

Variables

```
In [ ]: 3.14159265359
```

```
Out[ ]: 3.14159265359
```

หลักการตั้งชื่อตัวแปรเบื้องต้น

1. ตั้งให้สื่อ
2. ภาษาอังกฤษ
3. ใช้ตัวเลขได้แต่ห้ามขึ้นต้นด้วยตัวเลข
4. ห้ามเว้นวรรค

```
In [ ]: pi = 3.14159265359
```

```
In [ ]: pi
```

```
Out[ ]: 3.14159265359
```

int : จำนวนเต็ม

```
In [ ]: a = 10
```

```
In [ ]: print(a)
```

```
10
```

float : จำนวนจริง (ทศนิยม)

```
In [ ]: b = 10.  
print(b)
```

```
10.0
```

ตัวอักษร (char (character)) ข้อความ (text หรือ string)

.. # hashtag, sharp ใช้สำหรับคอมเมนต์

```
In [ ]: c = 'ธนพงศ์' # เราจะบอกว่าตัวแปรเป็นตัวอักษรหรือข้อความโดยการใช้ single quote ' หรือ double  
print(c)
```

```
ธนพงศ์
```

ตัวเลขที่เป็น string ไม่สามารถเอามา + - x / กับตัวเลขได้

```
In [ ]: d = '10'  
print(d)
```

```
10
```

```
In [ ]: d + 1
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-9-2d39be9af043> in <module>()  
----> 1 d + 1  
  
TypeError: must be str, not int
```

variable casting (การเปลี่ยนชนิดของข้อมูล)

```
In [ ]: int(d) + 1
```

```
Out[ ]: 11
```

```
In [ ]: print(a)  
        print(float(a))  
        print(str(a))
```

```
10  
10.0  
10
```

Operation (Operators +,-,*,/,%)

```
In [ ]: a+b
```

```
Out[ ]: 20.0
```

```
In [ ]: ab = a+b  
        print(ab)
```

```
20.0
```

% คือหมาย modulo

```
In [ ]: 5%3
```

```
Out[ ]: 2
```

```
In [ ]: 7%3
```

```
Out[ ]: 1
```

คำสั่ง print แบบพิเศษ (การ format string)

```
In [ ]: print('ตัวแปร') #สิ่งที่อยู่ข้างในวงเล็บคือ ตัวแปร และ string
```

```
ตัวแปร
```

```
In [ ]: print(f'% คือการหารเอาเศษ เช่น 7%3 = {7%3}') # เพิ่ม f หน้า 'string' และใช้ {} ใส่ code
```

```
% คือการหารเอาเศษ เช่น 7%3 = 1
```

```
In [ ]: print(f'% คือการหารเอาเศษ เช่น 7%3 = {7%3} \\  
        แต่\n/ คือการหารปกติ เช่น 7/3 = {7/3} \\  
        และ\n// คือการหารเอาส่วน เช่น 7//3 = {7//3} \\  
        หรือ\nใช้ int() เพื่อหารเอาส่วน เช่น int(7/3) = {int(7/3)}') # \n คือการขึ้นบรรทัดใหม่
```

% คือการหารเอาเศษ เช่น $7\%3 = 1$ \ แต่
/ คือการหารปกติ เช่น $7/3 = 2.3333333333333335$ \ และ
// คือการหารเอาส่วน เช่น $7//3 = 2$ \ หรือ
ใช้ `int()` เพื่อหารเอาส่วน เช่น `int(7/3) = 2`

DATA STRUCTURE (โครงสร้างข้อมูล)

List คือ การเอาตัวแปรหลายๆตัวมาเรียงกัน

list สามารถสร้างได้ 2 แบบ ดังนี้

แบบที่1 > square brackets

```
In [ ]: list_a = []  
print(list_a)
```

`[]`

```
In [ ]: list_b = [1,5,'v']  
print(list_b)
```

`[1, 5, 'v']`

ลำดับที่อยู่ใน list มีความสำคัญ (ลำดับใน list เริ่มจาก 0,1,2,...)

```
In [ ]: list_b[1]
```

Out[]: 5

แบบที่2

```
In [ ]: list_c = list()  
print(list_c)
```

`[]`

append() เพิ่มสมาชิกใน list

```
In [ ]: list_b.append('u')  
print(list_b)
```

`[1, 5, 'v', 'u']`

```
In [ ]: list_b.pop() ### ดึงสมาชิกที่สุดท้ายออกจาก List
```

Out[]: 'u'

```
In [ ]: list_b
```

Out[]: `[1, 5, 'v']`

```
In [ ]: list_b.append(list_a)  
print(list_b)
```

`[1, 5, 'v', []]`

String > list of characters

```
In [ ]: t = 'python is easy'
t
```

```
Out[ ]: 'python is easy'
```

```
In [ ]: len(list_b) # Len คือคำสั่งตรวจสอบความยาวของ list (จำนวนสมาชิก)
```

```
Out[ ]: 4
```

```
In [ ]: len(t)
```

```
Out[ ]: 14
```

access a member of a list (list&string)

โดยการเริ่มนับจะเริ่มนับจาก 0

0 คือสมาชิกตัวแรก , -1 คือสมาชิกตัวสุดท้าย

```
In [ ]: t[1]
```

```
Out[ ]: 'y'
```

```
In [ ]: t[-1]
```

```
Out[ ]: 'y'
```

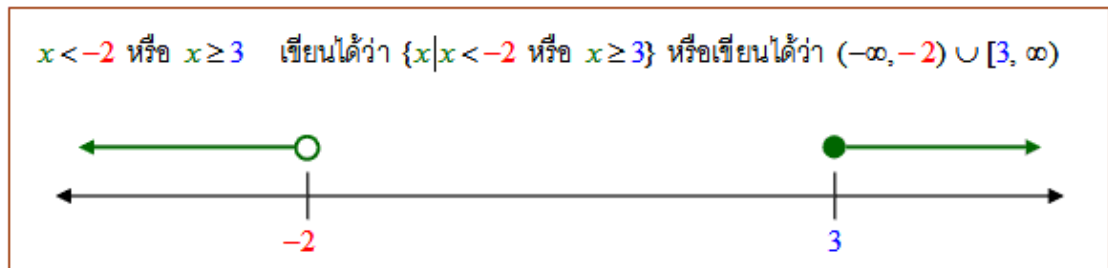
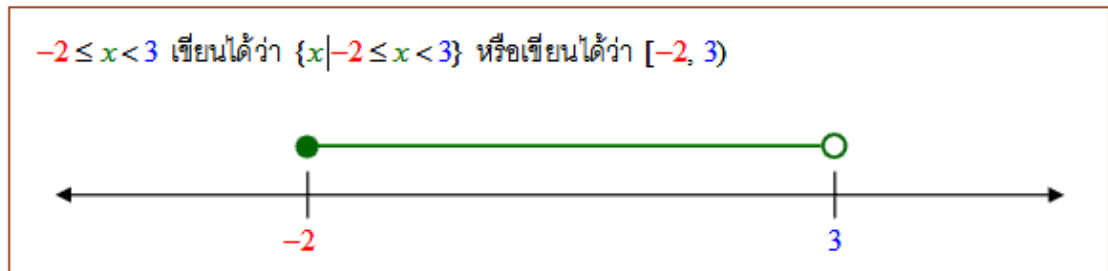
```
In [ ]: t[-4]
```

```
Out[ ]: 'e'
```

ตัวที่อยู่ข้างใน [] เราเรียกว่า index (ตัวชี้)

List slicing สามารถทำได้โดยใช้ colon :

[a:b] -> [a,b)



```
In [ ]: t = 'python is easy'
```

```
In [ ]: print(t)
        print(t[7:9])
```

```
python is easy
is
```

ถ้าเว้นว่างหน้า : หมายความว่า เริ่มตั้งแต่ตัวแรก

ถ้าเว้นว่างหลัง : หมายความว่า ไปจนถึงตัวสุดท้าย

```
In [ ]: print(t)
        print(t[:6])
        print(t[10:])
        print(t[-4:])
        print(t[:])
```

```
python is easy
python
easy
easy
python is easy
```

```
In [ ]: print(t[::2]) ## : ตัวที่สอง กำหนด step
```

```
pto ses
```

```
In [ ]: print(list(range(10)))
        print(list(range(10))[::2])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

```
In [ ]: print(t[2::2])
```

```
to ses
```

เราสามารถเอา list มาต่อกันได้ด้วย +

```
In [ ]: t + '??'
```

```
Out[ ]: 'python is easy??'
```

```
In [ ]: t + list_b ## ไม่สามารถเอา list ปกติมาต่อกับ string ได้
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-56-8ceeb7529eac> in <module>()  
----> 1 t + list_b ## ไม่สามารถเอา list ปกติมาต่อกับ string ได้  
TypeError: must be str, not list
```

```
In [ ]: list_b + list_a
```

```
Out[ ]: [1, 5, 'v', []]
```

การแบ่ง string ตามสัญลักษณ์ที่กำหนด -> split string

```
In [ ]: t.split(' ')
```

```
Out[ ]: ['python', 'is', 'easy']
```

```
In [ ]: time = '12:30:15'
```

```
In [ ]: time.split(':')
```

```
Out[ ]: ['12', '30', '15']
```

```
In [ ]: t_sp = t.split(' ')  
print(t_sp)
```

วิธีรวมกลับ

```
print(':' .join(t_sp))
```

```
['python', 'is', 'easy']  
python:is:easy
```

HW คำนวณเวลาเป็นวินาทีของเวลาต่อไปนี้โดยใช้คำสั่ง split() ช่วย (print ออกมาให้สวยงาม)

12:30:15

13:41:07

12:53:15

00:59:25

11:11:11

16:06:09

ให้สร้างฟังก์ชันคำนวณเวลาเป็นวินาทีของเวลาใดๆ และ print ออกมาให้สวยงาม

ให้ สร้าง list ของ เวลา

['12:30:15', '13:41:07',] แล้ววนลูปเรียกฟังก์ชันคำนวณเวลาเป็น วินาที

commit ว่า HW3

In []:

```
In [ ]: name = 'ภัทรวรรณ'
        surname = 'ใจเที่ยง'
        ID = '603021866-7'

        b = f'ชื่อ {name} นามสกุล {surname} รหัส {ID}'
        print(b)
```

ชื่อ ภัทรวรรณ นามสกุล ใจเที่ยง รหัส 603021866-7

Function

Function template

ทำหน้าที่รับ input มาประมวลผลออกมาเป็น output

$f(x) = y$

def คือการกำหนดฟังก์ชัน

backtick (`) ==> กด ~ ค้าง, alt - 9>6 (full keyboard with number)

tilde (~)

curly bracket ({ })

square bracket ([])

```
def function_name(_Input_) :
    do_something with _Input_ to get _Output_
    return _Output_
```

function มีส่วนสำคัญทั้งหมด 4 ส่วน

1. บอก python ว่าเราจะเขียนฟังก์ชัน ชื่ออะไร `def function_name():` (ขาดไม่ได้)
2. กำหนดตัวแปรที่จะเป็น input `_Input_` (ขาดได้)
3. ส่วนประมวลผล `do_something with _Input_ to get _Output_` (ขาดไม่ได้)
4. ส่วน output `return _Output_` (ขาดได้)

ตัวอย่างการเขียน normal function

```
In [ ]: def print_name(surname, ID, name):
        st = f'ชื่อ {name} นามสกุล {surname} รหัส {ID}'
        return st
```

```
In [ ]: print_name('อินทระ', '64xxxxxx', 'ธนพงศ์')
```

```
Out[ ]: 'ชื่อ ธนพงศ์ นามสกุล อินทระ รหัส 64xxxxxx'
```

```
In [ ]: print(print_name(name='กาญจนา', surname='ประสาธน์', ID='603021855-2'))
```

ชื่อ กาญจนา นามสกุล ประสาธน์ รหัส 603021855-2

เราใช้ เว้นวรรค (indent) เพื่อบอกขอบเขตของโปรแกรม

ฟังก์ชันไม่จำเป็นต้องมี output

```
In [ ]: def print_name2(surname,ID,name):  
        st = f'ชื่อ {name} นามสกุล {surname} รหัส {ID}'  
        print(st)
```

```
In [ ]: print_name2(name='กาญจนา',surname='ประสาธน์',ID='603021855-2')
```

ชื่อ กาญจนา นามสกุล ประสาธน์ รหัส 603021855-2

ฟังก์ชันไม่จำเป็นต้องมี input

```
In [ ]: def Pi():  
        return 3.14159265359
```

```
In [ ]: Pi()*(2**2) # หาพื้นที่ของวงกลมที่มีรัศมีเท่ากับ Pi * r^2
```

```
Out[ ]: 12.56637061436
```

เราสามารถกำหนดค่า default ให้กับฟังก์ชันได้

input ของ function ใน python มีสองแบบ input ที่จำเป็นต้องใส่ กับ input ที่ไม่จำเป็นต้องใส่ (มีค่า default)

เราต้องเรียง input ที่จำเป็นต้องใส่ขึ้นก่อน

```
In [ ]: def print_2lines_default(name,surname,ID,grade='F'):  
        st = f'ชื่อ {name} นามสกุล {surname} รหัส {ID}'  
        print (st)  
        st2=f'เกรดวิชา Data Viz >>> {grade}'  
        print(st2)
```

```
In [ ]: print_2lines_default('ธัญญการต์','พวงมาลัย','613020551-8')
```

ชื่อ ธัญญการต์ นามสกุล พวงมาลัย รหัส 613020551-8
เกรดวิชา Data Viz >>> F

```
In [ ]: print_2lines_default('ธัญญการต์','พวงมาลัย','613020551-8','A')
```

ชื่อ ธัญญการต์ นามสกุล พวงมาลัย รหัส 613020551-8
เกรดวิชา Data Viz >>> A

LOOP การวนซ้ำ

for เป็นคำที่ใช้บอก python ว่าเรากำลังเขียน loop โดย for จะวนถึงสมาชิกจาก `listA` มาทำ process `do_something`

```
for each_member in listA :  
    do_something
```

คำสั่ง for เป็นคำสั่งวนซ้ำที่ใช้ควบคุมการทำงานซ้ำๆ ในจำนวนรอบที่แน่นอน

```
In [ ]: for i in [1,2,3] :  
        o = i**2  
        print (f'this member = {i} after process = {o}')
```

```
this member = 1 after process = 1  
this member = 2 after process = 4  
this member = 3 after process = 9
```

จบวันที่ 7 มค 2564

ตัวอย่างการใช้คำสั่ง for ในการวนซ้ำค่าของ i เมื่อ i คือ 'Thanapong' และคำสั่ง `print(f'{i} -> /')` เพื่อกำหนดให้ผลลัพธ์แสดงสัญลักษณ์ -> / จากนั้นเพิ่มคำสั่ง `print(' ')` อีกครั้งเพื่อให้ผลลัพธ์มีการเว้นวรรค

```
In [ ]: for i in 'Thanapong':  
        print(f'{i} -> / ' )
```

```
T -> /  
h -> /  
a -> /  
n -> /  
a -> /  
p -> /  
o -> /  
n -> /  
g -> /
```

```
In [ ]: for i in 'Thanapong':  
        print(f'{i} -> / ' ,end = '') #สั่งให้ print แบบ ไม่เว้นบรรทัด
```

```
T -> / h -> / a -> / n -> / a -> / p -> / o -> / n -> / g -> /
```

```
In [ ]: for i in 'Thanapong':  
        print(f'{i} -> / ' ,end = ',') #สั่งให้ print แบบ ไม่เว้นบรรทัด
```

```
T -> / ,h -> / ,a -> / ,n -> / ,a -> / ,p -> / ,o -> / ,n -> / ,g -> / ,
```