



บทที่ 8 ฟังก์ชัน (Function)

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

- เข้าใจหลักการของฟังก์ชัน
- สามารถสร้างฟังก์ชันอย่างง่ายได้
- เข้าใจตัวแปรชนิด โกลบอลและโลคัล
- เข้าใจการส่งค่าระหว่างฟังก์ชันแบบต่างๆ
- สร้างฟังก์ชันที่มีการรับส่งค่าระหว่างฟังก์ชันได้

- ฟังก์ชัน คือ ชุดของการทำงาน ที่ถูกเขียนขึ้นให้โปรแกรมเมอร์สามารถเรียกใช้งานได้ง่าย

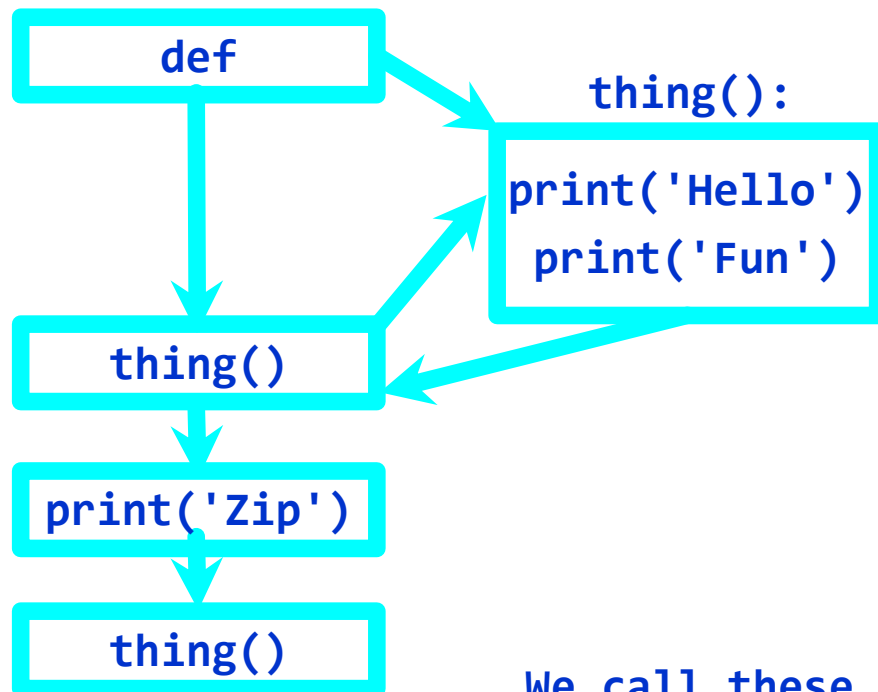
**** ฟังก์ชัน คือ ชุดของการทำงาน**

**** ฟังก์ชัน ถูกเรียกใช้งานได้**

- ทำให้โปรแกรมเมอร์สามารถพัฒนาโปรแกรมได้โดยง่าย โดยโปรแกรมเมอร์ไม่จำเป็นต้องทราบว่าการทำงานของฟังก์ชันทำงานอย่างไรทราบเพียงผลลัพธ์ของการทำงานเท่านั้น
- โปรแกรมเมอร์สามารถเขียนโปรแกรมให้มีการทำงานที่ซับซ้อนได้ โดยไม่จำเป็นต้องเขียนโปรแกรมส่วนที่ซับซ้อนนั้นหลายๆ ครั้ง
- โปรแกรมเมอร์สามารถออกแบบฟังก์ชันได้ตามความต้องการของโปรแกรมเมอร์

Stored (and reused) Steps

01006012 Computer Programming



Program:

```
def thing():
    print('Hello')
    print('Fun')
```

```
thing()
print('Zip')
thing()
```

Output:

```
Hello
Fun
Zip
Hello
Fun
```

We call these reusable pieces of code
“functions”

Python Functions



01006012 Computer Programming

There are two kinds of functions in Python.

- Built-in functions that are provided as part of Python - print(), input(), type(), float(), int() ...
- Functions that we define ourselves and then use

We treat the built-in function names as “new” reserved words (i.e., we avoid them as variable names)

Function Definition



01006012 Computer Programming

In Python a function is some reusable code that takes arguments(s) as input, does some computation, and then returns a result or results

We define a function using the def reserved word

We call/invoke the function by using the function name, parentheses, and arguments in an expression

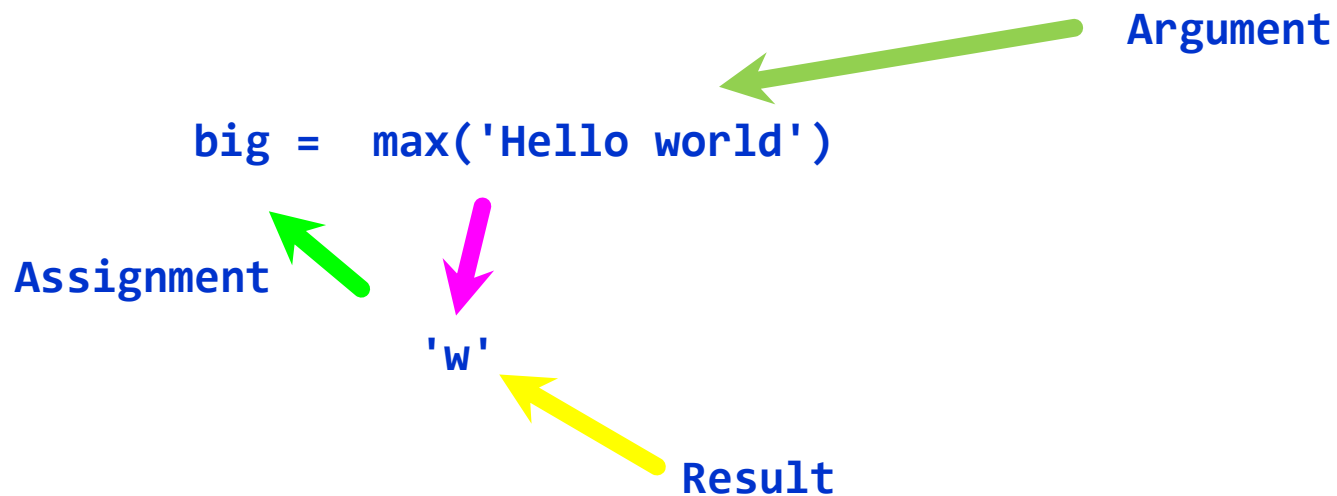
Argument

big = max('Hello world')

Assignment

'w'

Result



```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)

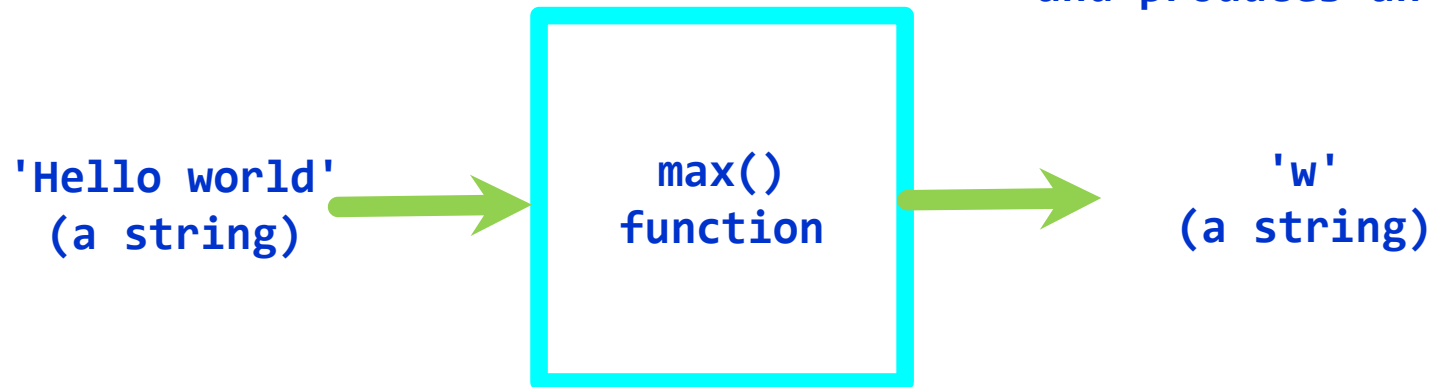
>>>
```


Max Function

01006012 Computer Programming

```
>>> big = max('Hello world')  
>>> print(big)  
w
```

A function is some stored code that we use. A function takes some input and produces an output.



Guido wrote this code

Max Function

01006012 Computer Programming

```
>>> big = max('Hello world')
>>> print(big)
w
```

A function is some stored code that we use. A function takes some input and produces an output.

'Hello world'
(a string)



```
def max(inp):
    blah
    blah
    for x in inp:
        blah
        blah
```



'w'
(a string)

Guido wrote this code



Building our Own Functions

01006012 Computer Programming

We create a new function using the `def` keyword followed by optional parameters in parentheses

We indent the body of the function

This **defines** the function but **does not** execute the body of the function

```
def print_lyrics():  
    print("I'm a lumberjack, and I'm okay.")  
    print('I sleep all night and I work all day.')
```



print_lyrics():

```
print("I'm a lumberjack, and I'm okay.")  
print('I sleep all night and I work all  
day.')
```

```
x = 5  
print('Hello')
```

```
def print_lyrics():  
    print("I'm a lumberjack, and I'm okay.")  
    print('I sleep all night and I work all day.')
```

```
print('Yo')  
x = x + 2  
print(x)
```

Hello
Yo
7

Definitions and Uses



01006012 Computer Programming

Once we have defined a function, we can call (or invoke) it as many times as we like

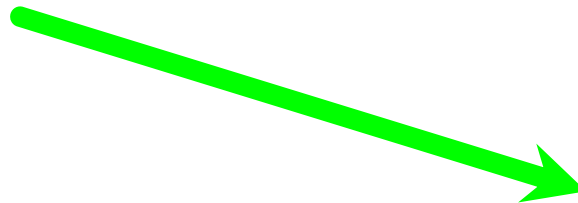
This is the store and reuse pattern



```
x = 5  
print('Hello')
```

```
def print_lyrics():  
    print("I'm a lumberjack, and I'm okay.")  
    print('I sleep all night and I work all day.')
```

```
print('Yo')  
print_lyrics()  
x = x + 2  
print(x)
```



```
Hello  
Yo  
I'm a lumberjack, and I'm okay.  
I sleep all night and I work all day.  
7
```

Arguments



01006012 Computer Programming

An argument is a value we pass into the function as its input when we call the function

We use arguments so we can direct the function to do different kinds of work when we call it at different times

We put the arguments in parentheses after the name of the function

big = max('Hello world')



Argument



Parameters

01006012 Computer Programming

A parameter is a variable which we use in the function definition. It is a “handle” that allows the code in the function to access the arguments for a particular function invocation.

```
>>> def greet(lang):  
...     if lang == 'es':  
...         print('Hola')  
...     elif lang == 'fr':  
...         print('Bonjour')  
...     else:  
...         print('Hello')  
...  
>>> greet('en')  
Hello  
>>> greet('es')  
Hola  
>>> greet('fr')  
Bonjour  
>>>
```


Return Values



01006012 Computer Programming

Often a function will take its arguments, do some computation, and return a value to be used as the value of the function call in the calling expression. The return keyword is used for this.

```
def greet():  
    return "Hello"
```

```
print(greet(), "Glenn")  
print(greet(), "Sally")
```

Hello Glenn
Hello Sally

Return Value



01006012 Computer Programming

A “fruitful” function is one that produces a result (or return value)

The return statement ends the function execution and “sends back” the result of the function

```
>>> def greet(lang):  
...     if lang == 'es':  
...         return 'Hola'  
...     elif lang == 'fr':  
...         return 'Bonjour'  
...     else:  
...         return 'Hello'  
...  
>>> print(greet('en'),'Glenn')  
Hello Glenn  
>>> print(greet('es'),'Sally')  
Hola Sally  
>>> print(greet('fr'),'Michael')  
Bonjour Michael  
>>>
```

Arguments, Parameters, and Results



01006012 Computer Programming

```
>>> big = max('Hello world')  
>>> print(big)  
w
```

Argument → 'Hello world'

Parameter

```
def max(inp):  
    blah  
    blah  
    for x in inp:  
        blah  
        blah  
    return 'w'
```

'w'
↑
Result

Multiple Parameters / Arguments



01006012 Computer Programming

We can define more than one parameter in the function definition

```
def addtwo(a, b):  
    added = a + b  
    return added
```

We simply add more arguments when we call the function

```
x = addtwo(3, 5)  
print(x)
```

We match the number and order of arguments and parameters

8



Void (non-fruitful) Functions

01006012 Computer Programming

When a function does not return a value, we call it a “void” function

Functions that return values are “fruitful” functions

Void functions are “not fruitful”

Default arguments

```
def add4(a,b,c=0,d=99):  
    print(f"a={a} b={b} c={c} d={d}")  
    return a+b+c+d  
print(add4(3,5,9,8))  
  
print(add4(3,5,9))  
  
print(add4(3,5))
```

a=3 b=5 c=9 d=8

25

a=3 b=5 c=9 d=99

116

a=3 b=5 c=0 d=99

107

Local variable

```
def fname():  
    name = "Yam"  
    print("in function",name)  
name = 'Rat'  
fname()  
print("in main",name)
```

```
in function Yam  
in main Rat
```

Global variable



```
def addx():  
    print(f"addx={x}")  
x = 5  
addx()  
print(f"mainx={x}")
```

```
addx=6  
mainx=6
```


Global variable



01006012 Computer Programming

```
def addx():  
    print(f"addx={x}")  
    x += 1  
    print(f"addx={x}")  
  
x = 5  
addx()  
print(f"mainx={x}")
```

```
File "e:\function\globalvar.py", line 7,  
in <module>
```

```
    addx()
```

```
File "e:\function\globalvar.py", line 2,  
in addx
```

```
    print(f"addx={x}")
```

^

```
UnboundLocalError: cannot access local  
variable 'x' where it is not associated  
with a value
```

Global variable



01006012 Computer Programming

```
def addx():  
    global x  
    print(f"addx1={x}")  
    x += 1  
    print(f"addx2={x}")
```

```
x = 5  
addx()  
print(f"mainx={x}")
```

```
addx1=5  
addx2=6  
mainx=6
```

Keyword argument

```
def add4(a=0,b=0,c=0,d=99):  
    print(f"a={a} b={b} c={c} d={d}")  
    return a+b+c+d
```

```
print(add4(3,5,9,8))
```

```
a=3 b=5 c=9 d=8
```

```
25
```

```
print(add4(3,5,9))
```

```
a=3 b=5 c=9 d=99
```

```
116
```

```
print(add4(c=911,a=7))
```

```
a=7 b=0 c=911 d=99
```

```
1017
```

```
print(add4(7,c=911))
```

```
a=7 b=0 c=911 d=99
```

```
1017
```

Returning more than one value

```
def getstu():  
    name = "Guido"  
    surname = "Rossom"  
    return name,surname
```

```
name, surname = getstu()  
print(f"name={name} surname={surname}")
```

```
name=Guido surname=Rossom
```

To function or not to function...



01006012 Computer Programming

Organize your code into “paragraphs” - capture a complete thought and “name it”

Don't repeat yourself - make it work once and then reuse it

If something gets too long or complex, break it up into logical chunks and put those chunks in functions

Make a library of common stuff that you do over and over - perhaps share this with your friends...

Summary



01006012 Computer Programming

Functions

Default argument

Built-In Functions

Keyword argument

Type conversion (int, float)

Local variable

String conversions

Global variable

Parameters
Arguments

Why use functions?

Results (fruitful functions)

Void (non-fruitful) functions