Advanced Python / Kurs rozszerzony języka Python

Martin Böhm (University of Wrocław)

October 2022

Lab 02: Pythonic and Unpythonic

Pythonic – what does it mean?

Pythonic – what does it mean?

- Code being in some (subjective, cultural) way being true to the spirit of Python.
- Code complying with the recommended style guides and PEPs.
- (PEP – Python Enhancement Proposal – technical report that *usually* discusses and advocates for an extension of Python.)
- Code using specific Python idioms that the current *online vocal supporters* appreciate.

Pythonic – what does it mean?

- Code being in some (subjective, cultural) way being true to the spirit of Python.
- Code complying with the recommended style guides and PEPs.
- (PEP – Python Enhancement Proposal – technical report that *usually* discusses and advocates for an extension of Python.)
- Code using specific Python idioms that the current *online vocal supporters* appreciate.

Q: Do I have to write the homework in a "Pythonic" way?

Pythonic – what does it mean?

- Code being in some (subjective, cultural) way being true to the spirit of Python.
- Code complying with the recommended style guides and PEPs.
- (PEP – Python Enhancement Proposal – technical report that *usually* discusses and advocates for an extension of Python.)
- Code using specific Python idioms that the current *online vocal supporters* appreciate.

Q: Do I have to write the homework in a "Pythonic" way?

A: Of course not! But it is good to have an idea of what people mean by it.

Pythonic – what does it mean?

- Code being in some (subjective, cultural) way being true to the spirit of Python.
- Code complying with the recommended style guides and PEPs.
- (PEP – Python Enhancement Proposal – technical report that *usually* discusses and advocates for an extension of Python.)
- Code using specific Python idioms that the current *online vocal supporters* appreciate.

Q: Do I have to write the homework in a “Pythonic” way?

A: Of course not! But it is good to have an idea of what people mean by it. In a way, you should *know your enemy*.

PEP 20 – the Zen of Python

`import this` – try this statement in Python!

PEP 20 – the Zen of Python

`import this` – try this statement in Python!

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one – and preferably only one – obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than **right** now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!

PEP 8 – Style Guide

`https://www.python.org/dev/peps/pep-0008/`

- (Other PEPs are usually serious, technical texts.)
- Describes *preferred* indentation (4 spaces), PascalCase for classes, lower_case for functions and variables ...
- A lot of it is actually handled by the IDE, if you use one.
- A controversial guide: *Limit all lines to a maximum of 79 characters.*
- The guide tells us what *style* is Pythonic, but not what *code* is Pythonic.

Interactive part

- I have a few examples ready for you – can you guess how to make the code within more Pythonic?

Sources, useful links

1. https://towardsdatascience.com/how-to-be-pythonic-and-why-you-should-care-188d63a5037
2. https://www.mattlayman.com/blog/2017/pythonic-code-the-property-decorator/
3. https://www.mattlayman.com/blog/2017/pythonic-code-the-list-comprehension/
4. https://programming.vip/docs/15-pythonic-code-examples.html