Pythonic Coding Style

C-START Python PD Workshop

A Foolish Consistency is the Hobgoblin of Little Minds

- Guido van Rossum (creator of Python) makes a point: *code is read more often than it is written*, so **readability counts**.

A Foolish Consistency is the Hobgoblin of Little Minds

- Guido van Rossum (creator of Python) makes a point: *code is read more often than it is written*, so **readability counts**.
- Python is one of the few languages with an official style guide (PEP-8) since there is a huge amount of Python code out there and the language's core principle is readability.

A Foolish Consistency is the Hobgoblin of Little Minds

- Guido van Rossum (creator of Python) makes a point: *code is read more often than it is written*, so **readability counts**.
- Python is one of the few languages with an official style guide (PEP-8) since there is a huge amount of Python code out there and the language's core principle is readability.
- Thus, it's important to follow Python's official style whenever possible

A Foolish Consistency is the Hobgoblin of Little Minds

- Guido van Rossum (creator of Python) makes a point: *code is read more often than it is written*, so **readability counts**.
- Python is one of the few languages with an official style guide (PEP-8) since there is a huge amount of Python code out there and the language's core principle is readability.
- Thus, it's important to follow Python's official style whenever possible

Legacy Code

It should be noted that when working on a project that was started before the ages of PEP-8 (before 2001), generally they have their own style guide and you should follow that instead. Otherwise, it would be generally considered unacceptable to not follow PEP-8.

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names
- You should avoid using underscores when possible to improve readability (e.g. `randint` is better than `rand_int`, as the naming is obvious without the underscore).

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names
- You should avoid using underscores when possible to improve readability (e.g. `randint` is better than `rand_int`, as the naming is obvious without the underscore).
- When there are conflicts with builtin keywords and a better name is not possible, an underscore should be appended to the variable name (e.g. `class_`)

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names
- You should avoid using underscores when possible to improve readability (e.g. `randint` is better than `rand_int`, as the naming is obvious without the underscore).
- When there are conflicts with builtin keywords and a better name is not possible, an underscore should be appended to the variable name (e.g. `class_`)
- Class names should be typed in `CapWords`

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names
- You should avoid using underscores when possible to improve readability (e.g. `randint` is better than `rand_int`, as the naming is obvious without the underscore).
- When there are conflicts with builtin keywords and a better name is not possible, an underscore should be appended to the variable name (e.g. `class_`)
- Class names should be typed in `CapWords`
- Function, method, and class names should describe the interface rather than the implementation.

Naming

- Python uses `snake_case` for variable names, function names, method names, and module names
- You should avoid using underscores when possible to improve readability (e.g. `randint` is better than `rand_int`, as the naming is obvious without the underscore).
- When there are conflicts with builtin keywords and a better name is not possible, an underscore should be appended to the variable name (e.g. `class_`)
- Class names should be typed in `CapWords`
- Function, method, and class names should describe the interface rather than the implementation.
- Private methods and variables should start with an underscore.

Indentation

As Python uses the indentation of the text to denote scope, consistency of indentation is critically important. PEP-8 recommends the following:

- Use 4 spaces per indentation level, **never use hard tabs**.

Indentation

As Python uses the indentation of the text to denote scope, consistency of indentation is critically important. PEP-8 recommends the following:

- Use 4 spaces per indentation level, **never use hard tabs**.
- On multiline function calls, list literals, etc., the arguments should be aligned and indented from the rest of the text. "Hanging indent" is acceptable as well.

Indentation

As Python uses the indentation of the text to denote scope, consistency of indentation is critically important. PEP-8 recommends the following:

- Use 4 spaces per indentation level, **never use hard tabs**.
- On multiline function calls, list literals, etc., the arguments should be aligned and indented from the rest of the text. "Hanging indent" is acceptable as well.
- Multiline `if`/`while` etc. should be indented to align with the top line

Other Pet Peeves

- Keep lines to 79 characters[1]

[1] It's OK to go to 90 or 100 if everyone in your project agrees.

Other Pet Peeves

- Keep lines to 79 characters[1]
- Avoid extraneous whitespace inside parentheses, brackets, and braces

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

[1] It's OK to go to 90 or 100 if everyone in your project agrees.

Other Pet Peeves

- Keep lines to 79 characters[1]
- Avoid extraneous whitespace inside parentheses, brackets, and braces

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

- Don't use parentheses on `if`/`while` etc. like you might in C-like languages

```
Yes: if i < 3:
No:  if(i < 3):
```

[1] It's OK to go to 90 or 100 if everyone in your project agrees.

Truthiness

Anything `False`, zero, or an empty sequence/mapping will implicity be false, and you *should* take advantage of that.

```
Ok:         if mybool == True:
Pythonic:   if mybool:

Ok:         if mynumber != 0:
Pythonic:   if mynumber:

Ok:         if len(mylist) == 0:
Pythonic:   if not mylist:
```

Comments

Every comment in the source code is a personal failure of the programmer, because it proves that he didn't manage to express the purpose of the code fragment with the programming language itself.

— Uncle Bob

Take Home: Comments are important when they are needed, but you should try and make your code readable instead.

Readability Counts!

No really, it is of utmost importance that Python code be readable **by following the guidelines of PEP-8**. You should read through PEP-8 before getting serious with Python.