Introduction to Python

Chapter 5 : Python User Defined Function

Akara Kijkarncharoensin

Computer Engineering and Financial Technology
University of the Thai Chamber of Commerce

Agenda

- Void Function : ฟังก์ชั่นแบบไม่ส่งค่ากลับ
- Value-return Function : ฟังก์ชั่นแบบคืนค่ากลับ
- Position Argument
- Keyword Argument
- Default Parameter
- Variadic Parameter
- Lambda Expression

Download เอกสาร **: https://github.com/akara-kij/Python101.git**

Void Function

- เป็นฟังก์ชั่นแบบไม่มีส่งค่าคืนกลับไปยังโปรแกรมที่เรียกใช้งาน

def ชื่อฟังก์ชั่น (พารามิเตอร์) **:**

Program code 1

- **def** ย่อมาจาก define ใช้ประกาศชื่อฟังก์ชั่น
- ชื่อฟังก์ชั่น เป็นไปตามการสร้างชื่อตัวแปรของภาษา Python
- พารามิเตอร์ เป็น input ที่จะส่งให้ฟังก์ชั่นใช้งาน

Void Function

- ตัวอย่างเช่น : ฟังก์ชั่นสำหรับบวกเลขสองจำนวน

ประกาศฟังก์ชั่น

Input parameter ของฟังก์ชั่น คือ **A** และ **B**

```
def sumAB(A,B) :
    Total = A + B
    print(Total)
```

การทำงานภายในฟังก์ชั่น

Indent (การเยื้อง)

- การใช้งานเพื่อหาผลรวมของเลข 5 และ 10
  - sumAB(5,10) # มีเลข 15 แสดงผลที่หน้าจอ

Value-return Function

- มีค่าคืนจากฟังก์ชั่นที่สร้างขึ้นกลับมายังส่วนที่เรียกใช้งาน

```
def sumAB(A,B) :
    Total = A + B
    return Total
```

ส่งค่ากลับไปยังโปรแกรมส่วนที่เรียกใช้

- คำสั่ง **return** จะส่งข้อมูลออกจากฟังก์ชั่นและหยุดทำงาน
- ตัวอย่างเช่น : x = sumAB(5,10) # ค่าของตัวแปร Total จะส่งออกมาสู่ตัวแปร x ซึ่งอยู่ภายนอก

Position Argument

- เป็น Input parameter ของฟังก์ชั่น โดยมีข้อกำหนดคือ
  - ลำดับ : ลำดับของ **Argument** ต้องตรงกับ ลำดับของ **Parameter**
  - จำนวน : จำนวน **Argument** ต้องตรงกับ จำนวนของ **Parameter**

Parameter

```
def sumAB(A,B) :
    Total = A + B
    return Total
```

ภายในฟังชั่น

Argument

```
x = sumAB(5,10)
```

ส่วนที่เรียกใช้งาน

Keyword Argument

- กำหนดชื่อและจับคู่ระหว่าง **Parameter** และ **Argument**
- หากระบุ **Argument** ด้วย **Keyword** ไม่จำเป็นต้องเรียงลำดับให้ตรงกับลำดับของ **Parameter**
- **Argument** ที่ไม่ได้ระบุ **Keyword** ยังต้องเรียงให้ตรงลำดับกับ **Parameter**

```
def sumABC(A,B,C) :
    Total = A+2*B+3*C
    return Total
```

ภายในฟังชั่น

```
x = sumABC(5,10,15)
x = sumABC(A=5,B=10,C=15)
x = sumABC(B=10,C=15,A=5)
x = sumABC(5,C=15,B=10)
```

Keyword Argument

เขียน **Position Argument** ก่อน

Default Parameter

- เป็นการกำหนดค่าล่วงหน้าให้แก่ **Parameter**
- หากได้กำหนดค่าไว้ล่วงหน้า ก็ไม่จำเป็นต้องใช้ **Argument** ส่งค่าให้แก่ฟังก์ชั่นอีก
- ภายในตัวฟังก์ชั่นให้เรียงลำดับ **Position Parameter** ก่อนแล้วจึงเป็น **Default Parameter**

```
def sumABC(A,B=10,C=15) :
    Total = A + B + C
    return Total
```

ถูกต้อง

```
def sumABC(A=5,B,C) :
    Total = A + B + C
    return Total
```

ผิดพลาด

Variadic Parameter

- Variadic Parameter สามารถรองรับ Argument ได้ไม่จำกัดจำนวน
- สามารถมี Variadic Parameter ได้เพียงหนึ่งตัวเท่านั้น
- นิยมวาง Vairadic Parameter ไว้ลำดับสุดท้าย
- ตัวอย่างเช่น

รองรับ **Argument** ได้ไม่จำกัดจำนวน

```
def length(*Vars) :
        return len( Vars )
```

นิยมวาง **Variadic Parameter** เป็นลำดับสุดท้าย

```
def ProductList(Name,*Vars) :
        print( Name + " has " + str( len(Vars) ) + " items" )
```

Lambda Expression

- เป็น Anonymous Function ประเภทหนึ่งซึ่งไม่ต้องระบุชื่อ หรือ ใช้ **def** ในการสร้างฟังก์ชั่น
- สามารถใช้ Lambda Expression เป็น Input Argument เข้าไปยังฟังก์ชั่นอื่นๆ ได้
- สามารถสร้างฟังก์ชั่นชนิดนี้ด้วย keyword "**lambda**"
- ตัวอย่าง :

```
def CalAB(A,B,Func) :
    return Func(A,B)
```

การทำงานเปลี่ยนไปตาม **input argument**

```
funcName = lambda a, b : a + b

Ans = CalAB( 5, 10, funcName ) # Ans = 15
```

ใช้ **Lambda Expression** เป็น **Argument**

UTCC University of the Thai Chamber of Commerce

มหาวิทยาลัยหอการค้าไทย