

w03-Lab1

Functions Part II

Assembled for 204111
by Kittipitch Kuptavanich

Variable Scope

```
04 def f(x):
05     y = 1
06     x = x + y
07     print("x = ", x)
08     return x
09
10
11 x = 3
12 y = 2
13 z = f(x)
14
15 print("z = ", z)
16 print("x = ", x)
17 print("y = ", y)
18
19
```

Global Scope

```
>>>
x = 4
z = 4
x = 3
y = 2
```

ในการเรียกใช้ฟังก์ชัน `f()` ตัวแปร `y` และ `x` ในฟังก์ชัน `f()` เป็น Local Variable และมี Scope (หรือ Name Space) ของ Variable แคภายในฟังก์ชัน `f()`

การ Assign ค่า หรือ Operation ใดๆ ในฟังก์ชัน `f()` ไม่มีผล ต่อ `x` และ `y` ที่อยู่ด้านนอก (`__main__`) เนื่องจากเป็น Variable คนละตัว

Variable Scope

- In each function, variables and parameters are **local**
- รู้จักเฉพาะในฟังก์ชัน เกิดและตายในฟังก์ชัน

Local variable

global variables

```
01 #!/usr/bin/env python
02
03 def hypotenuse(a, b):
04     h = ((a ** 2) + (b ** 2)) ** 0.5
05     return h
06
07 s1 = int(input("input a: "))
08 s2 = int(input("input b: "))
09
10 h = hypotenuse(s1, s2)
11
12 print("hypotenuse = {0:.2f}".format(h))
```

Think Python: How to Think Like a Computer Scientist

Variable Scope [2]

```
x = 8
def f():
    x = 5

f()
print(x)
```

```
>>>
8
```

For now: หลีกเลี่ยงการใช้ global variable

```
def f():
    print(x)

def g():
    print(x)
    x = 1
```

```
x = 3
f()
x = 3
g()
```

ทำให้ `x` มี scope เป็น local ดังนั้น `print(x)` จึงเกิด Error เพราะว่า `print()` ก่อน Assign

```
>>>
3
```

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

Example 1: Ones Digit

• Problem Statement

- ต้องการเขียนฟังก์ชันที่รับค่าเลขจำนวนเต็มใดๆ แล้วคืนค่าหลักหน่วย (ones digit) ของจำนวนนั้นๆ
- การวิเคราะห์ปัญหา
 - Input: (parameter)
 - จำนวนข้อมูล_____ชนิดข้อมูล_____
 - Output (return value)
 - จำนวนข้อมูล_____ชนิดข้อมูล_____
- ก่อนที่จะเริ่มเขียนฟังก์ชัน เราจะเริ่มจากการพิจารณา **Test Case** ต่างๆ ก่อน ให้แน่ใจว่าเราเข้าใจหน้าที่ของฟังก์ชันที่จะเขียน

5

Example 1: Ones Digit [2]

• Test Cases

- อะไรคือผลลัพธ์ของ `ones_digit(123)`?
 - 3 # หลักหน่วยของ 123
- `ones_digit(7890)`
 - 0
- `ones_digit(6)`
 - 6
- `ones_digit(-54)`
 - 4

6

Example 1: Ones Digit [3]

- เราจะสร้างฟังก์ชันเพื่อทดสอบฟังก์ชัน `ones_digit()` (ที่ยังไม่ได้เขียน ณ จุดนี้)
- เราจะสร้างฟังก์ชันชื่อ `test_ones_digit()`
- ในขั้นตอนที่เราจะนำ Test Case ที่สร้างไว้มาทดสอบโดยใช้ฟังก์ชัน `assert()`
 - ค่าที่ส่งเป็น argument ใน `assert()` function ควรเป็นค่าที่เป็น True
 - มิฉะนั้นฟังก์ชันจะฟ้องโดยการแสดง **AssertionError Exception** (แสดงว่าเกิดความผิดพลาด)

```
03 def test_ones_digit():
04     print("Testing ones_digit... ",end='')
05     assert(ones_digit(123) == 3)
06     assert(ones_digit(7890) == 0)
07     assert(ones_digit(6) == 6)
08     assert(ones_digit(-54) == 4)
09     print("Passed all tests!")
```

7

Example 1: Ones Digit [4]

• Stub Solution

- ในขั้นตอนที่เราจะเขียน Function Body ปลอมๆ (or "stub") ขึ้นมาซึ่งไม่ได้ทำหน้าที่แก้ปัญหาดตามโจทย์ **แต่ทำหน้าที่คืนค่าคำตอบปลอมๆ**
- Why? เพื่อที่จะได้ลอง run ฟังก์ชันทดสอบ

```
def ones_digit(x):
    return 3                # stub, for testing
test_ones_digit()          # actually run the test!
```

```
assert(ones_digit(7890) == 0)
AssertionError
```


 Surprised?

8

Example 1: Ones Digit [5]

- แก้ปัญหา, ทดสอบ, ทำซ้ำ
 - ตอนนี้เหลือเพียงขั้นตอนเดียวคือการแก้ปัญหา
 - And how do we do that?
 - Quite simple: the $x \% y$ gives the remainder
 - So 1's digit is just $x \% 10$

```
def ones_digit(x):
    return x % 10          # first attempt!
```

```
assert(ones_digit(-54) == 4)
AssertionError
```

Test cases:

123
7890
6
-54

9

Example 1: onesDigit [6]

- แก้ปัญหา, ทดสอบ, ทำซ้ำ [2]
 - (อย่างน้อยก็) ผ่าน 3 Test Case แรก
 - ดูเหมือนฟังก์ชันของเราจะใช้ได้กับจำนวนบวกแต่ทำงานพลาดถ้าเป็นจำนวนลบ
 - Why? ลองพิจารณาการทำงานของ modulo

```
def ones_digit(x):
    return abs(x) % 10      # second attempt!
```

```
Testing ones_digit... Passed all tests! ^_^
```

10

Top-Down Design (Recap)

- เขียนฟังก์ชันจากใหญ่ไปเล็ก
Write functions top-down
 - ให้สมมติว่า Helper Function (ฟังก์ชันย่อย) ต่าง ๆ เขียนเสร็จแล้ว
Assume helper functions already exist!
- ทำการ Test Function จากเล็กไปใหญ่
Test functions bottom-up
 - ไม่นำฟังก์ชันใด ๆ มาเรียกใช้ จนกว่าจะผ่านการ test อย่างถี่ถ้วน
Do not use a function before it has been thoroughly tested
- ในทางปฏิบัติสามารถเขียน Stub Function มาใช้ชั่วคราวขณะทำ Top-down Design
Practicality: May help to write stubs (simulated functions as temporary placeholders in top-down design)

11

Pseudocode

- Pseudocode หรือ รหัสเทียมคือข้อความที่เขียน ขึ้นเพื่อทำการวางโครงร่าง (Outline) ของโปรแกรม
 - ใช้ภาษาอังกฤษ (หรือ ภาษาธรรมชาติ – Non Formal) ไม่มีรูปแบบตายตัว – แต่มี Notation (เครื่องหมาย หรือ สัญลักษณ์) กลางเพื่อให้ผู้อ่านส่วนมากเข้าใจ
- Why Pseudocode?
 - ใช้เพื่อออกแบบ วางแผนและอธิบาย ขั้นตอนการทำงาน ของโปรแกรมก่อนที่จะมีการ Code จริง
 - หลังจาก Code แล้ว Pseudocode จะกลายเป็น Comments

12

Example 1: Sum of Integers

Example 1

• Problem Statement:

- เขียนโปรแกรมเพื่อรับจำนวนเต็มสองจำนวนจาก user แล้วแสดงผลบวกของตัวเลขทั้งสอง

• Pseudocode:

- Prompt the user to enter the first integer
รับจำนวนเต็มตัวแรกจาก User
- Prompt the user to enter a second integer
รับจำนวนเต็มตัวที่สองจาก User
- Compute the sum of the two user inputs and save to a variable
คำนวณผลบวกของทั้งสองจำนวน แล้วเก็บไว้ใน Variable
- Display an output prompt that explains the answer as the sum
แสดง Output โดยอธิบายว่าจำนวนที่จะแสดงคือผลบวก
- Display the result
แสดงค่าของ Variable ที่เก็บผลบวก

Think Python: How to Think Like a Computer Scientist

13

204111: Fundamentals of Computer Science

Example 2: Sphere Volume [2]

```

01 #!/usr/bin/env python
02
03
04
05 def main():
06     # รับข้อมูลพื้นที่ผิวจาก user
07
08     # นำพื้นที่ผิวที่ได้มาคำนวณหารัศมี
09
10     # หารัศมีที่คำนวณได้มาคำนวณหาปริมาตร
11
12     # แสดงปริมาตรทรงกลม
13
14
15 main()

```

sphere.py

Think Python: How to Think Like a Computer Scientist

15

Example 2: Sphere Volume

• Problem Statement (Lab03_1_5XXXXXXXXX.py)

- เขียนโปรแกรมเพื่อรับค่าพื้นที่ผิวของทรงกลมจาก user แล้วคำนวณปริมาตรของทรงกลมนั้น

• การวิเคราะห์ปัญหา

• Input:

- จำนวนข้อมูล _____ ชนิดข้อมูล _____

• Output

- จำนวนข้อมูล _____ ชนิดข้อมูล _____

Think Python: How to Think Like a Computer Scientist

14

204111: Fundamentals of Computer Science

Example 2: Sphere Volume [3]

```

01 #!/usr/bin/env python
02
03 import math
04
05 def main():
06     # รับข้อมูลพื้นที่ผิวจาก user
07     surface_area = float(input("input surface area: "))
08     # นำพื้นที่ผิวที่ได้มาคำนวณหารัศมี
09     radius = find_r_from_surface_area(surface_area)
10     # หารัศมีที่คำนวณได้มาคำนวณหาปริมาตร
11     volume = sphere_volume(radius)
12     # แสดงปริมาตรทรงกลม
13     print("volume = {:.2f}".format(volume))
14
15 main()

```

sphere.py

Think Python: How to Think Like a Computer Scientist

16

Writing Modules

- ฟังก์ชันที่เราเขียนขึ้นใน Script หนึ่ง ๆ สามารถนำไปใช้ใน Script อื่น ๆ ได้ โดยการใช้คำสั่ง `import` ในลักษณะเดียวกับการ `import math` Module
- Script ที่เราเขียนฟังก์ชันต่าง ๆ ไว้ เมื่อเรียกใช้งานจากอีก Script ก็ถือเป็น module เช่นกัน

```
import sphere
```

ใช้คำสั่ง import ตามด้วยชื่อไฟล์
โดยไม่ต้องใส่ .py

ชื่อ Module ต้องตรงกับชื่อไฟล์
(ตัวพิมพ์เล็ก และ underscore)

Think Python: How to Think Like a Computer Scientist

17

204111: Fundamentals of Computer Science

Example 2: Sphere Volume [4]

- Implement ฟังก์ชันดังต่อไปนี้เพิ่มจากไฟล์ใน slide 16
 - `find_r_from_surface_area()`
 - `sphere_volume()`
- และส่งงานในชื่อ Lab03_1_5XXXXXXX.py

Think Python: How to Think Like a Computer Scientist

19

Writing Modules [2]

- ทั้งนี้หากในไฟล์ `sphere.py` หากมีชุดคำสั่งอื่น ๆ นอกเหนือจาก Function Definition ใน Global Scope
 - เช่น การเรียกใช้ฟังก์ชันเพื่อการทำ testing หรือการเรียกใช้ฟังก์ชัน `main()`
 - ชุดคำสั่งเหล่านั้นก็จะถูกเรียกใช้งานไปด้วยเมื่อมีการ `import`
 - ได้ Output ที่ User ไม่ต้องการ (User เพียงต้องการเรียกใช้ฟังก์ชัน)
- เราสามารถป้องกันการถูกเรียกใช้งานดังกล่าว โดยการ ตั้งเงื่อนไข ให้ชุดคำสั่งดังกล่าว ถูกเรียกใช้ในกรณีที่ Script ถูก run โดยตรงเท่านั้น (ไม่ถูก run ในกรณี `import`) ดังแสดงด้านล่าง

```
26 if __name__ == '__main__':
27     main()
```

Think Python: How to Think Like a Computer Scientist

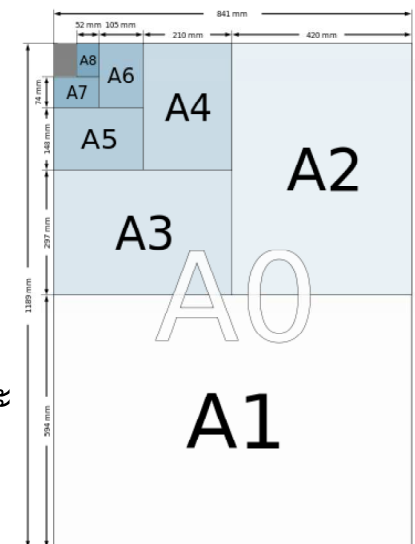
18

204111: Fundamentals of Computer Science

https://en.wikipedia.org/wiki/Paper_size

Example 3: Paper Size Standard

- ISO 216 เป็นมาตรฐานสากลว่าด้วยขนาดกระดาษซึ่งประเทศส่วนใหญ่ใช้กันในปัจจุบัน โดยมีขนาดกระดาษเป็นชุด A และ B
- ขนาดกระดาษชุด A เป็นที่นิยมใช้มากที่สุด โดยเฉพาะขนาด A4 (210 × 297 mm)

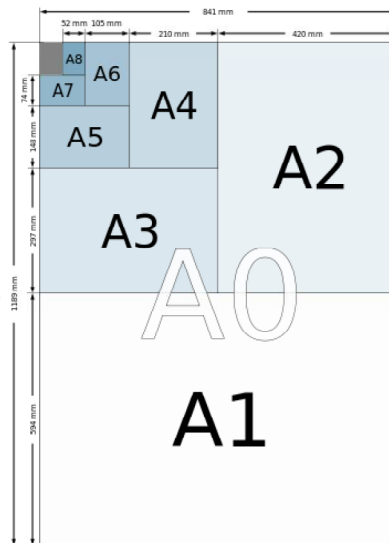


Think Python: How to Think Like a Computer Scientist

20

Example 3: Paper Size Standard [2]

- กระดาษในมาตรฐาน ISO 216 จะมีอัตราส่วนในลักษณะดังรูป คือ เมื่อพับกระดาษขนาด A0 ตามเส้นแนวความกว้าง จะได้กระดาษขนาด A1 ที่มีอัตราส่วนกว้าง (width): ยาว (height) เท่ากับอัตราส่วน กว้าง:ยาว ของกระดาษขนาด A0
- และเมื่อพับกระดาษขนาด A1 ตามเส้นแนวความกว้าง จะได้กระดาษขนาด A2 ที่มีอัตราส่วน กว้าง:ยาว เท่าเดิมเช่นกัน



Think Python: How to Think Like a Computer Scientist

21

Example 3: Paper Size Standard [4]

- Problem Solving**

Think Python: How to Think Like a Computer Scientist

23

Example 3: Paper Size Standard [3]

- Problem Statement** (Lab03_2_5XXXXXXXXX.py)
 - เขียนฟังก์ชัน `paper_width(h)` เพื่อรับค่าความยาว h ของกระดาษที่มีอัตราส่วนตามมาตรฐาน ISO 216 แล้วคำนวณความกว้าง (Width) ของกระดาษดังกล่าว
- การวิเคราะห์ปัญหา**
 - Input:**
 - จำนวนข้อมูล _____ ชนิดข้อมูล _____
 - Output**
 - จำนวนข้อมูล _____ ชนิดข้อมูล _____

Think Python: How to Think Like a Computer Scientist

22

References

- <http://www.greenteapress.com/thinkpython/thinkCSpy/html/chap03.html>
- <http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-writing-functions-examples.html>
- <https://docs.python.org/3/tutorial/controlflow.html#defining-functions>
- <http://www.minich.com/education/wyo/stylesheets/pse/udocode.htm>

Think Python: How to Think Like a Computer Scientist

24