CSS 112 Computer Programming

Lecture V: Functions in Python

Suvil Chomchaiya (Ph.D.)

Contents

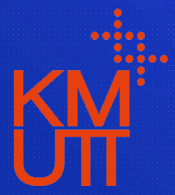
- Functions in Python
- Parameters and Arguments
- More Examples in Functions

Functions in Python

KMUTT

Functions in Python

- ฟังก์ชั่น (Function) คือ block ของ code ที่จะ Run เมื่อถูกเรียกใช้
- ฟังก์ชั่นสามารถรับ data ที่เรียกว่า parameters เข้าไปในฟังก์ชั่นได้ โดยฟังก์ชั่นจะคืนค่า (return) เป็นผลลัพธ์

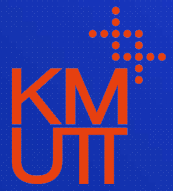


Functions in Python

- ฟังก์ชันออกเป็น 2 ประเภทหลักๆ คือ
  1. Pre-defined function (หรือ Built-in function)
     - ฟังก์ชันในกลุ่มนี้จะถูกสร้างไว้อยู่แล้ว
     - สามารถเรียกใช้ฟังก์ชันประเภทนี้ได้ทันที เช่น ฟังก์ชัน print(), range(), input() เป็นต้น
  2. User-defined Function
     - ฟังก์ชันที่ผู้เขียนโปรแกรมพัฒนาขึ้นมาใช้งานตามวัตถุประสงค์ของตนเอง
     - User-defined Function จะสร้างขึ้นเพื่อกระทำการต่างๆ เช่น การคำนวณหาผลลัพธ์ การแสดงผล หรือการอ่านค่าบางอย่าง
     - ลักษณะการทำงานที่จะนำมาสร้างเป็นฟังก์ชัน มักเป็นสิ่งที่ต้องทำซ้ำหลายๆ ครั้งในรูปแบบเดียวกัน

Pre-defined function (Built-in function)

- ฟังก์ชั่น (Function) คือ block ของ code ที่จะ Run เมื่อถูกเรียกใช้
- ฟังก์ชั่นสามารถรับ data ที่เรียกว่า parameters เข้าไปในฟังก์ชั่นได้ โดยฟังก์ชั่นจะคืนค่า (return) เป็นผลลัพธ์



Pre-defined function (Built-in function)

Function	Description
int(x [,base])	แปลงออบเจ็ค x จากฐานที่กำหนด base ให้เป็น Integer
long(x [,base])	แปลงออบเจ็ค x จากฐานที่กำหนด base ให้เป็น Long
float(x)	แปลงออบเจ็ค x ให้เป็น Floating point number
complex(real [,im])	สร้างตัวเลขจำนวนเชิงซ้อนจากค่า real และค่า imagine
str(x)	แปลงออบเจ็ค x ให้เป็น String
repr(x)	แปลงออบเจ็ค x ให้เป็น String expression
eval(str)	ประเมินค่าของ String
tuple(s)	แปลง Sequence ให้เป็น Tuple
list(s)	แปลง Sequence ให้เป็น List

Pre-defined function (Built-in function)

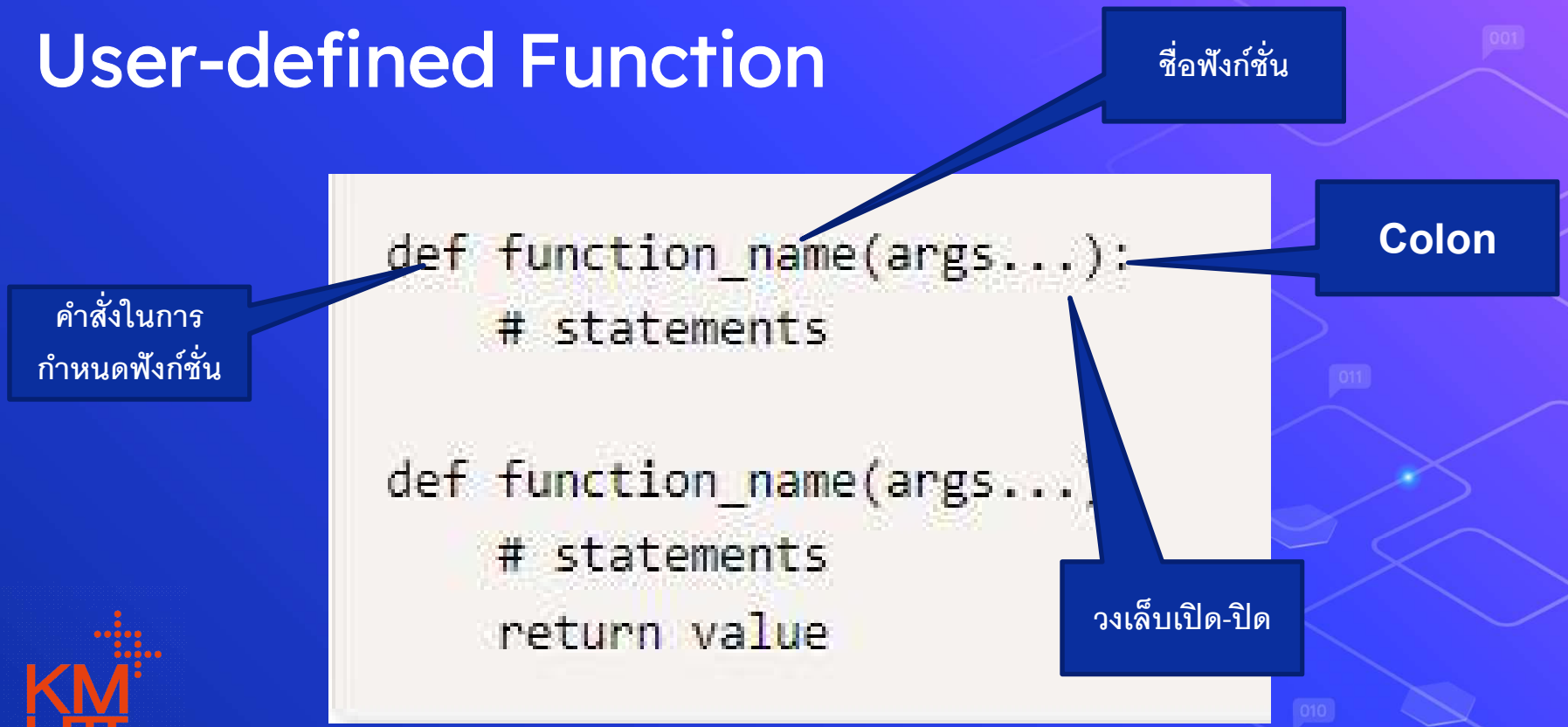
Function	Description
set(s)	แปลง Sequence ให้เป็น Tuple
dict(d)	แปลงออบเจ็คให้เป็น Dictionary
frozenset(s)	แปลงออบเจ็คให้เป็น Frozen set
chr(x)	แปลงค่าของ Integer ให้เป็น Unicode Char
ord(x)	แปลง Charterer ให้เป็นค่า Integer
hex(x)	แปลง Integer ให้เป็น Hex string
oct(x)	แปลง Integer ให้เป็น Oct string

User-defined Function

- รูปแบบของการสร้างก์ชั่นเพื่อใช้งานใน Python ใช้คำว่า def ตามด้วย ชื่อฟังก์ชั่นที่ต้องการ
- ชื่อฟังก์ชั่นที่สร้าง.....ห้ามซ้ำกับ Built-in Functions
- รูปแบบการสร้างฟังก์ชั่น…

```
def function_name(args...):
    # statements

def function_name(args...):
    # statements
    return value
```

User-defined Function



User-defined Function

- ใช้คำสั่ง def และหลังจากนั้น function_name เป็นชื่อของฟังก์ชัน
- ในวงเล็บ () เป็นการกำหนดพารามิเตอร์ของฟังก์ชัน
- พารามิเตอร์ของฟังก์ชันนั้นสามารถมีจำนวนเท่าไหร่ก็ได้หรือไม่มีก็ได้
- ฟังก์ชันอาจจะมีหรือไม่มีการส่งค่ากลับ สำหรับฟังก์ชันที่ไม่มีการ return ค่ากลับนั้น เรามักจะเรียกว่า โพรซีเยอร์ (Procedure)

User-defined Function - Example

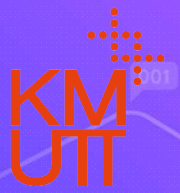
```
def hello(name):
    print('Hello %s' % name)

def count_vowel(str):
    vowel = 0
    for c in str:
        if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):
            vowel = vowel + 1
    return vowel

def area(width, height):
    c = width * height
    return c
```

User-defined Function - Example

```
def hello(name):
    print('Hello %s' % name)
```

- ฟังก์ชันแรกมีชื่อว่า hello() เป็นฟังก์ชันสำหรับแสดงข้อความทักทายจากที่ชื่อส่งเข้ามา
- ฟังก์ชันนี้มีหนึ่งพารามิเตอร์คือ name สำหรับรับชื่อที่ส่งเข้ามาในฟังก์ชัน

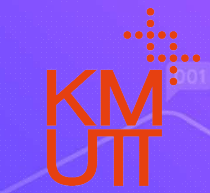
User-defined Function - Example




```
def count_vowel(str):
    vowel = 0
    for c in str:
        if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):
            vowel = vowel + 1
    return vowel
```

- ฟังก์ชัน count_vowel() เป็นฟังก์ชันสำหรับนับจำนวนสระใน String
- ฟังก์ชันนี้มีหนึ่ง String พารามิเตอร์
- ในการทำงานของฟังก์ชั่น ใช้คำสั่ง For loop ในการวนอ่านค่าทีละตัวอักษรเพื่อตรวจสอบว่าเป็นสระหรือไม่ด้วยคำสั่ง in
- ตัวแปร vowel นั้นใช้สำหรับนับจำนวนสระที่พบใน String ในตอนท้ายเราได้ส่งค่าของจำนวนสระที่นับได้กลับไปด้วยคำสั่ง return

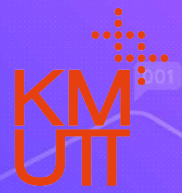
User-defined Function - Example




```
def area(width, height):
    c = width * height
    return c
```

- ฟังก์ชัน area() เป็นฟังก์ชันสำหรับหาพื้นที่ของรูปสี่เหลี่ยมด้านขนาน
- ฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม
- ฟังก์ชันทำการ return ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง return

User-defined Function - Example



- การเรียกใช้งานฟังก์ชั่น (Function Calling)
  - ในการเรียกใช้ฟังก์ชันนั้นเราจะใช้ชื่อของฟังก์ชันและส่ง Argument ให้สอดคล้องกับพารามิเตอร์ที่กำหนดไว้ในฟังก์ชัน
  - Argument คือค่าที่ส่งเข้าไปในฟังก์ชันตอนใช้งาน
  - พารามิเตอร์นั้นคือตัวแปรทีกำหนดไว้ในฟังก์ชันเพื่อรับค่าจาก Argument

User-defined Function - Example

- การเรียกใช้งานฟังก์ชั่น (Function Calling)

```
# calling functions
hello(‘Jake')
hello(‘Miyasaki')
print('Vowel in string = %d' % count_vowel(‘kmutt.ac.th'))
print('Vowel in string = %d' % count_vowel('Python'))
print('Area = %d' % area(18,14))
```

```
Hello Jake
Hello Miyasaki
Vowel in string 1 = 2
Vowel in string 2 = 1
Area = 738
```

Parameters and Arguments

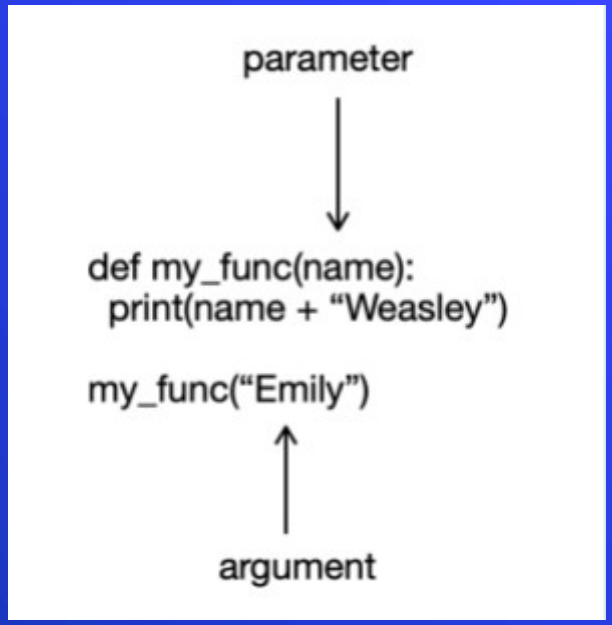
KMUTT

User-defined Function - Parameters & Arguments

- parameter และ argument ใช้ในเรียกข้อมูลที่ pass เข้าไปในฟังก์ชั่นเหมือนกัน โดยมีความแตกต่างกันดังนี้
  - Parameter คือตัวแปรที่ปรากฏในเครื่องหมายวงเล็บในฟังก์ชั่น
  - Argument คือค่าที่ถูกส่งเข้าไปในฟังก์ชั่นเมื่อถูกเรียก

User-defined Function - Parameters & Arguments

Parameters และ Arguments

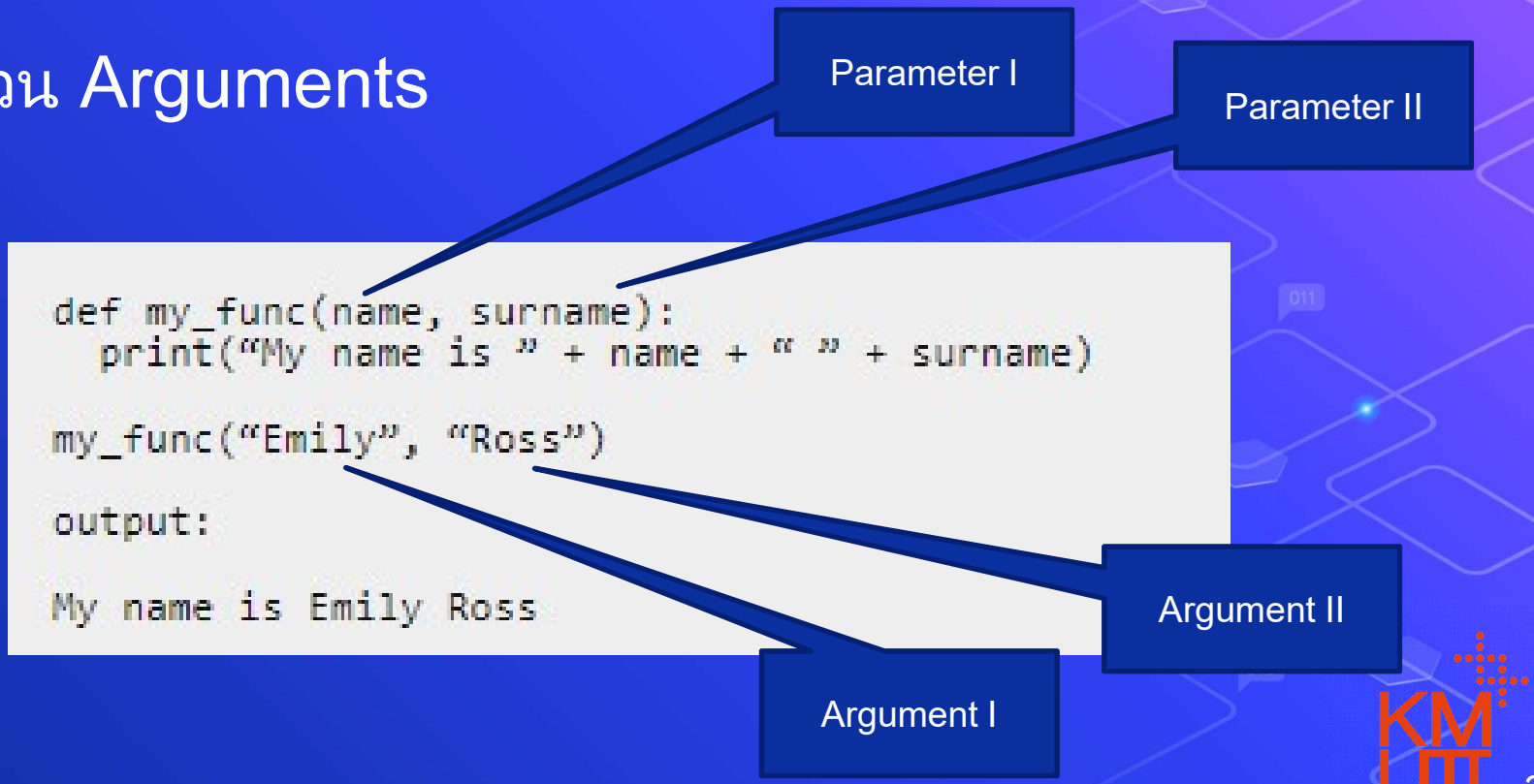


User-defined Function - Parameters & Arguments

- จำนวน Arguments
  - ฟังก์ชั่นต้องถูกเรียกด้วยจำนวน argument ที่ถูกต้อง
  - ตัวอย่างเช่น หากในฟังก์ชั่นกำหนดมาว่าต้องมี 2 argument เมื่อเรียกใช้ฟังก์ชั่น ก็ต้องเรียกใช้ฟังก์ชั่นพร้อม argument 2 จำนวนตามที่ฟังก์ชั่นกำหนด
  - หากไม่ได้ใส่ argument ตามจำนวนที่กำหนดก็จะเกิด error

User-defined Function - Parameters & Arguments

- จำนวน Arguments



User-defined Function - Parameters & Arguments

- Arbitrary Arguments (*args)
  - หากไม่ทราบจะว่าจำนวน argument ที่จะ pass ลงในฟังก์ชั่น มีจำนวนเท่าใด ให้ใส่เครื่องหมายดอกจันทร์ (asterisk) หน้า parameter ข้างในวงเล็บของนิยามฟังก์ชั่น
  - เมื่อใช้วิธีนี้ ฟังก์ชั่นนั้นจะรับค่า argument เป็นค่าแบบ tuple และสามารถเข้าถึงค่า parameter นั้นได้โดยเลือกหมายเลข index ที่ต้องการเข้าถึง

User-defined Function - Parameters & Arguments

- Arbitrary Arguments (*args)

Arbitrary Argument

```
def my_func(*dog):
  print("My favourite dog is " + dog [2])

my_func("Richard", "Ellis", "Spiky", "Timothy")
```

```
My favourite dog is Spiky
```

Output

User-defined Function - Parameters & Arguments

- Keyword Arguments
  - สามารถส่ง argument ด้วยไวยากรณ์ (syntax) key = value ได้
  - ด้วยวิธีการนี้ ลำดับของ argument จะไม่มีนัยสำคัญ

User-defined Function - Parameters & Arguments

- Keyword Arguments

```
def my_func(dog4, dog3, dog2, dog1):
  print("My favourite dog is " + dog3)

my_func(dog1 = "Richard", dog2 = "Ellis", dog3 = "Spiky", dog4 = "Timothy")
```

```
My favourite dog is Spiky
```

Output

User-defined Function - Parameters & Arguments

- Arbitrary Keyword Arguments (**kwargs)
  - สามารถใส่ asterisk 2 อันด้านหน้า paramerter ในขั้นตอนการนิยามฟังก์ชั่นหากไม่ทราบจำนวน argument ทั้งหมดที่จะ pass ข้อมูลเข้าไปในฟังก์ชั่น
  - เมื่อใช้วิธีนี้ ฟังก์ชั่นจะรับค่า argument เป็นค่าแบบ dictionary และสามารถเข้าถึงได้โดยระบุชื่อ key ของสมาชิกใน dictionary ที่ต้องการ

User-defined Function - Parameters & Arguments

- Arbitrary Keyword Arguments (**kwargs)

```
def my_func(**students):
  print("The student last name is " + students["lastName"])

my_func(firstName = "Emily", lastName = "Ross")
```

```
The student last name is Ross
```

Output

User-defined Function - Parameters & Arguments

- Default Parameter Value
  - Default Parameter คือ การตั้งค่าสำหรับฟังก์ชั่นในกรณีที่ฟังก์ชั่นถูกเรียกใช้งานโดยไม่มี argument
  - ฟังก์ชั่นจะใช้ค่า Default ที่ตั้งไว้แทนสำหรับฟังก์ชั่นที่ถูกเรียกโดยไม่มี argument

User-defined Function - Parameters & Arguments

- Default Parameter Value

Default Parameter Value

```
def my_func(country = "Thailand"):
  print("I am from " + country)

my_func("England")
my_func()
my_func("Spain")
```

```
I am from England
I am from Thailand
I am from Spain
```

Output

User-defined Function - Parameters & Arguments

- ส่งผ่าน List เป็น Argument
  - สามารถส่ง data ชนิดใดก็ได้เข้าไปเป็น argument ในฟังก์ชั่น
  - เมื่อถูกส่งผ่านเข้าไปแล้ว data เหล่านั้นจะถูกใช้ในแบบชนิดของเดิม เช่น หาก argument เป็น list เมื่อไปอยู่ในฟังก์ชั่นก็จะถูกใช้งานในรูปแบบ list

User-defined Function - Parameters & Arguments

- ส่งผ่าน List เป็น Argument

```
def my_func(food):
  for x in food:
    print(x)

fruits = ["mango", "pineapple", "apple"]

my_func(fruits)
```

```
mango
pineapple
apple
```

Output

User-defined Function - Parameters & Arguments

- การคืนค่ากลับ (Return Values)
  - การคืนค่าหรือการให้ฟังก์ชั่นคืนค่า จำเป็นต้องใช้ statement ที่เรียกว่า return

```
def my_func(x):
  return 2 + x

print(my_func(2))
print(my_func(3))
print(my_func(4))
```

```
4
5
6
```

Output

User-defined Function - Parameters & Arguments

- Lambda Expressions
  - Lambda Expressions คือ anonymous function ที่เป็นฟังก์ชันที่มีการทำงานขนาดเล็กอยู่ภายใน
  - สามารถมีได้เพียง Expression เดียวเท่านั้น
  - สามารถสร้างโดยใช้คำสั่ง lambda
  - สามารถใช้ Lambda Expressions สร้างออบเจ็คของฟังก์ชันได้ และค่า return จะเป็นค่าที่ได้จากผลลัพธ์ของ Expression ของฟังก์ชัน

User-defined Function - Parameters & Arguments

- Lambda Expressions

```
3
9
4.0
21.5
13
14
18
```

Output

```
f = lambda x: x + 1
print(f(2))
print(f(8))

g = lambda a, b: (a + b) / 2
print(g(3, 5))
print(g(10, 33))

def make_incrementor(n):
    return lambda x: x + n

f = make_incrementor(13)
print(f(0))
print(f(1))
print(f(5))
```

More Examples in Functions

KMUTT

User-defined Function

```python
def add(num1: int, num2: int) -> int:
    """Add two numbers"""
    num3 = num1 + num2

    return num3

# Driver code
num1, num2 = 5, 15
ans = add(num1, num2)
print(f"The addition of {num1} and {num2} results {ans}.")
```

Output

```
The addition of 5 and 15 results 20.
```

User-defined Function

```
#defining a function to print a number.
def number( ):
    num = 30
    print( "Value of num inside the function: ", num)

num = 20
number()
print( "Value of num outside the function:", num)
```

```
Value of num inside the function:  30
Value of num outside the function: 20
```

Output

User-defined Function

```
# Defining a function with return statement
def square( num ):
    return num**2

# Calling function and passing arguments.
print( "With return statement" )
print( square( 39 ) )

# Defining a function without return statement
def square( num ):
     num**2

# Calling function and passing arguments.
print( "Without return statement" )
print( square( 39 ) )
```

```
With return statement
1521
Without return statement
None
```

Output

The End.....

Questions or Comments????