใช้  
ฟังก์ชันเหล่านี้ให้ทำงานตามที่ต้องการ (ตัวโปรแกรมที่ต้องเขียน  
มีคำสั่งไม่น่าเกิน 6 คำสั่ง)



# แบบฝึกหัด: ทำความเข้าใจฟังก์ชันเหล่านี้

```
def read_answers():
    N = int(input())
    answers = []
    for k in range(N):
        sid, ans = input().split()
        answers.append([sid, ans])
    return answers

def marking(answer, solution):
    p = 0
    for i in range(len(answer)):
        if answer[i] == solution[i]:
            p += 1
    return p

def grading(score):
    g = [[80,"A"], [70,"B"], [60,"C"], [50,"D"]]
    for a,b in g:
        if score >= a:
            return b
    return "F"
```

# แบบฝึกหัด: ทำความเข้าใจฟังก์ชันเหล่านี้

```
def scoring(answers, solution):
    scores = []
    for [sid, ans] in answers:
        score = marking(ans, solution) / \
                len(solution) * 100
        grade = grading(score)
        scores.append([sid, score, grade])
    return scores

def report(scores):
    for [sid, score, grade] in scores:
        print(sid, grade)

def sort(scores):
    x = []
    for [sid, score, grade] in scores:
        x.append([score, sid, grade])
    x.sort()
    for i in range(len(x)):
        scores[i] = [x[i][1], x[i][0], x[i][2]]
```

# ข้อควรระวัง: อย่าแก้ไขค่าของพารามิเตอร์ด้วย =

```
def f1(x):
    # ตัวแปร x เปลี่ยนค่า แต่ตัวแปรที่ส่งค่ามาไม่เปลี่ยน
    x = 0

def f2(x):
    # ตัวแปร x เปลี่ยนค่า แต่ตัวแปรที่ส่งค่ามาไม่เปลี่ยน
    x = [0]*len(x)

def f3(x):
    for i in range(len(x)):
        x[i] = 0 # ตัวแปรที่ส่งค่ามากับเปลี่ยนด้วย

a = 9
f1(a)          # a เหมือนเดิม
b = [1,2,3,4]
f2(b)          # b เหมือนเดิม
f3(b)          # b เปลี่ยน
```

# ความผิดพลาดที่พบบ่อย

```
def hello(name):  
    print("Hello", name)
```

```
def double(u):  
    return 2*u
```

```
x = hello("Bell")      # ไม่มีผลคืนมา แต่มีตัวรับผล  
double( [1,2,3] )      # คืนผลแต่ไม่มีตัวรับผล
```

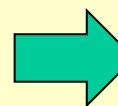
แบบนี้ x มีค่าเป็น None  
(None เป็นค่าพิเศษค่าหนึ่งใน Python)

```
def clip(x):  
    if x < 0:  
        return 0  
    x += 2          # มีคำสั่งอยู่ตามหลัง return  
    return x
```

```
def is_odd(x):         # บางกรณีคืนค่า บางกรณีไม่คืน  
    if x%2 == 1: return True
```

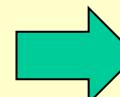
# Tips

```
def is_odd(n):  
    if n%2 == 1:  
        return True  
    else:  
        return False
```



```
def is_odd(n):  
    return n%2 == 1
```

```
def foo(n):  
    if n%2 == 1:  
        return 3*n+1  
    else:  
        return n//2
```



```
def foo(n):  
    if n%2 == 1:  
        return 3*n+1  
    return n//2
```

## แบบฝึกหัด: ระยะระหว่างจุดสองจุด

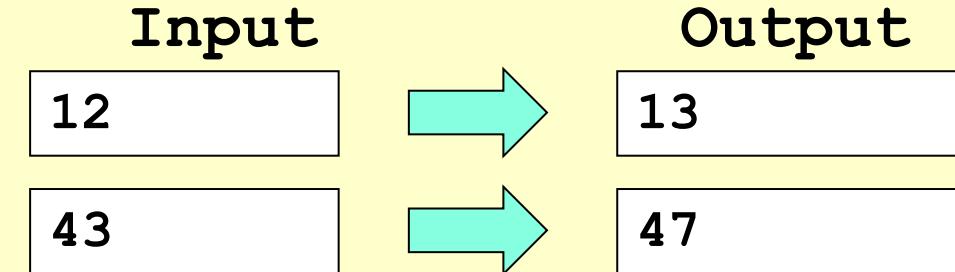
```
def distance(x1, y1, x2, y2):  
    # คำนวณระยะห่างระหว่างจุด (x1,y1) กับ (x2,y2)
```

```
d = distance(1, 1, 4, 5)  
print(d)
```

# แบบฝึกหัด: แสดงอะไร ?

```
def f(x) :  
    return 2*x  
  
def g(x) :  
    return x+5  
  
def h(x) :  
    return x//2  
  
x = 4  
y = f(g(h(x)))  
print(x, "-->", y)
```

# แบบฝึกหัด: หาจำนวนเฉพาะตัวแรกที่มีค่ามากกว่า N



```
def is_prime(n) :  
    if n <= 1:  
        return False  
    for k in range(2,n) :  
        if n%k == 0:  
            return False  
    return True  
def next_prime(N) :
```

# String and File Processing

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

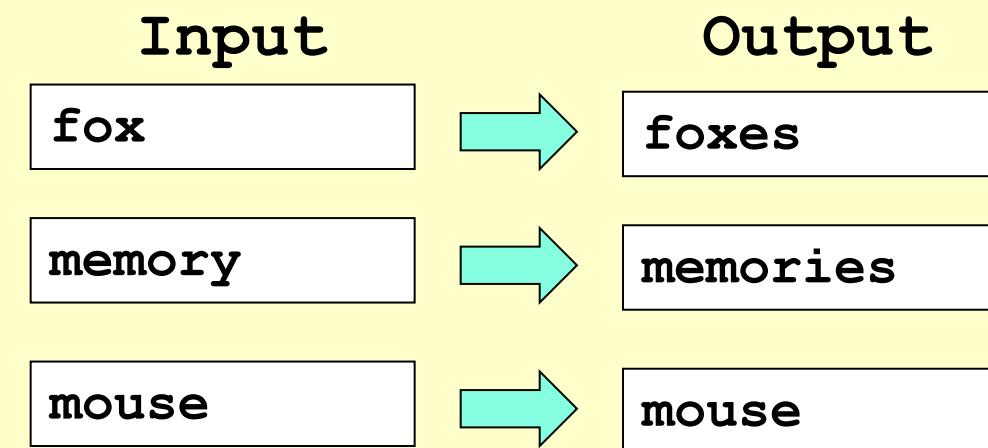
# ทบทวนเรื่องสตริง

```
s = "I'm a string"
t = 'I said "This is a string".'
chars = input()
c = 0
for ch in chars:
    if ch in s:
        c += 1
for i in range(len(t)):
    if t[i] in s:
        c += 1
r = ""
for k in range(2,10,2):
    r += str(k)      # 2468
r = 2*r              # 24682468
```

# แบบฝึกหัด: เอกพจน์ → พหุพจน์

## แบบง่าย ๆ

- ลงท้าย s, x, หรือ ch → เติม es ต่อท้าย
- ลงท้าย y แต่ตัวก่อน y ไม่ใช่สระ → เปลี่ยน y เป็น ies
- ถ้าไม่ตรงกับกฎสองข้อข้างบนนี้ → เติม s ต่อท้าย



# Espresso, Macchiato, Ristretto, Americano, Latte, Cappuccino, Mocha, Affogato

```
x = input().split(", ")
out = []
k = 0
for e in x:
    out.append((e + "*12)[:12])
    k += 1
    if k % 3 == 0:
        print("".join(out))
        out = []
print("".join(out))
```

Affogato  
Espresso  
Mocha

Americano  
Latte  
Ristretto

Cappuccino  
Macchiato



```
import urllib.request

def find(s, start, c):
    for i in range(start, len(s)):
        if s[i] == c: return i
    return -1

url = "http://air4thai.pcd.go.th/services/getNewAQI_XML.php?stationID=52t"
web = urllib.request.urlopen(url)
for line in web:
    line = line.decode()
    if "<PM25 value=" in line:
        i = find(line, 0, '"')
        j = find(line, i+1, '"')
        print("PM 2.5 =", line[i:j])
        break
```

# อักขระพิเศษ (Escape Characters)

```
s = "Hello"  
print(s)                      # Hello  
s = "\"Hello\""  
print(s)                      # "Hello"  
s = "'Hello'"  
print(s)                      # 'Hello'  
s = "\\\'\\Hello\\\\"  
print(s)                      # ''\Hello\  
s = "Hello\nPython"          # Hello  
                            # Python
```

\ " คือ "

\ ' คือ '

\ \ คือ \

\n คือ รหัสให้ขึ้นบรรทัดใหม่

## ตัวอย่าง: แทน " ' / \ ( ) , . : ; ด้วยช่องว่าง

```
def blank(t):
    result = ""
    for c in t:
        if c in "\\"\'/\\"\",.:;":
            result += " "
        else:
            result += c
    return result
```

# บริการของสตริง : String Methods

01234567890123

**s = " Hello World "**

- **len(s)** ได้ 14 , **len("")** ได้ 0
- **s.lower()** ได้ " hello world "
- **s.upper()** ได้ " HELLO WORLD "
- **s.strip()** ได้ "Hello World"
- **s.find("o")** ได้ 6      **s.find("ex")** ได้ -1
- **s.find("o",7)** ได้ 9

# ตัวอย่าง: การใช้งาน

```
t = input().strip()      # ป้องกันกรณีผู้ใช้เหลือ空格 space
if t.upper() != "YES":   # ตรวจได้ทั้ง Yes, yes, ...
    j = t.find("mailto:")
    if j >= 0:
        j += len("mailto:")
        k = t.find("@", j)
        if k >= 0:
            username = t[j:k]
        ...
    
```



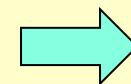
```
<a href="mailto:somchai@gmail.com">...
```

# แบบฝึกหัด: camelCase

camelCase เป็นรูปแบบการเขียนวิลีที่ไม่มีเครื่องวรดตอน โดยนำคำต่าง ๆ ในวลีมาเขียนติดกันหมด แต่เขียนให้แต่ละคำขึ้นต้นด้วยตัวใหญ่ยกเว้นเฉพาะคำแรกสุดเป็นตัวเล็กหมด (ตัวเลขคงไว้เหมือนเดิม)

**Input**

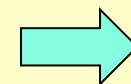
An example of "camel case".



**Output**

anExampleOfCamelCase

Emergency call 911



emergencyCall911



```
import urllib.request

url = "http://air4thai.pcd.go.th/services/getNewAQI_XML.php?stationID=52t"
web = urllib.request.urlopen(url)
for line in web:
    line = line.decode()
    pattern = '<PM25 value="'
    i = line.find(pattern)
    if i >= 0:
        i += len(pattern)
        j = line.find('"', i)
        print("PM 2.5 =", line[i:j])
        break
```

```
def split(t, s) :
    x = []
    k0 = 0
    k = t.find(s,0)
    while k >= 0:
        x.append(t[k0:k])
        k0 = k+1
        k = t.find(s,k0)
    x.append(t[k0:])
    return x
```

# Method Chaining vs Function Composition

```
x = math.radians(d)
s = math.sin(x)
y = abs(s)
r = round(y, 2)
```

```
r = round(abs(math.sin(math.radians(d))), 2)
```

```
line1 = input()
line2 = line1.strip()
line3 = line2.upper()
i      = line3.find("OK")
```

```
i = input().strip().upper().find("OK")
```

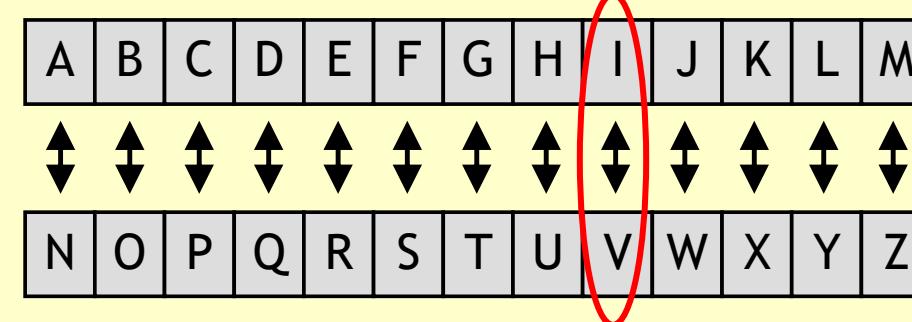
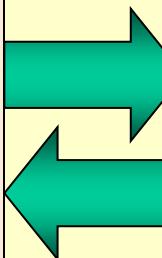
# ข้อควรระวัง

- ห้ามเปลี่ยนค่าภายในสตริง
  - `s[2] = "a"`
  - `s[3:7] = "-^o^-"`
- string methods ไม่เปลี่ยนตัวสตริง
  - `s = "Hello"`
  - `s.lower()` ได้ '`hello`' แต่ `s` เก็บค่าเดิม '`Hello`'
- แต่เปลี่ยนค่าชี้งเก็บในตัวแปรได้
  - `s = "Hello"`
  - `s = s.lower()` แบบนี้ `s` เก็บค่าใหม่ '`hello`'

# ตัวอย่าง : rot-13

I SEE TREES OF GREEN  
RED ROSES TOO  
I SEE EM BLOOM  
FOR ME AND FOR YOU  
AND I THINK TO MYSELF  
WHAT A WONDERFUL  
WORLD

V FRR GERRF BS TERRA  
ERQ EBFRF GBB  
V FRR RZ OYBBZ  
SBE ZR NAQ SBE LBH  
NAQ V GUVAZ GB ZLFRYS  
JUNG N JBAQRESHY  
JBEYQ



# ตัวอย่าง : rot-13

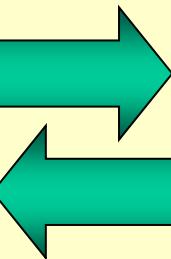
```
def rot13(txt_in):
    upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    alphabets = 2*upper
    txt_out = ""
    for ch in line:
        j = alphabets.find(ch)
        if j != -1:
            txt_out += alphabets[j + 13]
        else:
            txt_out += ch
    return txt_out
line = input()
while line.strip().upper() != "END":
    print(rot13(line))
    line = input()
```



# แบบฝึกหัด: rot-13

ปรับโปรแกรมให้เข้า/ออกรหัสได้ทั้งตัวอังกฤษเล็กและใหญ่

I see trees of green  
red roses too  
  
I see em bloom  
for me and for you  
  
And I think to myself  
what a wonderful world  
  
end



V frr gerrf bs terra  
erq ebfrf gbb  
  
V frr rz oybbz  
sbe zr naq sbe lbh  
  
Naq V guvax gb zlfrys  
jung n jbaqreshy jbeyq



# การอ่านข้อมูลจากแฟ้มข้อความ

```
fin = open( filename, "r" )  
line = fin.readline()  
...  
for line in fin:  
    ...  
fin.close()
```

"r" บอกว่าเปิด  
แฟ้มเพื่อ อ่าน

readline อ่านหนึ่งบรรทัด  
ถัดไปเข้ามาเป็นสตริง

for แบบนี้ อ่านทีละบรรทัด  
ถัดไปเป็นสตริงใส่ตัวแปร line  
รอบละหนึ่งบรรทัด จนหมดแฟ้ม

เลิกอ่านแฟ้ม  
แล้วก็ปิดแฟ้ม

# ตัวอย่าง: หาคะแนนเฉลี่ยนิสิต 3 คนแรกในแฟ้ม

```
sum_points = 0; n = 3

infile = open("data.txt", "r")
for k in range(n):
    line = infile.readline() # อ่านบรรทัดถัดไป
    x = line.split()
    sum_points += float(x[1])
infile.close()

print("Average =", sum_points/n)
```

	x[0]	x[1]	Average = 83.0
data.txt	6230012121	90	
	6230351221	80	
	6231027921	79	
	6230548121	70	

# ตัวอย่าง: อ่านแฟ้มคณ์แนมมาแสดงเรียงจากมากถึงน้อย

```
students = []

infile = open("data.txt", "r")
for line in infile: # อ่านรอบบรรทัดจนหมดแฟ้ม
    sid,point = line.strip().split()
    point = float(point)
    students.append([point, sid])
infile.close()

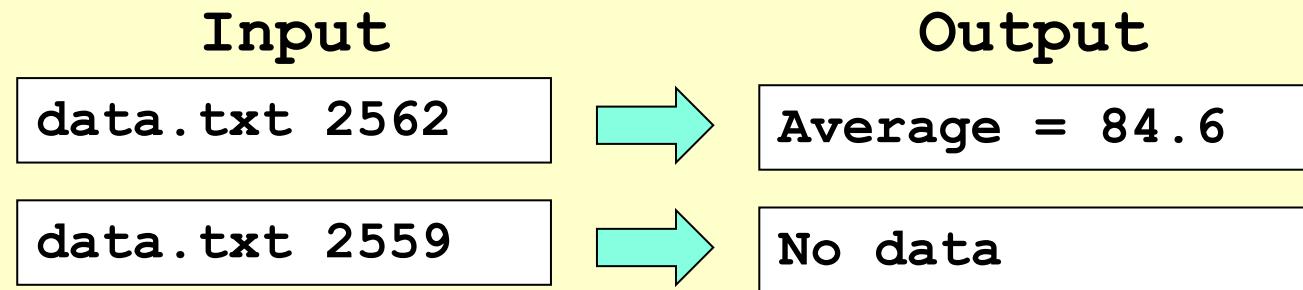
students.sort() # เรียงจากน้อยไปมากตามคณ์แนน

for [point,sid] in students[::-1]:
    print(sid, point)
```

data.txt

6230012121	90
6230351221	80
6231027921	79
6230548121	70

# แบบฝึกหัด: หาค่าแนวเฉลี่ยของนิสิตบางกลุ่มในไฟล์



data.txt

6230012121 90
6130351221 80
6231027921 79
5830548121 65
6031087221 70
6230550321 72
6230432721 87
6230215221 95
6130518321 72

# การบันทึกข้อมูลลงแฟ้มข้อความ

```
fout = open( filename, "w" )
```

"w" บอกว่าเปิด  
แฟ้มเพื่อ **เขียน**

```
fout.write("First line")
```

write บรรทัดสตริง  
ต่อท้ายในแฟ้ม

```
fout.write("Text\n")
```

ต้องการขึ้นบรรทัด  
ใหม่ต้องใส่ \n เอง

```
fout.close()
```

บันทึกเสร็จแล้ว  
ก็ปิดแฟ้ม

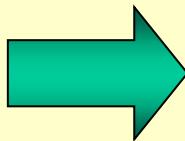
# ตัวอย่าง: บันทึก Triangular Number ร้อยตัวลงแฟ้ม

```
fout = open("tri_numbers.txt", "w")
for k in range(1,101):
    n = k*(k+1)//2      # triangular number
    fout.write(str(n)+" ")
    if k%10 == 0:        # ครบ 10 ตัว ขึ้นบรรทัดใหม่
        fout.write("\n")
fout.close()
```

1	3	6	10	15	21	28	36	45	55
66	78	91	105	120	136	153	171	190	210
231	253	276	300	325	351	378	406	435	465
496	528	561	595	630	666	703	741	780	820
861	903	946	990	1035	1081	1128	1176	1225	1275
1326	1378	1431	1485	1540	1596	1653	1711	1770	1830
1891	1953	2016	2080	2145	2211	2278	2346	2415	2485
2556	2628	2701	2775	2850	2926	3003	3081	3160	3240
3321	3403	3486	3570	3655	3741	3828	3916	4005	4095
4186	4278	4371	4465	4560	4656	4753	4851	4950	5050

# แบบฝึกหัด: สร้างแฟ้มที่เป็น rot-13 ของอีกแฟ้ม

I see trees of green  
red roses too  
  
I see em bloom  
for me and for you  
  
And I think to myself  
what a wonderful world



V frr gerrf bs terra  
erq ebfrf gbb  
  
V frr rz oybbz  
sbe zr naq sbe lbh  
  
Naq V guvax gb zlfrys  
jung n jbaqreshy jbeyq

**data.txt**

**rot13.txt**

Dict

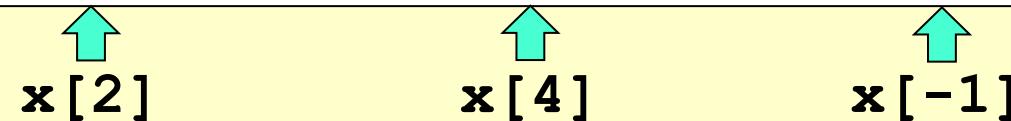
ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# list กับ dict

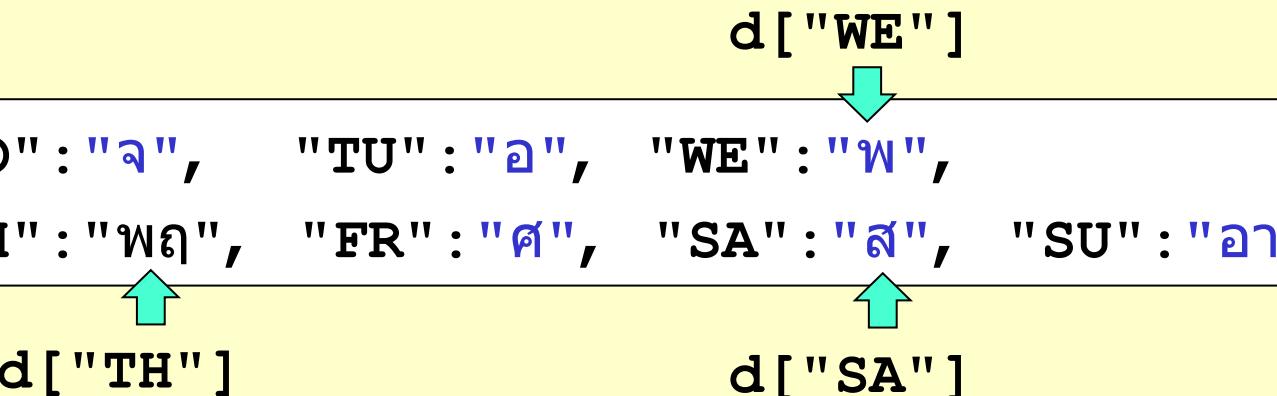
- list: เก็บข้อมูลเป็นรายการเรียงจากซ้ายไปขวา
  - ใช้จำนวนเต็มเป็นตัวระบุตำแหน่งในการใช้ข้อมูลใน list

```
x = ["MO", "TU", "WE", "TH", "FR", "SA", "SU"]
```



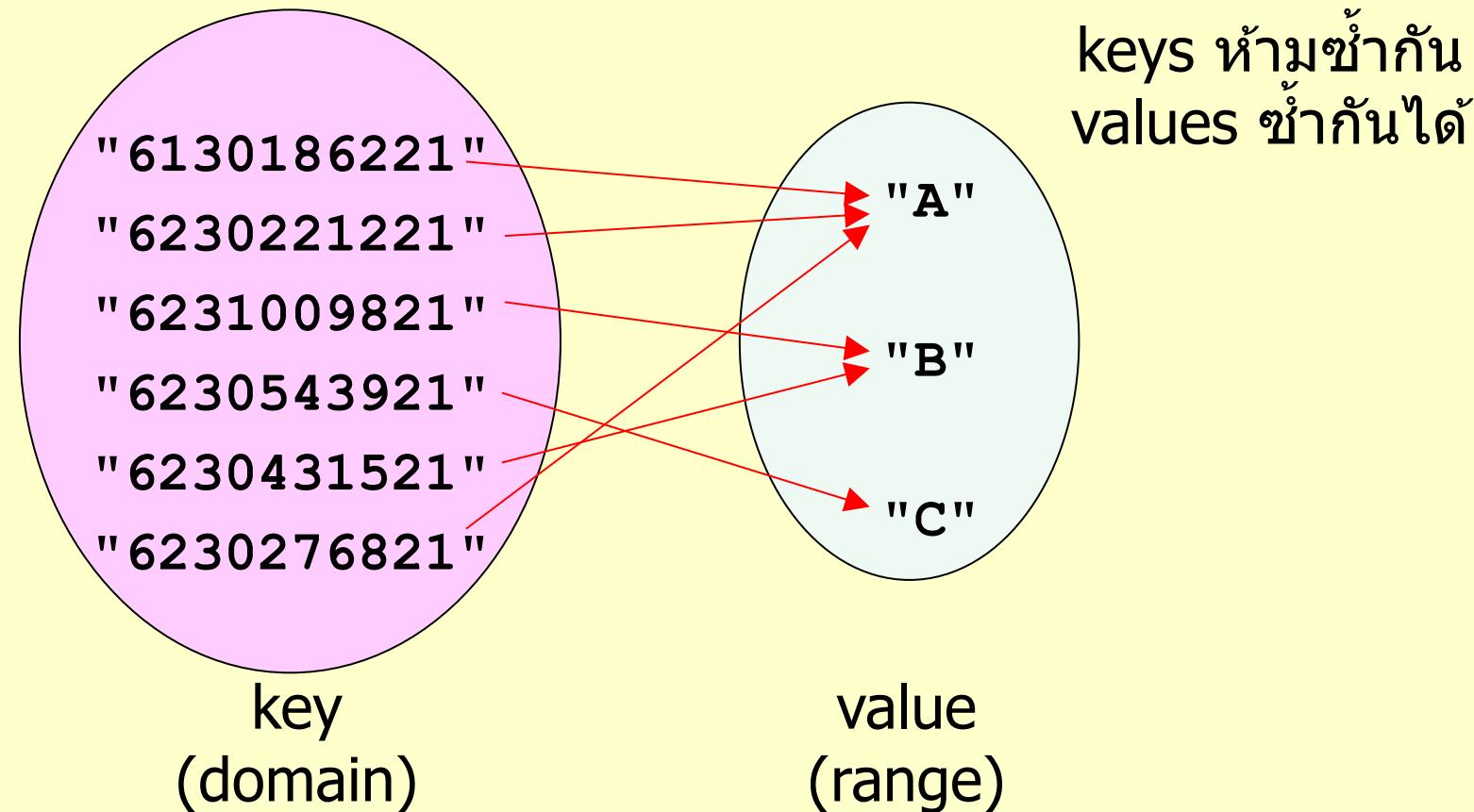
- dict: เก็บข้อมูลเป็นคู่ ๆ **key**: **value**
  - ใช้ **key** เป็นตัวระบุ **value** ที่ต้องการใน dict  
(key เป็นได้ทั้ง int, float, str, ...)

```
d = {"MO": "ຈ", "TU": "ອ", "WE": "ພ",
      "TH": "ພຖ", "FR": "ສ", "SA": "ສ", "SU": "ອາ"}
```



# dict เหมือน mapping function

```
grade = {"6130186221": "A", "6230221221": "A",
          "6231009821": "B", "6230543921": "C",
          "6230431521": "B", "6230276821": "A"}
```



# ให้ key กับ dict และได้ value

v = grade["6231010621"]

key      value

```
grade = {"6130186221": "A", "6230221221": "A",
          "6231009821": "B", "6230543921": "C",
          "6230431521": "B", "6230276821": "A"}
```

```
ID = input()
```

```
while ID != "q":
```

```
    print(ID, "-->", grade[ ID ])
```

```
    ID = input()
```

ถ้าใช้คีย์ที่ไม่มีอยู่ใน dict  
จะเกิดความผิดพลาด

Input

```
6231009821
6130186221
6230221221
q
```

output

```
6231009821 --> B
6130186221 --> A
6230221221 --> A
```

มีแต่ให้ key ได้ value  
ไม่มีแบบให้ value ได้ key

# การเพิ่ม/การเปลี่ยน value ใน dict

```
grade["6231010621"] = "A"
```

```
grade = {}  
grade["6231010621"] = "A"      # {"6231010621": "A"}  
  
grade["6231009821"] = "B"      # {"6231010621": "A",  
                                "6231009821": "B"}  
  
grade["6231009821"] = "C"      # {"6231010621": "A",  
                                "6231009821": "C"}
```

# การเพิ่ม/การเปลี่ยน value ใน dict

## Input

```
grade = {}      # dict ว่าง
n = int(input())
for k in range(n):
    ID, g = input().split()
    grade[ ID ] = g
print(grade)
```

4	เพิ่ม
6130186221 A	เพิ่ม
6230221221 A	เพิ่ม
6231009821 F	เพิ่ม
6231009821 B	เปลี่ยน

## Output

```
{'6130186221': 'A', '6230221221': 'A', '6231009821': 'B'}
```

# ตัวอย่าง: นับโหวต

```
BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]
votes = {}                      # สร้าง dict ว่าง
for name in BLACKPINK:
    votes[name] = 0              # เพิ่ม key: value ใหม่ใน dict
# {"Jisoo": 0, "Jennie": 0, "Rosé": 0, "Lisa": 0}

name = input()
while name != 'q':
    if name in BLACKPINK:
        votes[name] += 1          # เพิ่ม value
    name = input()

for name in BLACKPINK:
    print(name, "-->", votes[name]) # หยิบ value มาใช้
```

```
Lisa
Lisa
Lisa
Lisa
Jennie
Aum
Jisoo
Lisa
q
```

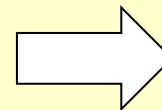
# แบบฝึกหัด: นับจำนวนของตัวอักษรแต่ละตัว

```
alphabets = "abcdefghijklmnopqrstuvwxyz"
```

<i>Input</i>	<b>ABCabcABZ</b>
<i>Output</i>	a -> 3 b -> 3 c -> 3 z -> 1

for *each\_key* in *a\_dict*

```
for k in dic:  
    if k == t:  
        v = dic[k]
```



```
v = dic[t]
```

*for each\_key in a\_dict*

เมื่อต้องการหยิบ key แต่ละตัวออกมาใช้งาน

```
ordinal = {"first": 1, "second": 2, "third": 3,
           "fourth": 4, "fifth": 5, "sixth": 6,
           "seventh": 7, "eighth": 8,
           "ninth": 9, "tenth": 10 }

for k in dic:
    if k == t:
        v = dic[k]

v = dic[k]
for key in ordinal:
    print(key, "-->", ordinal[key] )
```

ข้อสังเกต: key ที่ได้จาก  
for มีลำดับไม่แน่นอน

ถ้า run โปรแกรมนี้ด้วย Python 3.7 จะได้ key เรียงตามดอน  
สร้าง แต่ไม่รับประกันว่าพฤษติกรรมจะเป็นแบบนี้ในรุ่นถัด ๆ ไป  
(Python ใน Grader ก็เป็นแบบที่ได้ลำดับไม่แน่นอน)

tenth --> 10  
second --> 2  
fourth --> 4  
third --> 3  
sixth --> 6  
ninth --> 9  
seventh --> 7  
eighth --> 8  
fifth --> 5  
first --> 1

# ตัวอย่าง: พึงก์ชั้นหาค่าเฉลี่ยของ value ใน dict

```
def average( d ):      # d เป็น dict ที่มี value เป็นจำนวน  
    total = 0  
    for key in d:          # ลุยกะทุก key มาใช้  
        total += d[key]  
    return total / len(d)
```

`len(d)` คือจำนวนคู่  
key:value ใน dict `d`

```
gpa = {"6130186221": 3.15, "6230221221": 2.85,  
       "6231009821": 2.90, "6230543921": 3.20,  
       "6230431521": 3.35, "6230276821": 3.42}
```

```
print( average(gpa) ) # ได้ 3.145
```

# แบบฝึกหัด: ส่องฟังก์ชัน

```
def reverse( d ): # d เป็น dict ที่ประกันว่า value ไม่ซ้ำกัน
    r = { }
    รับ { "A": "ເອ", "B": "ບີ", "C": "ໜີ" }
    คืน { "ເອ": "A", "ບີ": "B", "ໜີ": "C" }
    return r
```

```
def keys( d, v ): # คืนลิสต์ของ keys ที่มี value เท่ากับ v
    x = []
    รับ d = { 2:33, 4:55, 9:33, 7:33, 8:55 }
    v = 33
    คืน [ 2, 7, 9 ]
    return x
```

*if key in dict หรือ if key not in dict*

เมื่อยกตัวมารู้ว่ามี *key* หรือไม่มี *key* ใน *dict* ไหม ?

```
grade = {"6130186221": "A", "6230221221": "A",
          "6231009821": "B", "6230543921": "C",
          "6230431521": "B", "6230276821": "A"}
```

```
ID = input()
```

```
while ID != "q":
```

```
    if ID in grade:
```

```
        print(ID, "-->", grade[ ID ])
```

```
    else:
```

```
        print("Not found")
```

```
    ID = input()
```

ถ้าไปหยิบ value ของ key ที่ไม่มีใน dict  
จะทำงานผิดพลาด จึงต้องทดสอบก่อน

*if key in dict* ทำงานเร็วกว่า *if elem in list*

```
import time
def search_all(X):
    b = time.time()
    n = len(X)
    for i in range(n):
        if i in X:      # True
            pass
    for i in range(n):
        if (n+1) in X: # False
            pass
    print(time.time() - b)
```

```
n = 50000
L = []
for i in range(n):
    L.append(i)
D = {}
for i in range(n):
    D[i] = i

search_all(L)
search_all(D)
```

การค้นใน dict  
เร็วกว่า list

มาก

40.90408134460449  
0.0156097412109375

หน่วยเป็นวินาที

# แบบฝึกหัด: ชื่อเล่นอะไร ชื่อจริงอะไร

เขียนโปรแกรม รับชื่อจริงแสดงชื่อเล่น  
รับชื่อเล่นแสดงชื่อจริง โดยใช้ dict

Robert      ⇔    Dick

William     ⇔    Bill

James        ⇔    Jim

John        ⇔    Jack

Margaret    ⇔    Peggy

Edward     ⇔    Ed

Sarah        ⇔    Sally

Andrew     ⇔ [Andy](http://www.kyoto-su.ac.jp/~trobb/nicklist.html)

# ตัวอย่าง: นับโหวต

```
BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]
votes = {"Jisoo":0, "Jennie":0, "Rosé":0, "Lisa":0}

name = input()
while name != 'q':
    if name in BLACKPINK:
        votes[name] += 1
    name = input()

for name in BLACKPINK:
    print(name, "-->", votes[name])
```

ตรวจจาก key ใน votes ก็ได้  
หยิบจาก key ใน votes ก็ได้  
(ดูหน้าถัดไป)

Lisa  
Lisa  
Lisa  
Lisa  
Jennie  
Aum  
Jisoo  
Lisa  
q

# ตัวอย่าง: นับโหวต (เขียนอีกแบบ)

```
votes = {"Jisoo":0, "Jennie":0, "Rosé":0, "Lisa":0}

name = input()
while name != 'q':
    if name in votes:
        votes[name] += 1
    name = input()

for name in votes:
    print(name, "-->", votes[name])
```

ตรวจก่อนว่าเป็นชื่อที่ถูกต้อง  
แล้วค่อยเพิ่มคะแนน

Lisa  
Lisa  
Lisa  
Lisa  
Jennie  
Aum  
Jisoo  
Lisa  
q

# ตัวอย่าง: นับโหวต (แบบนับทุกชื่อ)

```
votes = {}  
  
name = input()  
while name != 'q':  
    if name in votes:  
        votes[name] += 1 # เป็นชื่อที่มีอยู่แล้ว เพิ่ม 1 คะแนน  
    else:  
        votes[name] = 1 # เป็นชื่อใหม่ ให้ 1 คะแนน  
    name = input()  
  
for name in votes:  
    print(name, "-->", votes[name])
```

เราไม่รู้ว่า จะมีชื่อใดบ้าง  
ก็ต้องอ่านไป สร้างไป นับไป

Lisa  
Lisa  
Lisa  
Lisa  
Jennie  
Aum  
Jisoo  
Lisa  
q

# แบบฝึกหัด: ยอดขายไอศกรีม

	<i>Input</i>	<i>Output</i>
ราคา	5 Magnum 50 Cornetto 25 PaddlePop 15 AsianDelight 20 Calippo 15  9 Magnum 5 Cookie 4 MamaTomYum 3 Cornetto 5 AsianDelight 4 Calippo 4 Cornetto 6 MangoSheet 4 Calippo 4	Total ice cream sales = 725.0
จำนวนขาย		ยอดขาย = 5x50 +  5x25 + 4x20 + 4x15 + 6x25 +  4x15 = 725

# ย้ำอีกครั้ง

d เป็น dict, e เป็น key

```
if e in d :  
    ...
```

เร็ว

x เป็น list, e เป็นข้อมูล

```
if e in x :  
    ...
```

ช้า

```
k = x.index(e)
```

ช้า

k เป็นจำนวนเต็ม

```
x[k] = z
```

เร็ว

d[e] = z

เร็ว

```
z = d[e]
```

เร็ว

```
z = x[k]
```

เร็ว



# Nested Structures

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# More Data & Flow Controls

เรียนไปแล้ว

`int, float,  
str, bool  
list, dict`

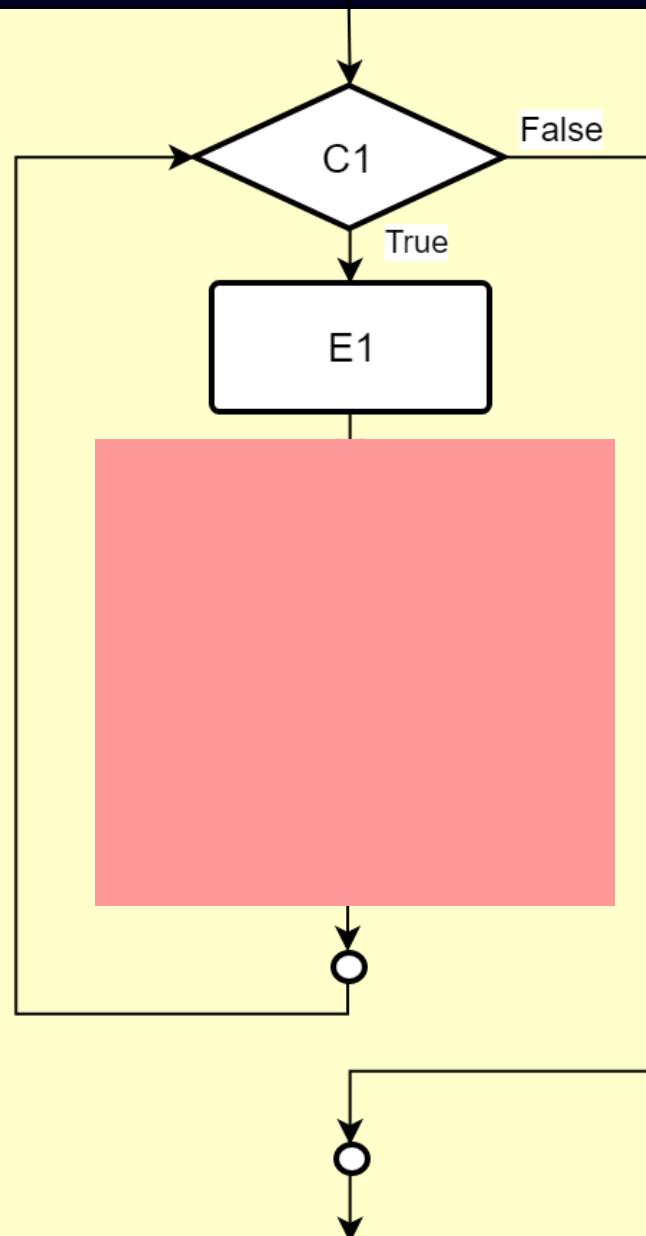
`if-elif-else  
while  
for  
break  
function`

จะเรียนต่อไป

`tuple, set,  
list, dict,  
numpy array,  
class/object`

`nested loop  
comprehension  
recursion`

# Nested Loops: while ช้อน while



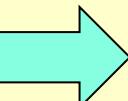
**while C1:**

**E1**

# ตัวอย่าง: หา หrm. ของจำนวนเต็มหลัก ๆ คู่

```
x = input().split()
while x[0] != 'q':
    a,b = int(x[0]),int(x[1])
    while b != 0:
        a,b = b, a % b
    print(a)
    x = input().split()
```

5 8  
143 65  
q



1  
13

a b  
143 65  
65 13  
13 0

# แบบฝึกหัด: Factorization

```
def factor(N):
    f = []
    k = 2
    while k <= N:
        # ถ้า k หาร N ลงตัว
        # ก็ วนหาร N ด้วย k
        # จน k ไม่เป็น factor ของ N
        # เพิ่ม k และจำนวนครั้งที่หาร ใส่ใน f
        k += 1
    return f
```

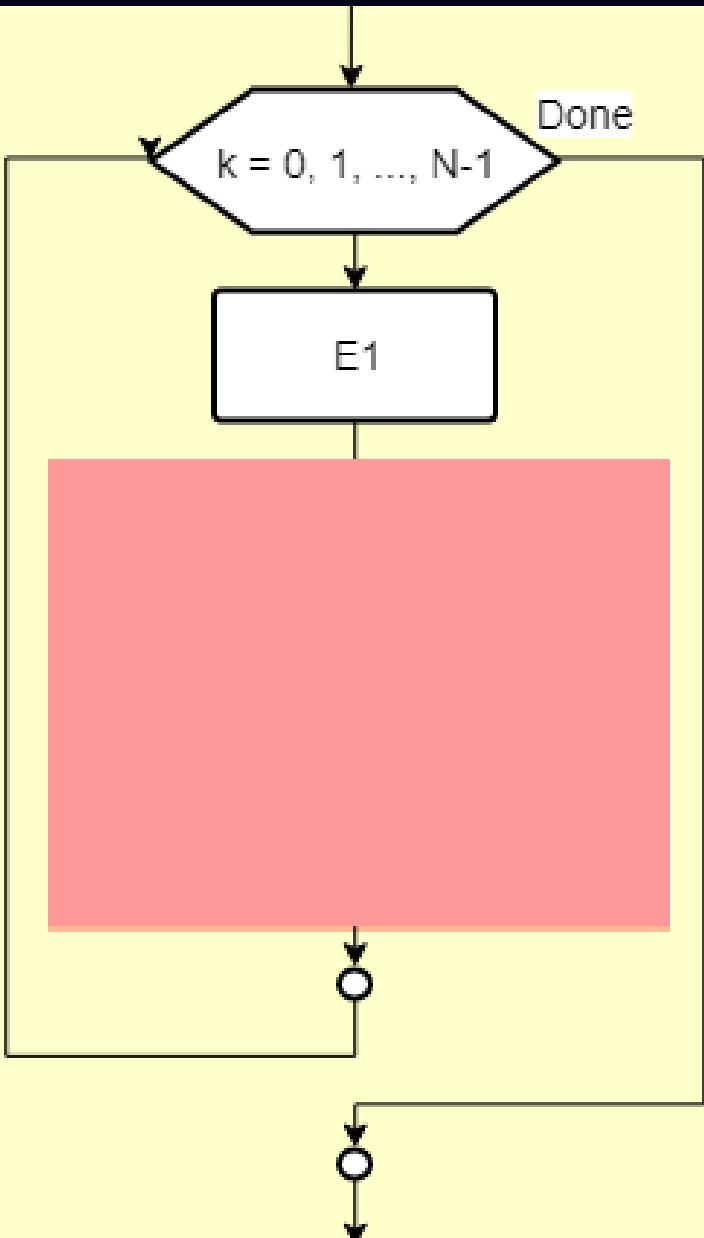
ເວັ້ນ ทำໄໝເຮົາໄນ່ໃຫ້  
`for k in range(2, N+1):`

$N = 200, k = 2, N = 200 \rightarrow 100 \rightarrow 50 \rightarrow 25$   
 $N = 25, k = 3$   
 $N = 25, k = 4$   
 $N = 25, k = 5, N = 25 \rightarrow 5 \rightarrow 1$

`f = [[2,3], [5,2]]`

$$200 = 2^3 \cdot 5^2$$

# Nested Loops: for ชื่อ for



```
for k in range(N) :
```

```
E1
```

# ตัวอย่าง

```
for i in range(3):  
    for j in range(4):  
        print(i, j)
```

i	j
0	0
1	
2	
3	
1	0
1	1
2	2
3	3
2	0
2	1
2	2
3	3

```
for i in range(3):  
    for j in range(i, 4):  
        print(i, j)
```

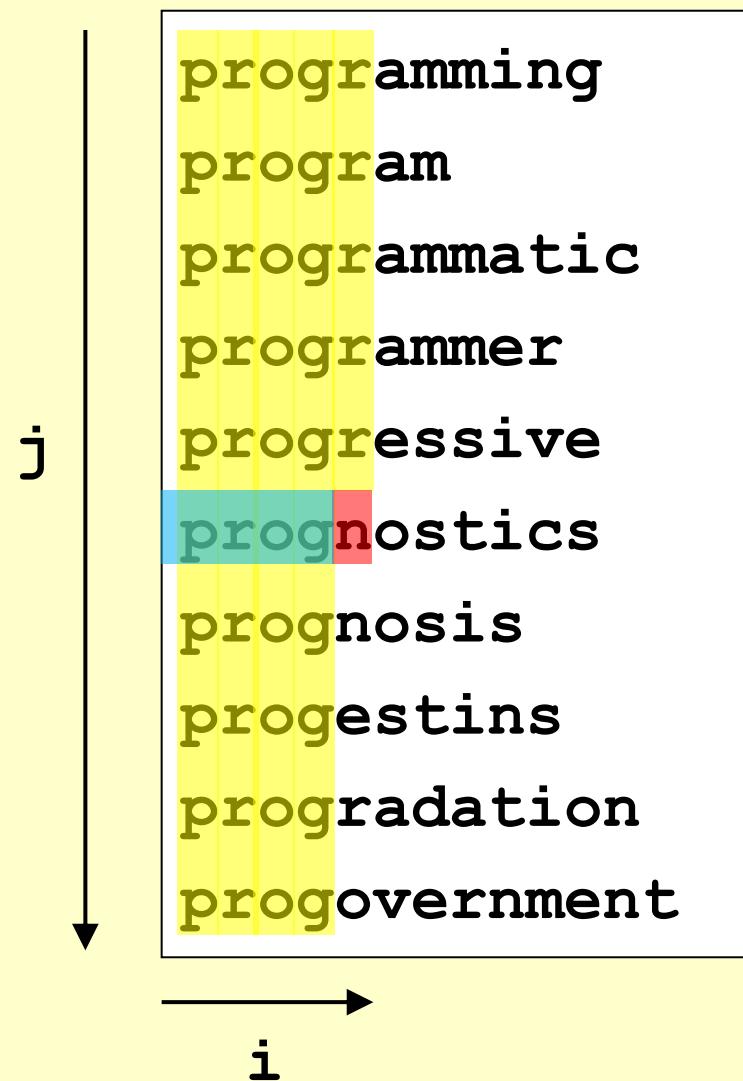
i	j
0	0
0	1
0	2
0	3
1	1
1	2
1	3
2	2
2	3
3	3

```
for i in range(3):  
    for j in range(i+1, 4):  
        print(i, j)
```

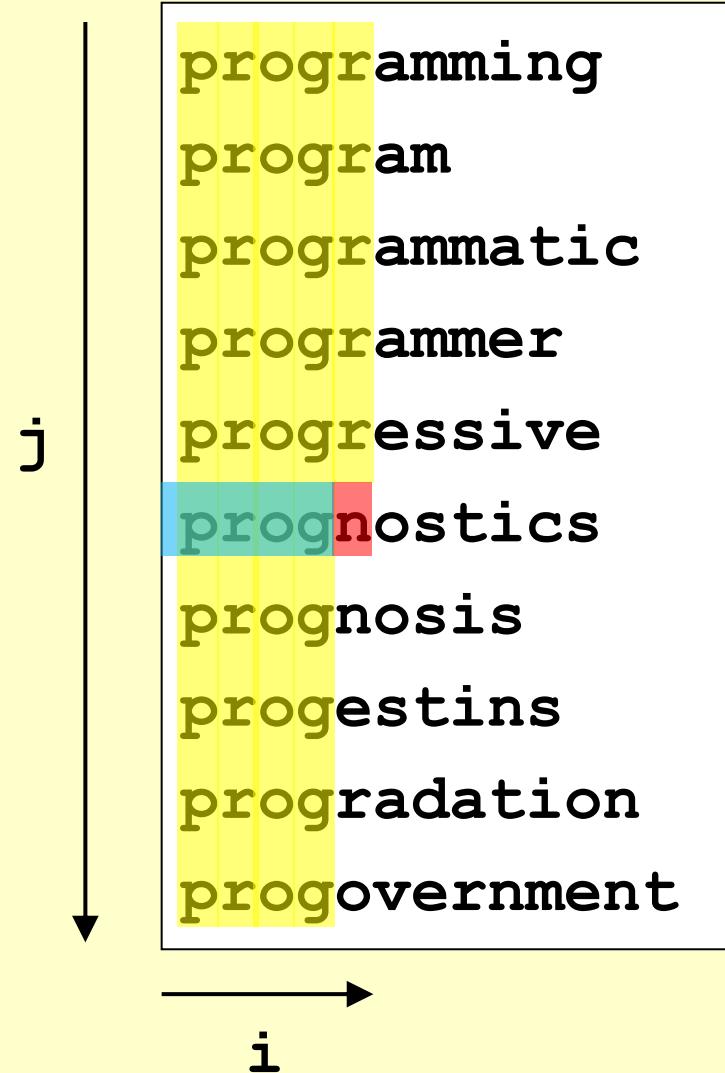
i	j
0	1
0	2
0	3
1	2
1	3
2	3

แจกแจง index ของ  
ทุกคู่ข้อมูลในลิสต์

# ตัวอย่าง: พึงก์ชั้นหา prefix ยาวสุดของรายการสตริง



# ตัวอย่าง: พึงก์ชั้นหา prefix ยาวสุดของรายการสตริง



```
def longest_prefix(words):  
    for i in range(len(words[0])):  
        c = words[0][i]  
        for j in range(1, len(words)):  
            if i >= len(words[j]) or \  
                c != words[j][i]:  
                    return words[j][:i]  
    return words[0]
```

# ตัวอย่าง: พัฒนาขั้นตอนตรวจสอบข้อมูลซ้ำกันในลิสต์

```
def has_duplicate( x ):  
    for i in range(len(x)-1):  
        for j in range(i+1, len(x)):  
            if x[i] == x[j]:  
                return True  
  
    return False
```

i	j
0	1
	2
	3
1	2
	3
2	3

x = [11, 34, 22, 34]

จุดตรวจทุกคู่

## ตัวอย่าง: Pairwise Coprime

- A set of integers can be called **coprime** if its elements share no common positive factor except 1.
- A stronger condition on a set of integers is **pairwise coprime**, which means that  $a$  and  $b$  are coprime for every pair  $(a, b)$  of different integers in the set.
- The set  $\{2, 3, 4\}$  is coprime, but it is not pairwise coprime since 2 and 4 are not relatively prime.

21 ,	15 ,	35
$3 \times 7$	$3 \times 5$	$5 \times 7$

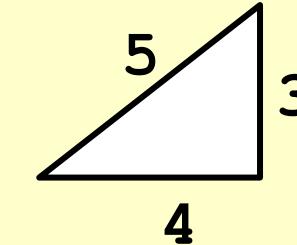
21 ,	10 ,	121
$3 \times 7$	$2 \times 5$	$11 \times 11$

# ตัวอย่าง: Pairwise Coprime

```
def gcd(a,b):  
    while b != 0:  
        a,b = b, a%b  
    return a  
  
def is_pairwise_coprime( d ):  
    for i in range(len(d)-1):  
        for j in range(i+1, len(d) ):  
            if gcd(d[i],d[j]) != 1:  
                return False  
    return True
```

# แบบฝึกหัด: Primitive Pythagorean Triple

- Pythagorean triple: จำนวนเต็ม  $a$ ,  $b$  และ  $c$  ที่  
 $a^2 + b^2 = c^2$  เช่น  $(3, 4, 5)$
- ถ้า  $(a, b, c)$  เป็น Pythagorean triple  
 $(ka, kb, kc)$  ก็เป็นด้วย โดยที่  $k = 1, 2, 3, 4, \dots$
- เราต้องการ Primitive Pythagorean triple คือ  
Pythagorean triple  $(a, b, c)$  ที่  $a, b$  และ  $c$  เป็น coprime (คือมี ห.ร.ม. เป็น 1)
- จะเขียนโปรแกรมหา Pythagorean triple ทุกค่าที่  
 $a \leq b \leq c \leq M$  โดยที่  $M$  คือ input เช่น ให้  $M = 20$   
จะได้



$[3, 4, 5], [5, 12, 13], [8, 15, 17]$

# แบบฝึกหัด: Primitive Pythagorean Triple

```
def gcd(a,b) :  
    while b != 0:  
        a,b = b, a%b  
    return a  
  
def is_coprime(a, b, c) :  
    ???  
  
def primitive_Pythagorean_triple( M ) :  
    triple = []  
    for ??? in range( ??? ) :  
        for ??? in range( ??? ) :  
            for ??? in range( ??? ) :  
                ???  
  
    return triple
```

# យោងនៅក្នុងការប្រព័ន្ធបាត់

```
def is_pairwise_coprime(d):
    for i in range(len(d)-1):
        for j in range(i+1, len(d)):
            a,b = d[i],d[j]
            while b != 0:
                a,b = b, a%b
            if a != 1:
                return False
    return True
```



```
def gcd(a,b):
    while b != 0:
        a,b = b, a%b
    return a
```

```
#-----
def is_pairwise_coprime( d ):
    for i in range(len(d)-1):
        for j in range(i+1, len(d)):
            if gcd(d[i],d[j]) != 1:
                return False
    return True
```

ខ្សោយការ

# យោងនៅលើកបញ្ជី

```
n = int(input())
count = 0
for k in range(n):
    t = input()
    c = 0
    for ch in t:
        if "0" <= ch <= "9":
            c += 1
    count += c
print(count)
```

បង្ហាញតម្លៃរបស់ខ្លួន  
ដោយប្រើប្រាស់លទ្ធផល

```
def count_digits(s):
    c = 0
    for ch in s:
        if "0" <= ch <= "9":
            c += 1
    return c
```

```
#-----
n = int(input())
count = 0
for k in range(n):
    t = input()
    count += count_digits(t)
print(count)
```

# break ออกไปหลาย ๆ ชั้นไม่ได้

```
...  
for ...  
...  
for ...  
...  
if condition:  
...  
break  
...  
...
```

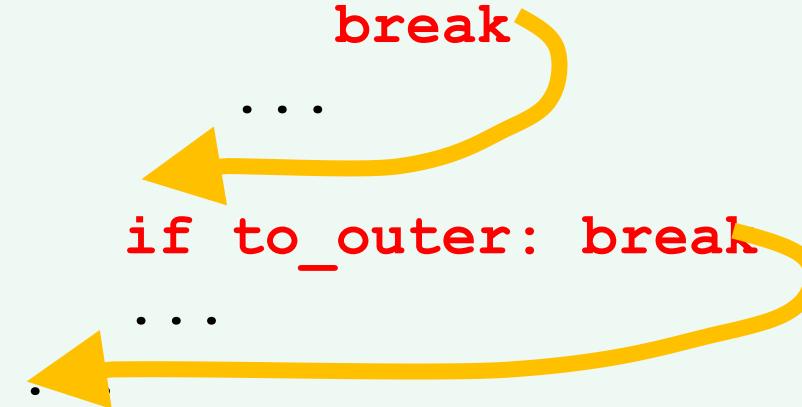
break จะกระโดด  
ออกมายังชั้นที่  
break อยู่

```
...  
for ...  
...  
for ...  
...  
if condition:  
...  
ออกไปนอกสุด  
...  
...
```

ถ้าต้องการให้ break  
กระโดดออกมาระหว่างนอก ๆ  
จะทำอย่างไร

# break ออกไปหลาย ๆ ชั้นด้วย ตัวแปรเสริม

```
...
to_outer = False
for ...
    ...
        for ...
            ...
                if condition:
                    ...
                        to_outer = True
                        break
...
if to_outer: break
...
.
```



# break ออกไปหลาย ๆ ชั้นด้วย ตัวแปรเสริม

```
prefix = words[0]
for i in range(len(words[0])):
    c = words[0][i]
    for j in range(1,len(words)):
        if i >= len(words[j]) or \
           c != words[j][i]:
            prefix = words[j][:i]
            อยากออกไปนอกสุด
            
```

หา longest prefix  
ของคำใน words

```
prefix = words[0]
found = False
for i in range(len(words[0])):
    c = words[0][i]
    for j in range(1,len(words)):
        if i >= len(words[j]) or \
           c != words[j][i]:
            prefix = words[j][:i]
            found = True
            break
            if found: break
            
```

**break** ออกไปหลาย ๆ ชั้นด้วย การแยกออกเป็นฟังก์ชัน

```
def func(...):
    for ...
        ...
        for ...
            ...
            if condition:
                ...
                ...
                return
            ...
        ...
    ...
#-----
```

The diagram illustrates the execution flow of a Python function. It starts with a call to `func(...)`. The function body begins with a `for` loop. Inside this loop, there is another `for` loop. Both loops have ellipses indicating they can contain multiple statements. Following the second `for` loop is an `if` condition with its own block of code. This block also contains ellipses. The `return` statement is located within this block. After the `return` statement, there are more ellipses. The function body ends with a final set of ellipses. The entire function definition is preceded by a dashed line, and the entire function body is shaded in light orange.

# break ออกไปหลาย ๆ ชั้นด้วย การแยกออกเป็นฟังก์ชัน

```
prefix = words[0]
for i in range(len(words[0])):
    c = words[0][i]
    for j in range(1,len(words)):
        if i >= len(words[j]) or \
            c != words[j][i]:
            prefix = words[j][:i]
```

อยากรอกรอไป



```
def longest_prefix(words):
    for i in range(len(words[0])):
        c = words[0][i]
        for j in range(1,len(words)):
            if i >= len(words[j]) or \
                c != words[j][i]:
                return words[j][:i]
    return words[0]
```

```
prefix = longest_prefix(words)
```

# Nested Lists: ลิสต์ช้อนในลิสต์

- เก็บข้อมูลที่ประกอบด้วยข้อมูลย่อยที่เป็นลิสต์
    - `["Ranee", 1989,  
 ["Plerng Boon", "Bubphe Sanniwat", "Krong Kam"]]`
  - เก็บข้อมูลหลาย ๆ ตัว ที่แต่ละตัวมีข้อมูลย่อย ๆ
    - `[ [6131001021, 3.8], [6130020221, 3.7] ]`
  - เก็บข้อมูลชั่วคราวเพื่อนำไปประมวลผล ( เช่น sort ตามความยาว)
    - `["your", "kiss", "is", "on", "my", "list"]  
[ [4, "your"], [4, "kiss"], [2, "is"],  
[2, "on"], [2, "my"], [4, "list"] ]`
  - เก็บเมทริกซ์
    - `[ [1, 2, 3, 0],  
[2, 3, 0, 1],  
[4, 1, 2, 2] ]` 
$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{bmatrix}$$

# สร้าง nested list

Ranee

1989

Plerng Boon, Bubphe Sanniwat, Krong Kam

```
name = input()
byear = int(input())
series = input().split(", ")
actress = [name, byear, series]
```

["Ranee", 1989,

  ["Plerng Boon", "Bubphe Sanniwat", "Krong Kam"] ]

# สร้าง nested list

```
3  
6131001021 3.8  
6130020221 3.7  
6130150721 2.7
```

```
n = int(input())  
students = []  
for i in range(n):  
    student_ID, gpax = input().split()  
    gpax = float(gpax)  
    students.append([student_ID, gpax])
```

```
[["6131001021", 3.8], ["6130020221", 3.7], ["6130150721", 2.7]]
```

# สร้าง nested list

```
["your", "kiss", "is", "on", "my", "list"]
```

```
[ [4, "your"], [4, "kiss"], [2, "is"],  
[2, "on"], [2, "my"], [4, "list"] ]
```

```
def sort_by_length( words ):  
    x = []  
    for w in words:  
        x.append( [len(w), w] )  
    x.sort()  
    for k in range(len(x)):  
        words[k] = x[k][1]
```

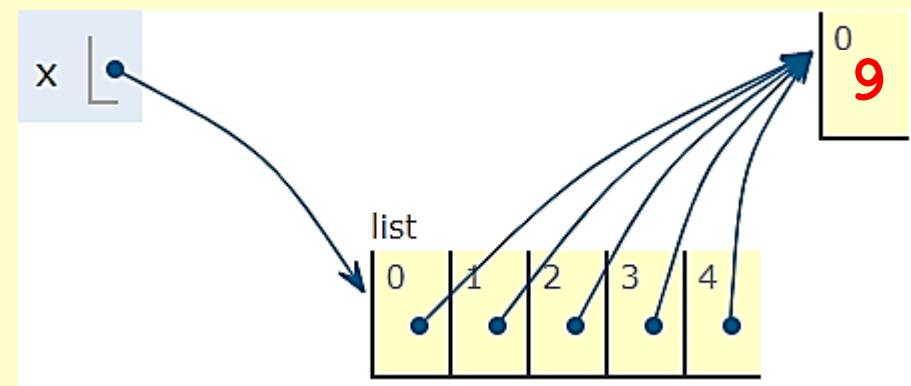
```
[ [2, "is"], [2, "my"], [2, "on"],  
[4, "kiss"], [4, "list"], [4, "your"] ]
```

```
["is", "my", "on", "kiss", "list", "your"]
```

# ข้อควรระวัง

```
x = [0]*5 ได้ [0, 0, 0, 0, 0]
```

```
x = [[0]]*5 ได้ [[0], [0], [0], [0], [0]]  
x[0][0] = 9 ได้ [[9], [9], [9], [9], [9]]
```



```
x = []  
for i in range(5):  
    x.append([0])
```

แบบนี้แต่ละช่องเป็นคนละลิสต์

## แบบฝึกหัด: First Fit / Best Fit

จะเขียนโปรแกรมแบ่งรายการของจำนวนเต็มไม่เกิน 100 ออกเป็นรายการย่อย ๆ แต่ละรายการมีผลรวมไม่เกิน 100 ให้ได้จำนวนรายการน้อย ๆ

[20,90,10,80]

First Fit

[ [20] ]

[ [20], [90] ]

[ [20,10], [90] ]

[ [20,10], [90], [80] ]

Best Fit

[ [20] ]

[ [20], [90] ]

[ [20], [90,10] ]

[ [20,80], [90,10] ]

เจอลิสต์แรกที่ใส่ได้ก็ใส่เลย

เลือกลิสต์อันที่ใส่แล้ว  
มีผลรวมใกล้ 100 ที่สุด

# Nested List as Matrix

```
3 [ 1.0, 2.0, 3.0, 0.0],  
1 2 3 0 [2.0, 3.0, 0.0, 1.0],  
2 3 0 1 [4.0, 1.0, 2.0, 2.0] ]  
4 1 2 2
```

```
def read_matrix():  
    m = []  
    nrows = int(input())  
    for k in range(nrows):  
        x = input().split()  
        r = []  
        for e in x:  
            r.append( float(e) )  
        m.append(r)  
    return m
```

# print\_matrix( M )

```
def print_matrix( M ):
    if len(M) == 1:
        print(M)
    else:
        print("[" + str(M[0]))
        for i in range(1,len(M)-1):
            print(" " + str(M[i]))
        print(" " + str(M[-1]) + "]")
```

```
M = [[1,2,3,4],[2,2,1,3],[2,6,7,7]]
print_matrix(M)
```

```
[[1, 2, 3, 4]
 [2, 2, 1, 3]
 [2, 6, 7, 7]]
```

## add\_matrix(A, B)

```
def add_matrix(A, B):
    C = []
    nrows = len(A)
    ncols = len(A[0])
    for i in range(nrows):
        C.append( [0.0]*ncols )
        for j in range(ncols):
            C[i][j] = A[i][j] + B[i][j]
    return C
```

```
A = read_matrix()
B = read_matrix()
C = add_matrix(A, B)
print_matrix(C)
```

## แบบฝึกหัด: mult(A, B)

```
def mult(A, B):  
    C = []  
  
    return C
```

$$C_{i,j} = \sum_{k=0}^{q-1} A_{i,k} B_{k,j}$$

$A$  มีขนาด  $p \times q$ ,  $B$  มีขนาด  $q \times r$ ,  $C$  มีขนาด  $p \times r$

# List Comprehension

วิธีการเขียนคำสั่งสร้างลิสต์ที่สั้น และมีประสิทธิภาพ

```
# ให้ x เป็นลิสต์ของ int  
t = []  
for e in x:  
    t.append(2*e)
```

```
# แปลงทุกค่าใน x เป็นอีกอย่าง  
t = [ 2*e   for e in x ]
```

```
t = []  
for e in x:  
    if e >= 0:  
        t.append(e)
```

```
# เลือกบางค่าในลิสต์ x  
t = [ e for e in x if e >= 0 ]
```

```
t = []  
for e in x:  
    if e >= 0:  
        t.append(2*e)
```

```
# เลือกบางค่าในลิสต์ x มาแปลง  
t = [2*e for e in x if e >= 0 ]
```

## ตัวอย่าง: อ่านรายการของจำนวนบรรทัดเดียวกัน

```
x = input().split()  
d = []  
for e in x:  
    d.append( int(e) )  
...
```

```
d = []  
for e in input().split():  
    d.append( int(e) )  
...
```

ใช้ list comprehension

```
d = [int(e) for e in input().split()]  
...
```

```
d = [float(e) for e in input().split()]  
...
```

## ตัวอย่าง: เรียงลำดับสตริงตามความยาว

```
def sorted_by_length(s):
    t = []
    for e in s:
        t.append( [len(e), e] )
    t.sort()
    r = []
    for n,e in t:
        r.append( e )
    return r
```

ใช้ list comprehension

```
def sorted_by_length(s):
    t = [[len(e),e] for e in s]
    t.sort()
    return [e for n,e in t]
```

# List Comprehension เร็วกว่า

```
import timeit

def for_loop(n):
    t = []
    for i in range(n):
        t.append(n)
    return t

def comprehension(n):
    return [n for i in range(n)]

def time(func):
    print(timeit.timeit(func+"(1000000)",
                        globals=globals(), number=100))

time("for_loop")      # 9.2609192
time("comprehension") # 4.7720880999999995
```

# Tuple – Set – Dict

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# Tuple

- tuple เหมือน list แต่สร้างแล้วเปลี่ยนแปลงไม่ได้
  - list: x1 = [1, 3, 4]; x2 = [9]
  - tuple: t1 = (1, 3, 4); t2 = (9,)
- มี operations ต่าง ๆ เหมือนลิสต์ (เฉพาะที่ไม่เปลี่ยนค่า)

```
t = (11,22,33)
print(len(t))          # 3
print(t[0],t[-1])     # 11 33
print(t[:2])           # (11,22)
print(33 in t)         # True
print(t.index(33))    # 2

# แก้ไขไม่ได้ แต่สร้างใหม่ได้
t[0] = 99              # ผิด
t = (99,) + t[1:]      # (99,22,33)
```

# Tuple vs. List

- มักใช้ list เก็บข้อมูลความหมายเหมือนกัน ชนิดเดียวกัน แต่ละตัวอาจเปลี่ยนค่า และลิสต์เปลี่ยนขนาดได้
  - ลิสต์ของนักเรียนเรียงตามคะแนนมากไปน้อย
  - ลิสต์ของอุณหภูมิที่วัดได้ทุกชั่วโมง
  - ลิสต์ของลูกค้าที่รอรับบริการ
- มักใช้ tuple เก็บข้อมูลความหมายต่างกัน อาจต่างชนิดกัน ข้อมูลไม่เปลี่ยนแปลง และทุกเปลี่ยนแปลงไม่เปลี่ยนขนาด
  - ทุกเปลี่ยนของเก็บพิกัด x, y แทนตำแหน่ง
  - ทุกเปลี่ยนของเก็บ เลขประจำตัว ชื่อ และ ปีเกิด
- ในทางเทคนิค tuple เล็กกว่า เร็วกว่า (นิดหน่อย)

ตัวอย่างของลิสต์ที่เคยนำเสนอมา ก่อนนี้ หลายอันใช้ทุกเปลี่ยนจะเหมาะสมกว่า

# ใช้ tuple เหมาะกว่า

```
days_of_week = ("SU", "MO", "TU", "WE", "TH", "FR", "SA")
```

```
[("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A")]
```

```
[(4, "your"), (4, "kiss"), (2, "is"), (2, "on"), (2, "my"), (4, "list")]
```

```
words = ["your", "kiss", "is", "on", "my", "list"]
x = [len(s), s] for s in words]
```

# ใช้ tuple เพื่อปกป้องการแก้ไข

```
[ ["6131001021", "A"], ["6130020221", "B"], ["6130150721", "A"] ]
```

```
[ ("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A") ]
```

```
( ("6131001021", "A"), ("6130020221", "B"), ("6130150721", "A") )
```

```
def print_grades(grades):
    for name,grade in grades:
        print(name, "-->" grade)
    grades[0][1] = "F"
```

```
    grades[0] = (grades[0][0], "F")
```

```
grades = ((grades[0][0], "F"),) + grades[1:]
```

# แบบฝึกหัด: Polynomial

$4x^2 + 3x - 1$  แทนด้วย [(4, 2), (3, 1), (-1, 0)]

```
def add_polynomial( p1, p2 ):  
    # คืนผลบวกของ p1 กับ p2
```

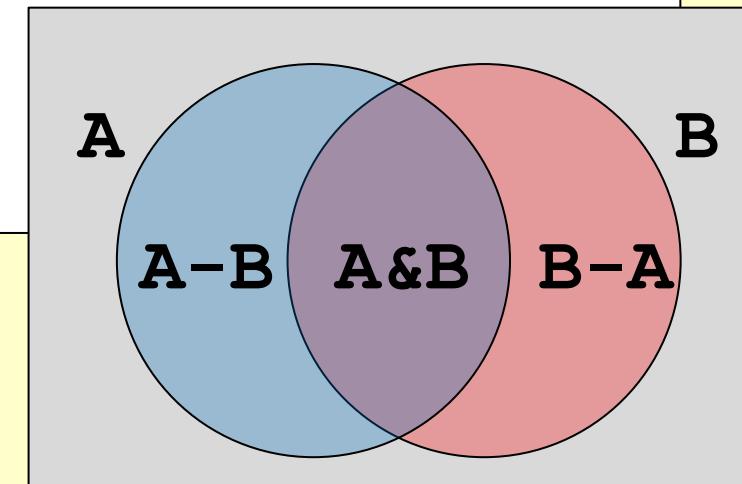
```
def mult_poly(p1, p2):  
    # คืนผลคูณของ p1 กับ p2
```

# Set

- set เป็นที่เก็บข้อมูลที่ไม่ซ้ำกัน และไม่มีลำดับ
- ข้อมูลที่เก็บใน set ต้องเปลี่ยนแปลงไม่ได้
  - เก็บ int, float, bool, str, tuple
  - เก็บ list, dict, set ไม่ได้
- $s = \{4, 3, 1, 2\}$
- $s = \text{set}()$  ได้เช็ตว่าง แต่ {} ได้ dict ว่าง
- ใช้  $\text{len}(s)$ ,  $\text{if } e \text{ in } s$ , และ  $\text{for } e \text{ in } s$  ได้

# Set Methods

```
A = {1,2,3,4,5}
B = {3,4,5,6,7}
C = A.union(B)          # {1,2,3,4,5,6,7}
C = A | B              # {1,2,3,4,5,6,7}
C = A.intersection(B) # {3,4,5}
C = A & B              # {3,4,5}
C = A.difference(B)   # {1,2}
C = A - B              # {1,2}
C.add(9)                # {1,2,9}
C.remove(1)              # {2,9}
print( A <= B, A.issubset(B) ) # True True
for e in A:
    print(e)
C = A ^ B
```



# การสร้างเซ็ตจากข้อมูลในที่เก็บต่าง ๆ

```
s = set(d)
```

เหมือนกับ

```
s = set()  
for e in d:  
    s.add(e)
```

<code>s = set([1,2,3,1])</code>	$\rightarrow$ <code>s = {1,2,3}</code>
<code>s = set((1,2,3,1))</code>	$\rightarrow$ <code>s = {1,2,3}</code>
<code>s = set("Mono")</code>	$\rightarrow$ <code>s = {"M", "o", "n"}</code>
<code>s = set({"A":2, "B":2})</code>	$\rightarrow$ <code>s = {"A", "B"}</code>
<code>s = set({1,2,3})</code>	$\rightarrow$ <code>s = {1,2,3}</code>
<code>s = set(range(1,7,2))</code>	$\rightarrow$ <code>s = {1,3,5}</code>

# การสร้างลิสต์จากข้อมูลในที่เก็บต่าง ๆ

```
x = list(d)
```

เหมือนกับ

```
x = []
for e in d:
    x.append(e)
```

```
x = list([1,2,3,1])
```

```
→ x = [1,2,3,1]
```

```
x = list((1,2,3,1))
```

```
→ x = [1,2,3,1]
```

```
x = list("Mono")
```

```
→ x = ["M", "o", "n", "o"]
```

```
x = list({"A":2, "B":2})
```

```
→ x = ["A", "B"]
```

```
x = list({1,2,3})
```

```
→ x = [1,2,3]
```

```
x = list(range(1,7,2))
```

```
→ x = [1,3,5]
```

# การสร้างทูเพิลจากข้อมูลในที่เก็บต่าง ๆ

```
t = tuple( [1,2,3,1] )           → t = (1,2,3,1)
t = tuple( (1,2,3,1) )          → t = (1,2,3,1)
t = tuple( "Mono" )             → t = ("M", "o", "n", "o")
t = tuple( {"A":2, "B":2} )       → t = ("A", "B")
t = tuple( {1,2,3} )              → t = (1,2,3)
t = tuple( range(1,7,2) )        → t = (1,3,5)
```

# sort vs. sorted

```
def sorted(x) :  
    out = []  
    for e in x:  
        out.append(e)  
    out.sort()  
    return out
```

sorted(x) เป็น built-in function ที่นำข้อมูลใน x มาเรียงลำดับข้อมูล แล้วคืนผลลัพธ์กลับมาเป็นลิสต์ ส่วน x ยังเหมือนเดิม

x = sorted( [1,4,3,2] )	→ x = [1, 2, 3, 4]
x = sorted( {1,4,3,2} )	→ x = [1, 2, 3, 4]
x = sorted( "hello" )	→ x = ["e", "h", "l", "l", "o"]
x = sorted( {22:2, 90:3, 3:23} )	→ x = [3, 22, 90]

```
x = [1, 4, 3, 2]  
x.sort()          # sort ใช้กับ list เท่านั้น
```

สังเกตความแตกต่างในการใช้ x.sort() กับ sorted( x )

*for elem in a\_set*

ข้อมูลถูกหยิบจากเซ็ตออกมายield ลำดับที่ไม่แน่นอน

```
S = {11111, 22222, 33333,  
     44444, 55555, 66666}  
for e in S:  
    print(e)
```

```
55555  
11111  
66666  
22222  
33333  
44444
```

*if e in set* ทำงานเร็วกว่า *if e in list*

```
import time
def search_all(X):
    b = time.time()
    n = len(X)
    for i in range(n):
        if i in X:      # True
            pass
    for i in range(n):
        if (n+1) in X: # False
            pass
    print(time.time() - b)
```

```
n = 50000
L = []
for i in range(n):
    L.append(i)
S = set()
for i in range(n):
    S.add(i)

search_all(L)
search_all(S)
```

การค้นใน set  
เร็วกว่า list

มาก

41.84556722640991  
0.0149579048156738

หน่วยเป็นวินาที

## ตัวอย่าง: พิ่งก์ชันตรวจสอบข้อมูลซ้ำกันในลิสต์

```
def has_duplicate( x ):  
    for i in range(len(x)-1):  
        for j in range(i+1, len(x)):  
            if x[i] == x[j]:  
                return True  
  
    return False
```

```
def has_duplicate( x ):  
    d = sorted(x)  
    for i in range(len(x)-1):  
        if d[i] == d[i+1]:  
            return True  
  
    return False
```

```
[2, 3, 4, 5, 6, 6, 6, 7, 8, 8, 8, 8, 9]
```

# ตัวอย่าง: พิ่งก์ชันตรวจสอบข้อมูลซ้ำกันในลิสต์

```
def has_duplicate( x ):  
    s = set( x )  
    return len(s) != len(x)
```

```
def has_duplicate( x ):  
    s = set()  
    for e in x:  
        if e in s:  
            return True  
        s.add(e)  
    return False
```

## ตัวอย่าง: หาสองจำนวนต่างกันที่รวมกันได้ k

```
x = [int(e) for e in input().split()]
d = set(x)
k = int(input())
soln = []
for e in d:
    e1 = k - e
    if e < e1 and e1 in d:
        soln.append((e, e1))
if len(soln)==0:
    print("Not found")
else:
    print(soln)
```

```
1 2 3 4 8 9 -8 -2
11
```

```
[ (2, 9), (3, 8) ]
```

# ตัวอย่าง: Sieve of Eratosthenes

```
S1 = { 2, 3, 5, 7,  
       11, 13, 17, 19,  
       23, 29,  
       31, 37,  
       41, 43, 47, }
```

จำนวนเฉพาะ  
ทุกตัวที่มีค่า<sup>\*</sup>  
ไม่เกิน N

```
S1 = set(range(2, N+1))  
for i in range(2, int(N**0.5) + 1):  
    S2 = set(range(2*i, N+1, i))  
    S1 = S1 - S2
```

S2

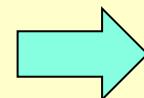
```
{ 4, 6, 8, 10, ..., 50}  
{ 6, 9, 12, 15, ..., 48}  
{ 8, 12, 16, 20, ..., 48}  
{10, 15, 20, 25, ..., 50}  
{12, 18, 24, 30, ..., 48}  
{14, 21, 28, 35, ..., 49}
```

## แบบฝึกหัด: Winner

ให้เขียนโปรแกรมเพื่อรับผลการแข่งขันฟุตบอล  
จากนั้นให้หาว่าทีมใดบ้างที่ไม่เคยแพ้เลย

5

Chelsea Liverpool  
ManU Liverpool  
Liverpool ManU  
Chelsea Arsenal  
Everton ManCity



```
[ 'Chelsea' , 'Everton' ]
```

# List vs. Dict

เก็บข้อมูลที่มีความสัมพันธ์กันได้ด้วย list หรือ dict

```
x = [ "6130102321", "6130238221", "6031022121" ]
p = [ [10, 9, 10], [9, 10, 8], [5, 8, 7] ]
```

```
x = [ [6130102321, [10, 9, 10]],
      [6130238221, [ 9, 10, 8]],
      [6031022121, [ 5, 8, 7]] ]
```

```
d = { "6130102321": [10, 9, 10],
      "6130238221": [ 9, 10, 8],
      "6031022121": [ 5, 8, 7] }
```

# List vs. Dict

```
x = [ "6130102321", "6130238221", "6031022121" ]
p = [ [10, 9, 10], [9, 10, 8], [5, 8, 7] ]
```

```
ID = input()
if ID in x:
    print( p[x.index(ID)] )
```

 ช้า

# `x.sort()` จะเรียงแต่ในลิสต์ x, ข้อมูลใน p จะไม่ตรงตาม x

```
t = []
for i in range(len(x)):
    t.append([x[i],p[i]])
t.sort()
for id, sc in t:
    print(id, sc)
```

 ยุ่ง

# List vs. Dict

```
x = [ [6130102321, [10, 9, 10]],  
      [6130238221, [ 9, 10, 8]],  
      [6031022121, [ 5, 8, 7]] ]
```

```
ID = input()  
for sid, scores in x:      # ค้น ID ต้องไล่ค้นเอง  
    if sid == ID:  
        print( scores )  
        break
```

 ช้า, ยุ่ง

```
x.sort()  # เรียงตาม ID  
for sid, sc in x:  
    print( sc )
```

 สะดวก

# List vs. Dict

```
d = { "6130102321": [10, 9, 10],  
      "6130238221": [ 9, 10, 8],  
      "6031022121": [ 5, 8, 7] }
```

```
ID = input()  
if ID in d:  
    print( d[ID] )
```



ເຮົາ

```
# ຕ້ອງການເຮັດວຽກ  
for id in sorted(d):  
    print( id, d[id] )
```



ສະດວກ

# แบบฝึกหัด: เรียงประเภทเพลงตามเวลารวม

## Input

9

Shake It Off, Taylor Swift, Pop, 3:39  
Rolling In The Deep, Adele, Pop, 3:48  
Chandelier, Sia, Pop, 3:36  
Roar, Katy Perry, Pop, 3:42  
Hotel California, Eagle, Rock, 6:30  
We Are the Champions, Queen, Rock, 2:59  
Hello Dolly, Louis Armstrong, Jazz, 2:27  
Bohemian Rhapsody, Queen, Rock, 5:55  
Coward of the County, Kenny Rogers, Country, 4:20

## Output

Rock --> 15:24  
Pop --> 14:45  
Country --> 4:20

แสดงประเภทเพลง ตามด้วย  
เวลารวมเป็นนาทีและวินาที  
เรียงตามลำดับเวลารวม 3  
อันดับแรกจากมากน้อย

# More on Dict

ให้ value ของ dict เป็น list, set หรือ dict ก็ได้

```
{  
    'Hello': ['Adele', 'Lionel Richie', 'Prince'],  
  
    'Shake It Off': ['Taylor Swift'],  
  
    'Chandelier': {'Sia'},  
                  {str: set}  
  
    "You've got a Friend": {'Carol King',  
                            'James Taylor'},  
  
    'What a Wonderful World': {'Anne Murray',  
                             'Louis Armstrong',  
                             'Rod Stewart'},  
}  
}
```

## ตัวอย่าง: dict

```
birthdate = {  
    "6130192221": (31,12,2000),  
    "6131022521": (28,2,2000),  
    "6230012121": (3,4,2001)  
}  
  
ID = input().strip()  
print("Birth date of", ID, "is", birthdate[ID])  
print("Birth year of", ID, "is", birthdate[ID][2])
```

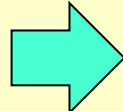
{str: tuple}

```
series = {  
    "Ranee": {"Roy Marn", "Plerng Boon"},  
    "Urassaya": {"Maya Tawan", "Kleun Cheewit"}  
}  
  
name = input().strip()  
print(name, "starred in", ", ".join(series[name]))
```

{str: set}

# ตัวอย่าง: ตัวการ์ตูน

Ted, bear  
Pongo, dog  
Fozzie, bear  
Winnie-the-Pooh, bear  
Nana, dog  
Scooby Doo, dog  
Garfield, cat  
Yogi, bear  
Tom, cat  
Sylvester, cat  
Figaro, cat  
Pluto, dog  
Baloo, bear  
Goofy, dog  
Felix, cat  
q



```
{ 'bear':  
    { 'Ted', 'Fozzie'  
     'Winnie-the-Pooh',  
     'Yogi', 'Baloo' },  
  'dog':  
    { 'Pongo', 'Nana',  
     'Scooby Doo',  
     'Pluto', 'Goofy' },  
  'cat':  
    { 'Garfield', 'Tom',  
     'Sylvester',  
     'Figaro', 'Felix' } }
```

```
{ str: set }
```

# ตัวอย่าง: ตัวการ์ตูน

```
cartoon = {}  
x = input()  
while x != 'q':  
    name, atype = x.split(", ")  
    if atype not in cartoon:  
        cartoon[atype] = {name} ←  
    else:  
        cartoon[atype].add(name)  
    x = input()  
print(cartoon)
```

{ str: set }

อย่าเขียน set(name)

# แบบฝึกหัด: แสดงข้อมูลการ์ตูน ตามลำดับที่อ่านเข้ามา

Ted, bear  
Pongo, dog  
Fozzie, bear  
Winnie-the-Pooh, bear  
Nana, dog  
~~Hello Kitty, cat~~  
Scooby Doo, dog  
Garfield, cat  
Yogi, bear  
Tom, cat  
Sylvester, cat  
Pluto, dog  
Goofy, dog  
q

เรียงก่อนหลัง  
ตามที่อ่านเข้ามา

เรียงก่อนหลัง  
ตามที่อ่าน  
เข้ามา

bear: Ted, Fozzie, Winnie-the-Pooh, Yogi  
dog: Pongo, Nana, Scooby Doo, Pluto, Goofy  
cat: ~~Hello Kitty~~, Garfield, Tom, Sylvester

# More on Dict: keys(), values(), items()

```
D = { "Vios": "Toyota", "Fortuner": "Toyota",  
      "Wave": "Honda", "Civic": "Honda" }
```

```
for k in D.keys():  
    print(k)
```

```
for v in D.values()  
    print(v)
```

```
for k,v in D.items():  
    print(k, v)
```

Civic
Fortuner
Vios
Wave
Honda
Toyota
Toyota
Honda
Civic Honda
Fortuner
Toyota
Vios Toyota
Wave Honda

# ตัวอย่าง: reverse mapping

```
def reverse( d ): # กรณีที่ value ไม่ซ้ำกัน
    r = { }
    for k,v in d.items():
        r[v] = k
    return r
```

```
if v not in r:
    r[v] = {k}
else:
    r[v].add(k)
```

```
def reverse( d ): # กรณีที่ value อาจซ้ำกัน
    r = { }
    for k,v in d.items():
        if v not in r:
            r[v] = set()
            r[v].add(k)
    return r
```

```
d { "Vios": "Toyota", "Fortuner": "Toyota",
    "Wave": "Honda", "Civic": "Honda" }
```

```
r { "Toyota": {"Vios", "Fortuner"}, 
    "Honda": {"Wave", "Civic" } }
```

# แบบฝึกหัด: ครรลองเพลงนี้

11

Input

Hello, Adele

Shake It Off, Taylor Swift

Chandelier, Sia

You've got a Friend, Carol King

Hello, Lionel Richie

What a Wonderful World, Anne Murray

Hello, Prince

What a Wonderful World, Louis Armstrong

You've got a Friend, James Taylor

What a Wonderful World, Rod Stewart

Hello, Sai Wa Si Bor Tim Gun, You've got a Friend

Hello -> Adele, Lionel Richie, Prince

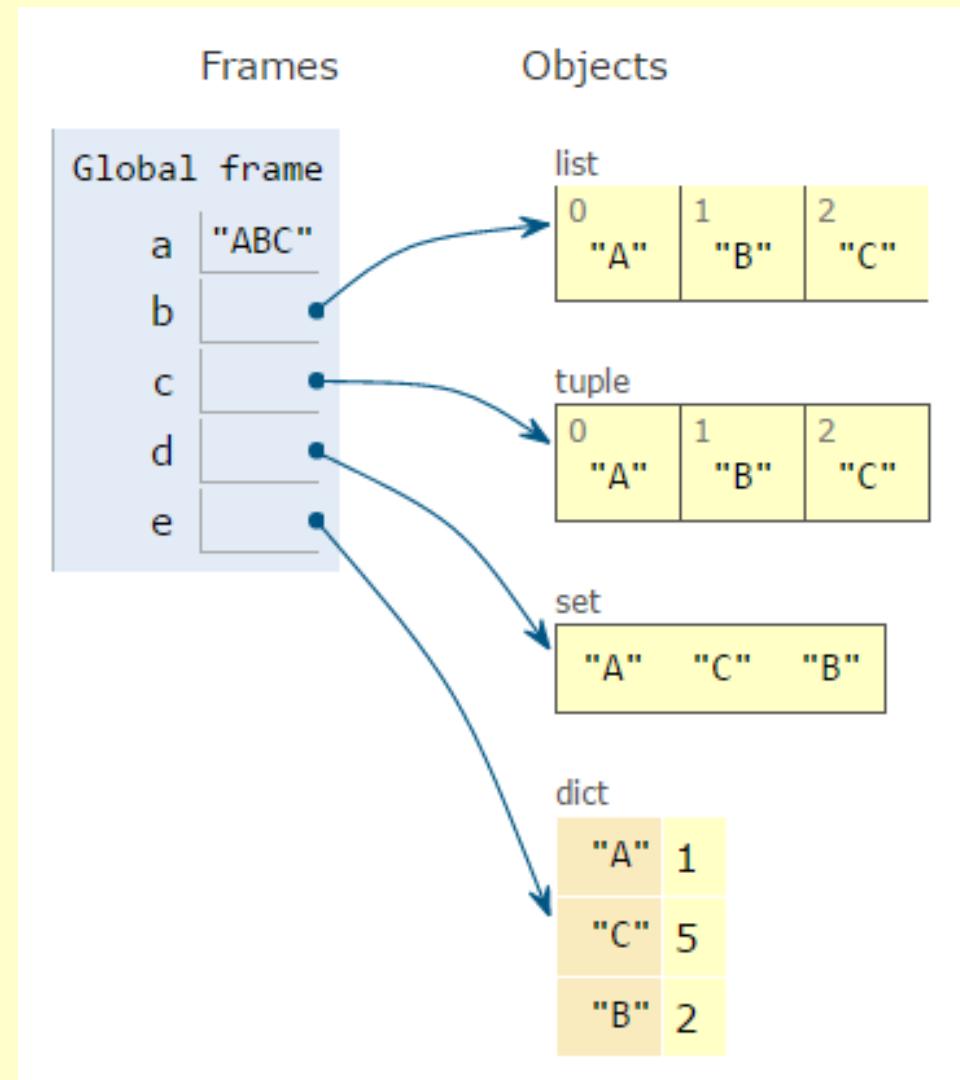
Output

Sai Wa Si Bor Tim Gun -> Not found

You've got a Friend -> Carol King, James Taylor

# สรุป string, list, tuple, set, dict

```
1 a="ABC"  
2 b= ["A", "B", "C"]  
3 c=( "A", "B", "C")  
4 d={ "A", "B", "C"}  
→ 5 e={"A":1, "B":2, "C":5}
```



# สรุปการใช้งาน list, tuple, dict, set

	<b>list</b>	<b>tuple</b>	<b>dict</b>	<b>set</b>
การใช้	<ul style="list-style-type: none"> <li>- ลำดับของข้อมูลมีความหมาย อาจมีการเปลี่ยนแปลง</li> <li>- ข้อมูลในรายการ มักมีความหมายเดียวกัน</li> </ul>	<ul style="list-style-type: none"> <li>- ลำดับของข้อมูลมีความหมาย</li> <li>- สร้างแล้วไม่เปลี่ยนแปลง</li> <li>- ข้อมูลใน tuple มักมีความหมายต่างกัน</li> </ul>	<ul style="list-style-type: none"> <li>- เก็บข้อมูลเป็นคู่ๆ key-value โดยใช้ key เช้าถึงข้อมูลเพื่อให้ได้ value มาใช้งาน</li> </ul>	<ul style="list-style-type: none"> <li>- เก็บข้อมูลไม่ซ้ำ ลำดับของข้อมูลไม่มีความหมาย เพื่อตรวจสอบว่า มีข้อมูลหรือไม่ รองรับ set operations</li> </ul>
การเข้าใช้ข้อมูล	ใช้จำนวนเต็มระบุตำแหน่ง <code>d[i]</code>	ใช้จำนวนเต็มระบุ <code>d[i]</code>	ใช้ key เป็นตัวระบุตำแหน่งข้อมูล <code>d[key]</code>	ต้อง <code>for...in...</code> เพื่อแจงข้อมูล
การค้นด้วย <code>in</code>	ค้นจากซ้ายไปขวา ช้า	ค้นจากซ้ายไปขวา ช้า	มีวิธีค้นที่เร็วมาก	มีวิธีค้นที่เร็วมาก
การสร้าง	<code>x = [1, 2, 3, 4]</code>	<code>t = (1, 2, 3, 4)</code>	<code>d = { "k1":1, "k2":2 }</code>	<code>s = {1, 2, 3, 4}</code>
การเพิ่มข้อมูล	<code>x.append(3)</code> <code>x.insert(1, 99)</code>	สร้างแล้วเปลี่ยนแปลงไม่ได้ ต้องสร้างใหม่ <code>t = t + (4, )</code>	<code>d["k1"] = 1</code> <code>d["k2"] = 2</code>	<code>s.add(3)</code>

# NumPy

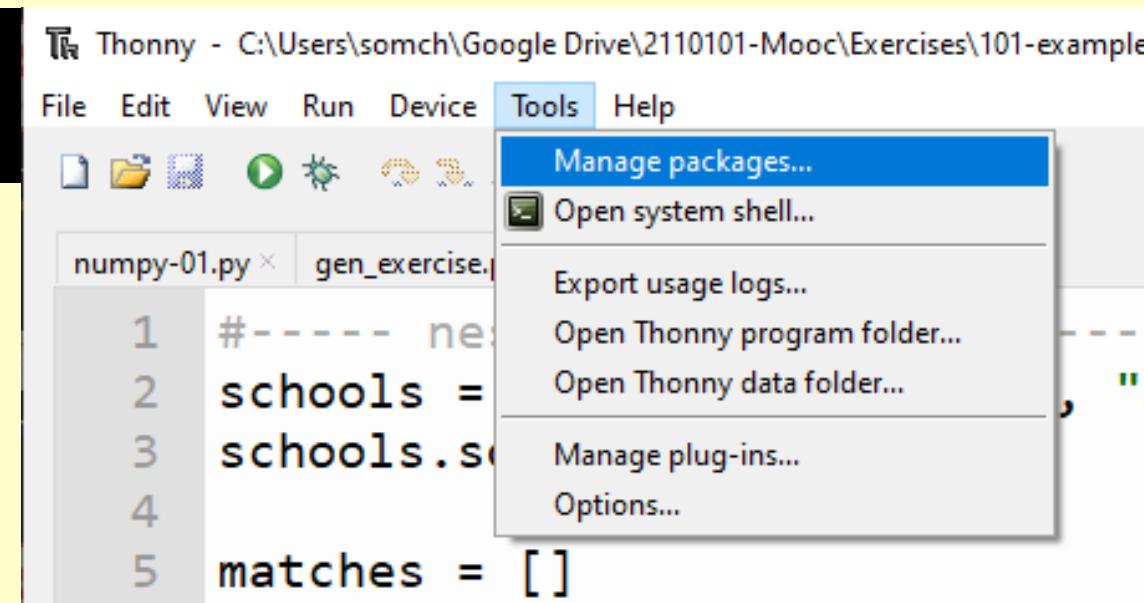
ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

# NumPy

- ชุดคำสั่งเพื่อการประมวลผลในงานทางวิทยาศาสตร์
- เขียนง่าย สั้น และทำงานเร็วมาก
- <https://www.numpy.org/>
- ไม่ได้มา กับ Python ต้องติดตั้งเพิ่ม

```
C:\>pip install numpy
```



# การเก็บข้อมูล: NumPy Array

Array: คือการเก็บข้อมูลเป็นแ阁ล์ดับเรียงกันไป  
คล้ายลิสต์ แต่มีข้อแตกต่าง เช่น

- ทุกช่องในอาร์เรย์เก็บข้อมูลเดียวกันหมด (มักเก็บจำนวน)
- เก็บข้อมูลได้หลายมิติ
  - 1 มิติ : vector  $[1, 2, 3, 4]$
  - 2 มิติ : matrix  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
  - n มิติ : tensor  $\begin{bmatrix} [1 & 0], [1 & 2] \\ [0 & 1], [3 & 4] \end{bmatrix}$
- ใช้ tuple เป็น index เพื่อใช้ข้อมูลในอาร์เรย์  
เช่น  $a$  เป็นอาร์เรย์ 2 มิติ  
เขียน  $a[(1, 2)]$  หรือจะเขียน  $a[1, 2]$  ก็ได้
- มี operators และ methods ให้ใช้งานมากมาย

# List vs. NumPy Array: ระยะทางทุกคู่จุด

```
def all_pair_distances(points) :  
    # points เป็น nested list เช่น [[0,0],[0,3],[4,0]]  
    n = len(points)  
    D = [[0.0]*n for i in range(n)]  
    for i in range(n) :  
        for j in range(i+1, n) :  
            dx = points[i][0] - point[j][0]  
            dy = points[i][1] - point[j][1]  
            D[i][j] = D[j][i] = (dx**2 + dy**2)**0.5  
    return D
```

List

```
def all_pair_distances(points) :
```

```
    # points เป็น NumPy array
```

```
    n = len(points)
```

```
    X = points[:, 0]
```

```
    Y = points[:, 1]
```

```
    dX = X - X.reshape(n,1)
```

```
    dY = Y - Y.reshape(n,1)
```

```
    D = (dX**2 + dY**2)**0.5
```

```
    return D
```

NumPy Array

การทดลอง n = 2000

List: 5.44 s.

NumPy: 0.337 s.

เข้าใจง่ายกว่า &  
ทำงานเร็วกว่ามาก

# NumPy Array ใน 2110101 (นิดเดียว)

การสร้างอาร์เรย์แบบต่าง ๆ

indexing

element-wise operations

broadcasting

ฟังก์ชันที่นำเสนอ

(sum, min, max, argmin, argmax, mean, std, dot)

# การสร้างอาร์เรย์

```
import numpy as np

a = np.array([1,2,3,4])                      # สร้างจากลิสต์
b = np.array([[1,2],[3,4]],float) # สร้างจากลิสต์
c = np.ndarray( (2,3) )                      # สร้างตามขนาด ค่าไม่รู้
d = np.ndarray( (2,3) , int)
e = np.zeros( (2,3) , int)                    # สร้างตามขนาด ค่า 0 หมด
f = np.ones( (2,3) , int)                     # สร้างตามขนาด ค่า 1 หมด
g = np.zeros_like (f, float) # ขนาดเหมือน f ค่า 0 หมด
h = np.ones_like( e, float) # ขนาดเหมือน e ค่า 1 หมด
I = np.identity( 4 , int) # identity matrix ขนาด 4x4
x = np.arange(0.0, 1.0, 0.1) # [0.0, 0.1, 0.2, ..., 0.9]
```

int ก็ได้, float ก็ได้ (range ได้แค่ int)

# array.shape

- array.shape คืน tuple บอกรายละเอียดของมิติ
  - `a = np.ones( (3, 4) )`  
จะได้ `a.shape` เป็น `(3, 4)`
    - `a.shape[0]` คือ 3 เป็นจำนวนแถว
    - `a.shape[1]` คือ 4 เป็นจำนวนคอลัมน์
- `len( array.shape )` เป็นขนาดของมิติ
  - `a = np.ones( (3, 4) )`  
จะได้ `len( a.shape )` เป็น 2

# array.reshape( newshape )

นำข้อมูลใน array มาจัดรูปแบบให้ตรงตาม shape

```
a = np.arange(8)          # [0 1 2 3 4 5 6 7]
b = a.reshape((2,4))      # [[0 1 2 3],
                          #  [4 5 6 7]]
c = b.reshape((4,2))      # [[0 1],
                          #  [2 3],
                          #  [4 5],
                          #  [6 7]]
d = c.reshape(8)          # [0 1 2 3 4 5 6 7]
d = c.reshape((2,3))      # ทำไม่ได้
```

# array.T

- $a.T$  คือ transpose ของอาร์เรย์  $a$
- $a$  มี 1 มิติ,  $a.T$  เหมือน  $a$
- $a$  มี 2 มิติ,  $a.T$  คือ transpose ของเมทริกซ์  $a$

```
a = np.arange(8)  
print(a)  
print(a.T)  
b = a.reshape((2, 4))  
print(b)  
print(b.T)  
c = a.reshape((1, 8))  
print(c)  
print(c.T)
```

[0 1 2 3 4 5 6 7]
[0 1 2 3 4 5 6 7]
[[0 1 2 3]]
[4 5 6 7]]
[[0 4]]
[1 5]]
[2 6]]
[3 7]]
[[0 1 2 3 4 5 6 7]]

[[0]]
[1]]
[2]]
[3]]
[4]]
[5]]
[6]]
[7]]

ข้อสังเกต:  $a$  ไม่เหมือน  $c$  (shape ไม่เหมือน)

# Indexing

ใช้ tuple ระบุตำแหน่งในอาร์เรย์

```
import numpy as np

def count_ones( A ):
    c = 0
    for i in range( A.shape[0] ):
        for j in range( A.shape[1] ):
            if A[i,j] == 1:
                c += 1
    return c
```

เขียน  $A[i, j]$  เหมือนกับ  $A[(i, j)]$

เดี๋ยวจะรู้ว่าการนับจำนวน 1 ใน  $A$  เขียน  $\text{np.sum}(A==1)$  ก็พอ

# Slicing: start : stop : step

- รูปแบบ: A[ เลือกแถว , เลือกคอลัมน์ ]
- ระวัง: A[ เลือกแถว ][ ตรงนี้ไม่ใช่เลือกคอลัมน์ ]

```
a = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]  
print( a[::-2] )                      # [[1,2,3], [7,8,9]]  
print( a[::-2][::-2] )                # [[1,2,3]]  
  
A = np.array(a)  
print( A[::-2] )                      # [[1 2 3]  
                                # [7 8 9]]  
print( A[::-2][::-2] )                # [[1 2 3]]  
print( A[::-2, ::2] )                # [[1 3]  
เลือกแถวคู่ คอลัมน์คู่      # [7 9]]  
print( A[::-1, ::-1] )               # [[12 11 10]  
                                # [ 9 8 7]  
                                # [ 6 5 4]  
                                # [ 3 2 1]]
```

# Fancy Indexing

```
# a = [0,10,20,30,40,50,60,70,80,90]
a = np.arange(0, 100, 10)
b = a[0::2]                      # b = [0 20 40 60 80]
c = a[ [8,1,9,0] ]               # c = [80 10 90 0]
d = c[ [True,False,False,True] ] # d=[80 0]

A = np.array([[1,2,3],[4,5,6],[7,8,9],[0,1,0]])
B = A[ [1,3,2], [2,0,1] ]
#
#           A[1,2] ,  A[3,0] ,  A[2,1]      B = [6 0 8]
```

## แบบฝึกหัด

```
# A is a 2-d array
def get_column_from_bottom_to_top( A, c ):
    return _____ # บรรทัดเดียว

def get_odd_rows( A ):
    return _____ # บรรทัดเดียว

def get_even_rows_last_column( A ):
    return _____ # บรรทัดเดียว

def get_diagonal1( A ): # A is a square matrix
    _____
    return _____ # ส่องบรรทัด

def get_diagonal2( A ): # A is a square matrix
    _____
    return _____ # ส่องบรรทัด
```

# การนำค่าสเกลาร์ใส่ในอาร์เรย์

ใส่ค่าสเกลาร์ให้กับทุกช่องทางซ้ายของ =

```
A = np.zeros(8)
```

```
A[2:5] = 9
```

```
[0 0 9 9 9 0 0 0]
```

```
A = np.ndarray((4,4), int)
```

```
A[:, :] = 9
```

นำ 9 ใส่ทุกແղວ, ทุกคอลัมน์

$$\begin{bmatrix} 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \end{bmatrix}$$

```
B = np.zeros((4,4), int)
```

```
B[:, 0::2] = 1
```

```
B[1::2, :] = 2
```

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

ใส่ 1 ในคอลัมน์คู่ของทุกແղວ

ใส่ 2 ในແղວคี่ของทุกคอลัมน์

# การคำนวณแต่ละค่าในอาร์เรย์กับค่าสเกลาร์

คำนวณให้ตัวต่อตัว และคืนผลเป็นอาร์เรย์

```
a = np.array([1, 2, 3, 4, 5])
b = a + 1      # [2 3 4 5 6]
c = a**2 + 1   # [2 5 10 17 26]
d = a/2        # [0.5 1.0 1.5 2.0 2.5]
```

```
def toCM( inches ):
    return inches * 2.54
```

```
d = np.array([0, 10, 12, 100])
print(toCM(d))
```

```
[ 0. 25.4 30.48 254. ]
```

## ห/library ที่มีใน math มีใน numpy ด้วย

```
a = np.array( [10, 100, 1000, 10000] )
b = np.log10(a)    # [1., 2., 3., 4.]

c = np.array( [np.pi, 2*np.pi, 3*np.pi] )
d = np.sin(c/2)
# [ 1.000000e+00,  1.2246468e-16, -1.000000e+00]
```

# การเปรียบเทียบค่าในอาร์เรย์กับสเกลาร์

เปรียบเทียบให้ตัวต่อตัว และคืนผลเป็นอาร์เรย์ True/False

```
a = np.array( [1, 2, 3, 4] )
b = a > 3      # ได้ [False False False True]
c = a%2 == 1   # ได้ [True False True False]
```

ต้องการนับจำนวนเลขคี่ในอาร์เรย์ a

```
def count_odds( a ):
    return sum( a%2 == 1 )

def get_odds( a ):
    return a[ a%2 == 1 ] # เลือกเฉพาะช่องที่ True

def get_odd_positions( a ):
    pos = np.arange(a.shape[0])
    return pos[ a%2 == 1 ]
```

ใน Python  
True มีค่า 1  
False มีค่า 0

## แบบฝึกหัด

```
def toCelsius( f ):  
    # f = [ temperature in Fahrenheit, ... ]
```

```
def BMI( wh ):  
    # [[w1,h1], [w2,h2], ...]
```

```
def distanceTo( P, p )  
    # distance from p to all points in P
```

# แบบฝึกหัด: Logistic Regression

สูตรทำนายโอกาส  $p(x)$  ที่นักเรียน  $x$  เรียนผ่านวิชานึง  
จากจำนวนโจทย์ที่ทำ ( $x_0$ ) กับเกรดเฉลี่ยที่มี ( $x_1$ )

$$p(x) = \frac{1}{1 + e^{-logit(x)}}$$

$$logit(x) = -3.98 + 0.1x_0 + 0.5x_1$$

จงเขียนโปรแกรมอ่านจำนวนโจทย์และเกรดเฉลี่ยของ  
นักเรียนกลุ่มนึง เพื่อคำนวณผลการทำนาย

# Element-Wise Operations

```
x = [1,2,3]
y = [4,5,6]
z = x + y      # concatenation → [1,2,3,4,5,6]
```

```
u = np.array([1,2,3])
v = np.array([4,5,6])
w = u + v      # element-wise addition
                # [1+4  2+5  3+6] = [5  7  9]
```

```
A = np.array([[1,2,3], [4,5,6], [7,8,9]])
I = np.identity(A.shape[0],int)
B = I*A        # element-wise multiplication
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

# Element-Wise Logical Operators

~~[ True, True, False, False ] and [ False, True, True, False ]~~

~~[ True, True, False, False ] or [ False, True, True, False ]~~

~~not [ False, True, True, False ]~~

[ True True False False ] & [ False True True False ]

[ True True False False ] | [ False True True False ]

~ [ False True True False ]

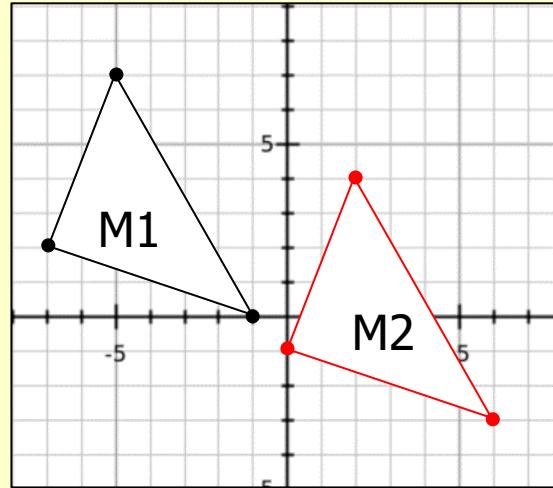
ต้องใช้เครื่องหมาย &, |, ~ สำหรับการทำ and, or, not แบบตัวต่อตัว

# Element-Wise Logical Operators

```
a = np.array( [9, 3, 0, 2, 6] )
b = a[ a < 5 ]
b = a[ [False, True, True, True, False] ]
# [            3,      0,      2          ]
b = a[ 2 < a < 5 ]           # ผิด
b = a[ 2 < a and a < 5 ]     # ผิด
b = a[ 2 < a   &   a < 5 ]    # ผิด
b = a[ (2 < a) & (a < 5) ]   # Ok
#a[[T,T,F,F,T] & [F,T,T,T,F]]
#a[  [F,T,F,F,F]  ]
# [3]
```

# ตัวอย่าง : Matrix Translation

x y  
[ [-7 2]  
[-5 7]  
[-1 0]]



[ [ 0 -1]  
[ 2 4]  
[ 6 -3]]

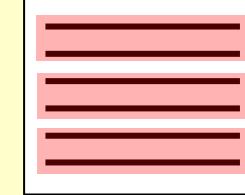
ต้องการย้ายทุกจุดไปทางขวา 7, ลงล่าง 3 คือบวกทุกจุดด้วย [7, -3]

```
M1 = np.array([ [-7,2], [-5,7], [-1,0] ])
T  = np.array([ [7,-3], [7,-3], [7,-3] ])
M2 = M1 + T
```

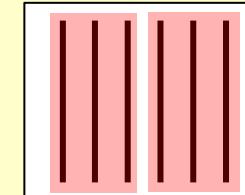
$$\begin{bmatrix} [-7 & 2] \\ [-5 & 7] \\ [-1 & 0] \end{bmatrix} + \begin{bmatrix} [7 & -3] \\ [7 & -3] \\ [7 & -3] \end{bmatrix} \text{ ได้ } \begin{bmatrix} [0 & -1] \\ [2 & 4] \\ [6 & -3] \end{bmatrix}$$

# แบบฝึกหัด

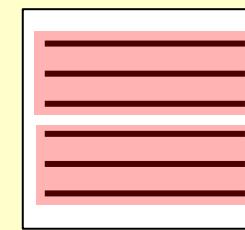
```
def sum_2_rows( M ):
```



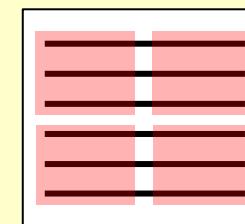
```
def sum_left_right( M ):
```



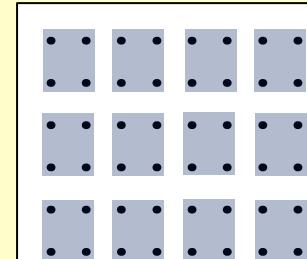
```
def sum_upper_lower( M ):
```



```
def sum_4_quadrants( M ):
```



```
def sum_4_cells( M ):
```



# Broadcasting

- เมื่อนำมาเรย์ 2 ตัวมาคำนวณแบบ element-wise แต่อาร์เรย์ทั้งสองมีขนาดไม่เท่ากัน
- ระบบจะ broadcast อาร์เรย์ตัวเล็ก (หรืออาจทำทั้งสองตัว) ให้มีขนาดเท่ากันก่อนทำงาน

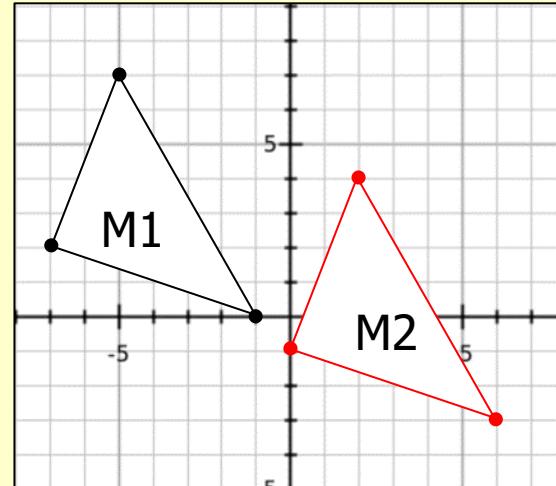
$$\begin{bmatrix} 2 & 3 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 2 & 3 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

- แต่บางครั้งก็ broadcast ไม่ได้

$$\begin{bmatrix} 2 & 3 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

# ตัวอย่าง: Matrix Translation

x y  
[ [-7 2]  
[-5 7]  
[-1 0]]



[ [ 0 -1]  
[ 2 4]  
[ 6 -3]]

ต้องการย้ายทุกจุดไปทางขวา 7, ลงล่าง 3 คือบวกทุกจุดด้วย [7, -3]

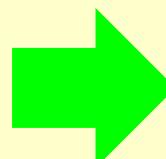
M1 กับ T  
มีขนาดเท่ากัน

```
M1 = np.array([ [-7,2], [-5,7], [-1,0] ])
T = np.array([ [7,-3], [7,-3], [7,-3] ])
M2 = M1 + T
```

เขียนแค่นี้ก็พอ

```
M2 = M1 + np.array( [7, -3] )
```

$$\begin{bmatrix} [-7 & 2] \\ [-5 & 7] \\ [-1 & 0] \end{bmatrix} + [7, -3]$$

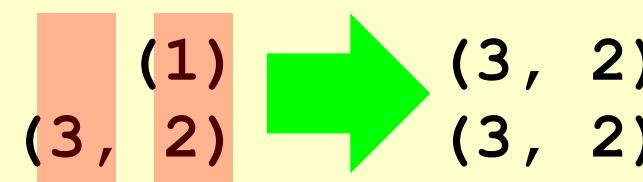


$$\begin{bmatrix} [-7 & 2] \\ [-5 & 7] \\ [-1 & 0] \end{bmatrix} + \begin{bmatrix} 7 & -3 \\ 7 & -3 \\ 7 & -3 \end{bmatrix}$$

# ตัวอย่าง: broadcast ตัวเล็ก ให้เท่าตัวใหญ่

```
x = np.array([[1,2], [3,4], [5,6]])  
u = np.array([2]) + x
```

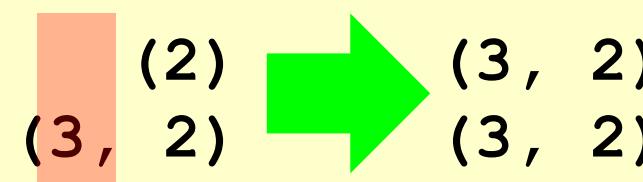
$$\begin{bmatrix} 2+1 & 2+2 \\ 2+3 & 2+4 \\ 2+5 & 2+6 \end{bmatrix}$$



# ตัวอย่าง: broadcast ตัวเล็ก ให้เท่าตัวใหญ่

```
x = np.array([[1,2],[3,4],[5,6]])  
w = np.array([10 20]) + x
```

$$\begin{bmatrix} 10 + 1 & 20 + 2 \\ 10 + 3 & 20 + 4 \\ 10 + 5 & 20 + 6 \end{bmatrix}$$



# ตัวอย่าง: broadcast ตัวเล็ก ให้เท่าตัวใหญ่

```
x = np.array([[1,2],[3,4],[5,6]])
v = np.array([[10],[20],[30]]) + x
```

$$\begin{bmatrix} 10 + 1 & 10 + 2 \\ 20 + 3 & 20 + 4 \\ 30 + 5 & 30 + 6 \end{bmatrix}$$

$$\begin{matrix} (3, 1) \\ (3, 2) \end{matrix} \xrightarrow{\hspace{1cm}} \begin{matrix} (3, 2) \\ (3, 2) \end{matrix}$$

## ตัวอย่าง: broadcast ทั้ง 2 ตัว

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + [4 \quad 5] \rightarrow \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix} + [4 \quad 5] \rightarrow \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix} + \begin{bmatrix} 4 & 5 \\ 4 & 5 \\ 4 & 5 \end{bmatrix}$$

$$(3, \begin{matrix} 1 \\ 2 \end{matrix})$$

$$(3, \begin{matrix} 2 \\ 2 \end{matrix})$$

$$(3, \begin{matrix} 2 \\ 2 \end{matrix})$$

# แบบฝึกหัด: Outer Product

จงเขียนโปรแกรมสร้างอาร์เรย์ที่เก็บสูตรคูณแม่ 1 ถึง 12 ข้างล่างนี้  
ด้วยคำสั่ง NumPy โดยไม่ต้องใช้วงวน

```
[ [ 1   2   3   4   5   6   7   8   9   10  11  12]
  [ 2   4   6   8   10  12  14  16  18  20  22  24]
  [ 3   6   9   12  15  18  21  24  27  30  33  36]
  [ 4   8   12  16  20  24  28  32  36  40  44  48]
  [ 5   10  15  20  25  30  35  40  45  50  55  60]
  [ 6   12  18  24  30  36  42  48  54  60  66  72]
  [ 7   14  21  28  35  42  49  56  63  70  77  84]
  [ 8   16  24  32  40  48  56  64  72  80  88  96]
  [ 9   18  27  36  45  54  63  72  81  90  99  108]
  [ 10  20  30  40  50  60  70  80  90  100 110 120]
  [ 11  22  33  44  55  66  77  88  99  110 121 132]
  [ 12  24  36  48  60  72  84  96  108 120 132 144] ]
```

# ฟังก์ชันที่นำเสนอของ NumPy

- np.sum
- np.max, np.argmax
- np.min, np.argmin
- np.mean, np.std
- np.dot

## np.sum

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 9 & 8 & 7 & 6 \\ 11 & 12 & 13 & 14 & 15 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$$

sum    sum    sum    sum    sum      np.sum(a, axis=1)

$$\begin{bmatrix} 42 \\ 42 \\ 42 \\ 42 \\ 42 \end{bmatrix}$$

np.sum(a, axis=0)

np.sum(a) ของทั้งหมดได้ 210

## np.min

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 9 & 8 & 7 & 6 \\ 11 & 12 & 13 & 14 & 15 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$$

min   min   min   min   min      np.min(a, axis=1)

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

np.min(a, axis=0)

np.min(a) ของทั้งหมดได้ 1

np.max ก็คล้าย np.min แต่ได้ค่ามากสุด

# np.argmin

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 9 & 8 & 7 & 6 \\ 11 & 12 & 13 & 14 & 15 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$$

min   min   min   min   min      np.argmin(a, axis=1)

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad [ \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad ]$$

np.argmin(a, axis=0)

np.argmin(a) ของทั้งหมดได้ 0

np.argmax ก็คล้าย np.argmin แต่ได้ตำแหน่งของค่ามากสุด

## np.mean

$$a = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 9 & 8 & 7 & 6 \\ 11 & 12 & 13 & 14 & 15 \\ 20 & 19 & 18 & 17 & 16 \end{bmatrix}$$

mean mean mean mean mean      np.mean(a, axis=1)

[ 10.5 10.5 10.5 10.5 10.5]

np.mean(a, axis=0)

np.mean(a) ของทั้งหมดได้ 10.5

np.std ก็คล้าย np.mean แต่ได้เบี่ยงเบนมาตรฐาน

## np.dot

- np.dot(vector, vector)

$$[1 \ 2 \ 3] \cdot [4 \ 5 \ 6] = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 2$$

- np.dot(vector, matrix)

$$\begin{aligned}[1 \ 2 \ 3] \cdot \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} &= [[1 \ 2 \ 3] \cdot [4 \ 5 \ 6] \ [1 \ 2 \ 3] \cdot [7 \ 8 \ 9]] \\ &= [32 \ 50]\end{aligned}$$

$$\begin{aligned}\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \cdot [2 \ 3] &= [[1 \ 2] \cdot [2 \ 3] \ [3 \ 4] \cdot [2 \ 3] \ [5 \ 6] \cdot [2 \ 3]] \\ &= [8 \ 18 \ 28]\end{aligned}$$

- np.dot(matrix, matrix) ก็คือการคูณ matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

# เขียน np.???(a,b) หรือ a.???(b) ได้

```
import numpy as np

x = np.array([[1,2,3],[4,5,6]])
y = np.array([[7,8],[9,10],[11,12]])

a = np.dot(x, y)
a = x.dot(y)

b = np.sum(x, axis=0)
b = x.sum(axis=0)

c = np.mean(x, axis=1)
c = x.mean(axis=1)
```

# ตัวอย่าง: รายได้รวมในสัปดาห์

ร้านขายอาหารตามสั่งมีราคาอาหารคือ

ข้าวแกง	25 บาท
ข้าวผัด	30 บาท
สุกี้ทะเล	45 บาท

ในสัปดาห์ที่ผ่านมาขายอาหารได้ดังนี้

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สุกี้ทะเล	50	45	70	65	50



$$[25 \ 30 \ 45]$$



$$\begin{bmatrix} 75 & 120 & 70 & 90 & 80 \\ 80 & 90 & 100 & 70 & 50 \\ 50 & 45 & 70 & 65 & 50 \end{bmatrix}$$

$$[25 \ 30 \ 45] \cdot \begin{bmatrix} 75 & 120 & 70 & 90 & 80 \\ 80 & 90 & 100 & 70 & 50 \\ 50 & 45 & 70 & 65 & 50 \end{bmatrix} = [6525 \ 7725 \ 7900 \ 7275 \ 5750]$$

daily income

weekly income

`np.sum( ... ) → 35175`

# รายงานรายได้ประจำสัปดาห์

ร้านขายอาหารตามสั่งมีราคาอาหารคือ<sup>1</sup>  
ข้าวแกง 25 บาท  
ข้าวผัด 30 บาท  
สก็อตติ่ง 45 บาท  
ในสัปดาห์ที่ผ่านมาขายอาหารได้ดังนี้

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สก็อตติ่ง	50	45	70	65	50

[6525 7725 7900 7275 5750]

MO --> 6525  
TU --> 7725  
WE --> 7900  
TH --> 7275  
FR --> 5750  
**weekly income = 35175**  
**daily average = 7035.0**  
**Best sales day = WE**  
**Sales loss on: MO, TH, FR**  
-----  
**Curry Rice --> 10875**  
**Fried Rice --> 11700**  
**Seafood Suki --> 12600**  
**Best menu = Seafood Suki**

ขาดทุนเมื่อ  
< 7500

`dailyincomes = np.dot(prices, dailysales)`

`weeklyincome = np.sum(dailyincomes)`

`dailyaverage = np.mean(dailyincomes)`

`best_day_index = np.argmax(dailyincomes)`

# รายงานรายได้ประจำสัปดาห์

```
def report(prices, dailysales, breakeven):
    days = ["MO", "TU", "WE", "TH", "FR"]
    menus = ["Curry Rice", "Fried Rice", "Seafood Suki"]

    dailyincomes = np.dot(prices, dailysales)
    for i in range(len(days)):
        print(days[i], '-->', dailyincomes[i])
    print("weekly income =", np.sum(dailyincomes))
    print("daily average =", np.mean(dailyincomes))
    print("Best sales day =", days[np.argmax(dailyincomes)])

    loss = np.array(days)[dailyincomes < breakeven]
    print("Sales loss on:", ", ".join(loss))
    print("-----")

    menuincomes = np.sum(dailysales, axis=1)*prices
    for i in range(len(menus)):
        print(menus[i], '-->', menuincomes[i])

    print("Best menu =", menus[np.argmax(menuincomes)])
```

## แบบฝึกหัด: โครงคะแนนรวมต่ำกว่าค่าคะแนนเฉลี่ย

	ID	Midterm	Final	Project
--	----	---------	-------	---------

```
data = np.array([[610011, 80, 90, 70],  
                [610022, 50, 80, 68],  
                [610033, 70, 85, 80],  
                [610044, 60, 50, 90],  
                [610055, 90, 74, 70]])  
  
weight = np.array([0.3, 0.5, 0.2])
```

$$\text{คะแนนรวมของ } 610011 = 0.3 \times 80 + 0.5 \times 90 + 0.2 \times 70 = 83.0$$

อยากรู้ว่าโครงบ้างที่คะแนนรวมต่ำกว่าค่าคะแนนเฉลี่ยของทั้งหมด

# Class / Object

ภาควิชาวิศวกรรมคอมพิวเตอร์  
จุฬาลงกรณ์มหาวิทยาลัย

# ประเภทข้อมูล

- ที่ผ่านมา ถ้าข้อมูลเป็น
  - จำนวน ใช้ int หรือ float
  - ข้อความ ใช้ str
  - จริง/เท็จ ใช้ boolean
  - กลุ่มข้อมูล ใช้ list, set, tuple, dict
  - ข้อมูลที่ประกอบด้วยข้อมูลย่อย ๆ ที่สัมพันธ์กัน
    - ใช้ list, tuple, dict

## หนังสือ

- ชื่อ
- หมายเลข ISBN
- ราคา
- สำนักพิมพ์
- ...

## บัตรประชาชน

- หมายเลข
- ชื่อ-สกุล
- ที่อยู่
- วันเกิด
- ...

# ใช้ tuple เก็บรายละเอียดต่าง ๆ ของหนังสือ 1 เล่ม

```
b1 = ("Data Science", "149190142X", 28.79)
b2 = ("Learning Python", "1449355730", 37.06)
b3 = ("Data Analysis", "1449319793", 27.68)

print(total_price( [b1, b2, b3] ))
```

```
def total_price( books ):
    s = 0
    for b in books:
        s += b[2]
    return s
```

**หนังสือ**

- ชื่อหนังสือ
- หมายเลข ISBN
- ราคา

```
def total_price( books ):
    s = 0
    for title, isbn, price in books:
        s += price
    return s
```

# ใช้ dict เก็บรายละเอียดต่าง ๆ ของหนังสือ 1 เล่ม

```
b1 = {"title": "Data Science", "isbn": "1408142X", "price": 28.79}  
b2 = {"title": "Easy Python", "isbn": "14455730", "price": 37.06}  
b3 = {"title": "Big Data", "isbn": "14493793", "price": 27.68}  
  
print(total_price( [b1, b2, b3] ))
```

```
def total_price( books ):  
    s = 0  
    for b in books:  
        s += b["price"]  
    return s
```

## หนังสือ

- ชื่อหนังสือ
- หมายเลข ISBN
- ราคา

# อีกแบบ: ใช้ class สร้างประเภทข้อมูลใหม่

```
class Book:  
    pass  
  
    def init(b, title, isbn, price):  
        b.title = title  
        b.isbn = isbn  
        b.price = price  
  
    b1 = Book()  
    init(b1, "Data Science", "149142X", 28.79)
```



ไม่เขียนแบบนี้

หนังสือ

- ชื่อหนังสือ
- หมายเลข ISBN
- ราคา

คลาส (class) คือประเภทข้อมูล  
อ็อบเจกต์ (object) คือตัวข้อมูล

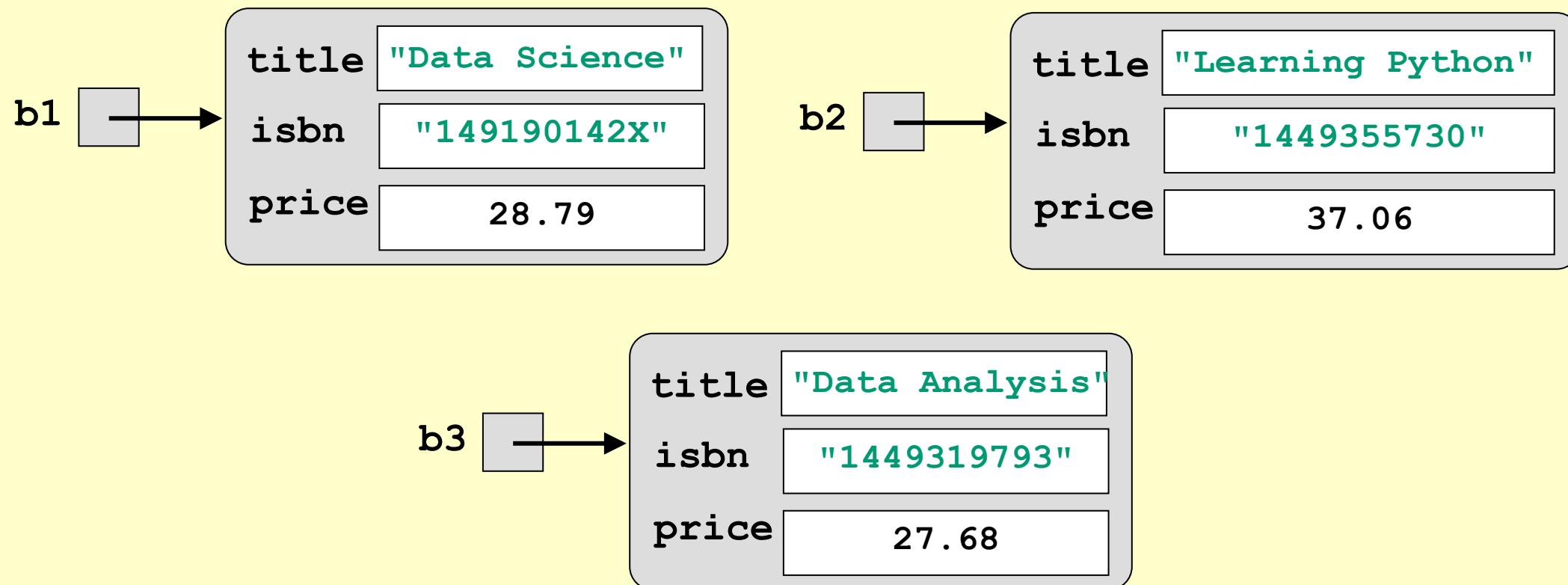
`b.title` คือ ตัวแปร  
`title` ของอ็อบเจกต์ `b`

`self` คือ อ็อบเจกต์  
ที่เพิ่งถูกสร้าง

```
class Book:  
    def __init__(self, title, isbn, price):  
        self.title = title  
        self.isbn = isbn  
        self.price = price  
  
    b1 = Book("Data Science", "149142X", 28.79)
```

# แต่ละอ้อมเจกต์มีตัวแปรประจำอ้อมเจกต์ของตัวเอง

```
b1 = Book ("Data Science", "149190142X", 28.79)  
b2 = Book ("Learning Python", "1449355730", 37.06)  
b3 = Book ("Data Analysis", "1449319793", 27.68)
```



# ตัวอย่าง : คลาสที่แทนประเภทข้อมูล

```
class Point:  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

```
p = Point(10, 23)
```

```
class Date:  
    def __init__(self, d, m, y):  
        self.day = d  
        self.month = m  
        self.year = y
```

```
d = Date(14, 2, 2562)
```

```
class Song:  
    def __init__(self, title, artist, lyrics):  
        self.title = title  
        self.artist = artist  
        self.lyrics = lyrics  
        self.nviews = 0
```

```
x = Song("Hello", "Adele", "")
```

# ตัวอย่าง : คลาสที่แทนประเภทข้อมูล

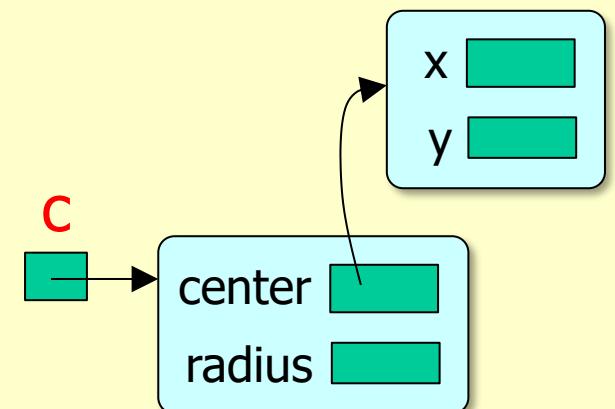
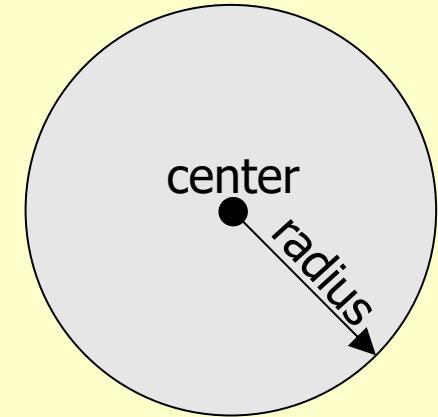
```
class BankAccount:  
    def __init__(self, acc_no, acc_name, balance):  
        self.acc_no = acc_no  
        self.acc_name = acc_name  
        self.balance = balance
```

```
class Rectangle:  
    def __init__(self, lower_left, w, h):  
        self.lower_left = lower_left  
        self.height = h  
        self.width = w
```

```
class Course:  
    def __init__(self, ID, name):  
        self.ID = ID  
        self.name = name  
        self.students = []
```

# การใช้ตัวแปรในอ้อมบเจกต์

```
class Point:  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
  
class Circle:  
    def __init__(self, p, r):  
        self.center = p  
        self.radius = r  
  
c = Circle( Point(20,30) , 100 )  
  
c.radius = 200  
  
c.center = Point(2, 4)  
  
c.center.x = 3      หน้าจุดเป็นอ้อมบเจกต์
```



อ้อมบเจกต์. ชื่อตัวแปรในอ้อมบเจกต์

# การใช้ตัวแปรในอ้อมบเจกต์

```
class Book:  
    def __init__(self, title, isbn, price):  
        self.title = title  
        self.isbn = isbn  
        self.price = price  
  
b1 = Book("Data Science", "149190142X", 28.79)  
  
print( b1.title )  
b1.price *= 0.80
```

อ้อมบเจกต์. ชื่อตัวแปรในอ้อมบเจกต์

# ตัวอย่างการใช้ตัวแปรในอ้อม JEK

```
class Book:  
    def __init__(self, title, isbn, price):  
        self.title = title  
        self.isbn = isbn  
        self.price = price  
  
    b1 = Book("Data Science", "149190142X", 28.79)  
    b2 = Book("Learning Python", "1449355730", 37.06)  
    b3 = Book("Data Analysis", "1449319793", 27.68)  
  
    print( total_price( [b1, b2, b3] ) )
```

```
def total_price( books ):  
    s = 0  
    for book in books:  
        s += book.price  
    return s
```

# ตัวอย่างการใช้ตัวแปรในอ้อม JEKET

```
class Book:  
    def __init__(self, title, isbn, price):  
        self.title = title  
        self.isbn = isbn  
        self.price = price
```

กำหนดให้ books คือ ลิสต์ที่เก็บอ้อม JEKET ของ Book

```
def discount( books, p ):  
    for b in books:  
        b.price *= (1 - p/100)
```

```
def get_min_price( books ):  
    return min([b.price for b in books])
```

```
def search( books, isbn ):  
    for b in books:  
        if isbn == b.isbn: return b  
    return None
```

# การเพิ่มบริการให้เรียกใช้กับอ็อบเจกต์

```
class Point:  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
  
    def distance(p1, p2):  
        dx = p1.x - p2.x  
        dy = p1.y - p2.y  
        return (dx**2+dy**2)**0.5  
  
    def to_str(p):  
        return "(" + str(p.x) + \  
                ", " + str(p.y) + ")"  
  
p1 = Point(2,4)  
p2 = Point(3,5)  
d = distance(p1, p2)  
print( to_str(p1), to_str(p2) )
```

functions

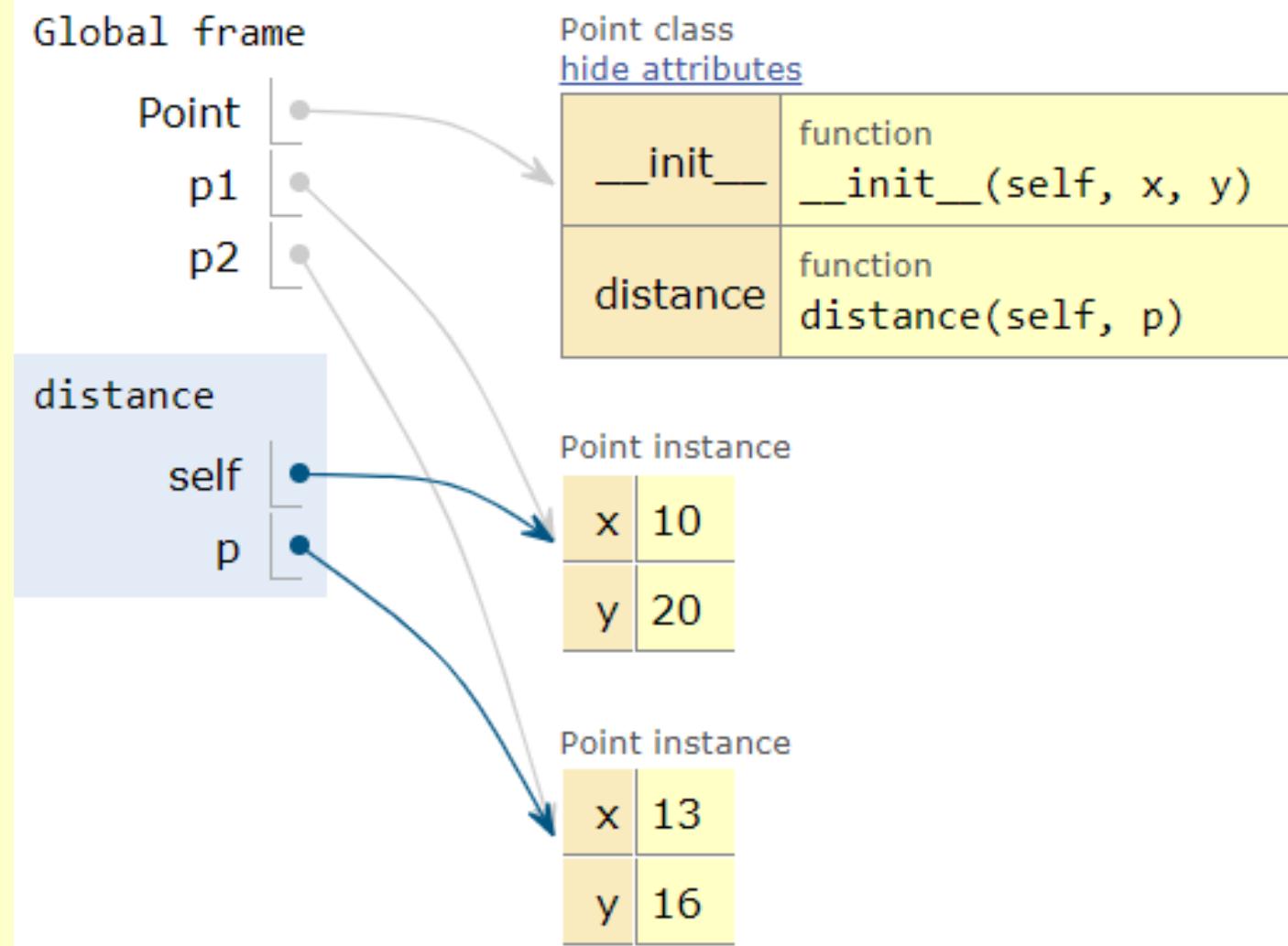
```
class Point:  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
  
    def distance(self, p):  
        dx = self.x - p.x  
        dy = self.y - p.y  
        return (dx**2+dy**2)**0.5  
  
    def to_str(self):  
        return "(" + str(self.x) + \  
                ", " + str(self.y) + ")"  
  
p1 = Point(2,4)  
p2 = Point(3,5)  
d = p1.distance(p2)  
print( p1.to_str(), p2.to_str() )
```

methods

อ็อบเจกต์.ชื่อเมธ็อด (...)

# อ้อมเจกต์ที่ self อ้างอิง

```
class Point:  
  
    def distance(self, p):  
        dx = self.x - p.x  
        ...  
  
    d = p1.distance(p2)
```



# เคยใช้ฟังก์ชันกับเมธอดนามากมาย

## functions

```
a = [1,2,3,4]
k = len(a)
print(a)
s = sum(a)
b = sorted(a)

# พังก์ชันใน module อื่น
k = math.sin(1)
d = np.ndarray((2,3))
```

## methods

```
a = [1,2,3,4]
a.sort()
a.append(99)
t = "aBc"
u = t.upper()
v = t.lower()
k = t.find("B")
s = set(a)
s = s.union([3,5])
```

# ตัวอย่าง : BankAccount

- คลาส BankAccount แทนบัญชีธนาคาร
  - หมายเลขบัญชี (acc\_no)
  - ชื่อบัญชี (acc\_name)
  - ยอดเงินปัจจุบัน (balance)
- เมธ็อดสำหรับ BankAccount
  - ฝาก : deposit( amount )
  - ถอน : withdraw( amount )

```
a1 = BankAccount("1-034-567-892",
                  "ปราณี รักเรียน", 500)
a1.deposit(1000)
a1.withdraw(150)
```

## ตัวอย่าง : BankAccount มีบริการฝาก/ถอน

```
class BankAccount:  
    def __init__(self, acc_no, acc_name, balance):  
        self.acc_no = acc_no  
        self.acc_name = acc_name  
        self.balance = balance  
  
    def deposit(self, amount):  
        if amount > 0:  
            self.balance += amount  
  
    def withdraw(self, amount):  
        if 0 < amount <= self.balance:  
            self.balance -= amount  
  
a1 = BankAccount("1-034-567-892", "ปราณี รักเรียน", 500)  
a1.deposit(1000)  
a1.withdraw(150)  
print(a1.acc_no, a1.balance)
```

# การเรียกใช้เมธอดภายในคลาสเดียวกัน

```
class BankAccount:  
    ...  
    def deposit(self, amount):  
        ...  
    def withdraw(self, amount):  
        ...  
  
    def transfer_to(self, acc, amount):  
        if 0 <= amount <= self.balance:  
            self.withdraw( amount )  
            acc.deposit( amount )
```

## การเรียกฟังก์ชันภายในคลาสเดียวกัน

```
class Rational:  
    def __init__(self, n, d):  
        self.n = n          # numerator เศษ  
        self.d = d          # denominator ส่วน  
  
r1 = Rational( 1, 2 )  # เก็บ 1/2  
r2 = Rational( 4, 8 )  # เก็บ 4/8
```

# การเรียกฟังก์ชันภายในคลาสเดียวกัน

```
class Rational:  
    def __init__(self, n, d):  
        g = Rational.gcd(n, d) # ชื่อคลาส.ชื่อเมท็อด (...)  
        self.n = n//g # numerator เศษ  
        self.d = d//g # denominator ส่วน
```

```
    def gcd(a, b):  
        while b != 0:  
            a, b = b, a%b  
        return a
```

```
r1 = Rational( 1, 2 ) # เก็บ 1/2  
r2 = Rational( 4, 8 ) # เก็บ 1/2
```

# ตัวอย่าง: จำนวนตรรกยะ

```
class Rational:  
    def gcd(a, b):  
        while b != 0:  
            a, b = b, a%b  
        return a  
  
    def __init__(self, n, d):  
        g = Rational.gcd(n, d)  
        self.n = n//g # numerator เช馈  
        self.d = d//g # denominator ส่วน  
  
    def mult(self, x):  
        n = self.n * x.n  
        d = self.d * x.d  
        return Rational(n, d)  
  
    def add(self, x):  
        d = self.d * x.d  
        n = self.n * x.d + x.n * self.d  
        return Rational(n, d)
```

```
        def to_float(self):  
            return self.n / self.d  
  
        def less_than(self, x):  
            return self.to_float() < x.to_float()  
  
        def to_str(self):  
            return str(self.n) + "/" + str(self.d)  
  
r1 = Rational(20,40)  
r2 = Rational(1,4)  
r3 = r1.add(r2)  
print(r3.to_str())  
r4 = r1.mult(r2)  
print(r4.to_str())  
print(r3.less_than(r4))
```

อยากเขียน  
r3 = r1 + r2  
print( str(r3) )  
r4 = r1 \* r2  
print( r3 < r4 )  
ทำไง ?

# เมธอดพิเศษของคลาสที่ทำให้ใช้งานง่าย

```
class Rational:  
    def __init__(self, n, d):  
        g = Rational.gcd(n, d)  
        self.nu = n//g # numerator เศษ  
        self.de = d//g # denominator ส่วน  
  
    def add(self, x):  
        ...  
    def mult(self, x):  
        ...  
    def to_float(self):  
        ...  
    def less_than(self, x):  
        ...  
    def to_str(self):  
        ...
```

```
r3 = r1.add(r2)  
print(to_str(r3), to_float(r3) )  
r4 = r1.mult(r2)  
print( r3.less_than(r4) )
```

```
class Rational:  
    def __init__(self, n, d):  
        g = Rational.gcd(n, d)  
        self.nu = n//g # numerator เศษ  
        self.de = d//g # denominator ส่วน  
  
    def __add__(self, x):  
        ...  
    def __mul__(self, x):  
        ...  
    def __float__(self):  
        ...  
    def __lt__(self, x):  
        ...  
    def __str__(self):  
        ...
```

```
r3 = r1 + r2  
print( str(r3), float(r3) )  
r4 = r1 * r2  
print( r3 < r4 )
```

# ตัวอย่าง: เมนูอาหาร และการสั่งอาหาร

```
class Item:  
    def __init__(self, name, price):  
        self.name = name  
        self.price = price
```

```
menu = [ Item("Fried rice", 45),  
         Item("Phat thai", 50),  
         Item("Congee", 30),  
         Item("Papaya salad", 40) ]
```

# ตัวอย่าง: เมนูอาหาร และการสั่งอาหาร

```
menu = [ Item("Fried rice", 45),  
         Item("Phat thai", 50),  
         Item("Congee", 30),  
         Item("Papaya salad", 40) ]
```

ข้าวผัดสอง ส้มตำหนึ่ง

```
o1 = Order();  
o1.add(menu[0], 2);  
o1.add(menu[3], 1)
```

```
class Order:  
    def __init__(self):  
        self.order_items = []  
        self.paid = False  
  
    def add(self, item, n):  
        for i in range(n):  
            self.order_items.append(item)  
  
    def total(self):  
        s = 0  
        for item in self.order_items:  
            s += item.price  
        return s
```

# ตัวอย่าง: สั่งอาหาร, จ่ายเงิน, รายรับรวม

```
class Item:  
    def __init__(self, name, price):  
        self.name = name  
        self.price = price
```

```
class Order:  
    def __init__(self):  
        self.order_items = []  
        self.paid = False  
  
    def add(self, item, n):  
        for i in range(n):  
            self.order_items.append(item)  
  
    def total(self):  
        return sum([item.price  
                   for item in self.order_items])
```

```
def get_total(orders):  
    return sum( [od.total() for od in orders if od.paid] )
```

```
m = [ Item("Fried rice", 45),  
      Item("Phat thai", 50),  
      Item("Congee", 30),  
      Item("Papaya salad", 40) ]  
  
o1 = Order()  
o1.add(m[0],2); o1.add(m[3],1)  
o2 = Order();  
o2.add(m[0],1); o2.add(m[1],2)  
o3 = Order();  
o3.add(m[1],1); o3.add(m[2],1)  
  
orders = [o1, o2, o3]  
o1.paid = True  
o2.paid = True  
  
print(get_total(orders))
```

# เติม `__lt__` และ `__str__` ให้ Item

```
class Item:  
    def __init__(self, name, price):  
        self.name = name  
        self.price = price  
  
    def __lt__(self, rhs):  
        return self.price < rhs.price  
  
    def __str__(self):  
        return self.name + ":" + str(self.price)  
  
x1 = Item("Congee", 30)  
x2 = Item("Phat thai", 50)  
print(x1 < x2)          # True  
print(x2 < x1)          # False  
print(str(x1), str(x2)) # Congee:30 Phat thai:50  
print(x1, x2)           # print จะไปเรียก str ให้
```

เปรียบเทียบด้วย  
ราคาอาหาร

# sort ใช้ lt 在การเรียงลำดับข้อมูล

```
menu = [ Item("Fried rice", 45),  
         Item("Phat thai", 50),  
         Item("Congee", 30),  
         Item("Papaya salad", 40) ]  
  
menu.sort()  
  
for item in menu:  
    print(item)
```

```
Congee:30  
Papaya salad:40  
Fried rice:45  
Phat thai:50
```

# ตัวอย่าง : Date

```
class Date:  
    def __init__(self, d, m, y):  
        self.d = d  
        self.m = m  
        self.y = y  
  
    def __lt__(self, rhs):  
        d1 = (self.y, self.m, self.d)  
        d2 = (rhs.y, rhs.m, rhs.d)  
        return d1 < d2  
  
    def __str__(self):  
        return str(self.d) + "/" + \  
               str(self.m) + "/" + \  
               str(self.y)  
  
d1 = Date(20, 1, 1990)  
d2 = Date(9, 12, 1990)  
print( d1 < d2 )  
print( d1, d2 )
```

True  
20/1/1990 9/12/1990