โครงงาน Python

เรื่อง

Function, Modules, Files I/O

เสนอ

อ.ดร.ทัศนวรรณ ศูนย์กลาง

Function

Function ใน python เราจะเริ่มต้นด้วยการใส่คำว่า def (ซึ่งเป็น format ของ python) ตามด้วย name_function , “ ( ) “ และปิดด้วย colon “ : ”

Syntax :

```
def name_function( parameters ) :

…
return expression
```

- def จะเปรียบเสมือน type function ของภาษาอื่น และ def สามารถเป็นได้ทุก type
- parameter ในภาษา python เราไม่จำเป็นจะต้องใส่ type ของ parameter function ก็สามารถเรียกใช้ได้

Example 1:

```
def printme( str ):
        ...
        print str
return


def printme( a, b ):
        ...
return a+b
```

- จะเห็นได้ว่า parameter จะไม่มี type ของข้อมูล แต่ก็สามารถใช้ได้

Calling a Function (การเรียกใช้ฟังก์ชัน)

Example function form example 1:

```
def printme( str ):
        …
        print str;
return;
# Now you can call printme function
printme(“My name is Python");
printme(“Ok , I know you");
```

Result :

My name is Python

Ok , I know you

Pass by reference vs value (เรียกใช้โดยการอ้างอิง)

Example 2:

```
def changeme( mylist ):
        …
        mylist.append([1,2,3,4]);
        print "Values inside the function: ", mylist
return
# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

Result :

```
Values inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values outside the function:  [10, 20, 30, [1, 2, 3, 4]]
```

- จะเห็นว่ามีการเรียก changeme นอกฟังก์ชันโดยส่ง mylist เป็น parameter และฟังก์ชัน changeme ทำการอ้างอิง mylist จากนอกฟังก์ชันเพื่อมาเรียกใช้
- append เป็นคำสั่งที่ใช้ในการต่อท้าย mylist

Example 3:

```
def changeme( mylist ):
        …
        mylist = [1, 2, 3, 4];
        print mylist
return
# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print mylist
```

Result :

```
[1, 2, 3, 4]
[10, 20, 30]
```

- ตัวอย่างนี้ก็จะทำการอ้างอิงเช่นเดียวกับ ตัวอย่างที่ 2
- ฟังก์ชันทำการพิมพ์ค่า mylist ที่อยู่ในฟังก์ชันเป็นลำดับแรก แล้วทำการพิมพ์ค่า mylist นอกฟังก์ชันเป็นลำดับถัดไป

Function Arguments(การส่งพารามิเตอร์)

มี 4 รูปแบบ คือ

1) Required arguments
2) Keyword arguments
3) Default arguments
4) Variable-length arguments

Required arguments

Example 4 :

```
def printme( str ):
        …
        print str;
return;
# Now you can call printme function
printme( );
```

Result :

มีข้อผิดพลาดในการรัน เนื่องการเรียกฟังก์ชันใช้ไม่มีตัวส่งผ่าน (parameter) ไปยังฟังก์ชัน

Keyword arguments

Example 5 :

```
def printme( str ):
        …
        print str;
return;
# Now you can call printme function
printme( str = "My string");
```

Result :

My string

- เป็นการส่งตัวหนังสือโดยพารามิเตอร์

Example 6 :

```
def printinfo( name, age ):
        …
        print "Name: ", name;
        print "Age ", age;
return;
# Now you can call printinfo function
printinfo( age=14, name=“python" );
```

Result :

Name: miki

Age 50

- เป็นการส่งตัวหนังสือกับตัวเลขโดยพารามิเตอร์

Default arguments

Example 7 :

```
def printinfo( name, age = 35 ):
        …
        print "Name: ", name;
        print "Age ", age;
return;
# Now you can call printinfo function
printinfo( age=50, name=“python" );
printinfo( name=“GGWP" );
```

Result :

```
Name:  python
Age  50
Name:  GGWP
Age  35
```

- เมื่อทำการเรียกใช้ฟังก์ชัน printinfo ที่มีพารามิเตอร์ส่งไปไม่ครบ ฟังก์ชันจะทำการนำค่า Default ที่เซ็ตไว้มาใส่ให้เอง

Variable-length arguments

Syntax :

```
def functionname([formal_args,] *var_args_tuple ):
        "function_docstring"
        function_suite
    return expression
```

- ” * ” วางไว้ข้างหน้า arguments ที่ต้องการกำหนด เป็น Variable-length arguments
- เป็นฟังก์ชันที่เราสามารถกำหนด arguments ตอนเรียกใช้งานฟังก์ชั่นกี่ตัวก็ได้

Example 8 :

```
def printinfo( arg1, *vartuple ):
        …
        print "Output is: "
        print arg1
        for var in vartuple:
                print var
    return;

# Now you can call printinfo function
printinfo( 10 );
printinfo( 70, 60, 50 );
```

Result :

```
Output is:
10
Output is:
70
60
50
```

The Anonymous Functions(ฟังก์ชันที่ไม่ต้องระบุชื่อ)

Syntax :

lambda [arg1 [,arg2,.....argn]]:expression

Example 9 :

```
sum = lambda arg1, arg2: arg1 + arg2;

sum2 = lambda arg3, arg4: arg3 * arg4;

# Now you can call sum as a function

print “Summation : ” , sum( 10, 20 )

print “Multiple : ” , sum2( 20, 20 )
```

Result :

Summation : 30

Multiple : 400

- เป็นการสร้างฟังก์ชันแบบไม่ต้องกำหนดชื่อ จะมี format คือ lambda เสมอ

The return Statement(ส่งค่าออกจากฟังก์ชั่น)

Example 10 :

```
def sum( arg1, arg2 ):
        …
        total = arg1 + arg2
        print "Inside the function : ", total
return total;
# Now you can call sum function
total = sum( 10, 20 );
print "Outside the function : ", total
```

Result :

```
Inside the function :  30
Outside the function :  30
```

- return สามารถใช้ในการคำนวน หรือ ลดการเขียนโค้ดที่ซ้ำๆ กันและมีการเรียกใช้กันไปมาระหว่างฟังก์ชั่นด้วยกันเอง หรือ เพื่อ กำหนดรูปแบบแสดงผลแบบต่างๆ

Scope of Variables(ขอบเขตของการใช้ตัวแปล)

มี 2 แบบ คือ ตัวแปลแบบ Global และตัวแปลแบบ Local

ตัวแปล Global จะสามารถใช้งานได้ทุกที่ ส่วนตัวแปล Local จะใช้งานได้แบบมีขอบเขต ถ้าตัวแปลนั้นอยู่ในฟังก์ชันก็จะสามารถใช้ได้แค่ในฟังก์ชันเท่านั้น

แบบฝึกหัด

1) ทำการสร้างฟังก์ชันแบบ Variable-length โดยให้พิมพ์ค่า 1, 2, 3, 4, 5 ออกมา

หมายเหตุ ควรทำการสร้างลูปเพื่อให้สามารถพิมพ์ค่าออกมาได้ทุกตัว

Syntax :

```
def functionname([formal_args,] *var_args_tuple ):
    "function_docstring"
    function_suite
return expression
```

2) จงสร้างฟังก์ชันที่ไม่ต้องการชื่อ (Anonymous Functions) มาหนึ่งฟังก์ชัน โดยกำหนดให้ใช้ตัวแปลที่เป็นตัวเลข 2 ตัว แล้วทำการบวกค่าในฟังก์ชัน

อ้างอิง

http://www.learnpython.org/en/Functions

http://www.tutorialspoint.com/python/python_functions.htm

Modules

Module คือ กลุ่มของ ตัวแปร ฟังก์ชั่น หรือ คลาส ที่อยู่ในไฟล์เดียวกัน โดยโมดูลมี 2 ประเภท ได้แก่

1. โมดูลที่มีอยู่แล้วใน Library ของ Python เช่น math , random , string , ..

2. โมดูลที่ผู้ใช้สร้างขึ้นเอง

การเรียกใช้ Module

จะต้อง **import** โมดูลเข้ามาก่อน ถึงจะสามารถเรียกใช้งานฟังก์ชัน หรือคลาสที่อยู่ภายในโมดูลได้ โดยมีรูปแบบ ดังนี้

import module1[, module2[,…moduleN]

การเรียกใช้ **Function** ในโมดูลที่ **import** เข้ามา

module.function()

ตัวอย่าง การใช้โมดูลที่มีอยู่แล้วใน Library ของ python

```
import math
print(math.pi)
```

ผลลัพธ์

```
3.141592653589793
```

ตัวอย่าง การใช้โมดูลที่ผู้ใช้สร้างขึ้นเอง

สร้างไฟล์ขึ้นมาใหม่ชื่อ **myModule.py** แล้วพิมพ์โค้ดด้านล่าง

```
def Addition(num1,num2)
    return num1+num2
```

แล้วทำการเรียกใช้งานโมดูล **myModule** จากอีกไฟล์

```
import myModule
print(myModule.Addition(10,20))
```

ผลลัพธ์

```
30
```

การใช้ from .. import ..

เป็นการเรียกใช้บาง functioon จาก module ที่ทำการ import เข้ามา โดยมีรูปแบบ ดังนี้

from module import function1[,… functionN][*]

ตัวอย่าง การเรียกใช้ฟังก์ชัน **randint()** (สุ่มเลขจำนวนจริง) ที่อยู่ในโมดูล **random** ซึ่งมีอยู่ใน Library ของ Python

```
from random import randint
print(randint(1,10))
```

ผลลัพธ์

(ตัวเลขสุ่มระหว่าง 1-10)

ตัวอย่าง การเรียกใช้ทุกๆฟังก์ชัน ที่อยู่ในโมดูล random ซึ่งมีอยู่ใน Library ของ Python โดยใช้ *

```
from random import *
print(randint(1,10))
```

ผลลัพธ์

(ตัวเลขสุ่มระหว่าง 1-10)

* ในการเรียกใช้ฟังก์ชันของ **from .. import ..** ไม่ต้องทำการอ้างถึงโมดูลของฟังก์ชันนั้นๆก่อน เช่น **random.randint()** แต่สามารถเรียกใช้ฟังก์ชัน **randint()** ได้โดยตรง

** การใช้ **from .. import ..** กับโมดูลที่ผู้ใช้สร้างขึ้นมาเอง ก็สามารถทำได้ในลักษณะเดียวกัน

dir() Function

dir() เป็นฟังก์ชันที่คืนค่าเป็นชื่อทั้งหมดที่ถูกกำหนดขึ้นในโมดูล ประกอบด้วย ชื่อโมดูล , ชื่อตัวแปล , ชื่อฟังก์ชัน

ตัวอย่าง

```
import math
print(dir(math))
```

ผลลัพธ์

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']

* จะต้องทำการ **import** โมดูลที่ต้องการใช้ฟังก์ชัน **dir()** ก่อน จึงจะสามารถใช้ฟังก์ชัน **dir()** ได้

Namespaces and Scoping

ชื่อตัวแปรและขอบเขตของตัวแปร โดยตัวแปรที่อยู่ในโปรแกรมหลักจะเรียกว่า ตัวแปรชนิด **Global** ส่วนตัวแปรที่อยู่ในฟังก์ชันจะเรียกว่า ตัวแปรชนิด **Local**

ตัวอย่าง

```
year = 2014
def NextYear():
    year = year + 1
print(year)
NextYear()
print(year)
```

ผลลัพธ์

```
UnboundLocalError: local variable 'year' referenced before assignment
```

* ผลลัพธ์ผิดเนื่องจาก **year** ที่อยู่นอกฟังก์ชัน(ตัวแปรชนิด Global) กับ **year** ที่อยู่ในฟังก์ชัน (ตัวแปรชนิด Local) ถือว่าเป็นตัวแปรคนละตัวกัน แม้ชื่อจะเหมือนกันก็ตาม ซึ่งมีวิธีแก้ คือ ใส่คำสั่ง **global** ไว้หน้าตัวแปร

ตัวอย่าง

```
year = 2014
def NextYear():
    global year
    year = year + 1
print(year)
NextYear()
print(year)
```

ผลลัพธ์

```
2014
2015
```

แบบฝึกหัด

1. สร้างไฟล์ชื่อ **myMath.py** ด้านในประกอบด้วยฟังก์ชัน โดยให้แต่ละฟังก์ชัน

รับ parameter เป็น **x** และ **y**

- **Addition()** คืนค่าเป็น ค่า **x** บวกด้วย ค่า **y**
- **Subtraction ()** คืนค่าเป็น ค่า **x** ลบด้วย ค่า **y**
- **Multiplication ()** คืนค่าเป็น ค่า **x** คูณด้วย ค่า **y**
- **Division ()** คืนค่าเป็น ค่า **x** หารด้วย ค่า **y**

แล้วสร้างอีกไฟล์ชื่อ **testMath.py** แล้วทำการเรียกใช้แต่ละฟังก์ชันจาก **myMath.py**

ที่มา : http://www.tutorialspoint.com/python/python_modules.htm

Files I/O

การรับ input มี 2 แบบ คือ รับค่าจาก keyboard และ ไฟล์ txt ส่วนการแสดง output จะมีรูปแบบเดียว คือ การใช้คำสั่ง print มีหัวข้อต่างๆที่เกี่ยวข้องดังนี้

- รูปแบบ input และ output
- File Object Attributes
- Reading and Writing Files
- File Positions

รูปแบบ input และ output

1.การรับ input และ แสดง output แบบทั่วไป

การรับ input จาก keyboard มี 2 รูปแบบ คือ

1.1การใช้ raw_input

เป็นการรับค่าเข้ามาแล้วคืนค่าเป็นstring

syntax : <variable> = raw_input();

ตัวอย่าง

name = raw_input(“Enter your name : “);

print “Your name is : “,name

input = Robert

ผลลัพธ์

Enter your name : Robert

Your name is : Robert

1.2การใช้ input

เป็นการรับค่าที่เป็นตัวเลข

syntax : <variable> = input();

ตัวอย่าง

number = input(“Enter number : “);

print “Number is : “,number*5

input = 15

ผลลัพธ์

Enter number : 15

Number is : 75

2.การรับ input จากไฟล์ txt

จะใช้ประกอบไปด้วย 2 Method คือ open กับ close

2.1 open() Method

มีหลักการทำงานคือจะมีตัว pointer ชี้ไปยังตำแหน่งต่างๆในไฟล์ ประกอบไปด้วย 3 ส่วน ดังนี้

1.file_name คือ ชื่อไฟล์ txt ที่ต้องการชี้ไปยังไฟล์นั้น

2.access_mode คือ โหมดของการเรียกใช้ ประกอบไปด้วยโหมดหลักๆ ดังนี้

- r คือ โหมดสำหรับอ่านข้อมูลที่อยู่ในไฟล์เท่านั้น โดยตัวชี้จะอยู่ที่ตำแหน่งแรกของไฟล์ และเป็นค่า default กรณีที่ไม่ได้กำหนด access_mode

- w คือ โหมดสำหรับเขียนข้อมูลลงในไฟล์เท่านั้น โดยจะเป็นการเขียนทับข้อมูลเดิม มีตัวชี้อยู่ที่ตำแหน่งแรก ถ้าไฟล์ที่เรียกไม่มีอยู่ จะสร้างไฟล์ใหม่ตามชื่อ file_name

- a คือ โหมดสำหรับเพิ่มข้อมูลลงไปในไฟล์ โดยเป็นการเพิ่มข้อมูลต่อท้ายข้อมูลเดิม มีตัวชี้อยู่ที่ตำแหน่งสุดท้ายของไฟล์ ถ้าไฟล์ที่เรียกไม่มีอยู่ จะสร้างไฟล์ใหม่ตามชื่อ file_name

- r+ คือ โหมดสำหรับอ่านและเขียนข้อมูล
- a+ คือ โหมดสำหรับอ่านและเพิ่มข้อมูล
- w+ คือ โหมดสำหรับอ่านและเขียนข้อมูล

3.buffering คือ การกำหนดการใช้ buffer โดยค่า 0 คือ ไม่ใช้ buffer ค่า 1 คือ ใช้ buffer

2.2 close() Method มีการทำงานคือถ้าเรียกใช้จะเป็นการปิดการใช้ไฟล์นั้น และถ้าใช้งานไฟล์อื่น ไฟล์เดิมจะถูกปิดโดยอัตโนมัติ

syntax : filObject.close();

File Object Attributes

เป็นการเรียกใช้เพื่อดูคุณสมบัติต่างๆของไฟล์ที่เปิดใช้งาน มีคำสั่งสำคัญดังนี้

1.file.closed – return ค่า true ถ้าไฟล์นั้นถูก close อยู่ ถ้าไม่ใช่ จะ return ค่า false

2.file.mode – return ชนิดของ access_mode ของไฟล์ที่เปิดใช้อยู่

3.file.name – return ชื่อไฟล์

ตัวอย่าง

```
file = open(“test.txt”, ”a”)
print "Name = ", test.name
print "Status = ", test.closed
print "Mode = : ", test.mode
```

ผลลัพธ์

Name = test.txt

Status = False

Mode = a

Reading and Writing Files

1.write() Method

syntax : fileObject.write(string);

ตัวอย่าง

```
file = open(“test.txt”, “w”)

file.write(“Python Java Banana”);

file.close()
```

ผลลัพธ์

จะได้ไฟล์ชื่อ test ที่มีข้อมูลอยู่คือ Python Java Banana

2.read() Method

syntax : fileObject.read(count);

ตัวอย่าง

```
file = open(“test.txt”, “r+”)

str = test.read(9);

print “Read String is : ”, str

file.close()
```

ผลลัพธ์

```
Read String is : Python Ja
```

File Positions

ประกอบด้วย method ที่เกี่ยวข้องกับตำแหน่งของตัว pointer ที่ชี้ไปยังตำแหน่งในไฟล์ ดังนี้

1.tell() Method - เป็น method ที่บอกตำแหน่งปัจจุบันของตัว pointer

2.seek(offset, from) Method – เป็น method ที่ใช้เปลี่ยนตำแหน่งตัว pointer โดย

offset คือ จำนวนตำแหน่งที่จะขยับไป

from คือ ตำแหน่งจุดเริ่มต้นที่จะทำการขยับ โดยให้

0 = ตำแหน่งเริ่มต้นของไฟล์

1 = ตำแหน่งปัจจุบัน

2 = ตำแหน่งสุดท้ายของไฟล์

ตัวอย่าง

```
file = open("test.txt", "r+")

str = test.read(9);

print "Read String is : ", str

position = test.tell();

print "Current file position : ", position

position = test.seek(4, 1);

str2 = test.read(4);

print "Again Read String is : ", str2

fo.close()
```

ผลลัพธ์

Read String is : Python Ja

Current file position : 9

Again Read String is : Bana

ข้อความ : Python Java Banana

แบบฝึกหัด

1.สร้างไฟล์ txt ชื่อ newfile และทำการใส่ข้อมูลลงไป ดังนี้

“Python Programming”

จงแสดงคำว่า gram ออกมาทางจอภาพ

2.จงรับข้อมูลชื่อของนักศึกษา และปีพ.ศ.เกิด เข้ามา จากนั้นแสดง

ชื่อ และอายุของนักศึกษา

ที่มา : http://www.tutorialspoint.com/python/python_files_io.htm

รายชื่อผู้จัดทำ

นายธณัชวัฒน์ เจตนา 07550437 เรื่อง Function

นายธนพล สุภาพ 07550439 เรื่อง Modules

นายธีรศักดิ์ ทะเลทอง 07550446 เรื่อง Files I/O