Function 1

Friday, October 10, 2025 8:00 AM

Function

- เคยเจอแล้ว: print(), input(), int(), range(), len() (ครั้งที่แล้ว), ... sum(), min()
- เราจะสร้าง function เอง!!

↳ why?
- code เราอ่านได้ง่ายขึ้น
- เราจะลด code ซ้ำๆ ลงได้

การสร้าง Function

ข้อมูลเข้าไปใน function → parameters

```
def function_name(optional parameters):
    statement
    statement
    statement
```

function block ↓ รันทุกครั้งที่เรียก function

ช่องว่างบอกว่าอยู่ใน function

การใช้งาน

```
function_name( )
```

จะไปรัน code ข้างใน

ใส่ถ้า function มี parameters

intuition: function ≈ mini-program

Simple function: จะแสดงผลชื่อ

สร้าง function = save file ✓

```
def print_name():
    print("Nut")
```

ใช้ function ✓

```
print_name()
```

เข้าไปใน function program

Output:

Nut

สร้าง ✓

```
def greeting():
    print("Welcome")
    print("Nice day")
    print("Good bye")
```

ใช้ ✓

```
greeting()
```

Welcome

Nice day

Good bye

```python
greeting()          # เรียกใช้
for i in range(3):
    greeting()      # loop1, loop2, loop3
```

Output:
Welcome
Nice day
Good bye } loop#1
Welcome
Nice day
Good bye } loop#2
Welcome
Nice day
Good bye } loop#3

```python
def print_namex3():         # สร้าง
    for i in range(3):
        print("Nut")
print_namex3()              # เรียกใช้
```

Nut
Nut
Nut

- **One parameter (input) function**

↳ จะส่งข้อมูลจากภายนอก เข้าไปใน function ใช้

Syntax:

```python
def function_name(param):
    statement
    statement
```

param = 20

param ← ตัวแปร — สามารถใช้ได้ใน function!! เก็บค่าเวลาเรียกใช้

Usage: `function_name(20)` (เก็บไว้ใน param)

Ex

```python
def greeting(name):               # สร้าง   name = "Nut"
    print('Welcome', name)        # "Nut"
greeting("Nut")                   # เรียกใช้
```

Welcome Nut

- อยู่ใน function นี้เท่านั้น!!
- ไปเรียกที่อื่น แล้ว error!

→ไป เรียกชื่ออื่น แล้ว error!

Multiple parameters

↳ คั่น ชื่อตัวแปรด้วย ,

"ABC" "XYZ" → parameters

```
def print_fullname(first, last):
    print(f"My name is {first} {last}")   → My name is ABCXYZ
print_fullname("ABC", "XYZ")
```

arguments

* map ค่าเข้า parameters ตามลำดับ (ตำแหน่ง) จากซ้ายไปขวา (positional arguments)

Ex สร้าง functions หาผลบวกเลข

```
def add(a, b):        # a = 5, b = 7
    total = a + b     # 5 + 7 = 12
    print(total)      → 12
add(5, 7)
```

```
def do_something(a, b, c):      # a = 1, b = 2, c = 3
    print(a + b * c)            # 1 + 2 * 3 → 7
do_something(1, 2, 3)
```

Keyword arguments

↳ เรียกชื่อตัวแปรเลย

ตอนเรียก! เราบอกเลย

Ex → do_something(a=1, b=2, c=3)

```
def do_x(a, b):
```

a = 5 , b = 7

```
def do_x(a,b):
    print(a*b)

do_x(a=2,b=3)

x = 5
y = 7
do_x(a=x, b=y)
```

a = 5 , b = 7

6 print(5*7)

do_x(b=3, a=2) ✓

35

* การใช้งาน position กับ key word ร่วมกัน
  - ↳ position ต้อง มาก่อน key word !!

do_x(5, b=7) ✓

do_x(a=5, 7) ✗

Default value

↳ ความสามารถ set ค่าตั้งต้นของ parameter ได้

Syntax:
```
def my_func(a=10)
    statement
```
↑ default

ถ้าไม่ใส่ค่า a เข้ามา จะใช้ค่านี้ !!

แต่ถ้าใส่ค่าใช้ค่านั้นแทน

EX
```
def do_y(a=10):
    print(a*2)

do_y()
do_y(20)
```
10 → 20

a=20 → 40

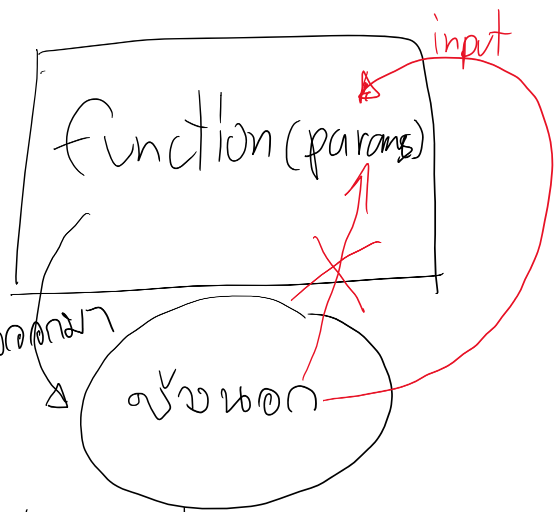
EX
```
def do_this(a, b=7, c=3):
```
5

Ex

```
def do_this(a, b=2, c=3):
    total = a + b*c
    print(total)            → 26

do_this(a=5, b=7)
```

(annotations: a=5, b=7; 5 + 7*3)

* default ต้อง มา หลัง parameters ที่ไม่มี default



เช่น len() ← ส่งออกมา
ส่งตัวเลขออกมา
หรือ input()

return statement

↳ ส่ง ค่า ออกมา จาก function → แล้ว ออกจาก function เลย

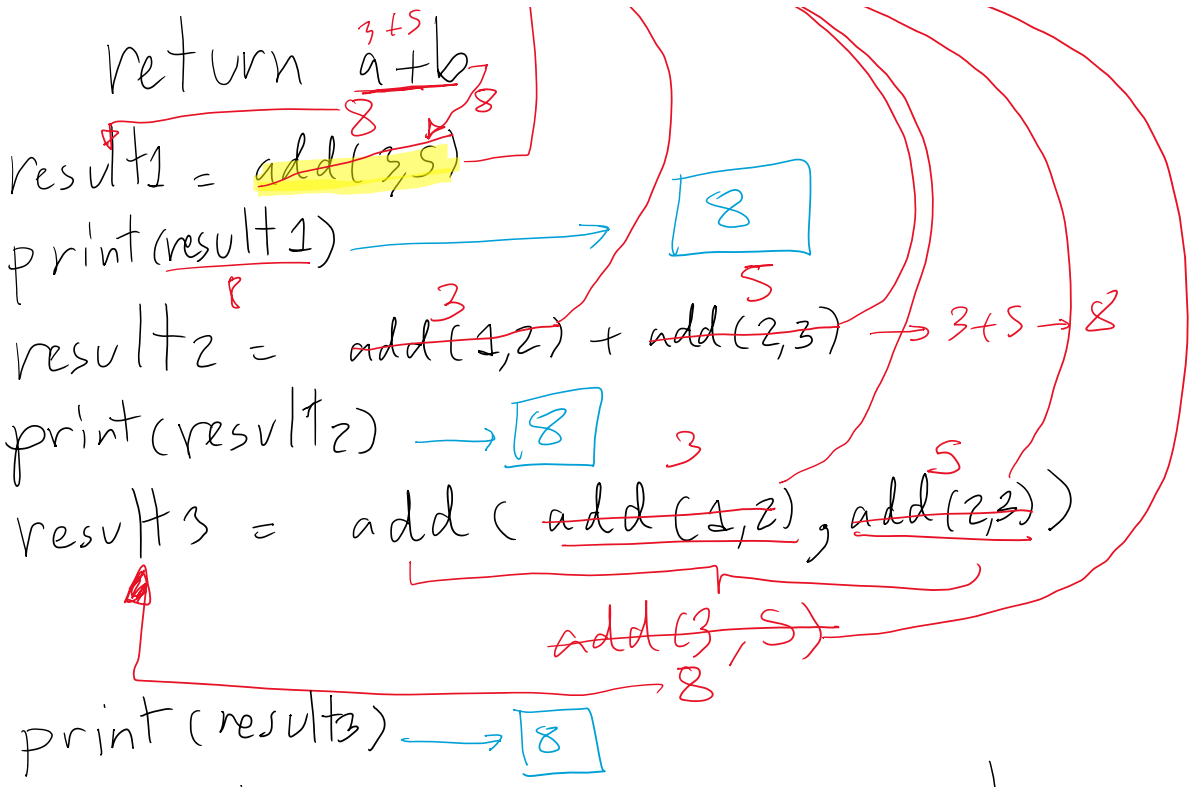
Ex

```
✓ def send_5():
      return 5

✓ output = send_5()
  print(output)           → 5

  send_5()                → 5
```

→ ไม่ได้แสดงผล
→ ไม่ได้เอาไปเก็บไว้!

```
def add(a, b):
    return a+b
```

(annotations: a=3, b=5; 3+5)



return vs print

```
def add(a,b)
    return a+b
result = add(3,5)
print(result)
print(add(3,5))
```

8 8 8

```
def print_add(a,b)
    print(a+b)
print_add(3,5)
```

print_add(3,5) → 8

```
def do_x(a,b):
    print("inside do_x")
    return a*b
result = do_x(3,5)
print(result)
```

3 5 3*5=15 15

inside do_x

15

print(result) → 15

```
def doy():
    print("A")
    return "B"
    print("C")
out = doy()
print(out)
```

A

B

Ex

```
def our_sum(lst):
    total = 0
    for item in lst:
        total += item
    return total
result = our_sum([1,2,3])
print(result)
```

[1,2,3] → 6 → 6

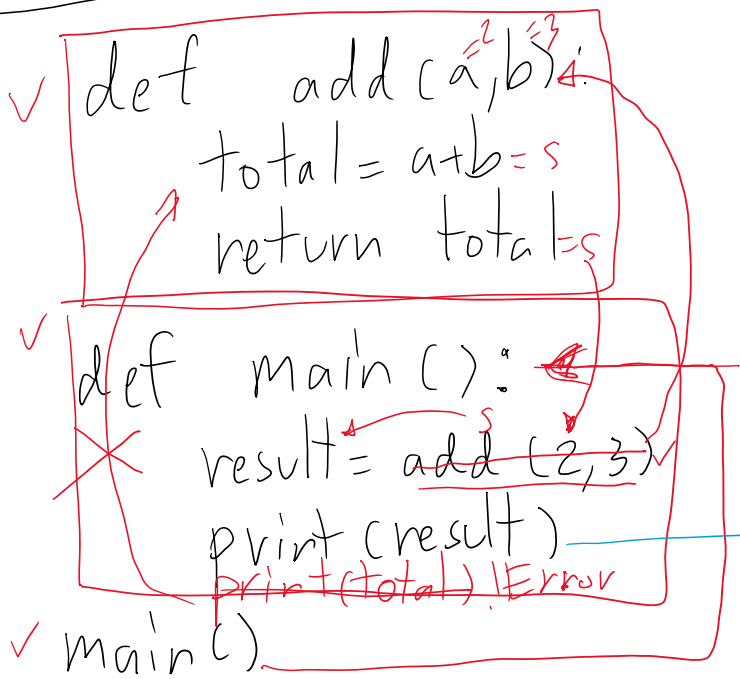
Ex input 內建 function

```
def get_user_input(msg):
    user_input = input(msg)
    return user_input
user = get_user_input("Enter your name:")
print(user)
```

user = ...

print(user)

EX



ใช้ เป็น main logic/ จุดเริ่มต้น

---

multiple returns

↳ ต้อง ด้วย commas

```
def multi()
    return 2, 3, 4
```

Usage:

1) สร้าง ตัว แปร จำนวน เท่า กับ ของ ที่ออก

a, b, c = ~~multi()~~ (2,3,4)

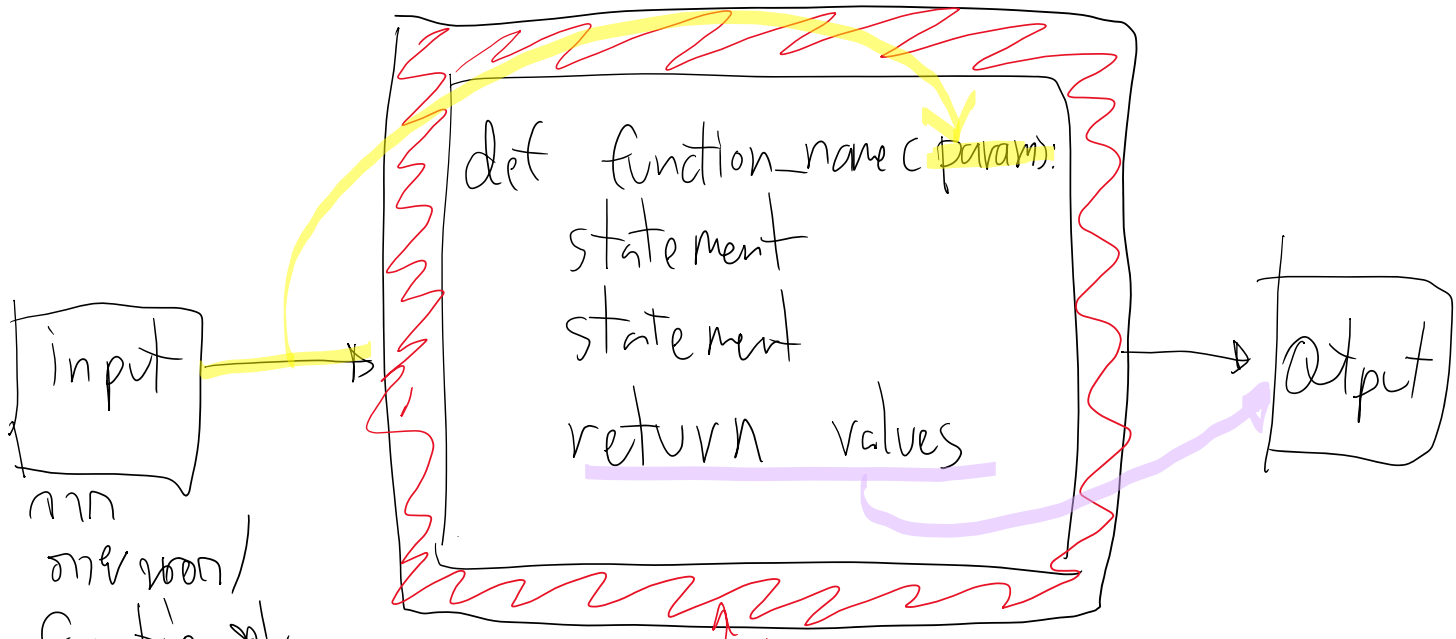
2) สร้าง ตัว แปร ตัว เดียว จะ เก็บ เป็น "tuple" ไว้ (คล้าย list แต่ แก้ไขค่า ไม่ได้)

result = ~~multi()~~ (2,3,4)

print(result[0]) → 2

(1, 2, 3, 4)

print(result[0]) → 2



ค่าที่ส่งออกจาก function

เข้าคู่กับตัวแปรใน function