w02-Lec3

Functions Part II

for 204111

by Kittipitch Kuptavanich

Function Call (Recap)

- **Keywords**

หากมี Argument หรือ Parameter มากกว่า 1 ตัว จะต้องคั่นด้วยเครื่องหมาย comma (สังเกตการใช้ Space ก่อนและหลัง Comma)

local

parameter

global

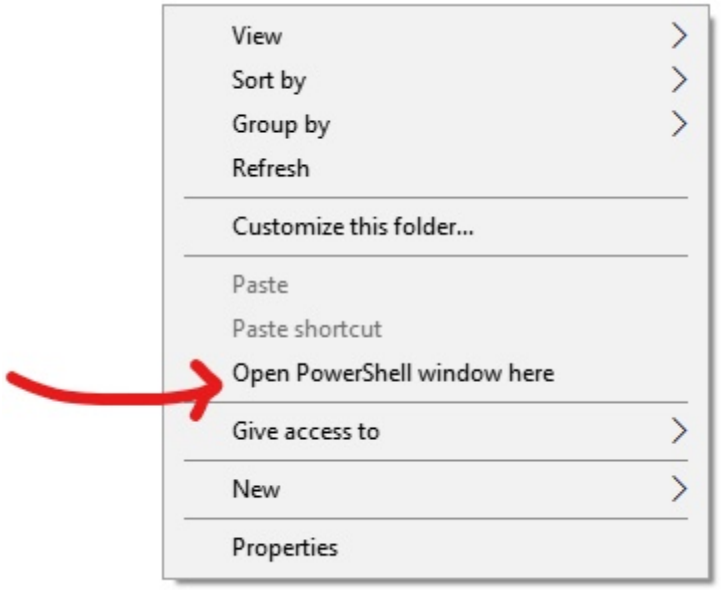
argument

hypotenuse.py

```
01 #!/usr/bin/env python3
02
03 def hypotenuse(a, b):
04     h = ((a ** 2) + (b ** 2)) ** 0.5
05     return h
06
07 s1 = float(input("input a: "))
08 s2 = float(input("input b: "))
09
10 h = hypotenuse(s1, s2)
11
12 print("hypotenuse = {0:.2f}".format(h))
```

How to Launch PowerShell

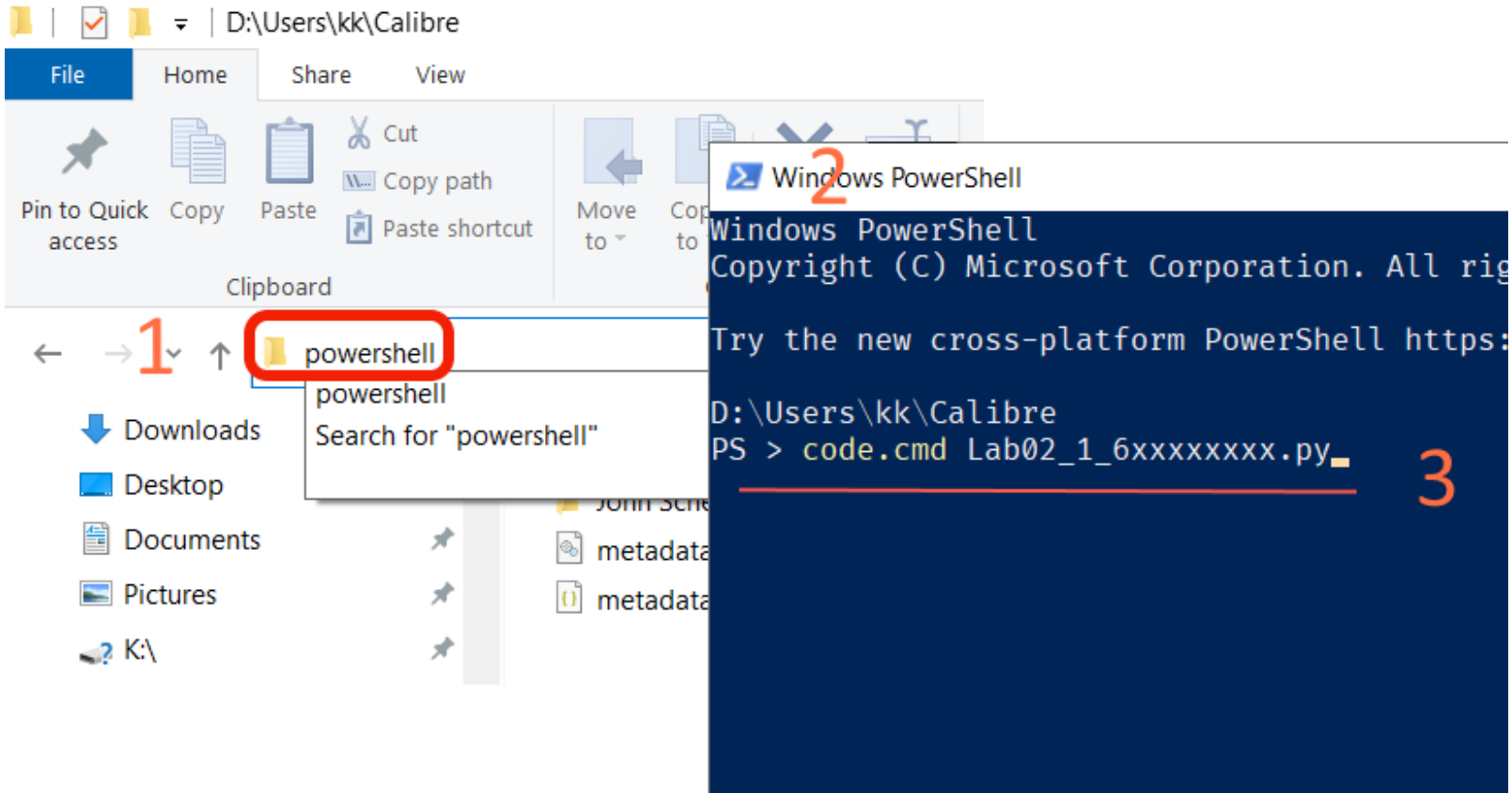
- **Shift + Right Click**
- **Then open VS Code from PowerShell**




```
PS D:\Users\kk\Desktop> code.cmd Lab01_0_6xxxxxxxx.py
```

How to Launch PowerShell [2]

- **Another way to launch PowerShell and Open VS Code**



FUNCTIONS PART II

Top-Down Design

- 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)

Variable Scope

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

Local variable

global variables

```
01 #!/usr/bin/env python3
02
03 def hypotenuse(a, b):
04     h = ((a ** 2) + (b ** 2)) ** 0.5
05     return h
06
07 s1 = float(input("input a: "))
08 s2 = float(input("input b: "))
09
10 h = hypotenuse(s1, s2)
11
12 print("hypotenuse = {0:.2f}".format(h))
```

Variable Scope [2]

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

```
>>>
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** แค่ภายในฟังก์ชัน `f()`

การ **Assign** ค่า หรือ **Operation** ใด ๆ ในฟังก์ชัน `f()` ไม่มีผล ต่อ x และ y ที่อยู่ด้านนอก เนื่องจากเป็น **Variable** คนละตัว

Variable Scope [3]

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

f()
print(x)
```

```
>>>
8
```

```
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
```

For now: หลีกเลี่ยงการใช้ global variable

http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-writing-functions-examples.html

Example 1: Ones Digit

- **Problem Statement**
  - ต้องการเขียนฟังก์ชันที่รับค่าเลขจำนวนเต็มใด ๆ แล้วคืนค่า*หลักหน่วย* (ones digit) ของจำนวนนั้น ๆ
  - การวิเคราะห์ปัญหา
    - Input: (parameter)
      - จำนวนข้อมูล 1 ชนิดข้อมูล int
    - Output (return value)
      - จำนวนข้อมูล 1 ชนิดข้อมูล int
  - ก่อนที่จะเริ่มเขียนฟังก์ชัน เราจะเริ่มจากการพิจารณา Test Case ต่าง ๆ ก่อน ให้แน่ใจว่าเราเข้าใจหน้าที่ของฟังก์ชันที่จะเขียน

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

Example 1: Ones Digit [3]

- เราจะสร้างฟังก์ชันเพื่อทดสอบฟังก์ชัน `ones_digit()` (ที่ยังไม่ได้เขียน ณ จุดนี้)
  - เราจะสร้างฟังก์ชันชื่อ `test_ones_digit()`
  - ในขั้นตอนนี้เราจะนำ Test Case ที่สร้างไว้มาทดสอบโดยการใช้ statement `assert` ซึ่งทำหน้าที่ตรวจสอบค่าความจริงของ expression
    - Expression ที่อยู่หลัง statement `assert` ควรเป็นค่าที่เป็น True
    - มิฉะนั้น assert จะเกิดการ Throw *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!")
```

Example 1: Ones Digit [4]

- **Stub Solution**
  - ในขั้นตอนนี้เราจะเขียน **Function Body** ปลอม ๆ (or "stub") ขึ้นมาซึ่งไม่ได้ทำหน้าที่แก้ปัญหาตามโจทย์ แต่ทำหน้าที่คืนค่าคำตอบปลอม ๆ
  - **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?

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

Test cases:
123
7890
6
-54

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

```
Assert ones_digit(-54) == 4
AssertionError                    -_-
```

Example 1: Ones Digit [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!    ^_^
```

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 มาใช้ชั่วคราว**ขณะทำ **Top-down Design**
  Practicality: May help to write stubs (simulated functions as temporary placeholders in top-down design)

Pseudocode

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

Example 1: Sum of Integers

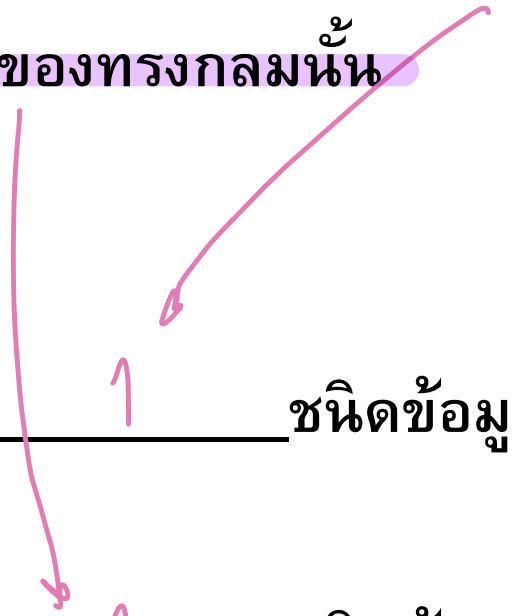
Example 1

- **Problem Statement:**
  - **เขียนโปรแกรม เพื่อรับจำนวนเต็มสองจำนวน จาก user แล้ว แสดงผลบวกของตัวเลขทั้งสอง**
- **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

http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-writing-functions-examples.html

Example 2: Sphere Volume

- **Problem Statement**
  - เขียนโปรแกรมเพื่อรับค่าพื้นที่ผิวของทรงกลมจาก user แล้วคำนวณปริมาตรของทรงกลมนั้น
- การวิเคราะห์ปัญหา
  - **Input:**
    - จำนวนข้อมูล____1____ชนิดข้อมูล___float___
  - **Output**
    - จำนวนข้อมูล____1____ชนิดข้อมูล___float___



Example 2: Sphere Volume [2]

sphere.py

```
01 #!/usr/bin/env python3
02
03
04
05 def main():
07     # รับข้อมูลพื้นที่ผิวจาก user
08
09     # นำพื้นที่ผิวที่ได้มาคำนวณหารัศมี
10
11     # นำรัศมีที่คำนวณได้มาคำนวณหาปริมาตร
12
13     # แสดงปริมาตรทรงกลม
14
15
16
17 if __name__ == '__main__':
18     main()
```

Example 2: Sphere Volume [3]

sphere.py

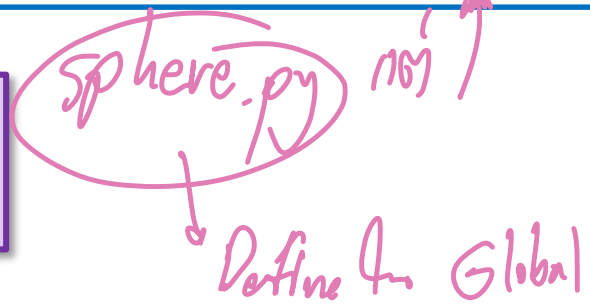
```
01 #!/usr/bin/env python3
02
03 import math
04
05 def main():
07     # รับข้อมูลพื้นที่ผิวจาก user
08     surface_area = float(input("input surface area: "))
09     # นำพื้นที่ผิวที่ได้มาคำนวณหารัศมี
10     radius = find_r_from_surface_area(surface_area)
11     # นำรัศมีที่คำนวณได้มาคำนวณหาปริมาตร
12     volume = sphere_volume(radius)
13     # แสดงปริมาตรทรงกลม
14     print("volume = {0:.2f}".format(volume))
15
16
17 if __name__ == '__main__':
18     main()
```

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)



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

```
17 if __name__ == '__main__':
18     main()
```

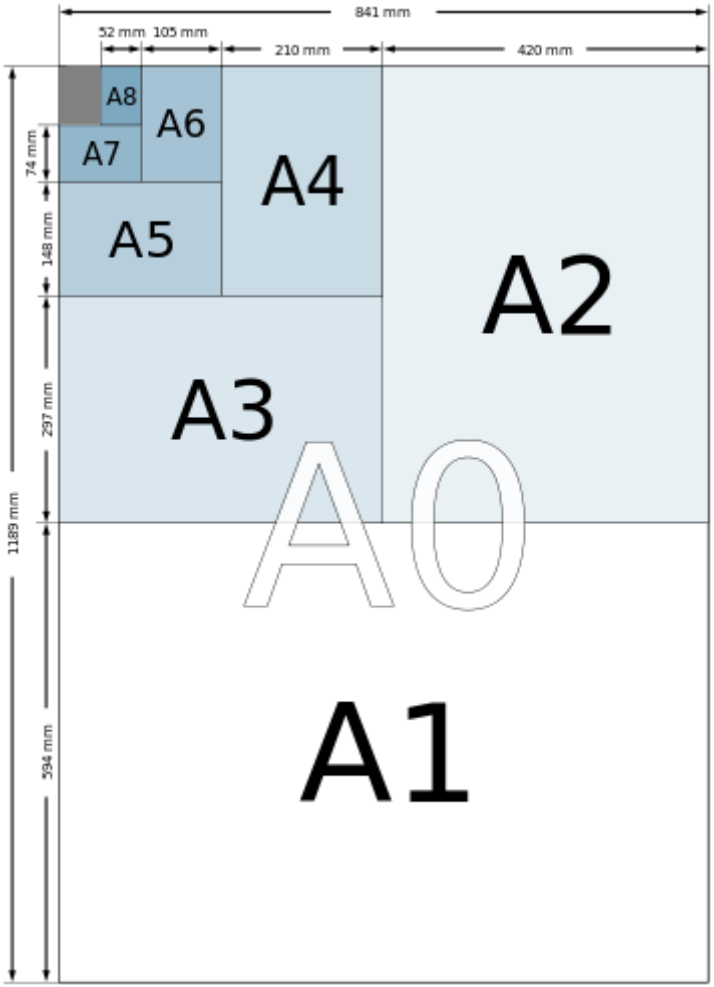
Example 2: Sphere Volume [4]

- **Implement ฟังก์ชันดังต่อไปนี้เพิ่มจากไฟล์ใน slide 17**
  - `find_r_from_surface_area()`
  - `sphere_volume()`

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

Example 3: Paper Size Standard

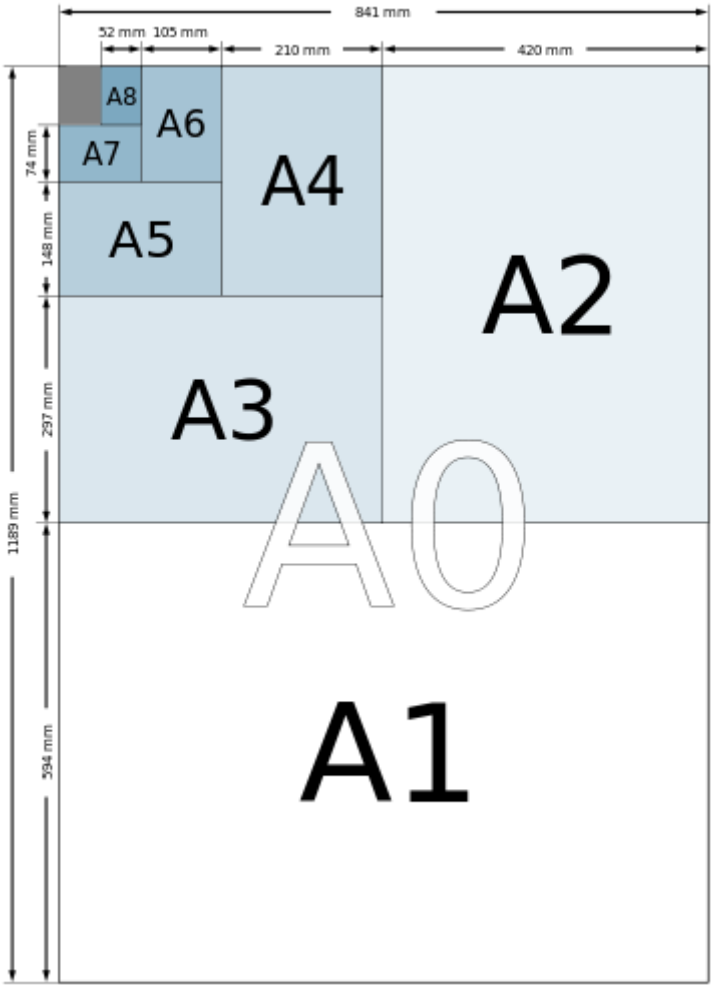
- **ISO 216 เป็นมาตรฐานสากลว่าด้วยขนาดกระดาษซึ่งประเทศส่วนใหญ่ใช้กันในปัจจุบัน โดยมีขนาดกระดาษเป็นชุด A และ B**
  - **ขนาดกระดาษชุด A เป็นที่นิยมใช้มากที่สุด โดยเฉพาะขนาด A4 (210 × 297 mm)**



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

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** ที่มีอัตราส่วนกว้าง:ยาว เท่าเดิมเช่นกัน



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

Example 3: Paper Size Standard [3]

- **Problem Statement**
  - เขียนฟังก์ชัน `paper_width(h)` เพื่อรับค่าความยาว h ของกระดาษที่มีอัตราส่วนตามมาตรฐาน ISO 216 แล้วคำนวณความกว้าง (Width) ของกระดาษดังกล่าว
- การวิเคราะห์ปัญหา
  - **Input:**
    - จำนวนข้อมูล 1 ชนิดข้อมูล float
  - **Output**
    - จำนวนข้อมูล 1 ชนิดข้อมูล float

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

Example 3: Paper Size Standard [4]

- **Problem Solving**

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/pseudocode.htm

Think Python: How to Think Like a Computer Scientist

References [2]

- http://www.cs.cmu.edu/~./15110/lectures/lec4-ProgrammingPart1.pdf
- http://www.cs.cmu.edu/~./15110/lectures/lec5-ProgrammingPart2.pdf
- https://docs.python.org/3/tutorial/inputoutput.html
- https://docs.python.org/3/library/stdtypes.html#old-string-formatting
- https://docs.python.org/3.4/library/functions.html