

**ITD62-123 COMPUTER PROGRAMMING**

**IMI62-122 FUNDAMENTAL OF COMPUTER PROGRAMMING**

Theerat Saichoo & Yunyong Punsawad  
School of Informatics, Walailak University



# Chapter 4

## Functions



# Recursion Functions

---

- Python also accepts function recursion, which means a defined function can call itself.
- Recursion is a common mathematical and programming concept.
- That a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.
- The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power.
- However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

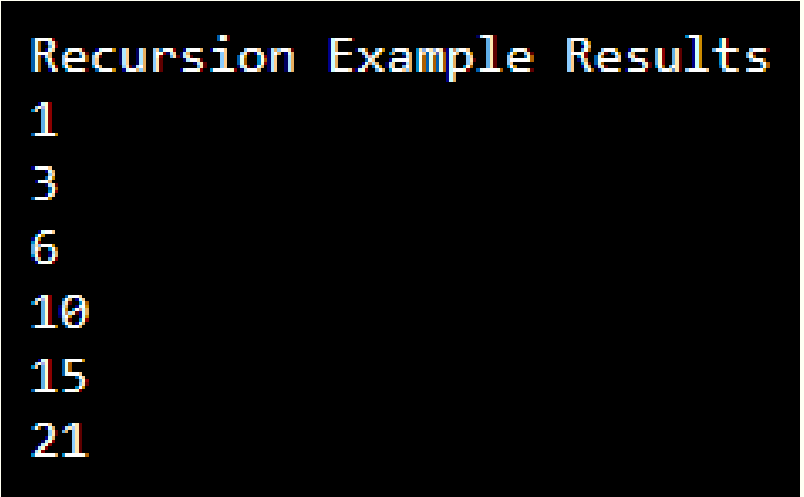
# Recursion Functions

---

*In this example, tri\_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).*

## ■ Example

```
def tri_recursion(k):  
    if(k > 0):  
        result = k + tri_recursion(k - 1)  
        print(result)  
    else:  
        result = 0  
    return result  
  
print("\n\nRecursion Example Results")  
tri_recursion(6)
```



```
Recursion Example Results  
1  
3  
6  
10  
15  
21
```

# Python Modules

---

- Consider a module to be the same as a code library.
- A file containing a set of functions you want to include in your application.

# Create a Module

---

- To create a module just save the code you want in a file with the file extension .py
- Example
  - Save this code in a file named mymodule.py

```
def greeting(name):  
    print("Hello, " + name)
```

# Use a Module

---

- Now we can use the module we just created, by using the import statement:
- Example
  - Import the module named mymodule, and call the greeting function:

```
import mymodule
```

```
mymodule.greeting("Jonathan")
```

# Naming a Module

---

- You can name the module file whatever you like, but it must have the file extension .py
- Re-naming a Module
  - You can create an alias when you import a module, by using the as keyword:
- Example
  - Create an alias for mymodule called mx:

```
import mymodule as mx
```

```
a = mx.person1["age"]  
print(a)
```



# Built-in Modules

---

- There are several built-in modules in Python, which you can import whenever you like.
- Example
  - Import and use the platform module:

```
import platform
```

```
x = platform.system()  
print(x)
```

# Built-in Modules

---

- `import datetime`
- `import math`
- `import json`
- `import re`
- `import numpy`
- `import scipy`
- and more.....

# Python PIP

---

- **What is PIP?**

- PIP is a package manager for Python packages, or modules if you like.

- **What is a Package?**

- A package contains all the files you need for a module.
  - Modules are Python code libraries you can include in your project.

# Python PIP

---

- **Download a Package**

- Downloading a package is very easy.
- Open the command line interface and tell PIP to download the package you want.
- Navigate your command line to the location of Python's script directory, and type the following:

- **Example**

- Download a package named "camelcase":

*pip install camelcase*

# Python PIP

---

- **Using a Package**

- Once the package is installed, it is ready to use.
- Import the "camelcase" package into your project.

- **Example**

- Import and use "camelcase":

```
import camelcase

c = camelcase.CamelCase()
txt = "hello world"
print(c.hump(txt))
```

# Python PIP

---

- **Remove a Package**

- Use the uninstall command to remove a package:

- **Example**

- Uninstall the package named "camelcase":

*pip uninstall camelcase*

# Python PIP

---

- **List Packages**

- Use the list command to list all the packages installed on your system:

- **Example**

- List installed packages:

*pip list*

THE END

**Special thanks to W3Schools**



# Class activity 7

---

เขียน Function แบบ Recursion ทำงานดังนี้

1. รับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 3 ค่า
2. คำนวณค่าเฉลี่ยของตัวเลขจำนวนเต็มทั้ง 3
3. หากค่าเฉลี่ยที่คำนวณได้ มีค่ามากกว่าผลรวมของค่าที่ 1 และค่าที่ 2 ให้แสดงข้อความว่า “The end” แล้วจบการทำงาน
4. แต่หากเงื่อนไขในข้อ 3 ไม่เป็นจริง ให้เรียกใช้งานฟังก์ชันซ้ำไปเรื่อย ๆ