

การใช้ Function ใน Python

Shutchon Premchaisawatt

Function ใน Python เป็นส่วนสำคัญของการเขียนโปรแกรมแบบเชิงโครงสร้าง (structured programming) ช่วยให้การเขียนโปรแกรมมีประสิทธิภาพมากขึ้น มีโครงสร้างที่ชัดเจน และง่ายต่อการอ่านและแก้ไข ในบทความนี้ เราจะเรียนรู้เกี่ยวกับฟังก์ชันใน Python ตั้งแต่พื้นฐานไปจนถึงเทคนิคขั้นสูง

```
class BigFile:
    def __init__(self, datadir, ndims):
        idfile = os.path.join(datadir, "id.txt")
        self.names = [x.strip() for x in str.split(open(idfile).read()) if x.strip()]
        self.name2index = dict(zip(self.names, range(len(self.names))))
        self.ndims = ndims
        self.featurefile = os.path.join(datadir, "feature.bin")
        print "[BigFile] %d features, %d dimensions" % (len(self.names), self.ndims)
        print "        binary: %s" % self.featurefile
        print "        txt: %s" % idfile

    def read(self, requested, isname=True):
        if isname:
            index_name_array = [(self.name2index[x], x) for x in requested if x in self.names]
        else:
            assert(min(requested) >= 0)
            assert(max(requested) < len(self.names))
            index_name_array = [(x, self.names[x]) for x in requested]
            index_name_array.sort()

        vecs = seq_read(self.featurefile, self.ndims, [x[0] for x in index_name_array])
        return [x[1] for x in index_name_array], vecs

    def shape(self):
        return [len(self.names), self.ndims]
```



ประโยชน์ของการใช้ Function

การใช้ Function ใน Python นั้นมีประโยชน์มากมาย

- 1 การนำกลับมาใช้ใหม่ (Reusability)**
Function ช่วยให้เราสามารถนำโค้ดส่วนหนึ่งมาใช้ซ้ำๆ ได้หลายครั้ง โดยไม่ต้องเขียนโค้ดใหม่ทั้งหมด ตัวอย่างเช่น ถ้าเราต้องการเขียนโค้ดเพื่อคำนวณค่าเฉลี่ย เราสามารถสร้าง Function ที่รับ input เป็น list ของตัวเลข แล้ว return ค่าเฉลี่ย เราสามารถเรียกใช้ Function นี้ได้หลายครั้ง โดยส่ง list ที่ต่างกันไปเป็น input
- 2 ความชัดเจนและความเป็นระเบียบ (Clarity & Organization)**
Function ช่วยให้โค้ดของเรามีโครงสร้างที่ชัดเจน ทำให้โค้ดดูง่ายขึ้น และเข้าใจได้ง่ายขึ้น โดยเฉพาะในกรณีที่โค้ดมีขนาดใหญ่
- 3 การแก้ไขที่ง่ายขึ้น (Easier Debugging)**
Function ช่วยให้เราสามารถแยกส่วนโค้ดออกเป็นส่วนๆ ทำให้เราสามารถแก้ไขโค้ดได้ง่ายขึ้น โดยที่ไม่ต้องแก้ไขโค้ดทั้งหมด
- 4 ลดความซับซ้อน (Reduce Complexity)**
Function ช่วยลดความซับซ้อนของโค้ด โดยการแบ่งโค้ดออกเป็นส่วนๆ ทำให้โค้ดดูง่ายขึ้น และเข้าใจได้ง่ายขึ้น

การสร้าง Function

การสร้าง Function ใน Python นั้นทำได้ง่าย เราใช้คำสั่ง def ตามด้วยชื่อ Function และวงเล็บ

ภายในวงเล็บ เราสามารถใส่ตัวแปรที่ Function รับเป็น input (parameter) ได้

รหัสภายใน Function ควรเว้นวรรคจากขอบด้านซ้าย

ใช้คำสั่ง return เพื่อส่งค่ากลับจาก Function

```
def my_function(x, y):  
    """This function adds two numbers."""  
    return x + y
```

การส่งค่าไปยัง Function

เมื่อเราต้องการเรียกใช้ Function เราจะใส่ชื่อ Function ตามด้วยวงเล็บ

ภายในวงเล็บ เราสามารถส่งค่าไปยัง Function ได้ โดยค่าที่ส่งไปจะถูกเก็บไว้ในตัวแปรที่เราได้กำหนดไว้ใน Function

```
def my_function(x, y):  
    """This function adds two numbers."""  
    return x + y
```

```
result = my_function(5, 3)  
print(result) # Output: 8
```




Function Call – An Example

- If the function returns a value, then the returned value need to be assigned to a variable so that it can be stored

```
int GetUserInput (void); /* function prototype */  
int main(void)  
{  
    int input;  
    input = GetUserInput( );  
    return(0); /* return 0; */  
}
```

- However, it is perfectly okay (syntax wise) to just call the function without assigning it to any variable if we want to ignore the returned value
- We can also call a function inside another function
printf("User input is: %d", GetUserInput());



```
double sum(int num1, int num2)
{
    double result;
```

การรับค่าจาก Function

Function สามารถส่งค่ากลับไปยังส่วนโค้ดหลักที่เรียกใช้ Function โดยใช้คำสั่ง return

ค่าที่ส่งกลับจาก Function จะถูกเก็บไว้ในตัวแปร

```
def my_function(x, y):
    """This function adds two numbers."""
    return x + y
```

```
result = my_function(5, 3)
print(result) # Output: 8
```

การใช้ Default Parameter ใน Function

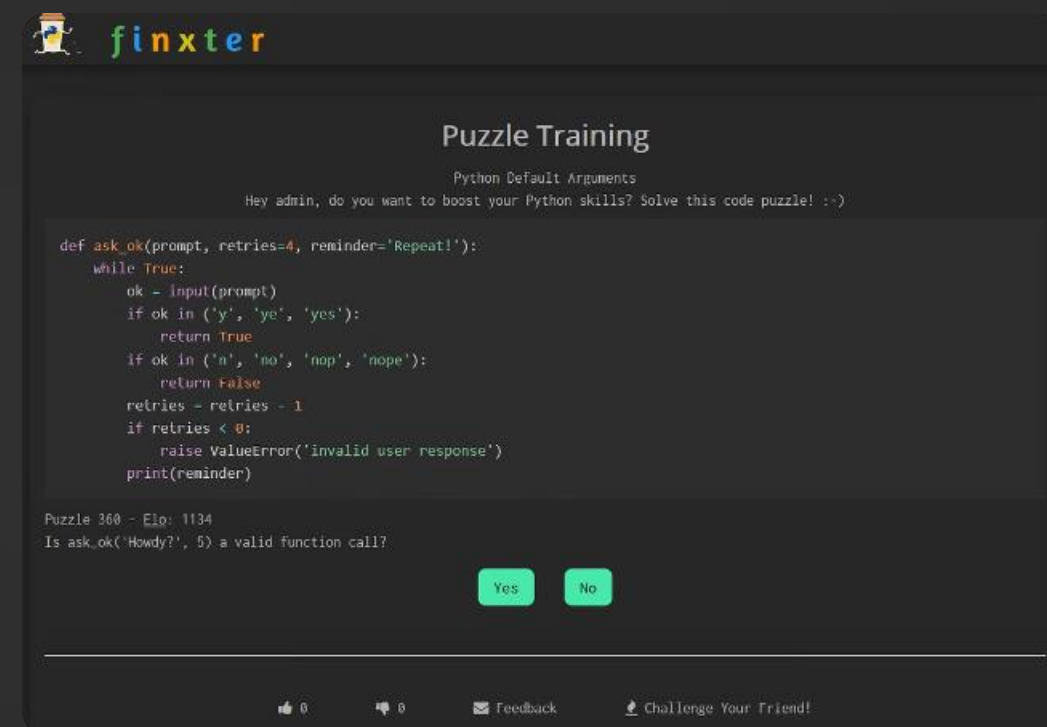
เราสามารถกำหนดค่าเริ่มต้นให้กับ parameter ใน Function โดยใช้เครื่องหมายเท่ากับ

หากเราไม่ส่งค่าไปยัง parameter ที่มีค่าเริ่มต้น Function จะใช้ค่าเริ่มต้น

```
def my_function(x, y=5):  
    """This function adds two numbers. If y is not provided,  
    it defaults to 5."""  
    return x + y
```

```
result1 = my_function(3) # y defaults to 5  
print(result1) # Output: 8
```

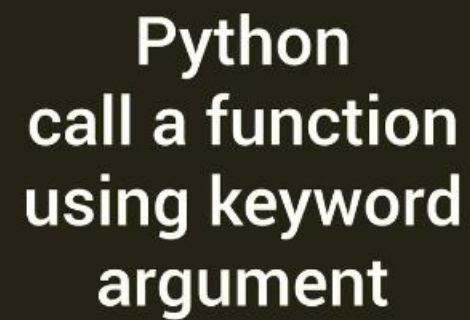
```
result2 = my_function(3, 10) # y is explicitly set to 10  
print(result2) # Output: 13
```



การใช้ Keyword Argument ใน Function

Keyword Argument ช่วยให้เราสามารถส่งค่าไปยัง Function โดยใช้ชื่อของ parameter

Keyword Argument ช่วยให้เราระบุค่าของ parameter ได้อย่างชัดเจน โดยเฉพาะในกรณีที่ Function มี parameter จำนวนมาก



Python
call a function
using keyword
argument

codevscolor.com

```
def my_function(x, y, z):  
    """This function takes three arguments."""  
    print(x, y, z)
```

```
my_function(x=1, z=3, y=2) # Keyword arguments  
allow us to specify values in any order  
# Output: 1 2 3
```


การใช้ Function ที่มีการรับค่าไม่จำกัด

Function ใน Python สามารถรับค่า input จำนวนไม่จำกัดได้ โดยใช้เครื่องหมายดอกจัน (*)

ค่า input ทั้งหมดจะถูกเก็บไว้ใน tuple



**Functions in Python
- Arbitrary Number
of Positional Parameters**

```
def my_function(*args):  
    """This function takes any number of arguments."""  
    for arg in args:  
        print(arg)  
  
my_function(1, 2, 3, 4, 5) # Pass any number of arguments  
# Output:  
# 1  
# 2  
# 3  
# 4  
# 5
```

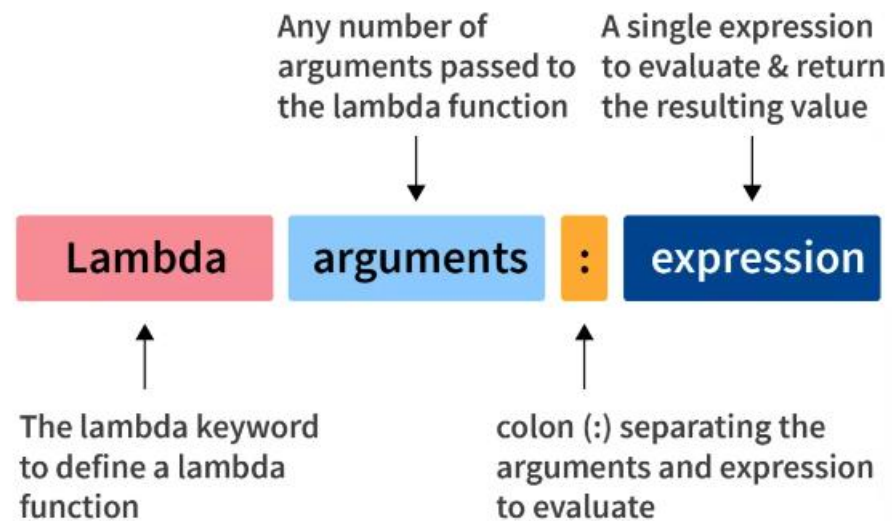

การใช้ Lambda Function

Lambda Function ใน Python เป็น Function ที่ไม่มีชื่อ

Lambda Function มักใช้ในกรณีที่เราต้องการ Function ที่มีขนาดเล็ก และใช้เพียงครั้งเดียว

Lambda Function ใช้สำหรับการเขียน Function แบบสั้นๆ โดยใช้แค่บรรทัดเดียว

```
double = lambda x: x * 2
print(double(5)) # Output: 10
```



```
@classmethod
def find_spec(cls, name, path, target=None):
    print(f"Module {name!r} not installed. Attempting to pip install")
    cmd = f"{sys.executable} -m pip install {name}"
    try:
        subprocess.run(cmd.split(), check=True)
    except subprocess.CalledProcessError:
        return None

    return util.find_spec(name)
```

การเรียกใช้ Function ที่อยู่ในโมดูลอื่น

Function ที่อยู่ในโมดูลอื่น สามารถเรียกใช้ได้โดยการ import โมดูลนั้น

การใช้คำสั่ง import จะทำให้เราสามารถเข้าถึง Function และ Class ทั้งหมดในโมดูลนั้น

```
import math
```

```
print(math.sqrt(16)) # Output: 4.0
```