WRITING QUALITY CODE

IDEAS, TECHNIQUES AND TOOLS FOR IMPROVING THE QUALITY OF WRITTEN CODE

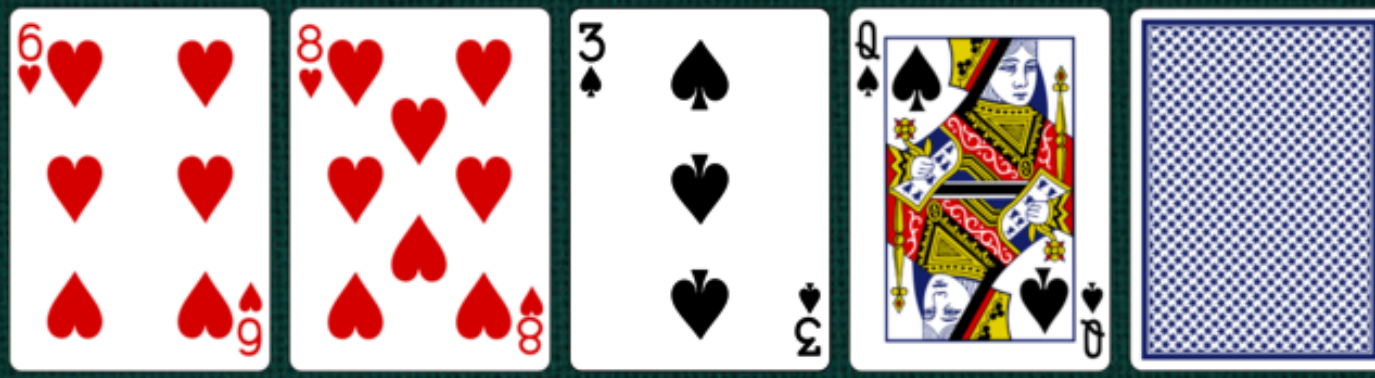
Radosław Jankiewicz / @radek_j / radekj

WHO AM I

- Programming in Python since 2007
- Web applications (mostly)
- STX NEXT

RANDOM FACT...

STX Hackathon v3.0

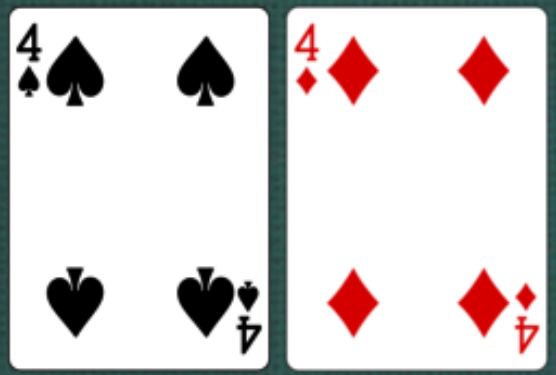


6

Player 2

422

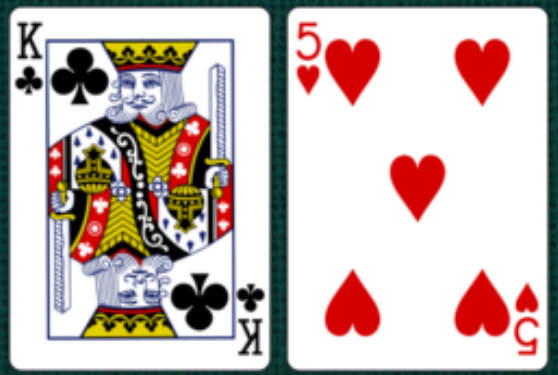
24



Player 1

597

22



Player 4

808

AGENDA

1. Definition of code quality
2. Why is it important
3. How to measure the quality of code
4. How to improve quality of written code
    - Good practices
    - Useful tools
    - Other hints
5. Q&A

HOW TO DEFINE HIGH QUALITY CODE?

CHARACTERISTICS OF SOFTWARE QUALITY

- External
- Internal

External characteristics of software quality

- Corectness
- Usability
- Efficiency
- Reliability
- Integrity
- Adaptability
- Accuracy
- Robustness

Internal characteristics software quality

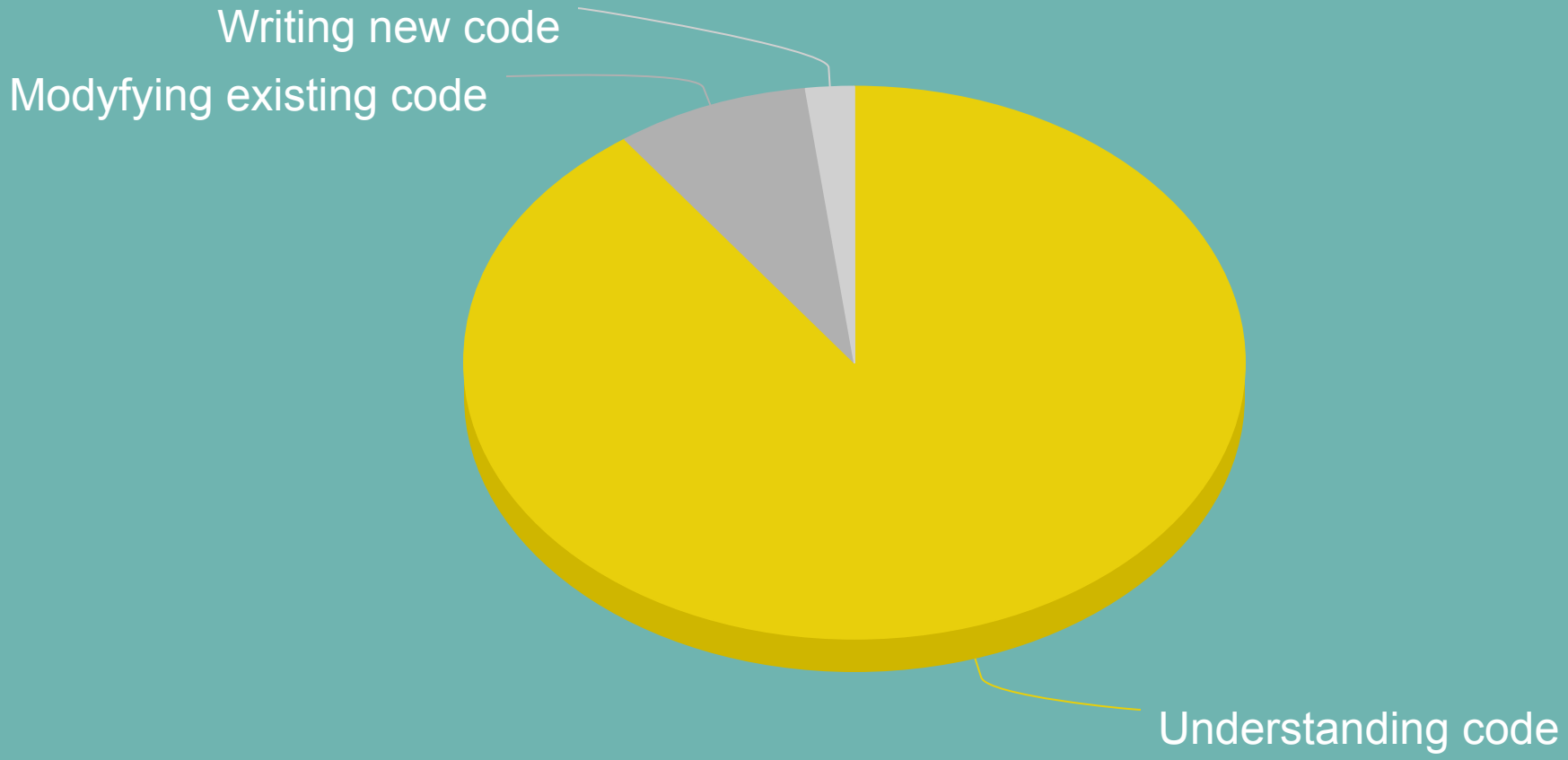
- Maintainability
- Flexibility
- Portability
- Reusability
- Readability
- Testability
- Understandability

GOOD CODE FROM THE DEVELOPER'S POINT OF VIEW:

UNDERSTANDABLE

UNDERSTANDABLE

Time spent by a programmer

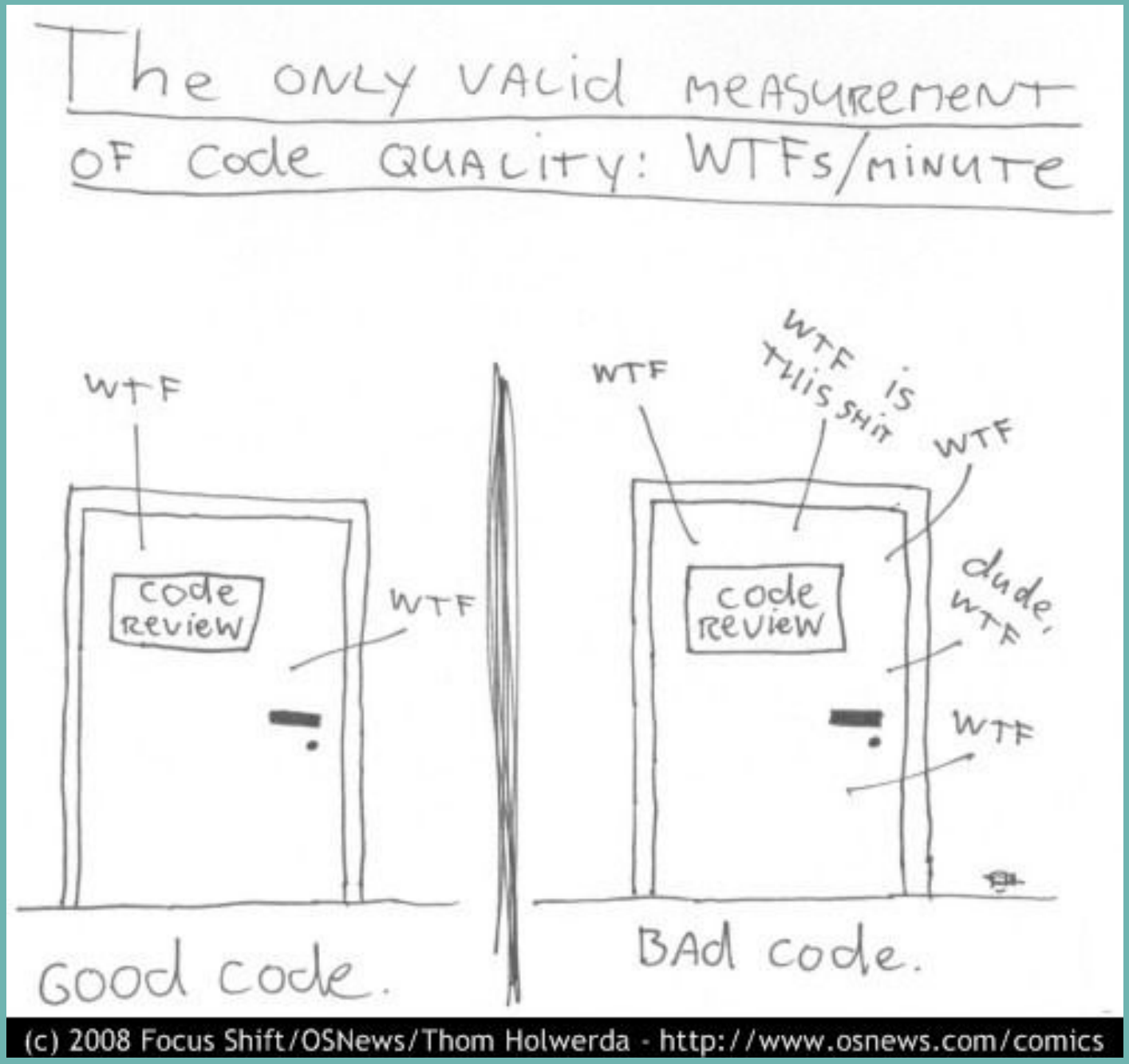


http://blog.codinghorror.com/when-understanding-means-rewriting

HOW IMPORTANT IS HIGH QUALITY OF CODE

POOR QUALITY CODE COSTS

HOW TO MEASURE THE QUALITY OF CODE


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

SOFTWARE QUALITY METRICS

- Cyclomatic Complexity
- Halstead complexity measures
- Maintainability Index

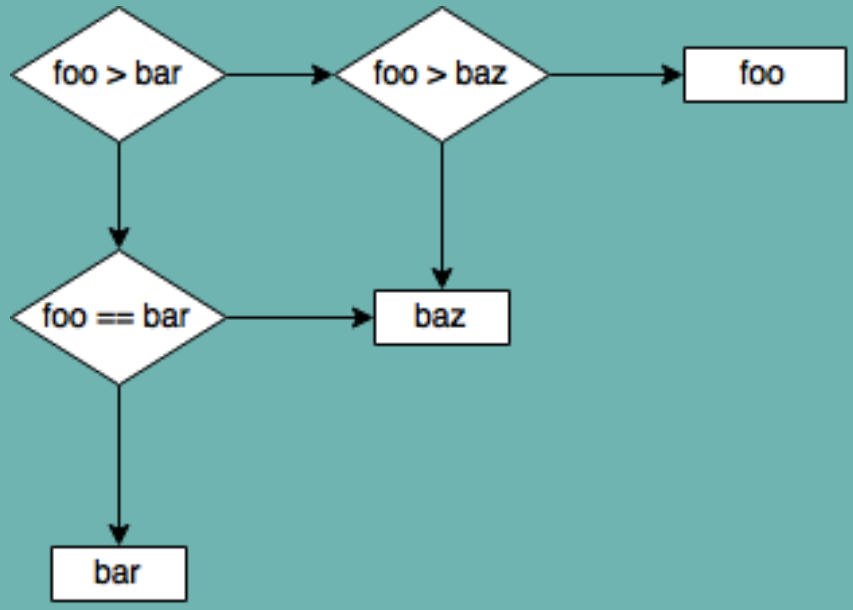
CYCLOMATIC COMPLEXITY

Construct	Effect on CC	Reasoning
if	+1	An *if* statement is a single decision.
else	+0	The *else* statement does not cause a new decision.
for	+1	There is a decision at the start of the loop.
Boolean Operator	+1	Every boolean operator (*and, or) adds a decision point.*

Full table: https://radon.readthedocs.org/en/latest/intro.html

CYCLOMATIC COMPLEXITY

```
def example(foo, bar, baz):
    if foo > bar:
        if foo > baz:
            return foo
        else:
            return baz
    elif foo == bar:
        return bar
    else:
        return baz
```



CC = 4

HALSTEAD METRICS

- η1 = the number of distinct operators
- η2 = the number of distinct operands
- N1 = the total number of operators
- N2 = the total number of operands

HALSTEAD METRICS

```
def example(foo, bar, baz):
    if foo > bar:
        if foo > baz:
            return foo
        else:
            return (baz / 3)
    elif foo == bar:
        return bar
    else:
        return baz
```

- η1 = 7 (example, if, else, elif, (,), >, ==, /, return)
- η2 = 4 (foo, bar, baz, 3)
- N1 = 16 (all operators)
- N2 = 14 (all operands)

HALSTEAD METRICS

- Program vocabulary: $\eta = \eta_1 + \eta_2$
- Program length: $N = N_1 + N_2$
- Calculated program length: $\widehat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$
- Volume: $V = N \log_2 \eta$
- Difficulty: $D = \frac{\eta_1}{2} \cdot \frac{N_2}{\eta_2}$
- Effort: $E = D \cdot V$
- Time required to program: $T = \frac{E}{18}$ seconds

MAINTAINABILITY INDEX

$$MI = 171 - 5.2 \ln V - 0.23G - 16.2 \ln L$$

- V is the Halstead Volume
- G is the total Cyclomatic Complexity
- L is the number of Source Lines of Code (SLOC)

RADON

CC number

```
$ radon cc ./url.py -s
./url.py
    M 287:4 URLMethodsMixin.resource_url - C (18)
    M 35:4 URLMethodsMixin._partial_application_url - C (17)
    M 85:4 URLMethodsMixin.route_url - C (16)
    C 31:0 URLMethodsMixin - B (7)
    M 539:4 URLMethodsMixin.static_url - A (5)
    F 753:0 static_url - A (3)
```

MI index

```
$ radon mi ./url*.py -s
./urldispatch.py - A (56.71)
./url.py - A (46.64)
```

RADON - CC RESULTS

CC score	Rank	Risk
1 - 5	A	low - simple block
6 - 10	B	low - well structured and stable block
11 - 20	C	moderate - slightly complex block
21 - 30	D	more than moderate - more complex block
31 - 40	E	high - complex block, alarming
41+	F	very high - error-prone, unstable block

RADON - MI RESULTS

MI score	Rank	Maintainability
100 - 20	A	Very high
19 - 10	B	Medium
9 - 0	C	Extremely low

WEB FRAMEWORKS - MI RESULTS

Rank	Pyramid (187 files)	Flask (61 files)	Django (836 files)
A	97.8%	100%	98.3%
B	1.6%	0%	0.3%
C	0.5%	0%	1.3%

PYLINT

- Static code analysis
- Coding Standard
- Error detection
- Refactoring help
- Fully customizable
- Editor/IDE integration

PYLINT

```
def example(foo, a, blah):
    qux = 123
    if foo > a:
        return foo
    else:
        return datetime.now()
```

```
************* Module a
C:  1, 0: Missing module docstring (missing-docstring)
C:  1, 0: Black listed name "foo" (blacklisted-name)
C:  1, 0: Invalid argument name "a" (invalid-name)
C:  1, 0: Missing function docstring (missing-docstring)
E:  6,15: Undefined variable 'datetime' (undefined-variable)
W:  1,20: Unused argument 'blah' (unused-argument)
W:  2, 4: Unused variable 'qux' (unused-variable)

Global evaluation
----------------
Your code has been rated at -8.33/10
```

MORE TOOLS

R. Ganczarek

Code Quality in Python - tools and reasons

Tomorrow, 16:45 (Barria 2 room)

HOW TO IMPROVE QUALITY OF WRITTEN CODE

PEER CODE REVIEW

- Decreases number of bugs
- Enforces writting neat code
- Speeds up learning
- Enhances the team culture

CODE REVIEW - USEFUL RULES

- All changesets get code reviewed
- Automate everything you can
- Everyone makes code reviews / everybody gets code reviewed

CODE REVIEW TOOLS

- Pull requests inline comments (Github / Bitbucket / ...)
- Gerrit
- Crucible
- Phabricator
- many more...

READABILITY COUNTS

CODING CONVENTION

Keep the code consistent with the project's convention.

Use automatic syntax/code style guide checking.

PEP-8 is a good option.

NAMING VARIABLES, CLASSES, METHODS...

“There are only two hard things in Computer Science: cache invalidation and naming things.”

Phil Karlton

Variable name for the maximum number of people in a car

At first - it should be descriptive

```
x = 5  # bad
data = 5  # bad
max = 5  # very bad
```

... but not too long ...

```
maximum_number_of_people_in_the_car = 123  # bad
```

abbreviations are acceptable

```
num_seats = 5  # not that bad
total_seats = 5  # good
max_passengers = 5  # good
```

Avoid double negative boolean logic

```
seat.is_not_occupied = True  # bad
```

```
seat.is_empty = True  # ok
```

DOCSTRINGS

- MUST be valid. Wrong docstring is worse than no docstring at all. Keep it up to date.
- Do not explain the implementation details
- Summarize the function's behaviour
- Document function's arguments, return value(s), side effects, exceptions raised, and restrictions on when it can be called (all if applicable)

COMMENTS

- MUST be valid. Wrong comment is worse than no comment at all
- Inline comments are unnecessary and in fact distracting if they state the obvious. Don't do this:

```
x = x + 1  # Increment x
```

KEEP YOUR TESTS CLEAN

“If you let the tests rot, then your code will rot too. Keep your tests clean.”

Rober C. Martin - "Clean Code"

SELF EFFORT

What else could you do to increase the quality of written code?

KNOW PYTHON IDIOMS

```
>>> x > 10 and x <= 20
```

More pythonic:

```
>>> 10 < x <= 20
```

1

KNOW PYTHON STANDARD LIBRARY

```
>>> colors = ['blue', 'red', 'green', 'red', 'blue', 'red']
```

```
>>> [(x, colors.count(x)) for x in set(colors)]
[('blue', 2), ('green', 1), ('red', 3)]
```

More pythonic:

```
>>> from collections import Counter
>>> Counter(colors)
Counter({'red': 3, 'blue': 2, 'green': 1})
```

KNOW PYTHON'S SYNTAX EXPRESSIONS

```
>>> from contextlib import contextmanager
>>> @contextmanager
... def tag(name):
...     print "<%s>" % name
...     yield
...     print "</%s>" % name
...
>>> with tag("h1"):
...    print "foo"
...
<h1>
foo
</h1>
```

SELF EFFORT

Read valuable books

Read documentation

Practice a lot

CHECKIO.ORG

Clear

Hide description

You are given a text, which contains different english letters and punctuation symbols. You should find the most frequent letter in the text. The letter returned must be in lower case.
While checking for the most wanted letter, casing does not matter, so for the purpose of your search, "A" == "a". Make sure you do not count punctuation symbols, digits and whitespaces, only letters.

If you have **two or more letters with the same frequency**, then return the letter which comes first in the latin alphabet. For example -- "one" contains "o", "n", "e" only once for each, thus we choose "e".

Input: A text for analysis as a string (unicode for py2.7).

Output: The most frequent letter in lower case as a string.

Example:

```
checkio("Hello World!") == "l"
checkio("How do you do?") == "o"
checkio("One") == "e"
checkio("Oops!") == "o"
checkio("AAaooo!!!!") == "a"
checkio("abe") == "a"
```

Back **First**

10 LVL

radekj

```python
import string
from collections import Counter


def checkio(text):
    delete_chars = string.punctuation + string.whitespace + string.digits
    trans = text.maketrans("", "", delete_chars)
    clean_text = text.translate(trans)
    counter = Counter(clean_text.lower())
    max_occurances = max(counter.values())
    most_common_letters = [k for k, v in counter.items() if v == max_occurances]
    return sorted(most_common_letters)[0]
```

Solutions for "The Most Wanted Letter"

HOME
200 46 %

Clear / Creative / Speedy / Uncategorized

Most voted / Newest / Most commented

1		20 LVL bryukh	max-count Python 3.3 Feb 14, 2014	68 +9 635	Open
2		19 LVL veky	Key Python 3.3 Jan 04, 2015	11 +7 49	Open
3		9 LVL ForeverYoung	First Python 3.3 Dec 18, 2013	6 +1 34	Open
4		11 LVL Happiness	First Python 2.7 New! Feb 13, 2015	1 +1 23	Open
5		11 LVL yama_k_1101	First Python 3.3 Jan 02, 2015	20	Open
6		10 LVL GoodOldBoy	First Python 3.3 Nov 09, 2014	17	Open
7		14 LVL panaro32	First Python 3.3 New! Apr 30, 2014	2 +2 15	Open
8		12 LVL walkingpendulum	First Python 2.7 New! Mar 25, 2015	15	Open
9		14 LVL yukirin	First Python 3.3 New! Mar 29, 2015	1 +1 13	Open
			Boy howdy		

- Story
- Solve it
- Discuss (98)
- Timeline
- Solutions
- Random
- Python 3.3
- Get next task

Author

20 LVL
bryukh

Show description

Back **max-count**

20 LVL

bryukh

```python
import string

def checkio(text):
    """
    We iterate through latyn alphabet and count each letter in the text.
    Then 'max' selects the most frequent letter.
    For the case when we have several equal letter,
    'max' selects the first from they.
    """
    text = text.lower()
    return max(string.ascii_lowercase, key=text.count)
```

QUESTIONS?

THANKS AND CREDITS

- reveal.js by Hakim El Hattab (MIT)
- Steve McConnell "Code complete 2"
- Robert Martin - "The clean coder"
- http://blog.codinghorror.com
- http://docs.python-guide.org
- https://radon.readthedocs.org
- http://www.pylint.org
- http://www.checkio.org
- STX Next
- My wife Karolina