

w04-Lec1

Python Style Guide

for 204111

by Kittipitch Kuptavanich

Python Coding Convention

- **Code is read much more often than it is written**
- **Aims to improve readability**
- **Consistency**
 - **Project Level**
 - **Module Level**
- **ใช้ style guide จาก Python Enhancement Proposals (PEPs) หัวข้อที่ 0008**
 - **เรียกว่า PEP8**

Code Layout

- **Tabs or Spaces?**
 - **Space only**
- **Maximum Line Length**
 - **79 ตัวอักษร**
- **Source File Encoding**
 - **UTF-8 ใน Python3**

Tab or Space

- **Indent (ย่อหน้า) ที่ละ 4 space**
 - **ไม่ควรใช้ tab character (\t)**
 - **Python 3 ไม่อนุญาตให้ใช้ \t และ space ปนกันในไฟล์เดียว**
 - **ให้ตั้ง preference ของ editor ให้ใช้ space เท่านั้น**

Indentation

Continuation lines

- กรณีชื่อฟังก์ชันหรือชื่อ **parameter** ยาวและจำเป็นต้องเขียนข้ามบรรทัด

```
def long_function_name(var_one,
                        var_two,
                        var_three,
                        var_four):
    print(var_one)
```

แบบที่ 1:

- จัดระดับการย่อหน้าให้เท่ากับ parameter ตัวแรก

```
def long_function_name(
    var_one, var_two,
    var_three,
    var_four):
    print(var_one)
```

แบบที่ 2:

- **Hanging indent** คือการที่เปิดวงเล็บเฉย ๆ ที่บรรทัดแรก
- สังเกตการเพิ่ม level การ indent ที่บรรทัดถัดไป

Indentation

Closing parentheses `)` - brackets `]` – braces `}`

- วงเล็บปิด สามารถปิดที่บรรทัดใหม่ได้ โดยต้อง indent ให้ตรงอักษรแรกของบรรทัดสุดท้ายก่อนบรรทัดวงเล็บปิด

```
result = some_function(
    'a', 'b', 'c',
    'd', 'e', 'f'
)
```

- หรือ indent ให้ตรงอักษรแรกของบรรทัดที่เริ่ม statement

```
result = some_function(
    'a', 'b', 'c',
    'd', 'e', 'f'
)
```

Indentation

Conditional – `if`

```
# No extra indentation.  
if (this_is_one_thing and  
    that_is_another_thing):  
    do_something()
```

```
# Add a comment, which will provide some distinction in editors w/  
# syntax highlighting  
if (this_is_one_thing and  
    that_is_another_thing):  
    # Since both conditions are true,  
    do_something()
```

```
# Add some extra indentation on the conditional continuation line.  
if (this_is_one_thing  
    and that_is_another_thing):  
    do_something()
```

Blank Line

- ถ้าเป็น **function definition** หรือ **class definition** ภายใน **module** ที่อยู่ **level** นอกสุด
 - ให้เพิ่มบรรทัดว่างสองบรรทัด ระหว่าง **function definition** หรือ ระหว่าง **class definition**
- ถ้าเป็น **method** (ฟังก์ชันภายใน **class**) ให้คั่นด้วยบรรทัดว่างเพียง 1 บรรทัด
- พิจารณาการเพิ่มบรรทัดว่างภายในฟังก์ชัน เพื่อแยก **block** ของ **code** ออกจากกัน เพื่อให้อ่านง่ายขึ้น

Imports

- **Module** ละหนึ่งบรรทัด

Yes: `import os`
`import sys`

No: `import sys, os`

- หาก ต้องการ **import** เฉพาะฟังก์ชัน หรือ **class** จาก **Module** ใด ๆ สามารถเขียน ชื่อฟังก์ชันและ **class** จาก **Module เดียวกัน** ในบรรทัดเดียวกันได้

`from subprocess import Popen, PIPE`

- **Import** ควรอยู่ส่วนบนสุดของแต่ละไฟล์ โดยเรียงลำดับดังนี้
 - Standard library imports
 - Related third party imports
 - Local application/library specific imports

Note: หลีกเลี่ยงการใช้ wildcard import เช่น

No: `from math import *`
`import math`

Yes: `import math.pi`
 (harder to maintain)

White Space in Expression and Statements

- ไม่เว้นวรรคก่อนอักขระตัวแรก ภายใน **() []** หรือ **{}**

Yes: `spam(ham[1], {eggs: 2})`

No: `spam(ham[1], { eggs: 2 })`

- ไม่เว้นวรรคก่อน เครื่องหมาย **comma ','** **semicolon ';'** หรือ **colon ':'**

Yes: `if x == 4: print x, y; x, y = y, x`

No: `if x == 4 : print x , y ; x , y = y , x`

- ไม่เว้นวรรคก่อนวงเล็บ **()** ของ **function call** หรือ **[]** ใน **index**

Yes: `spam(1)`

No: `spam (1)`

Yes: `dct['key'] = lst[index]`

No: `dct ['key'] = lst [index]`

White Space in Expression and Statements [2]

- ไม่ใช่ space มากกว่าหนึ่งอักขระใน การทำ assignment หรือ operator อื่น ๆ เพื่อจัดย่อหน้า

Yes:

```
x = 1
y = 2
long_variable = 3
```

No:

```
x          = 1
y          = 2
long_variable = 3
```

- ควรใส่ space ระหว่าง binary operator และ operand
 - Assignment (=)
 - Augmented assignment (+= , -= etc.)
 - Comparisons (== , < , > , != , <= , >= , in , not in , is , is not)
 - Booleans (and , or , not)

Line Width

- แต่ละบรรทัดไม่ควรมีความยาวเกิน 79 อักขระ
- กรณีจำเป็นต้องเขียน **statement** ที่มีความยาวมากเกิน 79 อักขระ สามารถขึ้นบรรทัดใหม่ได้ โดยใช้เครื่องหมาย **backslash** `'\'` เพื่อแสดงว่า **statement** ยังไม่จบ เช่น

```
with open('/path/to/some/file/you/want/to/read') as file_1, \  
    open('/path/to/some/file/being/written', 'w') as file_2:  
    file_2.write(file_1.read())
```

Naming Styles

วิธีการตั้งชื่อที่พบได้ทั่วไปในภาษาสำหรับเขียนโปรแกรม
ต่าง ๆ

- `b` (single lowercase letter)
- `B` (single uppercase letter)
- `lowercase`
- `lower_case_with_underscores` (or `snake_case`)
- `UPPERCASE`
- `UPPER_CASE_WITH_UNDERSCORES`

Naming Styles [2]

- CapitalizedWords (**or** CapWords, **or** CamelCase **or** StudlyCaps)
 - ถ้ามีอักษรย่อใน **Capwords** ควรใช้ตัวใหญ่ทั้งหมดในตัวย่อ

Yes: **HTTP**ServerError

No: **Http**ServerError

- mixedCase (differs **from** CapitalizedWords by initial lowercase character!)
- Capitalized_Words_With_Underscores (**ugly!**)

Naming Styles [3]

- **Avoid using:**
 - **'1' , '0', 'I' as single character variable names.**
 - ในบาง font อักษรเหล่านี้ดูเหมือนเลข 1 หรือ 0
 - 'I' และ '1' ดูคล้ายกันมาก ในบาง font เช่น 'l' vs '1'
- ถ้าชื่อที่ต้องการใช้ ชำกกับ **reserved words** ควรเพิ่ม **underscore** ไปที่หลังชื่อ แทนการย่อคำหรือเพี้ยนตัวสะกด
 - **class_** is better than **class**

Naming Styles [4]

Package Names	<ul style="list-style-type: none"> • lowercase
Module Names	<ul style="list-style-type: none"> • lowercase • lower_case_with_underscores • Prefer shorter names
Class Names	<ul style="list-style-type: none"> • CapWords
Function Names	<ul style="list-style-type: none"> • lowercase • lower_case_with_underscores
Method Names	<ul style="list-style-type: none"> • lowercase • lower_case_with_underscores • Leading underscore for non public method
Variable Names	<ul style="list-style-type: none"> • lowercase • lower_case_with_underscores
Exceptions Names	<ul style="list-style-type: none"> • CapWords
Constant Names	<ul style="list-style-type: none"> • UPPERCASE • UPPER_CASE_WITH_UNDERSCORES

Programming Recommendation

- เมื่อมีการเปรียบเทียบกับค่า `None` ควรใช้ keyword `is` หรือ `is not` และไม่ควรใช้เครื่องหมาย `==`

Yes:

```
if foo is None:
```

No:

```
if foo == None:
```

Yes:

```
if foo is not None:
```

No:

```
if not foo is None:
```

No:

```
if foo != None:
```

No:

```
if foo
```

Programming Recommendation [2]

- พิจารณาการใช้คำสั่ง `return` ให้เหมือนกันทั้งฟังก์ชัน (consistent)
 - หากเป็น non-void function ต้องมีการ `return` ทุกกรณี

Yes:

```
def foo(x):  
    if x >= 0:  
        return math.sqrt(x)  
    else:  
        return None  
  
def bar(x):  
    if x < 0:  
        return None  
    return math.sqrt(x)
```

No:

```
def foo(x):  
    if x >= 0:  
        return math.sqrt(x)  
  
def bar(x):  
    if x < 0:  
        return  
    return math.sqrt(x)
```

Programming Recommendation [3]

- ไม่เปรียบเทียบ Boolean value กับค่า **True** หรือ **False** ด้วยเครื่องหมาย **==**

Yes:

```
if greeting:  
    do_something()
```

No:

```
if greeting == True:  
    do_something()
```

Worse:

```
if greeting is True:  
    do_something()
```