Iteration (for loop)
Function

for

Using range

- Function range
- for … in range(…)

Files

- Open and close files
- Read a line from a file
- For loop to read each line from a file

Using range

Function range

for … in range(…)

Function range

Function range

- ฟังก์ชัน range เป็นฟังก์ชันแจงนับของจำนวนเต็ม โดยจะแจกแจงจำนวนตั้งแต่ค่าเริ่มต้น (start) ไปจนถึงจำนวนเต็มสุดท้ายก่อนค่าขอบปลาย (end)
- ค่าที่ได้อยู่ในช่วง [start, end)
- range(end) เริ่มต้นที่ 0 และสิ้นสุดที่ end-1
- range(start, end) เริ่มต้นที่ start และสิ้นสุดที่ end
- range(start, end, step) เริ่มต้นที่ start ค่าเพิ่มขึ้นทีละ step และสิ้นสุดก่อนถึง end

*** start, end, และ step จะต้องเป็นจำนวนเต็ม

ตัวอย่าง range

- range(end) เริ่มต้นที่ 0 และสิ้นสุดที่ end-1
- range(start, end) เริ่มต้นที่ start และสิ้นสุดที่ end
- range(start, end, step) เริ่มต้นที่ start ค่าเพิ่มขึ้นทีละ step และสิ้นสุดก่อนถึง end

• range(10)	**0**, 1, 2, 3, 4, 5, 6, 7, 8, 9
• range(1, 10)	**1**, 2, 3, 4, 5, 6, 7, 8, 9
• range(1, 10, 2)	1, 3, 5, 7, 9
• range(10, 0, -1)	10, 9, 8, 7, 6, 5, 4, 3, 2, 1
• Range(11, 11)	ไม่ทำสักรอบ เพราะ step เป็นบวกและ start >= end
• range(9, 10, -1)	ไม่ทำสักรอบ เพราะ step ติดลบและ start <= end

for … in range(…)

for … in range(…)

โครงสร้าง for

- โครงสร้าง **for** เป็นการวนซ้ำ โดยที่การทำงานแต่ละรอบจะดึงสมาชิกในข้อมูลแบบกลุ่ม (collection) ออกมา
- การทำงานมีลักษณะเป็นการวนซ้ำ **มีตัวแปรหนึ่งตัว** สำหรับเชื่อมกับค่าของสมาชิกในกลุ่ม
- ในแต่ละรอบของการวนซ้ำ **ตัวแปรเชื่อมค่า** จะเปลี่ยนการเชื่อมค่ากับสมาชิกไปทีละตัว จากตัวแรกไปจนถึงตัวสุดท้าย
- ดัชนี 0 ถึงดัชนี len(กลุ่ม) – 1 (ส่วนนี้ เก็บไว้เรียนในบท list)

ฟังก์ชัน range กับโครงสร้าง for

```
for i in range (10):
    print(i)
```

อ่านได้ว่า สำหรับ i ในช่วง [0,10)

ในแต่ละรอบ ตัวแปร i จะเชื่อมกับค่า 0 1 2 3 4 5 6 7 8 9 10 ตามลำดับและจบการทำงาน

```
n=10
for i in range (n):
    print(i)
```

Question: ในแต่ละรอบ ตัวแปร i จะเชื่อมกับค่า

การนับรอบ

สามารถใช้ for ร่วมกับ range ทำงานได้เหมือนกับการเขียน while แบบ loop n times

x=0
while x<=5:
  print (x)
  x=x+1
	for x in range(6):
	  print (x)
	จะมีการวนซ้ำทั้งหมด 6 รอบ และจบการทำงาน ในแต่ละรอบ ตัวแปร x จะมีค่า 0 - 5 ตามลำดับ

x=1
while x<=5:
  print (x)
  x=x+1
	for x in range(1, 6):
	  print (x)
	จะมีการวนซ้ำทั้งหมด 5 รอบ และจบการทำงาน ในแต่ละรอบ ตัวแปร x จะมีค่า 1 - 5 ตามลำดับ

ตัวอย่าง : โปรแกรมหาเลขจำนวนเต็มคู่ ตั้งแต่ 1–10

Idea

Input : จำนวนเต็มเก็บไว้ที่ตัวแปร n

Process: วนค่าหาจำนวนเต็มคู่ที่อยู่ในช่วง 1 – 10

Output:ค่าของจำนวนเต็มที่หาได้

PYTHON CODE:

```
n = 10
for i in range(1,n):
    if i%2 ==0:
        print(i)
```

ตัวอย่าง : หาเลขยกกำลัง $2^1, \ldots, 2^{10}$

Idea

Input : จำนวนเต็มเก็บไว้ที่ตัวแปร n

Process: วนค่าหา

Output: ค่าของจำนวนเต็มที่หาได้

PYTHON CODE:

```
n = 10
for i in range(1, n+1):
    print(2**i)
```

การเปลี่ยนค่าตัวแจงนับของ for

- ในแต่ละรอบของการวนซ้ำ ตัวแปรเชื่อมค่าจะเปลี่ยนการเชื่อมค่ากับสมาชิกไปทีละตัว จากตัวแรกไปจนถึงตัวสุดท้าย
- ซึ่งจะเป็นกำหนดค่าเตรียมไว้แล้วจนจบการทำงานของลูป
- หากภายในลูป มีการเปลี่ยนค่าของตัวแปรเชื่อมค่า เมื่อการทำงานย้อนกลับขึ้นไปที่ต้นลูป โปรแกรมจะใช้ค่าสมาชิกที่มันได้เตรียมแจงนับไว้ล่วงหน้าแล้ว เพื่อทำงานต่อไปตามปกติ

```
for x in range(1, 6):
    print (x)
```

```
for x in range(1, 6):
    print (x)
    x = 10
    print('[', x, ']')
```

การเปลี่ยนค่าตัวแจงนับของ for

```
for x in range(1,6):
    print(i)
```

ผลการรัน

1
2
3
4
5

```
for x in range(1,6):
    print(i)
    x=10
    print('[',x, ']')
```

ผลการรัน

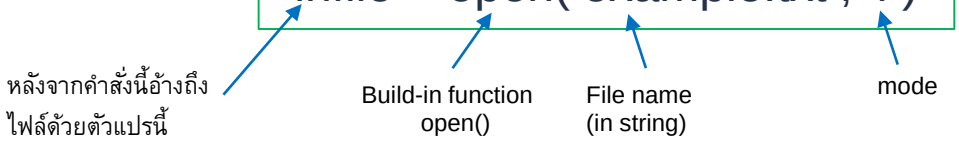
1
[10]
2
[10]
3
[10]
4
[10]
5
[10]

Files

- Open and close files
- Read a line from a file
- Write a string to a file
- For loop to read each line from a file

การเปิดและปิดไฟล์

```
infile = open('example.txt', 'r')
```



เปิด (open) ไฟล์ ก่อนใช้

Mode:

r : read only ถ้าไม่ใส่ไว้ โปรแกรมจะถือว่า เปิดไฟล์เพื่ออ่านอย่างเดียว

w : write only (เขียนทับข้อมูลที่มีอยู่ก่อน)

a : append (เขียนต่อท้ายจากข้อมูลที่มีอยู่ก่อน)

r+ : read and write

```
infile.close()
```

ปิด (close) ไฟล์ เมื่อเลิกใช้

การอ่านข้อมูลทีละบรรทัดจากไฟล์มาเก็บในสตริง

```
s = infile.readline()
```

ตัวแปรที่เก็บ stringที่อ่านได้ | ตัวแปรไฟล์ | function

test.txt

```
This is a test\n
abcdefgh\n
123456789\n
```

```
fp = open( 'test.txt', 'r')
str1 = fp.readline()
str2 = fp.readline()
str3 = fp.readline()
str4 = fp.readline()
fp.close()
```

ค่าในตัวแปร

str1 : This is a test

str2 : abcdefgh

Str3 : 123456789

str4 :

PYTHON CODE

```
fp=open('test.txt','r')
str1=fp.readline()
str2=fp.readline()
str3=fp.readline()
str4=fp.readline()
fp.close()
print(str1)
print(str2)
print(str3)
print(str4)
```

Output

```
This is a test

abcdefgh

123456789
```

การอ่านข้อมูลจนจบไฟล์

```
s = infile.readlines()
```

ตัวแปรที่เก็บ stringที่อ่านได้ — ตัวแปรไฟล์ — function

test.txt

```
This is a test\n
abcdefgh\n
123456789\n
```

```
fp = open( 'test.txt', 'r')
str = fp.readlines()
fp.close()
```

ค่าในตัวแปร

str1 : ['This is a test\n', 'abcdefgh\n', '123456789']

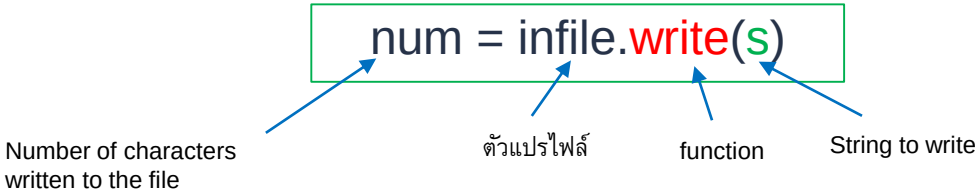
PYTHON CODE

```
fp=open('test.txt','r')
s=fp.readlines()
print(s)
```

```
['This is a test\n', 'abcdefgh\n', '123456789']
```

Output

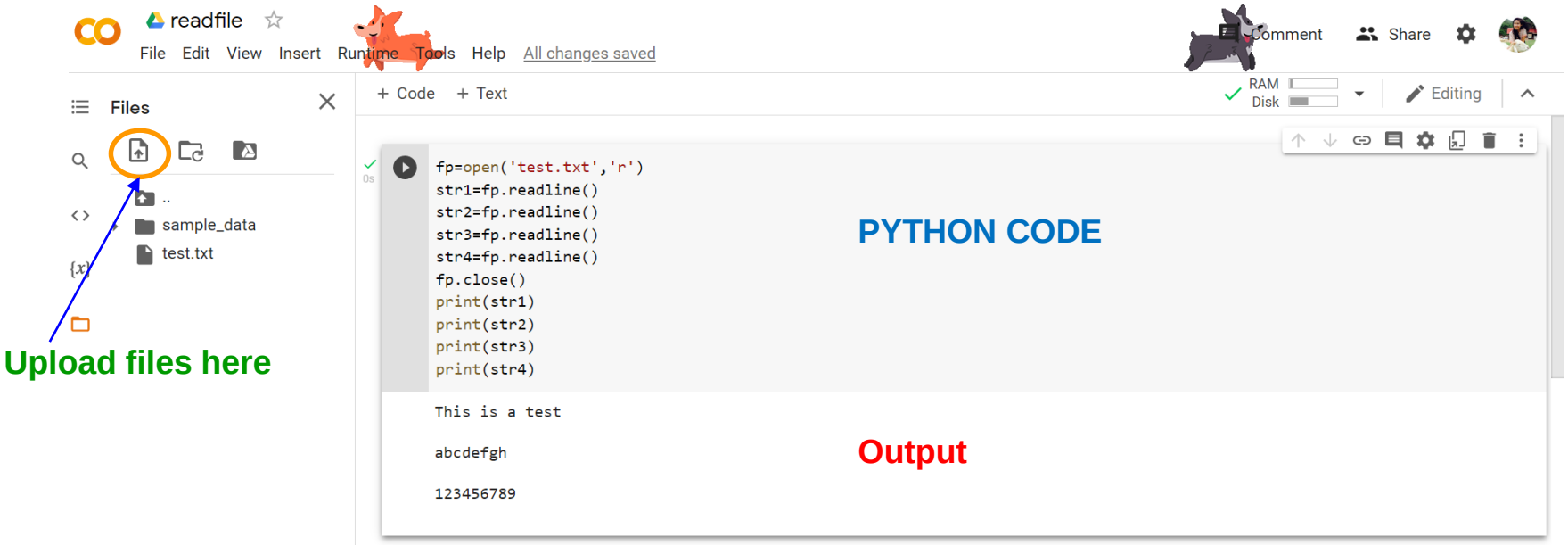
การเขียนสตริงเก็บลงไฟล์



เขียนสตริงลงในไฟล์เมื่อ write ครั้งต่อไป จะเขียนต่อจากครั้งก่อน

- ถ้าเปิดไฟล์ด้วย mode w จะเริ่มเขียนจากต้นไฟล์ โดยทับข้อมูลเก่าที่มีอยู่
- ถ้าเปิดไฟล์ด้วย mode w จะเขียนต่อจากไฟล์เดิม

ตัวอย่างการอ่านไฟล์



ตัวอย่างการเขียนไฟล์

โครงสร้าง for กับการอ่านไฟล์

```
for line in file:
    ....
```

ตัวแปร line เก็บสตริงหนึ่งบรรทัดที่อ่านมาจากไฟล์ที่อ้างถึงตัวแปร file

- โครงสร้าง for ที่ใช้กับการอ่านไฟล์จะเป็นการวนซ้ำโดยที่การทำงานแต่ละรอบจะดึงข้อมูลในไฟล์ออกมาทีละบรรทัด
- การทำงานมีลักษณะเป็นการวนซ้ำ มีตัวแปรหนึ่งตัว สำหรับเชื่อมกับข้อมูลแต่ละบรรทัดในไฟล์
- ในแต่ละรอบของการวนซ้ำ ตัวแปรเชื่อมค่าจะเปลี่ยนการเชื่อมค่ากับข้อมูลในไฟล์ทีละบรรทัดจากบรรทัดแรกไปจนถึงบรรทัดสุดท้าย

โครงสร้าง for กับการอ่านไฟล์

อ่านแฟ้มข้อความทีละบรรทัดใช้ readline

```
file = open(filename, 'r')
str1 = file.readline()
str2 = file.readline()
```



```
for line in file:
    #คำสั่ง for อ่านจาก file รอบละหนึ่งบรรทัด
    #มาเก็บเป็นสตริงใน line จนหมดแฟ้ม
    …
file.close() #ไม่อ่านแล้ว ก็ปิดแฟ้ม
```

ถ้าบรรทัดที่อ่านเข้ามาเป็นบรรทัดว่างๆเก็บนะ t จะได้ t='\n'

หรือ len(t) เป็น 1 แต่ถ้าอ่านตอนที่แฟ้มไม่มีข้อมูลให้อ่านแล้วจะได้ t='' หรือ len(t) เป็น 0

```
for line in file:
    ….
```

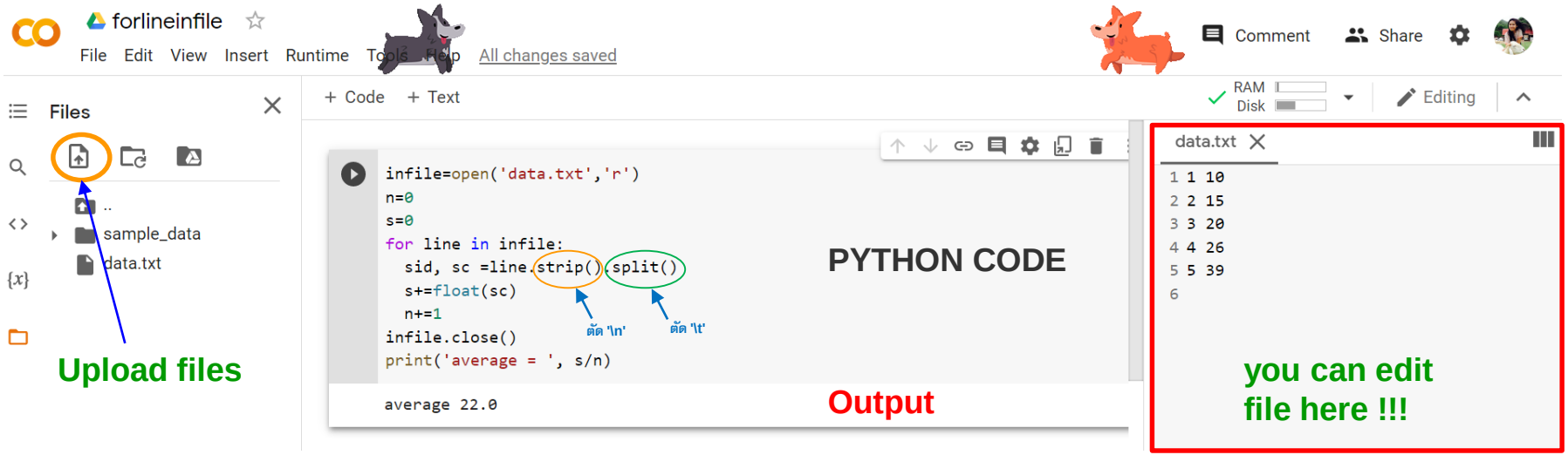
เหมือนกับ

```
line = file.readline()
```

ตัวอย่างการหาค่าเฉลี่ยคะแนนสอบเด็ก 5 คน

strip คือการตัดอักขระว่างต้นและท้ายข้อความเมื่ออ่านข้อความไฟล์

split คือการตัดข้อความด้วยอักขระคั่น



แบบฝึกหัด

1. จงเขียนโปรแกรมเพื่อหาผลบวกของอนุกรม 1-2+3-4+5-6+7-8+9-10 แสดงเป็นผลลัพธ์ (ผลลัพธ์ที่ได้คือ -5)

2. จงเขียนโปรแกรมเพื่อรับค่าจำนวนแถวเพื่อพิมพ์สามเหลี่ยมดังรูป แสดงผลลัพธ์เป็น (Hint!!! for 2 ชั้น)

```
*
**
***
****
*****
```

แบบฝึกหัด

3. จงเขียนโปรแกรมเพื่อนับจำนวนวิชาที่ลงทะเบียนและจำนวนหน่วยกิตรวม โดยอ่านข้อมูลการลงทะเบียนจากไฟล์ register.txt โดยมีตัวอย่างข้อมูลดังนี้

register.txt

```
08141101 3
08141102 3
08141103 3
08141104 1
08141105 4
```

ผลการรันโปรแกรม

```
register 5 courses
Total credit: 14
```

Functions

Functions

Function definition

- parameter
- return
- local variable
- global variable

Functions

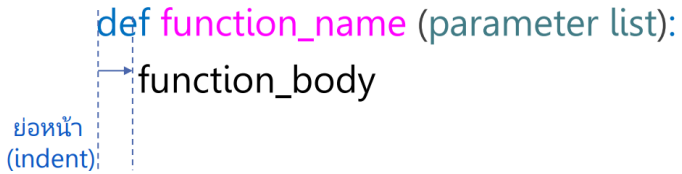
การใช้งานฟังก์ชัน

- เรียกว่า subprogram หรือ subroutine
- เป็นการแยกคำสั่งที่ซ้ำๆ กัน หรือที่เข้าใจยาก ออกมาจากโปรแกรมหลัก
- ทำให้โปรแกรมอ่านง่าย เข้าใจง่าย
- ทำให้โปรแกรมหลักเรียกใช้ฟังก์ชันได้ โดยไม่ต้องเขียนคำสั่งหลายๆ รอบ
- ฟังก์ชันควรมีหน้าที่การทำงานที่ชัดเจน เช่น ฟังก์ชันการหาค่ารากที่สอง ฟังก์ชันการหาค่าเฉลี่ย
- ต้องเขียนฟังก์ชันไว้ก่อนส่วนที่จะเรียกใช้

ในภาษาไพทอนมีฟังก์ชันให้ใช้งาน(Built-in function) อยู่แล้วมากมาย ในบทนี้เราจะมาเรียนการสร้างฟังก์ชันเอง

Function definition (การนิยามฟังก์ชัน)

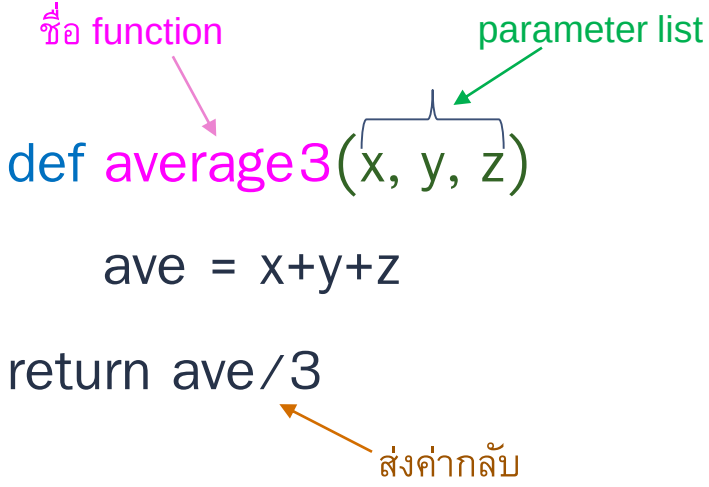
โครงสร้างของฟังก์ชัน



- **def** คือ keyword ที่บอกว่า เป็นจุดเริ่มต้นการนิยามฟังก์ชัน
- **function name** ชื่อฟังก์ชัน มีกฎการตั้งชื่อเหมือนกันกับการตั้งชื่อตัวแปร
- **parameter list** ภายในวงเล็บ คือ ชื่อตัวแปรที่รับค่าที่ส่งเข้ามาทำงานภายในฟังก์ชัน
  - ถ้าไม่มีให้ปล่อยว่างไว้
  - ถ้ามีมากกว่า 1 ตัวแปรให้คั่นด้วยเครื่องหมาย comma (,)
- จบบรรทัด **def** ด้วยเครื่องหมาย colon (:)
- **function_body** คือชุดคำสั่งโปรแกรมที่จะทำงานให้ได้ผลลัพธ์ตามที่ต้องการ เขียนคำสั่งภายในฟังก์ชันโดยต้องเยื้องย่อหน้า

ตัวอย่าง Function

Function เพื่อหาค่าเฉลี่ยของเลขจำนวนเต็ม 3 จำนวน



ค่าที่รับเข้ามา หรือ พารามิเตอร์ และการคืนค่าจากฟังก์ชันด้วยคำสั่ง return (ไม่จำเป็นต้องมีก็ได้)

Parameter list

- รายการของชื่อตัวแปรที่รับค่ามาจากคำสั่งการเรียกใช้ฟังก์ชันที่ส่งค่าข้อมูลเข้ามาทำงานภายในฟังก์ชัน
  - ถ้าไม่มีให้ปล่อยว่างไว้
  - ถ้ามีมากกว่า 1 ตัวแปรให้คั่นด้วยเครื่องหมาย comma (,)

- ถ้าตัวแปร parameter มีชนิดเป็น int, float, string, boolean จะมีการสร้างเนื้อหาที่ใหม่ในหน่วยความจำเพื่อเก็บค่าของตัวแปร
  - **ถ้าตั้งชื่อตัวแปร parameter ซ้ำกับชื่อตัวแปรในโปรแกรมหลัก โปรแกรมจะมองเป็นคนละตัวแปรกัน**
  - เมื่อทำงานจนจบฟังก์ชันแล้ว จะคืนเนื้อที่ในหน่วยความจำที่จองไว้เก็บค่าตัวแปร parameter เพื่อให้โปรแกรมอื่นนำเนื้อที่นั้นไปใช้งานต่อไป

- ถ้าตัวแปร parameter มีชนิดเป็น list, dict จะมีการเชื่อมตัวแปรพารามิเตอร์นี้ไปผูกกับค่าตัวแปรที่ผู้เรียกใช้ฟังก์ชันส่งเข้ามา
  - **ถ้ามีการแก้ค่าในฟังก์ชัน ก็จะแก้ค่าตัวแปรนั้นในส่วนของผู้เรียกใช้ฟังก์ชันด้วย**

Local variable

ตัวแปรในฟังก์ชัน local variable

- ตั้งชื่อซ้ำกับตัวแปรในฟังก์ชันอื่นได้ ถือว่าเป็นคนละตัวกัน
- ขอบเขตการมองเห็นตัวแปรแบบ local ถูกเรียกใช้ได้จากคำสั่งที่อยู่ภายในฟังก์ชันเท่านั้น
- เมื่อทำงานจนจบฟังก์ชันแล้ว จะมีการคืนเนื้อที่ในหน่วยความจำที่จองไว้เก็บค่าตัวแปรแบบ local variables เหล่านี้ เพื่อให้โปรแกรมอื่นนำเนื้อที่นั้นไปใช้งานต่อไป

return statement

การคืนการทำงานจากฟังก์ชันและคืนค่าจากฟังก์ชัน

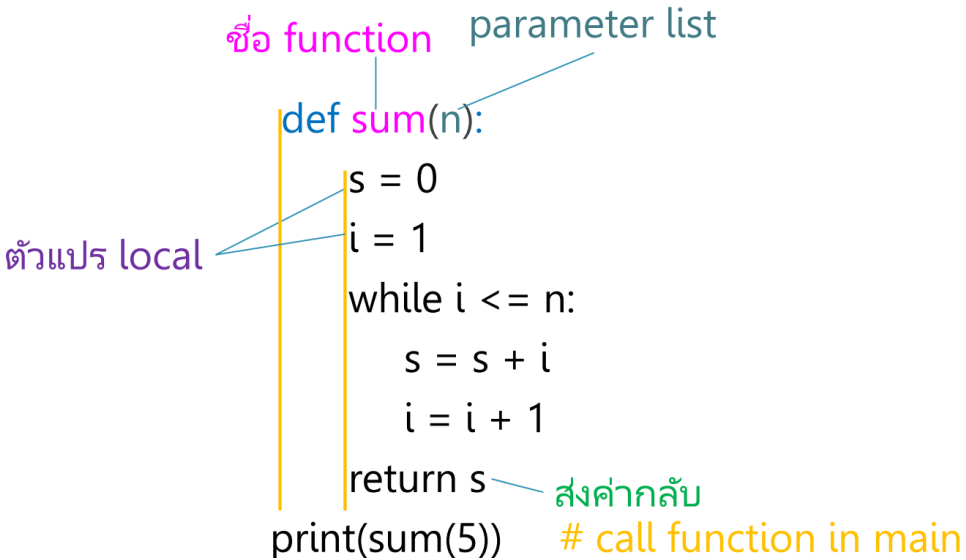
- สามารถใช้คำสั่ง return ได้หลายที่ในฟังก์ชัน
- เมื่อทำคำสั่ง return แล้ว จะหยุดการทำงานของฟังก์ชันทันที และกลับไปทำงานต่อหลังจุดที่เรียกใช้ฟังก์ชัน
- ฟังก์ชันคืนค่าได้ 1 ค่าเท่านั้น ถ้าต้องการคืนค่าหลายค่า ให้ใช้ **list**, tuple เช่น

  return –a **#เช่น a=10** return –x **#เช่น x=[1, 2, 3]** return (answer1, answer2)

- หากคำสั่งสุดท้ายของฟังก์ชันไม่ใช่ return ระบบจะเพิ่มคำสั่ง return (ไม่คืนค่าใดๆ) ที่ท้ายในฟังก์ชัน
- คำสั่ง return ที่ไม่ได้กำหนดให้คืนค่าใดๆ ระบบจะคืนค่า None (None เป็นค่าพิเศษในระบบ ไม่ใช่สตริง 'None')

ตัวอย่าง Function

เขียน function เพื่อหาผลรวมของ 1 ถึง n



ตัวอย่าง Function เพื่อหาผลรวมของ 1 ถึง n

การรันโปรแกรม

```
def sum(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
#in main program
print(sum(5))      # เรียกใช้ function ในโปรแกรมหลัก
```

```
#ไม่ใช้ฟังก์ชัน
s = 0
i = 1
while i <= 5:
   s = s+i
   i = i+1
print(s)
```

ตัวอย่าง Function เพื่อหา n!

```
def find_fac(n):
        fac = 1
        for x in range(1, n+1):
            fac = fac * x
        return fac
#in main program
n = int(input("Enter n : "))
print(find_fac(n))        # เรียกใช้ function
```

```
#ไม่ใช้ฟังก์ชัน
n = int(input('Enter n : '))
fac = 1
for x in range(1, n+1):
  fac = fac * x
print(fac)
```

```
#ในโปรแกรมหลักอาจเขียนคำสั่งเป็นแบบนี้
n = int(input("Enter n : "))
fac = find_fac(n) # เรียกใช้ function
print(fac)
```

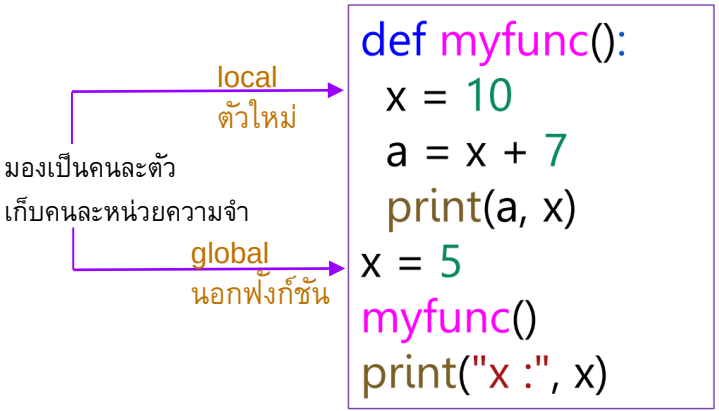
ลองรันโปรแกรม สังเกต ตัวแปร n, fac

Global variables

- ตัวแปรที่อยู่นอกขอบเขตของฟังก์ชัน
- ฟังก์ชันสามารถใช้งานตัวแปร global ได้
- ถ้าในฟังก์ชันต้องการเอาค่าของตัวแปร global มาใช้งาน สามารถอ้างชื่อตัวแปร global ได้

ข้อควรระวังแบบนี้ไม่ได้ใช้ตัวแปร global

ถ้าในฟังก์ชันมีการสร้างตัวแปรชื่อเดียวกันกับตัวแปร global ที่อยู่นอกฟังก์ชัน จะถือว่าเป็นการสร้างตัวแปร local ใหม่



ผลลัพธ์ที่ได้คือ
17 10
x : 5

ค่า x ไม่มีการอัพเดตเพราะเก็บคนละหน่วยความจำ

Global variables

- ถ้าในฟังก์ชันต้องการแก้ไขหรือกำหนดค่าของตัวแปร global คือให้มีการแก้ค่าภายในฟังก์ชันและส่งผลกับตัวแปร global ด้วย
- จะต้องประกาศให้ชัดเจน ว่าจะใช่ตัวแปร **global** นั้น

```
def myfunc():
  global x
  x = 10
  a = x + 7
  print(a, x)
x = 5
myfunc()
print("x :", x)
```

```
ผลลัพธ์ที่ได้คือ
17 10
x : 10
```

ค่า x เป็น global มีการอัพเดตค่าได้เพราะเก็บไว้ที่หน่วยความจำเดียวกัน

ตัวอย่าง: แสดงผลลัพธ์ที่ได้

def f(a, b):
  a = 10
  b = 0
  return a+b
 	
x = 5
y = 7
f(x,y)
print(x, y)
	def f(a, b):
	  a = 10
	  b = 0
	  return a+b
	
	x = 5
	y = 7
	a = f(x,y)
	print(a, x, y)
		def f(a, b):
		  a = 10
		  b = 0
		  return a+b
		
		a = 5
		b = 7
		c = f(a,b)
		print(a, b, c)

5 7	10 5 7	5 7 10

Exercise

จงแสดงผลลัพธ์ที่ได้

```
def f(a, b):
    a = 10
    b = 0
    x = 0
    y = 1
    print(x, y)
    return a-b

x = 5
y = 7
z = f(x, y)
print(x, y, z)
```

```
def f(a, b):
    global x, y
    a = 10
    b = 0
    x = 0
    y = 1
    print(x, y)
    return a-b

x = 5
y = 7
z = f(x, y)
print(x, y, z)
```

จงแสดงผลลัพธ์ที่ได้

```
def f(a, b):
    a = 10
    b = 0
    x = 0
    y = 1
    print(x, y)
    return a-b

x = 5
y = 7
z = f(x, y)
print(x, y, z)
```

```
def f(a, b):
    global x, y
    a = 10
    b = 0
    x = 0
    y = 1
    print(x, y)
    return a-b

x = 5
y = 7
z = f(x, y)
print(x, y, z)
```

แบบฝึกหัด

1. จงเขียนฟังก์ชันรับพารามิเตอร์เป็นจำนวนเต็ม 1 จำนวน แล้วส่งค่ากลับเป็นค่ากำลัง 2 ของจำนวนเต็มนั้น และให้เขียนโปรแกรมในส่วนของโปรแกรมหลักเพื่อรับค่าจำนวนจริง 1 จำนวนแล้วเรียกใช้ฟังก์ชันและเอาผลที่ได้จากฟังก์ชันมาแสดงเป็นผลลัพธ์

   โครงสร้าง def double(…):

2. จงเขียนฟังก์ชันรับพารามิเตอร์เป็นอุณหภูมิแบบองศาเซลเซียสแล้วเปลี่ยนอุณหภูมิเป็นหน่วยเคลวิน

   ตามสูตร $f = \frac{9}{5} * c + 32$

   โครงสร้าง def kelvin(…):

แบบฝึกหัด

3. จงเขียนโปรแกรมที่แสดงตัวเลือกให้ผู้ใช้ 5 ตัวเลือกได้แก่

1 คำนวณผลบวก 2 คำนวณผลลบ 3 คำนวณผลหาร 4 คำนวณผลคูณ 5 ออกจากโปรแกรม จากนั้นวนรับตัวเลือกจากผู้ใช้และทำงานตามตัวเลือก โดยให้สร้างเป็นฟังก์ชันสำหรับแต่ละตัวเลือก

ตัวอย่างการทำงาน

```
What would you like to do?
1: add
2: minus
3: divide
4: multiply
5: exit

Start value  500
Enter your choice: 1
Enter a number for adding: 300
Enter your choice: 2
Enter a number for subtracting: 100
Enter your choice: 3
Enter a number for dividing (not zero): 5
Enter your choice: 4
Enter a number for multiplying (not zero): 10
Enter your choice: 5
Remaining value : 800
```

ตัวอย่างการทำงาน

```
What would you like to do?
1: add
2: minus
3: divide
4: multiply
5: exit
Start value500
Enter your choice: 50
Invalid choice
Enter your choice: 5
Remaining value : 500
```

End