Onward! 2018
Boston Park Plaza Hotel
Boston, Massachusetts, United States
07 November 2018

On the Usage of Pythonic Idioms

Carol V. Alexandru[1], José J. Merchante[2], Sebastiano Panichella[1,3]
Sebastian Proksch[1], Harald C. Gall[1], Gregorio Robles[2]

[1]Software Evolution and Architecture Lab, University of Zurich, Switzerland
{alexandru,proksch,gall}@ifi.uzh.ch

[2]Grupo de Sistemas y Comunicaciones, Universidad Rey Juan Carlos, Spain
jj.merchante@alumnos.urjc.es, grex@gsyc.urjc.es

[3]Service Prototype Lab, Zurich University of Applied Sciences, Switzerland
panc@zhaw.ch

METROPOLIS

Carol V. Alexandru

Sebastiano Panichella

Sebastian Proksch

Harald C. Gall

José J. Merchante

Gregorio Robles

Things to know about Python

Things to know about Python

- Created by Guido van Rossum

Things to know about Python

- Created by Guido van Rossum
  - He is (was) the “Benevolent Dictator for Life”
  - Makes the final decisions when necessary

Things to know about Python

- Created by Guido van Rossum
  - He is (was) the "Benevolent Dictator for Life"
  - Makes the final decisions when necessary
- Strong principles
  - The "Zen of Python" `>>> import this`

Things to know about Python

```
>>> import this
The Zen of Python, by Tim Peters

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
```

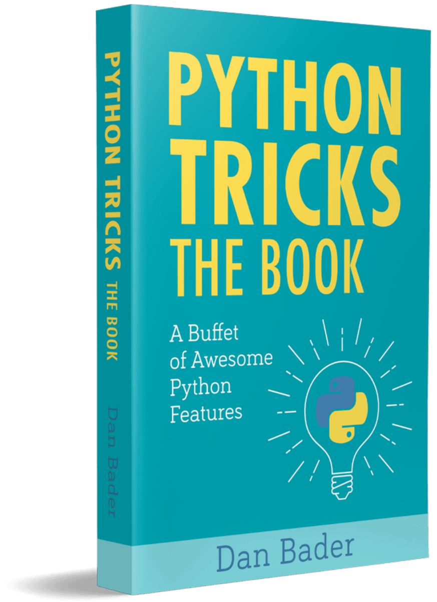
Things to know about Python

- Created by Guido van Rossum
  - He is (was) the “Benevolent Dictator for Life”
  - Makes the final decisions when necessary
- Strong principles
  - The “Zen of Python” >>> `import this`
  - Python Enhancement Proposals (PEPs)

Things to know about Python

- Created by Guido van Rossum
  - He is (was) the “Benevolent Dictator for Life”
  - Makes the final decisions when necessary
- Strong principles
  - The “Zen of Python” >>> `import this`
  - Python Enhancement Proposals (PEPs)
- Widespread adoption across many fields

Pythonic appears to be 'a thing'

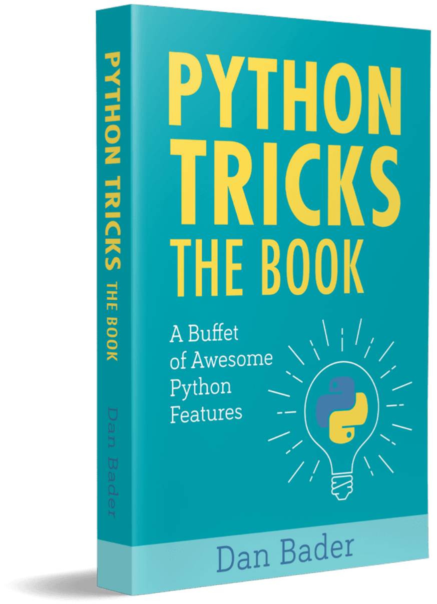


Mailman
GNU

#irc

stackoverflow

Pythonic appears to be 'a thing'



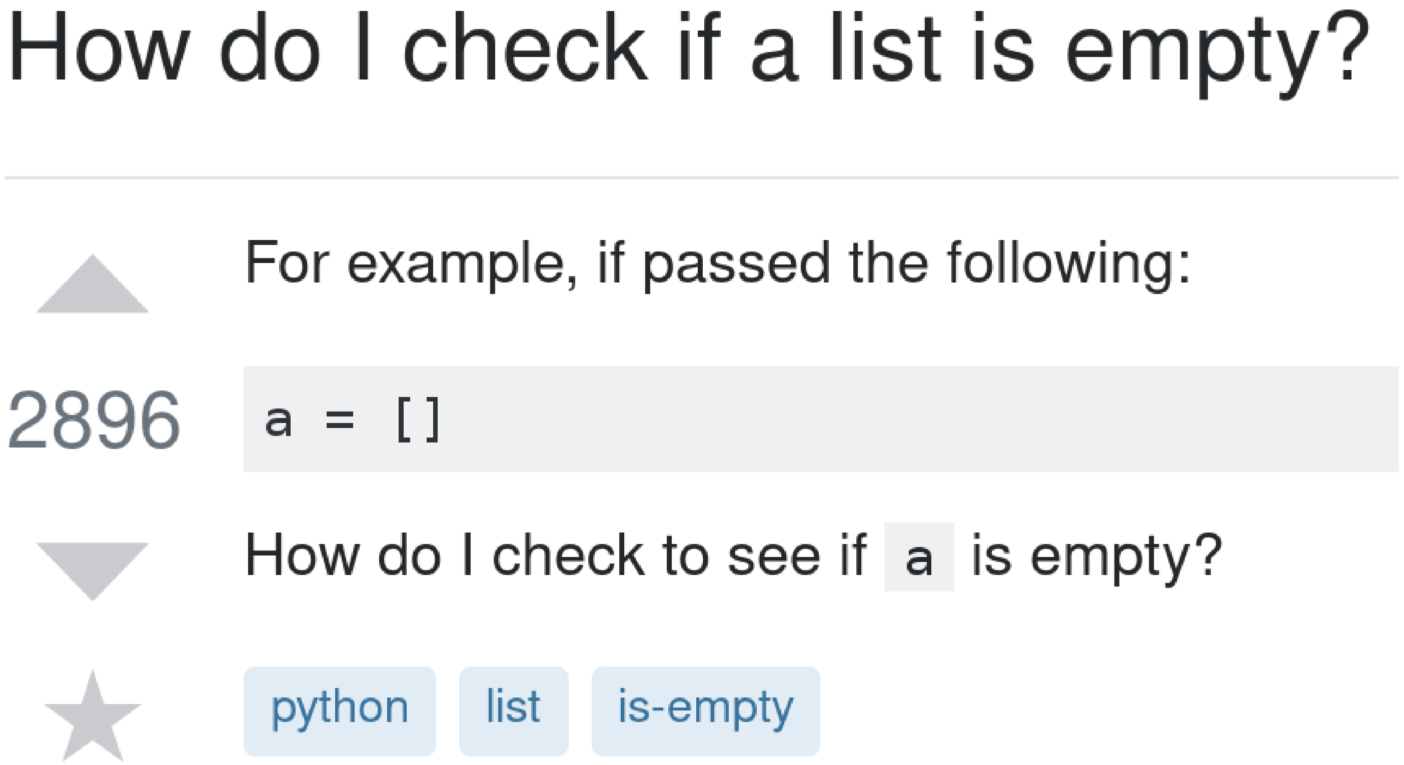
Mailman GNU

#irc

stackoverflow

“It would be more **pythonic** to...”

A simple question


How do I check if a list is empty?

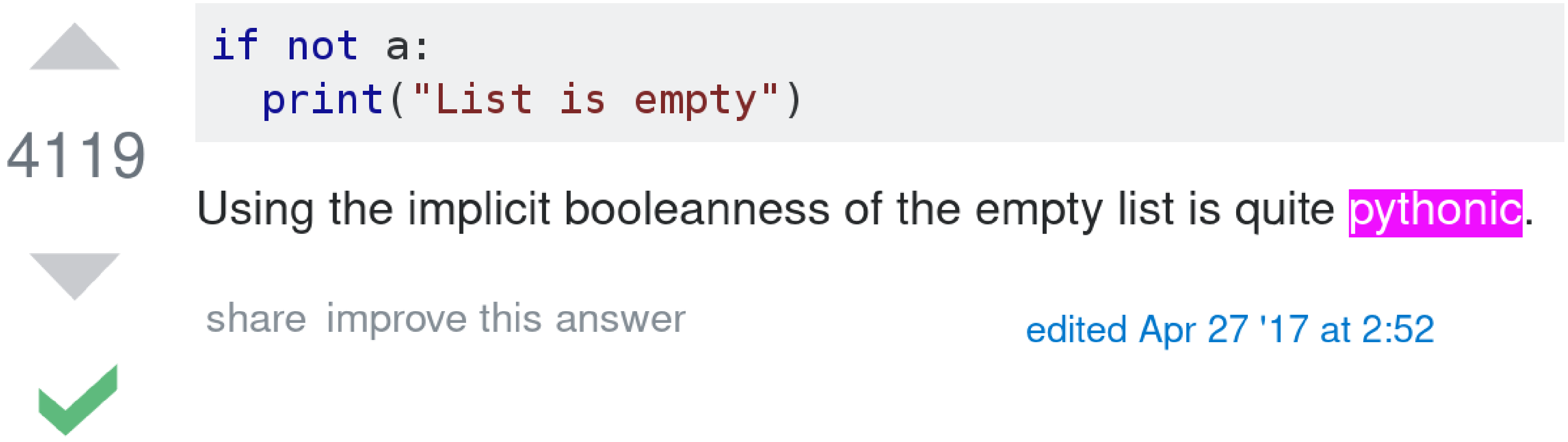
2896

For example, if passed the following:

```
a = []
```

How do I check to see if `a` is empty?

Top Answer claims to be "pythonic"...



4119

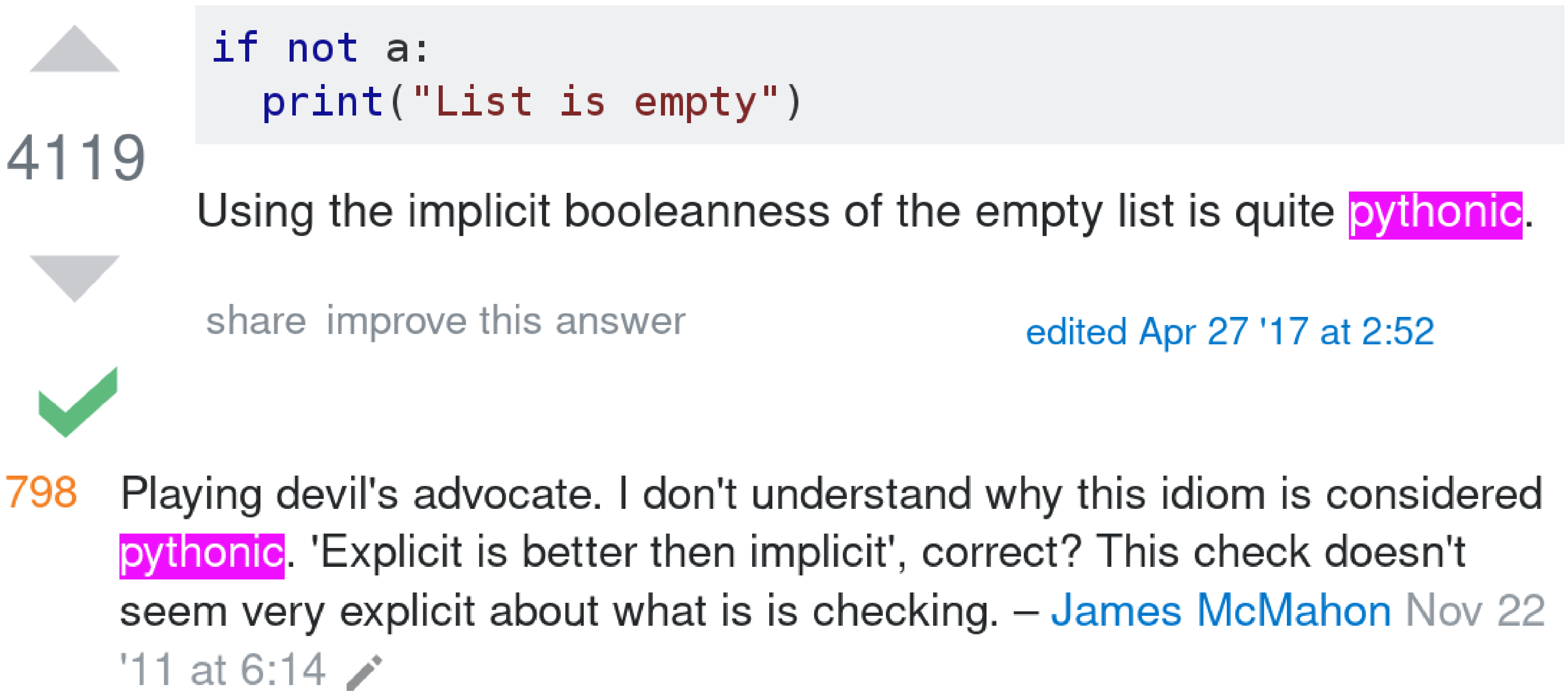
```
if not a:
  print("List is empty")
```

Using the implicit booleanness of the empty list is quite pythonic.

share improve this answer

edited Apr 27 '17 at 2:52

...but 100s of people seem sceptical



4119

```
if not a:
  print("List is empty")
```

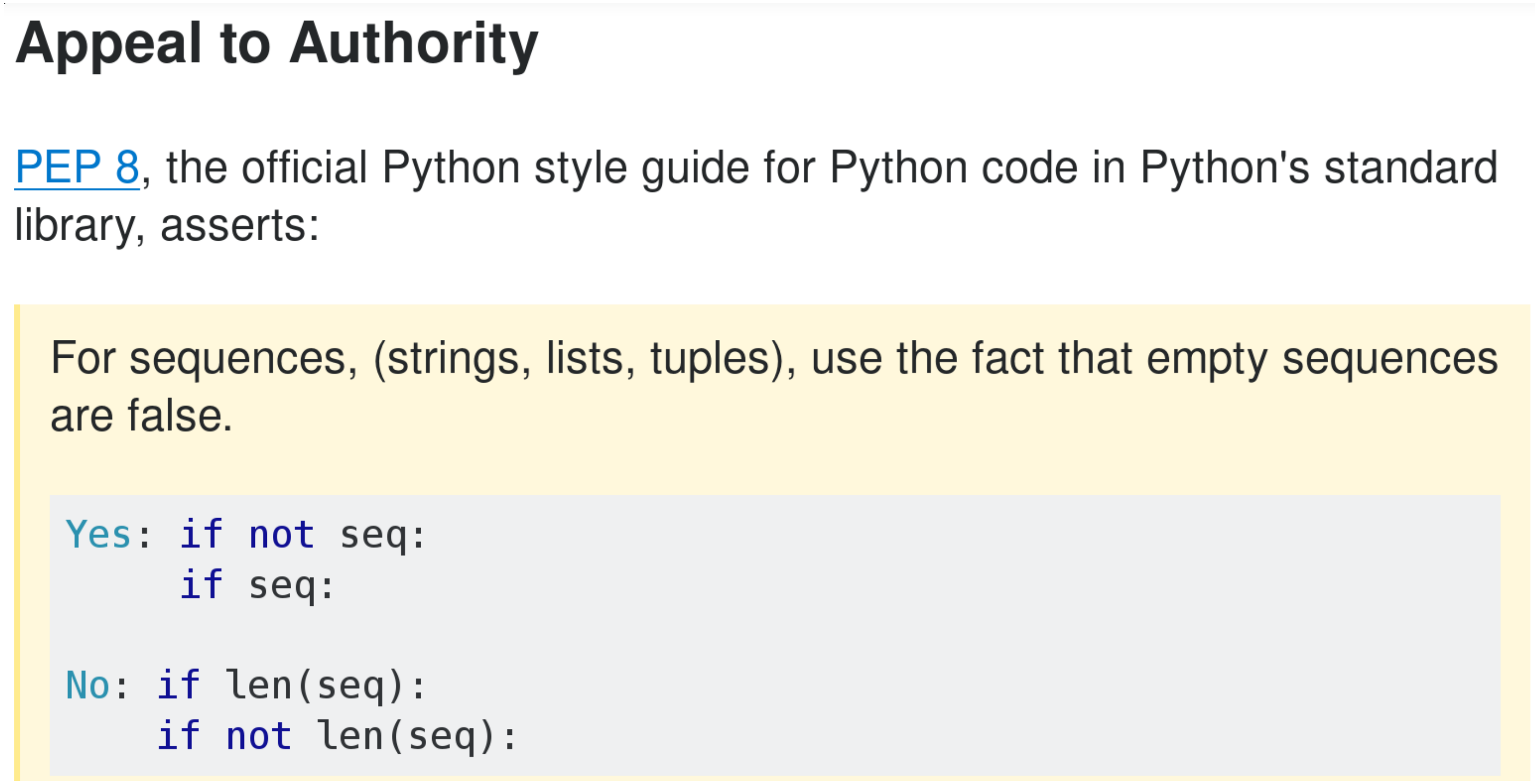
Using the implicit booleanness of the empty list is quite pythonic.

share improve this answer

edited Apr 27 '17 at 2:52

798 Playing devil's advocate. I don't understand why this idiom is considered pythonic. 'Explicit is better then implicit', correct? This check doesn't seem very explicit about what is is checking. – James McMahon Nov 22 '11 at 6:14

"Pythonic" because of the style guide (PEP 8)



Appeal to Authority

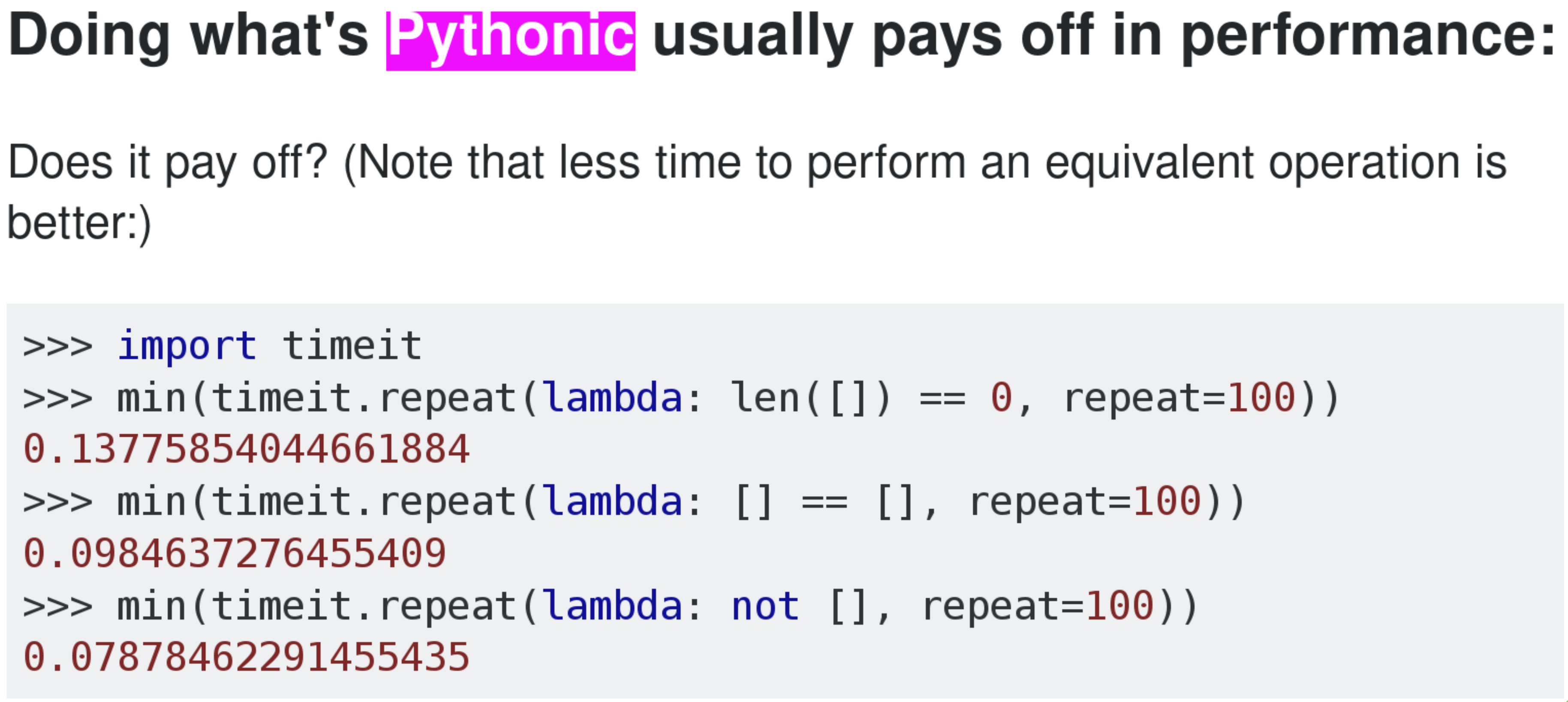
PEP 8, the official Python style guide for Python code in Python's standard library, asserts:

> For sequences, (strings, lists, tuples), use the fact that empty sequences are false.

```
Yes: if not seq:
     if seq:

No: if len(seq):
    if not len(seq):
```

“Pythonic” because of performance

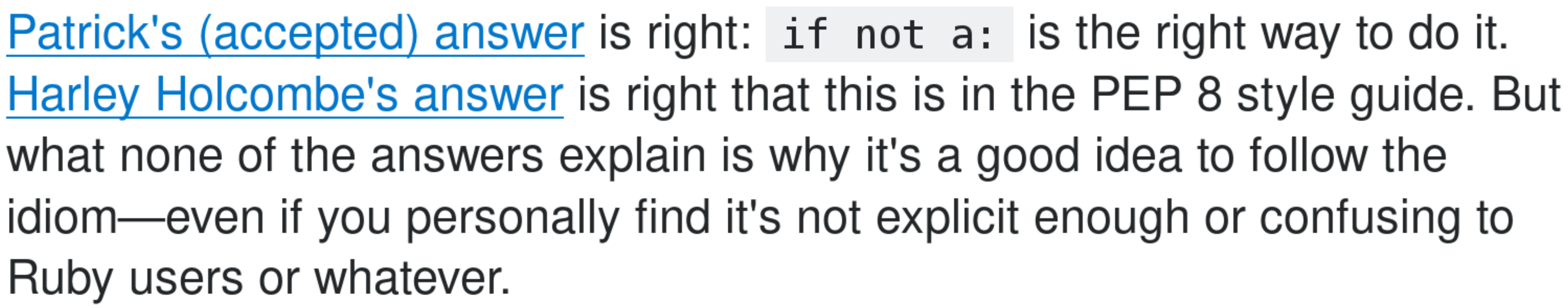


Doing what's Pythonic usually pays off in performance:

Does it pay off? (Note that less time to perform an equivalent operation is better:)

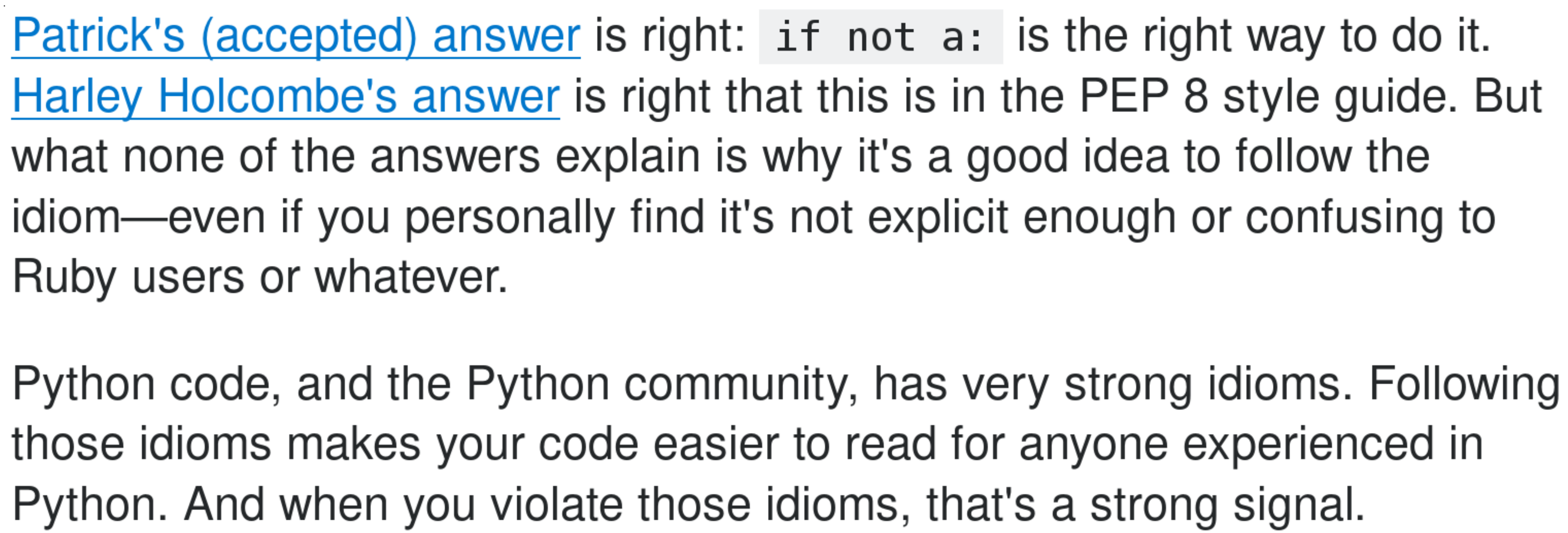
```
>>> import timeit
>>> min(timeit.repeat(lambda: len([]) == 0, repeat=100))
0.13775854044661884
>>> min(timeit.repeat(lambda: [] == [], repeat=100))
0.0984637276455409
>>> min(timeit.repeat(lambda: not [], repeat=100))
0.07878462291455435
```

Idioms and signaling



Patrick's (accepted) answer is right: `if not a:` is the right way to do it. Harley Holcombe's answer is right that this is in the PEP 8 style guide. But what none of the answers explain is why it's a good idea to follow the idiom—even if you personally find it's not explicit enough or confusing to Ruby users or whatever.

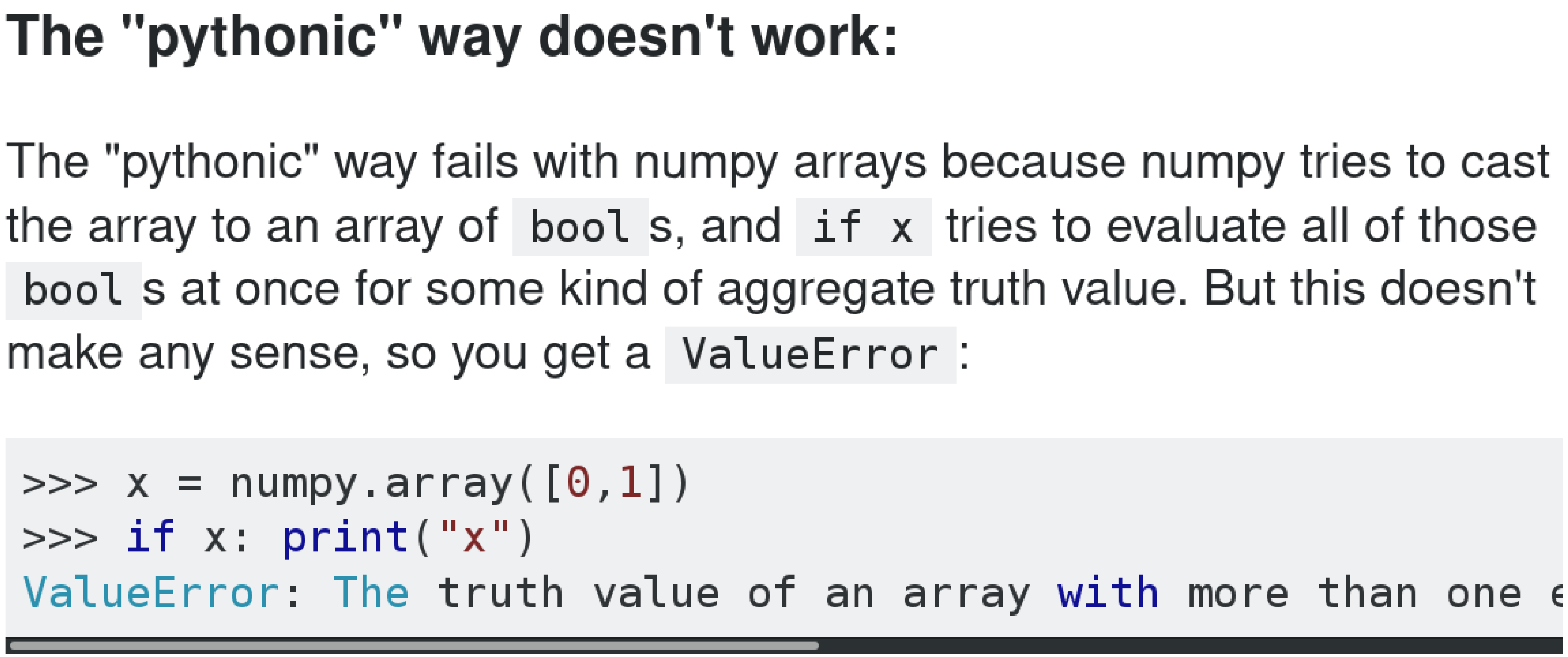
Idioms and signaling



Patrick's (accepted) answer is right: `if not a:` is the right way to do it. Harley Holcombe's answer is right that this is in the PEP 8 style guide. But what none of the answers explain is why it's a good idea to follow the idiom—even if you personally find it's not explicit enough or confusing to Ruby users or whatever.

Python code, and the Python community, has very strong idioms. Following those idioms makes your code easier to read for anyone experienced in Python. And when you violate those idioms, that's a strong signal.

And there are exceptions

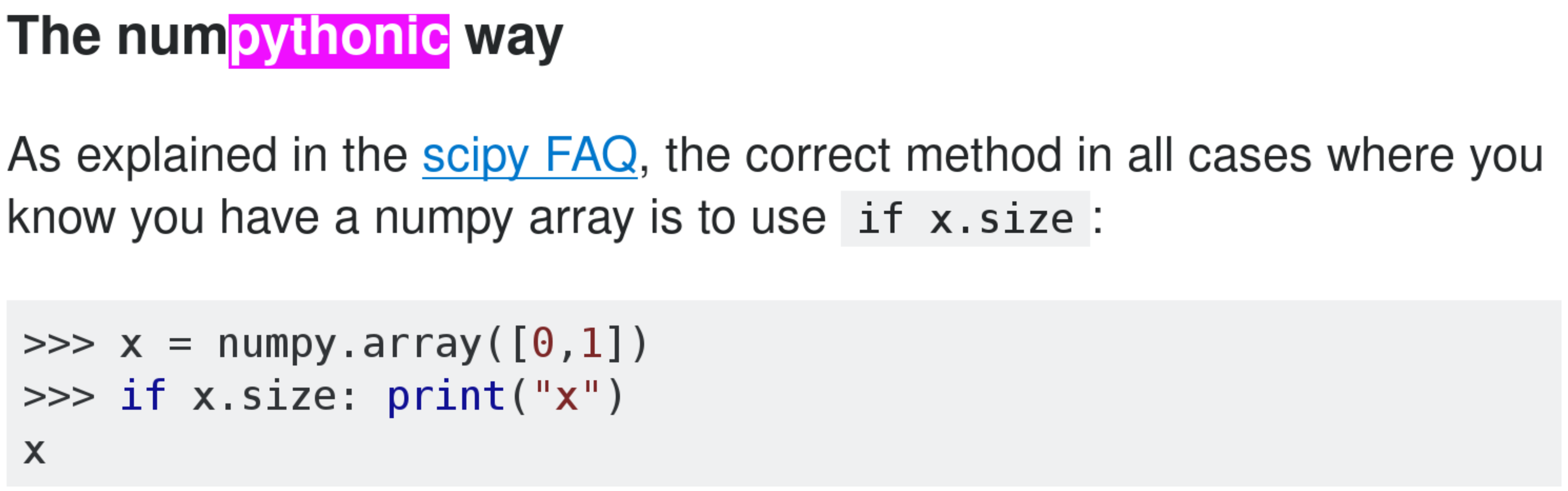


The "pythonic" way doesn't work:

The "pythonic" way fails with numpy arrays because numpy tries to cast the array to an array of `bool`s, and `if x` tries to evaluate all of those `bool`s at once for some kind of aggregate truth value. But this doesn't make any sense, so you get a `ValueError`:

```
>>> x = numpy.array([0,1])
>>> if x: print("x")
ValueError: The truth value of an array with more than one e
```

Creative use of the word "Pythonic"



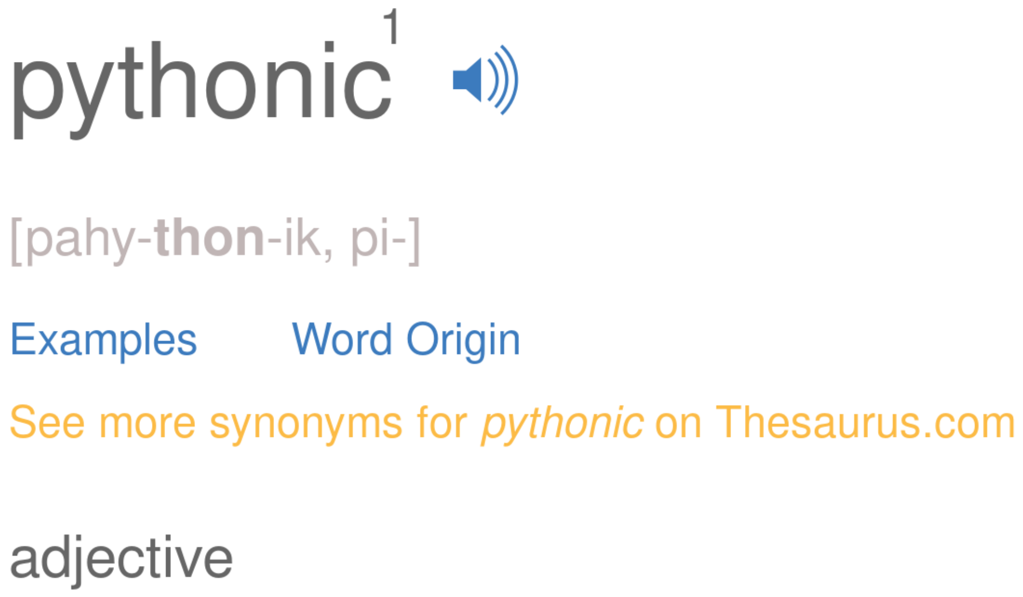
The numpythonic way

As explained in the scipy FAQ, the correct method in all cases where you know you have a numpy array is to use `if x.size`:

```
>>> x = numpy.array([0,1])
>>> if x.size: print("x")
x
```

Is there a definition for "Pythonic"?

Is there a definition for "Pythonic"?



pythonic^1

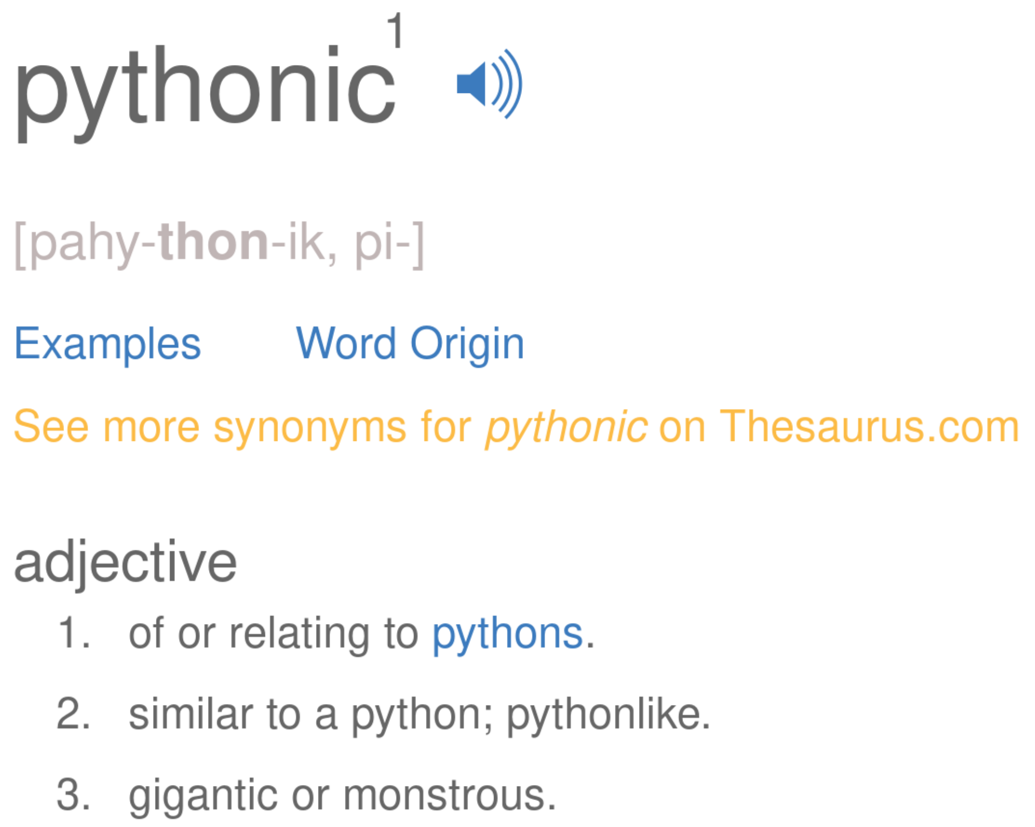
[pahy-**thon**-ik, pi-]

Examples Word Origin

See more synonyms for *pythonic* on Thesaurus.com

adjective

Is there a definition for “Pythonic”?



pythonic[1]

[pahy-**thon**-ik, pi-]

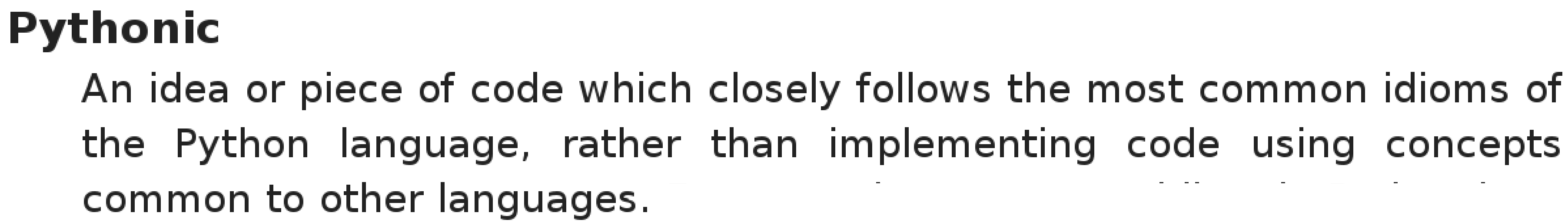
Examples Word Origin

See more synonyms for *pythonic* on Thesaurus.com

adjective

1. of or relating to pythons.
2. similar to a python; pythonlike.
3. gigantic or monstrous.

Is there a definition for “Pythonic”?



Pythonic

An idea or piece of code which closely follows the most common idioms of the Python language, rather than implementing code using concepts common to other languages.

Pythonic vs. Non-pythonic

```python
# numbers from 1 to 999
xs = range(1, 1000)
```

Pythonic vs. Non-pythonic

```
# numbers from 1 to 999
xs = range(1, 1000)

# Non-pythonic
res = []
for index in range(0, len(xs)):
    if xs[index] % 2 == 0:
        res.append(xs[index] * 3)
```

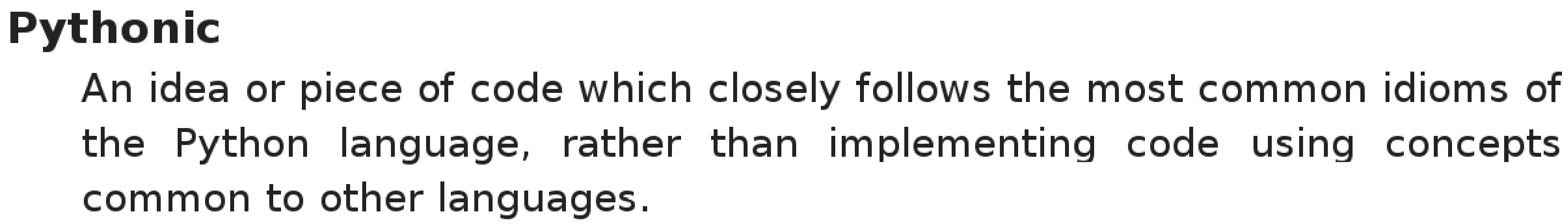
Pythonic vs. Non-pythonic

```python
# numbers from 1 to 999
xs = range(1, 1000)

# Non-pythonic
res = []
for index in range(0, len(xs)):
    if xs[index] % 2 == 0:
        res.append(xs[index] * 3)

# Pythonic
res = [x * 3 for x in xs if x % 2 == 0]
```

Is there a definition for “Pythonic”?



Pythonic

An idea or piece of code which closely follows the most common idioms of the Python language, rather than implementing code using concepts common to other languages.

So it's “Using Python-specific syntax and concepts”, right?

But what do developers believe?

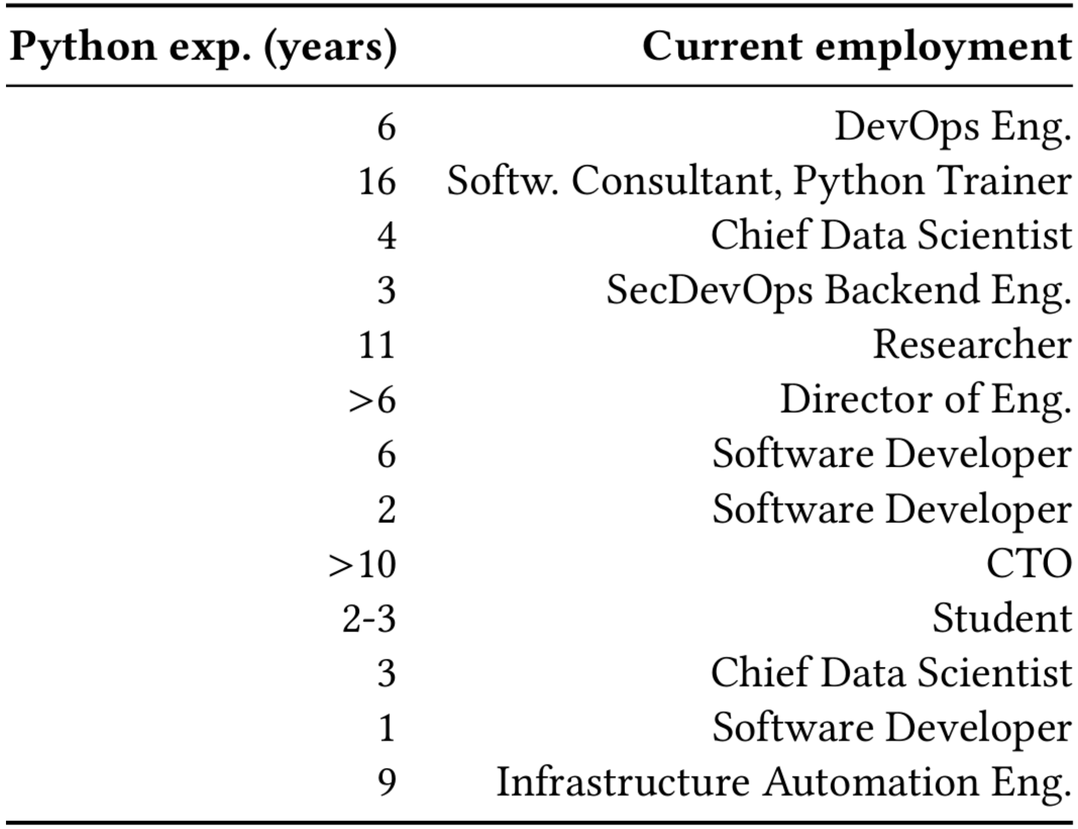
Let's ask a few developers

Interviews done

- in Person
- at a Python conference
- in Spain
- using open questions

Let's ask a few developers

Interviews done
- in Person
- at a Python conference
- in Spain
- using open questions



Python exp. (years)	Current employment
6	DevOps Eng.
16	Softw. Consultant, Python Trainer
4	Chief Data Scientist
3	SecDevOps Backend Eng.
11	Researcher
>6	Director of Eng.
6	Software Developer
2	Software Developer
>10	CTO
2-3	Student
3	Chief Data Scientist
1	Software Developer
9	Infrastructure Automation Eng.

What does Pythonic mean?

What does Pythonic mean?

“*elegant* and *readable* code”

“makes code easier to *understand* and *maintain*”

“boosts *readability* and *performance*”

What does Pythonic mean?

"*elegant* and *readable* code"

"makes code easier to *understand* and *maintain*"

"boosts *readability* and *performance*"

"using *features provided by the language* or standard library"

What does Pythonic mean?

“*elegant* and *readable* code”

“makes code easier to *understand* and *maintain*”

“boosts *readability* and *performance*”

“simply the *most accepted way* of writing python”

“using *features provided by the language* or standard library”

Using Python idioms != Pythonic

> *While there are many idioms in Python, using them does not mean that you're writing pythonic code. Sometimes, idioms make the code less readable, or more complicated.*

Using Python idioms != Pythonic

> *While there are many idioms in Python, using them does not mean that you're writing pythonic code. Sometimes, idioms make the code less readable, or more complicated.*

→ Using idioms != Pythonic code

→ Using idioms != always more readable

Novice vs. Pro

Novice:

- Better style
- Fewer lines of code

Pro:

- Using built-in functionality
- Efficient execution

Novice vs. Pro

Novice:

- Better style
- Fewer lines of code
- Using idioms

Pro:

- Using built-in functionality
- Efficient execution
- Writing elegant code

Novice vs. Pro

Novice:

- Better style
- Fewer lines of code
- Using idioms
- Simpler interpretation

Pro:

- Using built-in functionality
- Efficient execution
- Writing elegant code
- Less concrete interpretation

Learning 'Pythonic'?

Learning 'Pythonic'?

“StackOverflow shows you *multiple points of view* of people, and you always learn.”

Learning 'Pythonic'?

“StackOverflow shows you *multiple points of view* of people, and you always learn.”

“*reading code* in repositories of other projects”

“saw them in *documentation*”

Learning 'Pythonic'?

“StackOverflow shows you *multiple points of view* of people, and you always learn.”

“*reading code* in repositories of other projects”

“saw them in *documentation*”

“from colleagues
during *code review”*

Learning 'Pythonic'?

“StackOverflow shows you *multiple points of view* of people, and you always learn.”

“*reading code* in repositories of other projects”

“saw them in *documentation*”

“My code became more pythonic *year after year*”

“from colleagues during *code review”*

“Becoming a pythonic programmer *takes time*”

Learning 'Pythonic'?

“StackOverflow shows you *multiple points of view* of people, and you always learn.”

“*reading code* in repositories of other projects”

“saw them in *documentation*”

“My code became more pythonic *year after year*”

“from colleagues during *code review”*

“Becoming a pythonic programmer *takes time*”

→ Pythonic *not* taught in books or lectures

→ Seems to creep in with experience

Do you care? Do your peers care?

Do you care? Do your peers care?

“Pythonic code is *positively viewed* but *not required*”

Do you care? Do your peers care?

“Pythonic code is *positively viewed* but *not required*”

> “If you're a novice Python programmer, better *focus on general programming skills*”

Do you care? Do your peers care?

“Pythonic code is *positively viewed* but *not required*”

“Pythonic idioms can at least be used to *measure a developer’s knowledge*.”

“If you're a novice Python programmer, better *focus on general programming skills*”

Do you care? Do your peers care?

“Pythonic code is *positively viewed* but *not required*”

“Pythonic idioms can at least be used to *measure a developer’s knowledge*.”

“If you're a novice Python programmer, better *focus on general programming skills*”

“If I learn a new idiom I add it to my toolbox and then when I touch something, I modify it and leave it better, but it’s not an obsession.

Do you care? Do your peers care?

“Pythonic code is *positively viewed* but *not required*”

“Pythonic idioms can at least be used to *measure a developer’s knowledge*.”

“If you're a novice Python programmer, better *focus on general programming skills*”

“If I learn a new idiom I add it to my toolbox and then when I touch something, I modify it and leave it better, but it’s not an obsession.

→ Pythonic important, but not formally

→ Pythonic signals expertise and garners respect

A catalogue of Pythonic Idioms

A catalogue of Pythonic Idioms

- Published online at http://pythonic.libresoft.info/catalogue
  - including examples, references and benchmarks

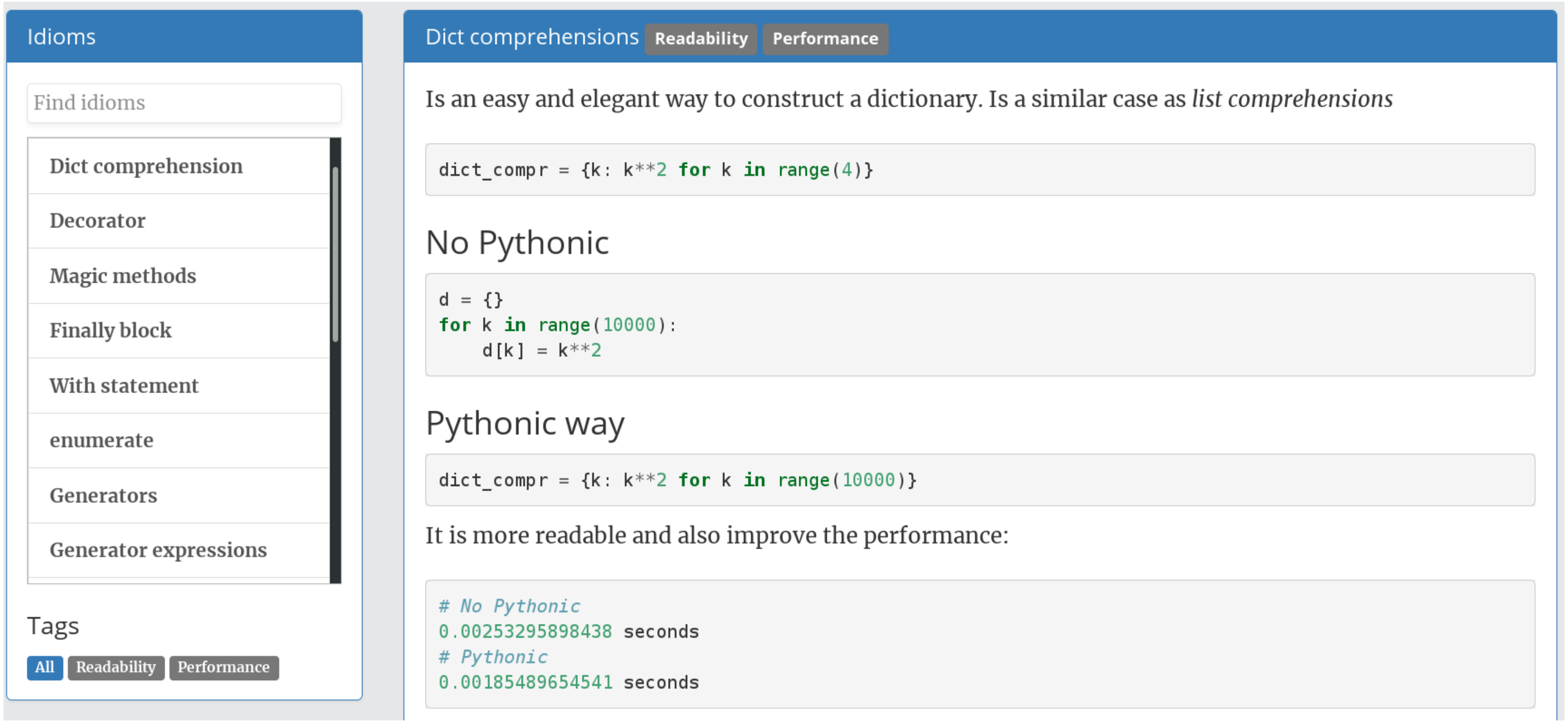
A catalogue of Pythonic Idioms

- Published online at http://pythonic.libresoft.info/catalogue
  - including examples, references and benchmarks
- Compiled from
  - Several books (on learning *and* applying Python)
  - Online statements by influential and renowned Python developers

A catalogue of Pythonic Idioms

- Published online at http://pythonic.libresoft.info/catalogue
  - including examples, references and benchmarks
- Compiled from
  - Several books (on learning *and* applying Python)
  - Online statements by influential and renowned Python developers
- Classified into “performance” and “readability”
  - performance measured using benchmarks

A catalogue of Pythonic Idioms



Dict comprehensions Readability Performance

Is an easy and elegant way to construct a dictionary. Is a similar case as *list comprehensions*

```
dict_compr = {k: k**2 for k in range(4)}
```

No Pythonic

```
d = {}
for k in range(10000):
    d[k] = k**2
```

Pythonic way

```
dict_compr = {k: k**2 for k in range(10000)}
```

It is more readable and also improve the performance:

```
# No Pythonic
0.00253295898438 seconds
# Pythonic
0.00185489654541 seconds
```

Empirical study

Empirical study setup

- Most recent commit in 1000 Python projects from GitHub
  - >1mb, sorted by stars, cleaned for books etc., no forks, not archived
  - 178,735 files, 38,505,577 lines of code

Empirical study setup

- Most recent commit in 1000 Python projects from GitHub
  - >1mb, sorted by stars, cleaned for books etc., no forks, not archived
  - 178,735 files, 38,505,577 lines of code
- Analyzed using LISA (http://t.uzh.ch/Fk)
  - AST-based detection of all idioms

Empirical study setup

- Most recent commit in 1000 Python projects from GitHub
  - >1mb, sorted by stars, cleaned for books etc., no forks, not archived
  - 178,735 files, 38,505,577 lines of code
- Analyzed using LISA (http://t.uzh.ch/Fk)
  - AST-based detection of all idioms
- Idiom occurences in #projects and total occurence count

Empirical study - a quick look

Idiom	# projects (out of 1000)	# of occurences
List comprehension	866	75,466
Generator expressions	709	33,038
Dict comprehension	146	796

Empirical study - a quick look

Idiom	# projects (out of 1000)	# of occurences
List comprehension	866	75,466
Generator expressions	709	33,038
Dict comprehension	146	796
Simple magic methods		
Intermediate magic methods		
Advanced magic methods		

Empirical study - a quick look

Idiom	# projects (out of 1000)	# of occurences
List comprehension	866	75,466
Generator expressions	709	33,038
Dict comprehension	146	796
Simple magic methods		
Intermediate magic methods		
Advanced magic methods		

Magic methods

`__nonzero__(self)`
Defines behavior for when bool() is called on an instance of your class. Should return True or False, depending on whether you would want to consider the instance to be True or False.

Empirical study - a quick look

Idiom	# projects (out of 1000)	# of occurences
List comprehension	866	75,466
Generator expressions	709	33,038
Dict comprehension	146	796
Simple magic methods	759	78,376
Intermediate magic methods	417	13,255
Advanced magic methods	190	2,613

Empirical study - a quick look

Idiom	# projects (out of 1000)	# of occurences
List comprehension	866	75,466
Generator expressions	709	33,038
Dict comprehension	146	796
Simple magic methods	759	78,376
Intermediate magic methods	417	13,255
Advanced magic methods	190	2,613

→ More in-depth research is needed

→ Detecting anti-idioms is difficult

What we learned so far

- “Pythonic” is important - somehow

What we learned so far

- “Pythonic” is important - somehow
- “Pythonic” encompasses more than just programming idioms

What we learned so far

- “Pythonic” is important - somehow
- “Pythonic” encompasses more than just programming idioms
- Using “Pythonic idioms”…
  - makes you appear more knowledgeable
  - alone does **not** necessarily make your code better or more pythonic

What we learned so far

- “Pythonic” is important - somehow
- “Pythonic” encompasses more than just programming idioms
- Using “Pythonic idioms”…
  - makes you appear more knowledgeable
  - alone does **not** necessarily make your code better or more pythonic
- “Pythonic” does not always mean “more readable” for everyone

What we learned so far

- “Pythonic” is important - somehow
- “Pythonic” encompasses more than just programming idioms
- Using “Pythonic idioms”…
  - makes you appear more knowledgeable
  - alone does **not** necessarily make your code better or more pythonic
- “Pythonic” does not always mean “more readable” for everyone
- “Pythonic” is not learned systematically

Questions remain...

- Is there something special in the culture of Python?
  - The Zen of Python / “one way to do it”
  - Why are there no words like “Rubyist”, “Javanese” or “C#y”

Questions remain…

- Is there something special in the culture of Python?
  - The Zen of Python / “one way to do it”
  - Why are there no words like “Rubyist”, “Javanese” or “C#y”
- If “Pythonic” is not just syntax and idioms, what is it?

Questions remain…

- Is there something special in the culture of Python?
  - The Zen of Python / “one way to do it”
  - Why are there no words like “Rubyist”, “Javanese” or “C#y”
- If “Pythonic” is not just syntax and idioms, what is it?
- Does “Pythonic” correlate with code quality?

Questions remain…

- Is there something special in the culture of Python?
  - The Zen of Python / “one way to do it”
  - Why are there no words like “Rubyist”, “Javanese” or “C#y”
- If “Pythonic” is not just syntax and idioms, what is it?
- Does “Pythonic” correlate with code quality?
- Is “Pythonic” just a posh mark of pride serving to create a two-tier society within the Python community?

On the Usage of Pythonic Idioms

University of Zurich UZH
Universidad Rey Juan Carlos
Zürcher Hochschule für Angewandte Wissenschaften zhaw

Carol V. Alexandru[1], José J. Merchante[2], Sebastiano Panichella[1,3]
Sebastian Proksch[1], Harald C. Gall[1], Gregorio Robles[2]

Read the paper: http://t.uzh.ch/S7

Get the slides: http://t.uzh.ch/Sb

Browse the catalogue: http://pythonic.libresoft.info/catalogue