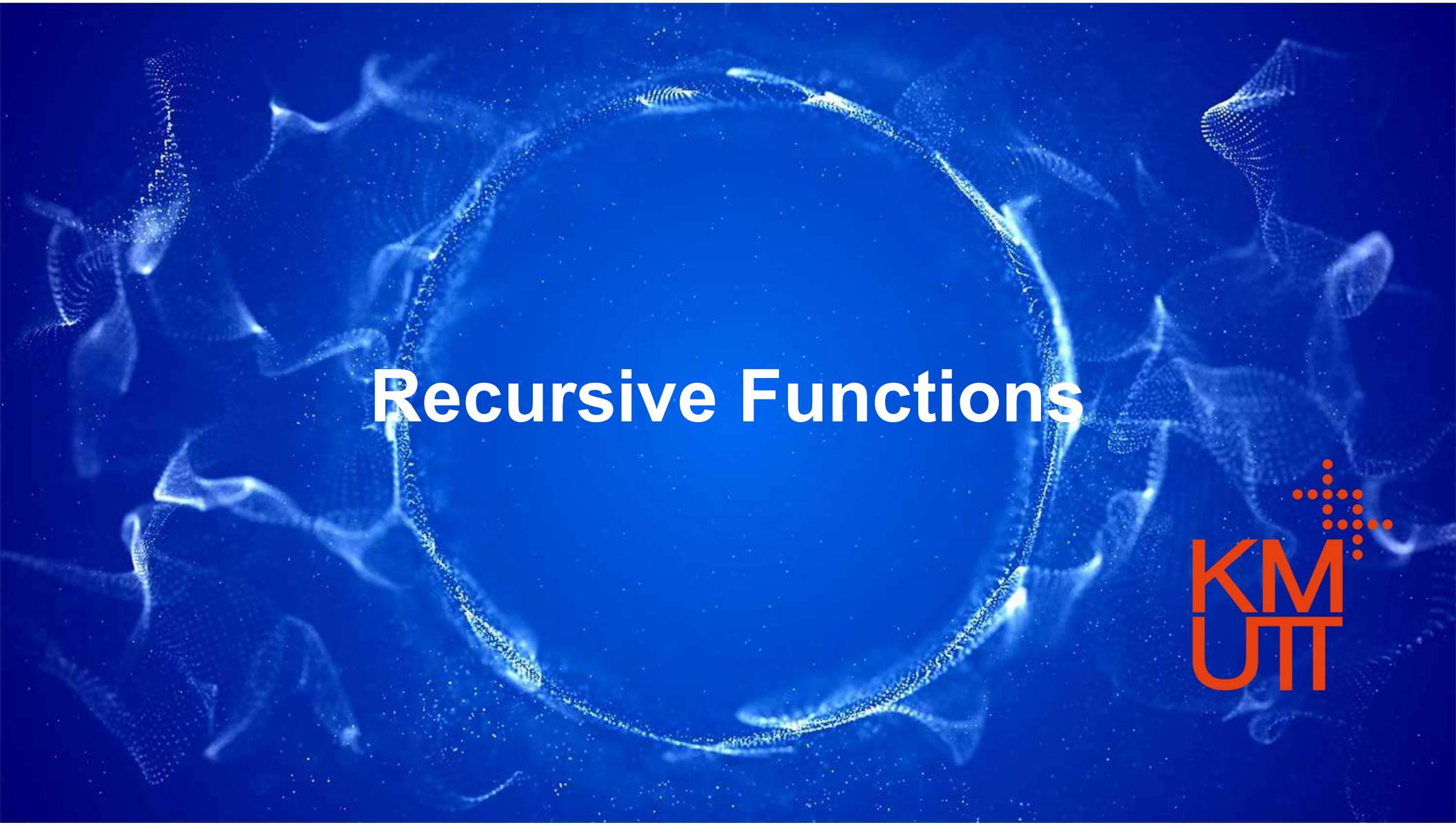
CSS 112 Computer Programming

Lecture VI: Advanced Functions in Python

Suvil Chomchaiya (Ph.D.)

Contents

- Recursive Functions
- Nested Functions
- Functions with Multiple Returns


Recursive Functions
KMUTT

Recursive Functions

- Recursive functions คือ “ฟังก์ชันแบบเวียนเกิด” หรือฟังก์ชันที่เรียกตัวเอง โดยแต่ละครั้งที่ฟังก์ชันถูกเรียก จะเกิดตัวแปรอัตโนมัติชุดใหม่ที่ไม่เกี่ยวข้องกับชุดเดิม
- หลักการ Recursive functions คือ เขียนโปรแกรมวนซ้ำเพื่อลดปัญหาของโปรแกรมที่ซับซ้อน
- ในการเขียนโปรแกรมด้วย Recursive functions จำเป็นจะต้องมี if statement เพื่อตัดสินใจว่าควรเรียกตัวเองต่อไปหรือไม่

Recursive Functions

- Recursive functions เป็นการสั่งให้ฟังก์ชันเรียกใช้งานตัวเองขึ้นมาทำงานซ้ำไปเรื่อยๆ
- ภายในฟังก์ชัน ให้เพิ่มการตรวจสอบผลลัพธ์ที่เกิดขึ้นก่อนจะส่งกลับออกไป ถ้าไม่ตรงตามต้องการ ก็แค่เรียกตัวมันเองขึ้นมาทำงานซ้ำ เหมือนการเรียกใช้ฟังก์ชันทั่วไป

Recursive Functions


```
def fac(x):
    f = 1                                  # ตั้งต้นที่ 1
    for i in range(2,x+1):                 # ใช้ for วนซ้ำ ไล่ตั้งแต่ 2
        f = f*i                            # คูณเพิ่มไปเรื่อยๆ
    return f                               # คืนผลลัพธ์ที่ได้กลับไป
num=int(input("Enter a number: "))
print(fac(num))
```

```
Enter a number: 5
120
```

Output

Recursive Functions

```
def fac(x):
    if(x>1):
        return fac(x-1)*x
    else:
        return 1
num=int(input("Enter a number: "))
print(fac(num))
```

```
Enter a number: 5
120
```

Output

Recursive Functions


```
def f(k):
  if(k > 0):
    result = k + f(k - 1)
    print(result)
  else:
    result = 0
  return result
num = int(input("Enter a number: "))
f(num)
```

```
Enter a number: 5
1
3
6
10
15
15
```

Output

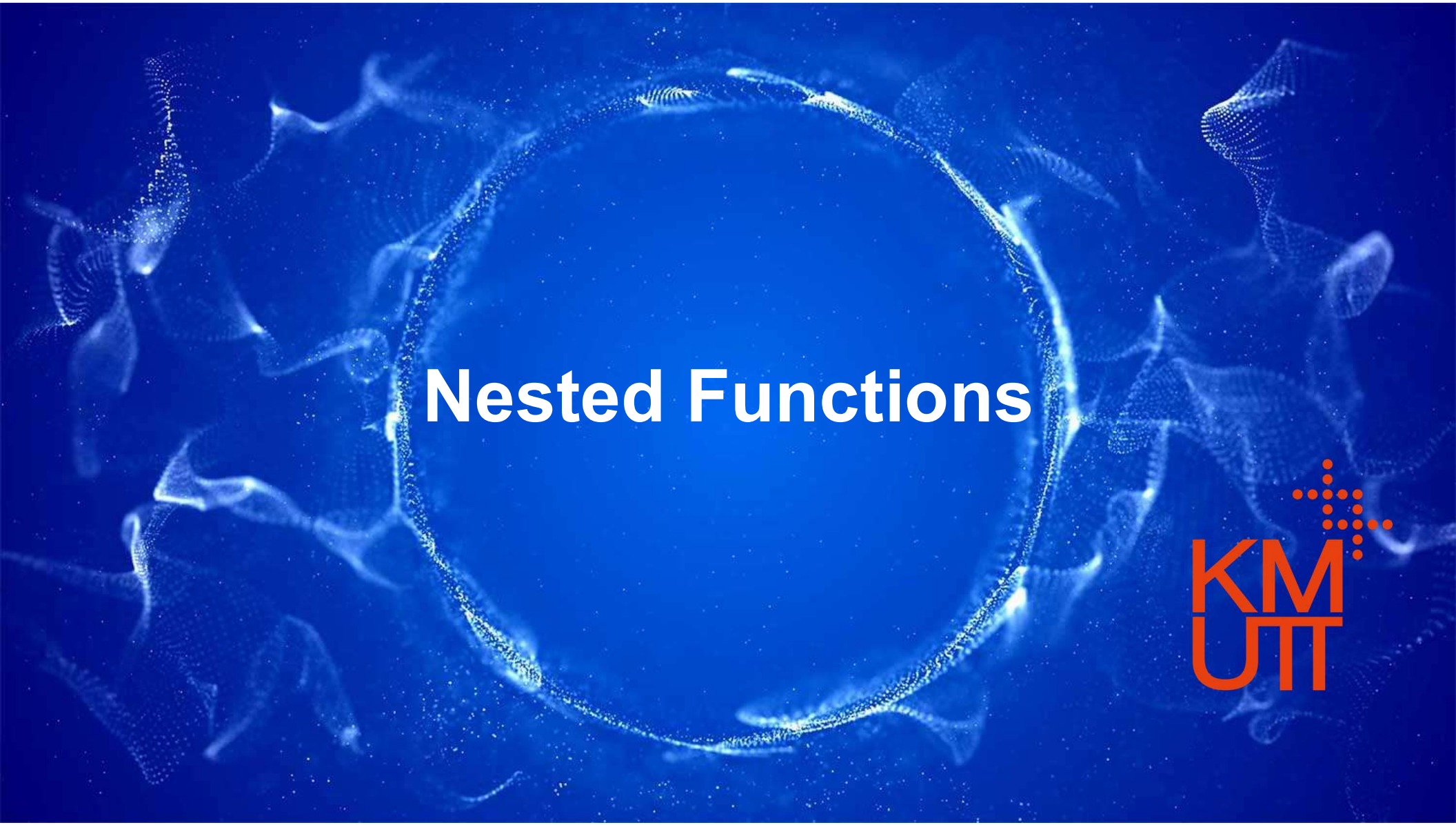
Recursive Functions


```
def countdown(n):
    print(n)
    if n == 0:
        return                  # Terminate recursion
    else:
        countdown(n - 1)  # Recursive call
num = int(input("Enter a number: "))
countdown(num)
```

```
Enter a number: 5
5
4
3
2
1
0
```

Output

Nested Functions



Nested Functions

- Recursive functions คือ “ฟังก์ชันแบบเวียนเกิด” หรือฟังก์ชันที่เรียกตัวเอง โดยแต่ละครั้งที่ฟังก์ชันถูกเรียก จะเกิดตัวแปรอัตโนมัติชุดใหม่ที่ไม่เกี่ยวข้องกับชุดเดิม
- หลักการ Recursive functions คือ เขียนโปรแกรมวนซ้ำเพื่อลดปัญหาของโปรแกรมที่ซับซ้อน
- ในการเขียนโปรแกรมด้วย Recursive functions จำเป็นจะต้องมี if statement เพื่อตัดสินใจว่าควรเรียกตัวเองต่อไปหรือไม่

Nested Functions


```
def function1():              # outer function
    print ("Hello from outer function")

    def function2():          # inner function
        print ("Hello from inner function")
    function2()

function1()
```

```
Hello from outer function
Hello from inner function
```

Output

Nested Functions

```
def num1(x):
    def num2(y):
        return x * y
    return num2


res = num1(10)
print(res(5))
```

```
50
```

Output

Nested Functions

```
def f1():
    s = 'I love KMUTT.'
    def f2():
        print(s)
    f2()
f1()
```

```
I love KMUTT.
```

Output

Nested Functions

```
def greeting(first, last):

  def getFullName():
    return first + " " + last
  print("Hi, " + getFullName() + "!")

Name=input("What is your name?: ")
Last=input("What is your last name?: ")

greeting(Name, Last)
```

```
What is your name?: Jack
What is your last name?: Welch
Hi, Jack Welch!
```

Output

Functions with Multiple Returns

Functions with Multiple Returns

- Python บางกรณีการใช้งานอาจต้องการให้มีการคืนค่าตั้งแต่สองค่าขึ้นไป
- หากต้องการให้คืนกลับหลายตัวก็ทำได้ด้วยการให้คืนกลับเป็นข้อมูลชนิดกลุ่ม เช่น ลำดับ, ทูเพิล, ดิกชันนารี

Functions with Multiple Returns

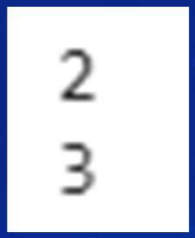
```
def pow(x,n):
    return [x**i for i in range(1,n+1)]
print(pow(2,12))
print(pow(3,7))
```

```
[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
[3, 9, 27, 81, 243, 729, 2187]
```

Output

Functions with Multiple Returns

```
def test_function(value1, value2):
    value1 += 1
    value2 += 1
    return value1, value2
result1, result2 = test_function(1, 2)
print(result1)
print(result2)
```



Output

Functions with Multiple Returns

ใช้ Comma-separated values (Tuples)

- ใช้ Python เพื่อคืนค่าหลายค่าโดยเพียงแค่คั่นด้วยเครื่องหมายจุลภาค (,)
- Python ใช้ tuple เพื่อทำสิ่งนี้

Functions with Multiple Returns

#Returning Multiple Values using Tuples

```
def fun():
    str = “CSS 112 Computer Programming"
    x   = 20
    return str, x;  # Return tuple, we could also


str, x = fun()      # Assign returned tuple
print(str)
print(x)
```

```
CSS 112 Computer Programming
20
```

Output

Functions with Multiple Returns

```
#Returning Multiple Values using Tuples
def name():
    return "John","Armin“

# print the tuple with the returned values
print(name())

# get the individual items
name_1, name_2 = name()
print(name_1, name_2)
```


```
('John', 'Armin')
John Armin
```

Output

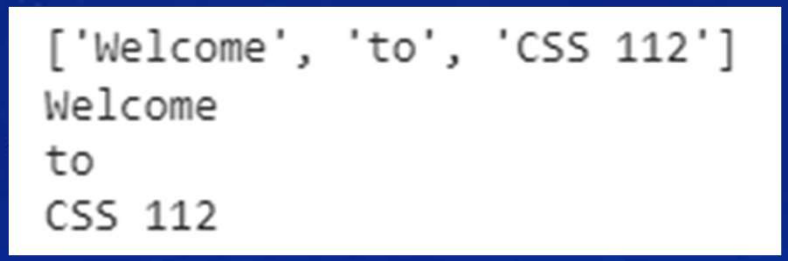
Functions with Multiple Returns

```
#Returning Multiple Values using Tuples
def getValues():
    value1 = 10
    value2 = 11
    value3 = 12
    return value1, value2, value3
value1, value2, value3 = getValues()
print(value1, value2, value3)
```

Output

Functions with Multiple Returns

```
#Returning Multiple Values using Tuples
def my_string():
    return “Welcome", "to", “CSS 112"
value1, value2, value3 = my_string()
print(value1)
print(value2)
print(value3)
```



Output

Functions with Multiple Returns


```
#Returning Multiple Values using List
def my_string():
    return [“Welcome", "to", "CSS 112"]
values = my_string()
print(values)
print( values[0] )
print( values[1] )
print( values[2] )
```

```
['Welcome', 'to', 'CSS 112']
Welcome
to
CSS 112
```

Output

Functions with Multiple Returns

```
#Returning Multiple Values using Dictionary
# Define a function - my_string() that
# returns 3 values inside a dictionary
def my_string():
    return {1:  "welcome",
            2:  "to",
            3:  “CSS 112"}
# Call the function - my_string()
print(my_string())
```


```
{1: 'Welcome', 2: 'to', 3: 'CSS 112'}
```

Output

Functions with Multiple Returns

```
#Returning Multiple Values using Set
# Define a function that returns 6 values with some duplicate values
def my_string():
    return "welcome", "to", “CSS 112", "welcome", "to", “CSS 112"
# Call the function - my_string() and
# get returned values in the form of a set
print( set(my_string()) )
```

```
{'Welcome', 'to', 'CSS 112'}
```

Output

THANK YOU