

easy_automation.py

@yashaka



AUTOMICIAN

01.2017

Plan

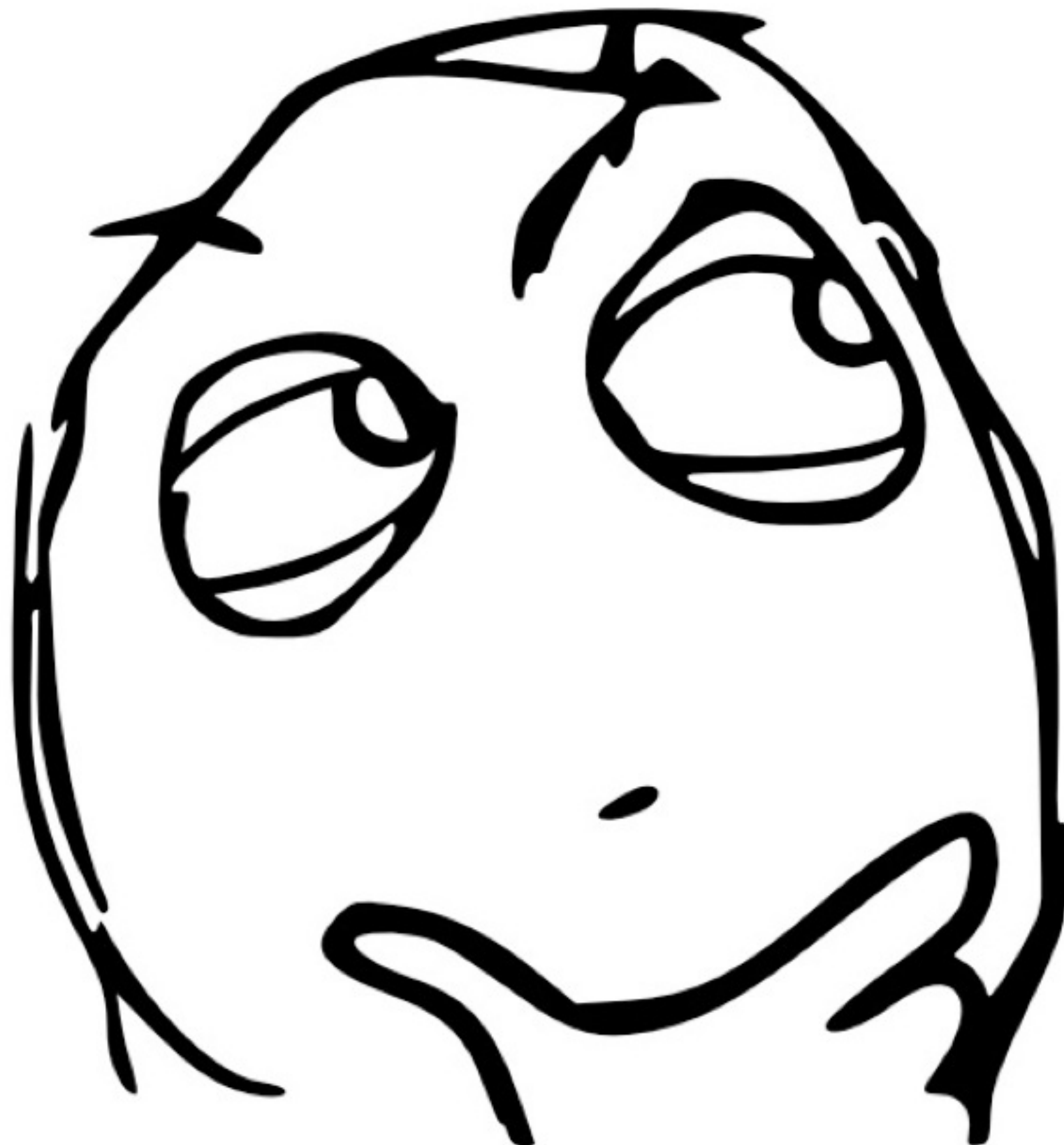
Intro

Live Coding Session

Easy Automation Tools Retrospective

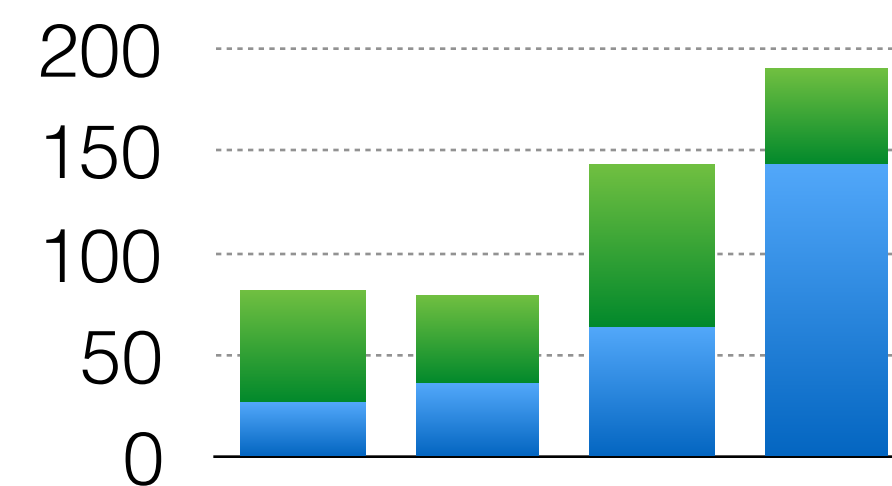
Q&A

Why Automation?



Testing
=
Assessing Quality

Effective Testing => **Assessing Quality**



Automation

=

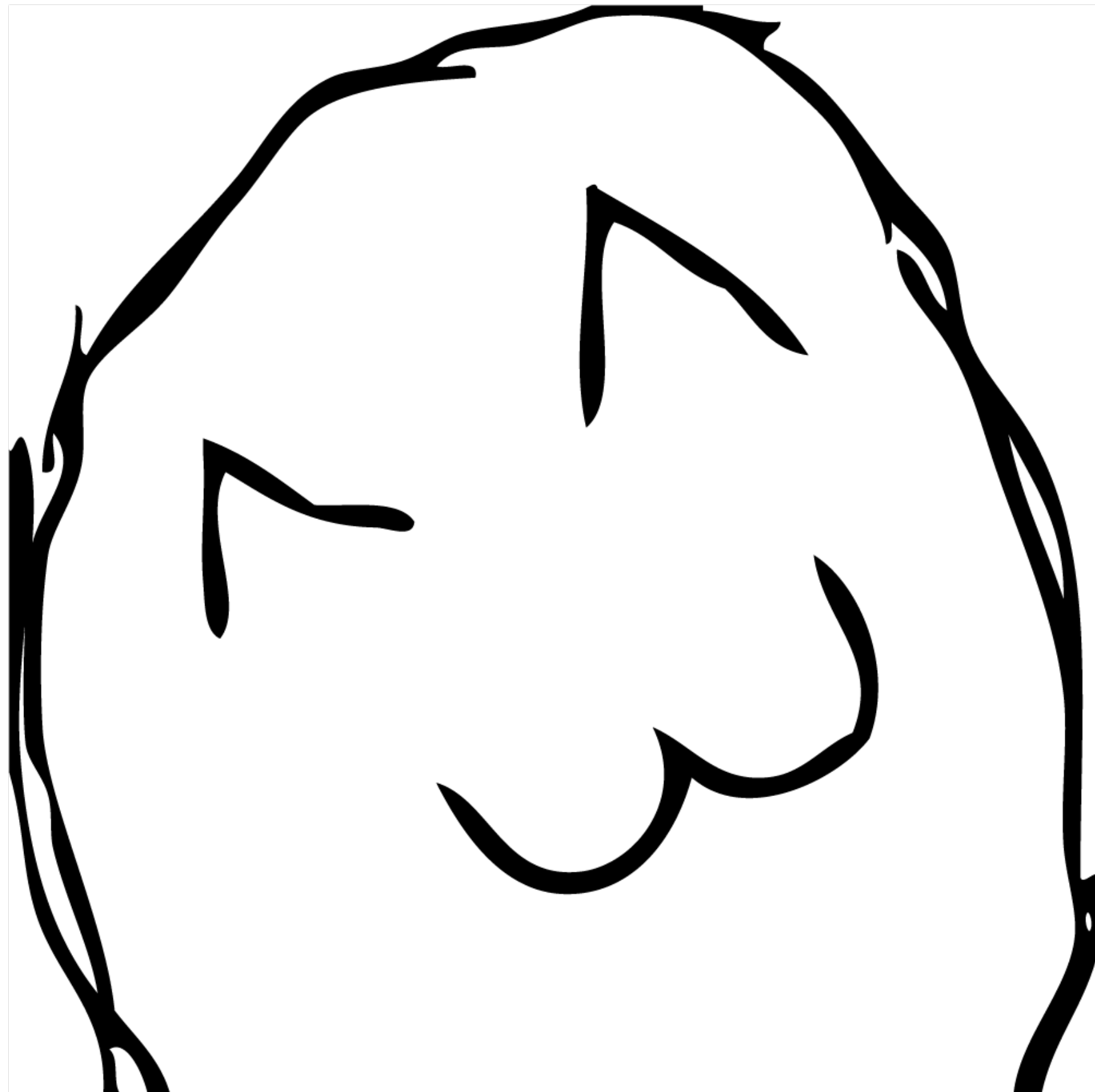
Automating manual routine work

Effective Testing

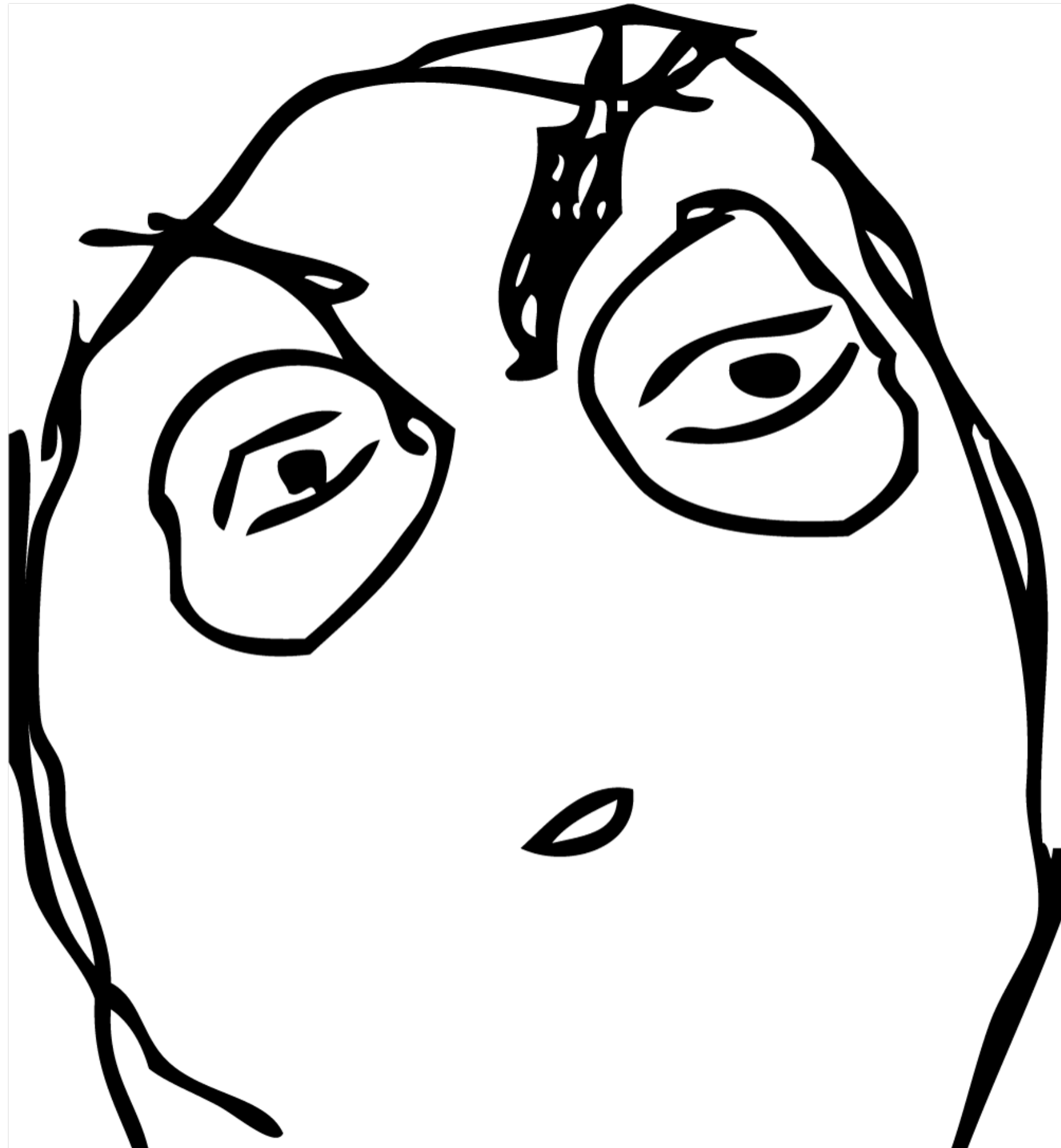
* =

Testing with automated routine
work

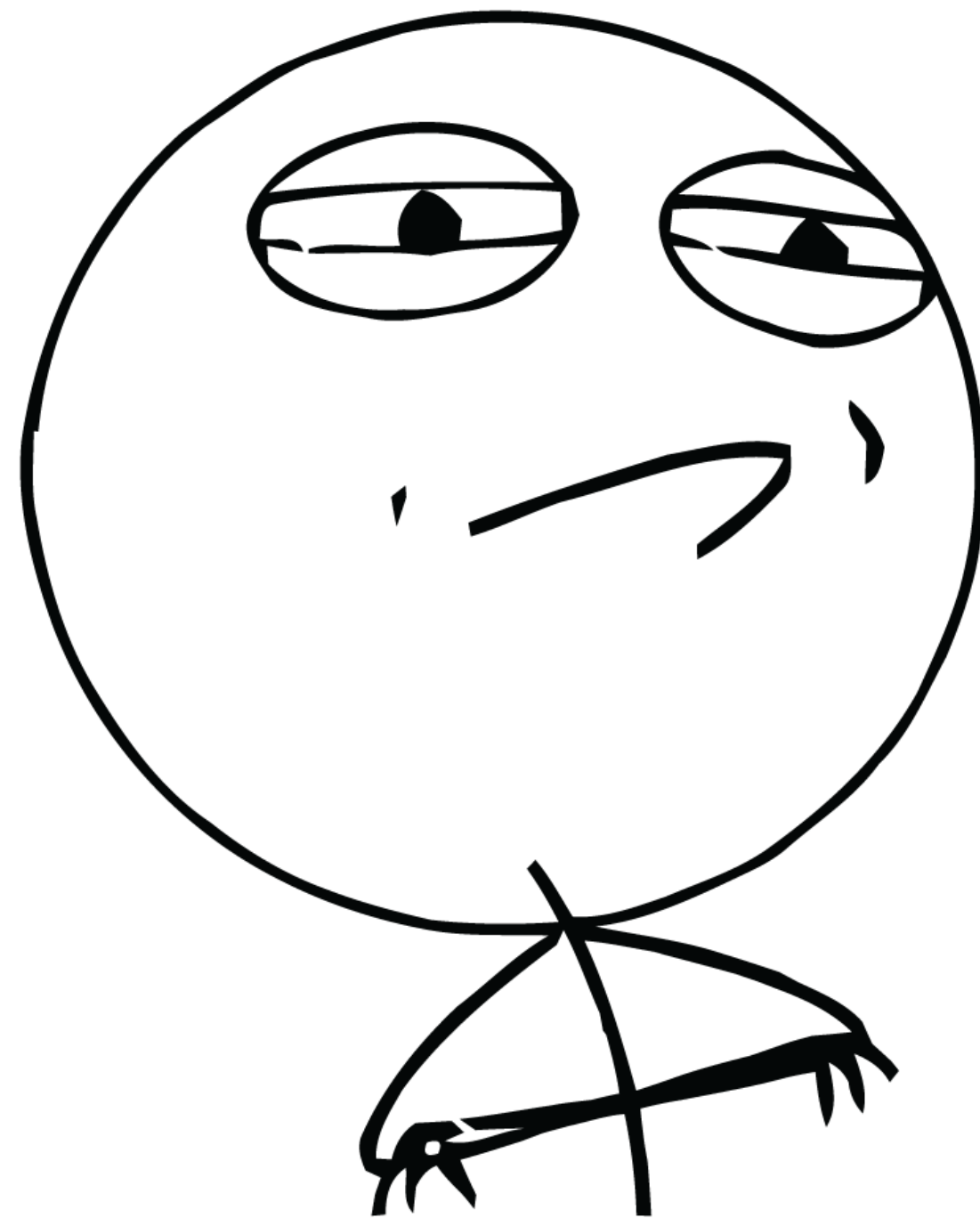
Automation!



But, it's hard, no?



no:p



Coding Session

<https://github.com/yashaka/talks/tree/master/easy-automation-py>

Summary

Coverage Styles

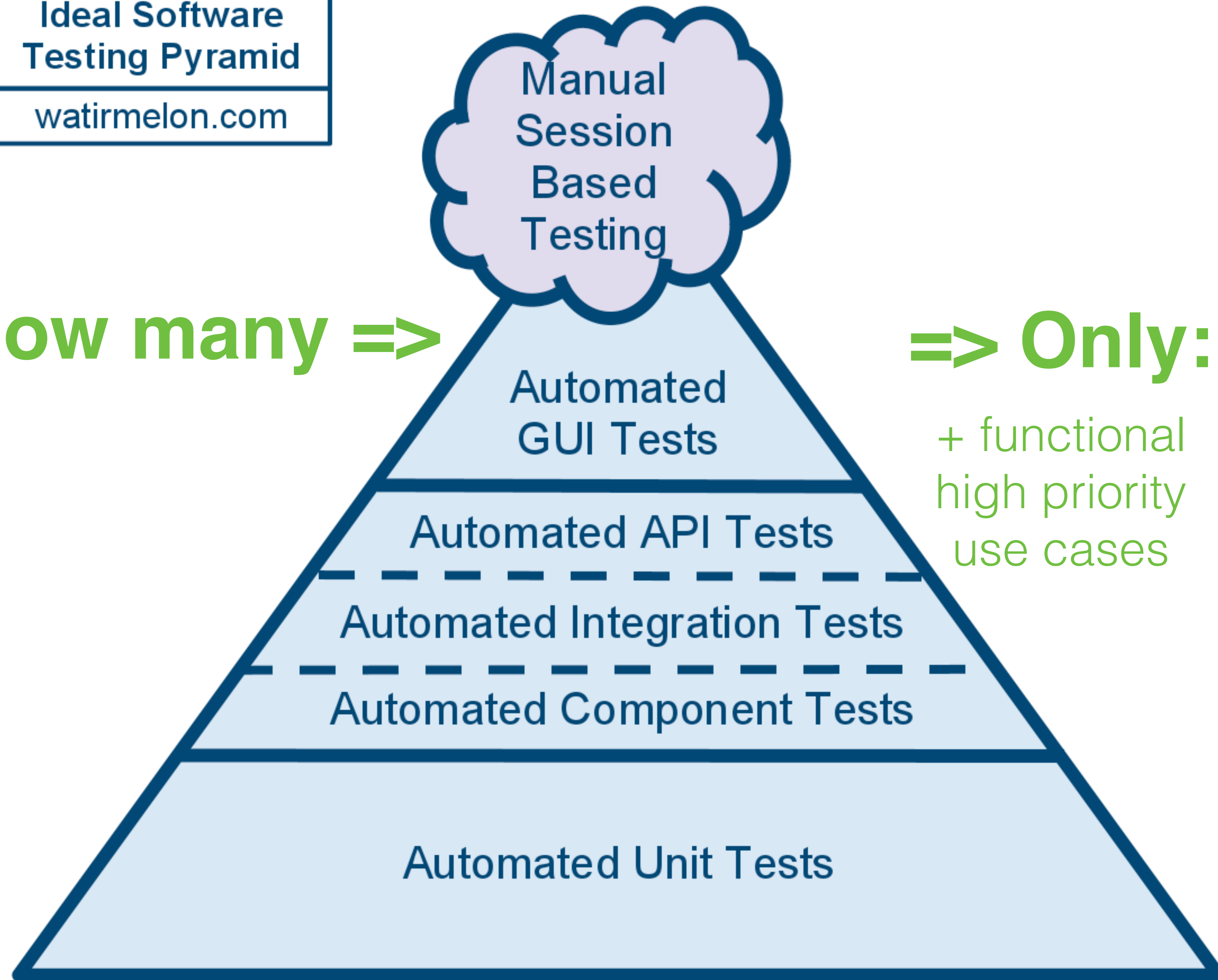
End to End style

- + more coverage in less time with less efforts during implementation
- + integration coverage

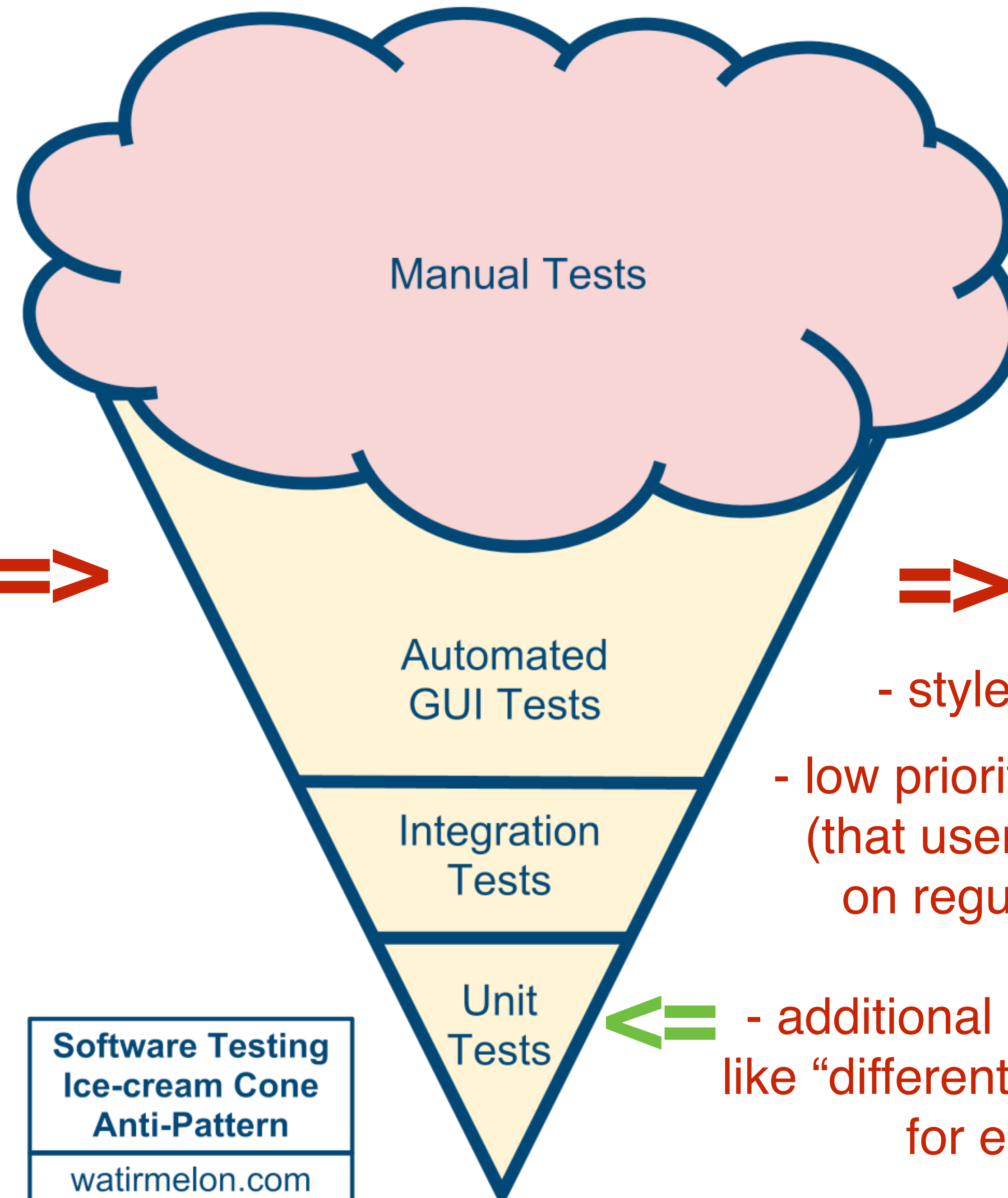
Unit/One-feature-per-test style

- + in case of bugs, gives more complete report
 - + easier to identify reason from the report
- =>
- + less time and efforts in support

How many =>



Avoid =>



=> No

- style checks
- low priority test cases (that user will not do on regular basis)
- additional requirements, like “different kind of names for entities”

Software Testing
Ice-cream Cone
Anti-Pattern

watirmelon.com

Implementing Automation

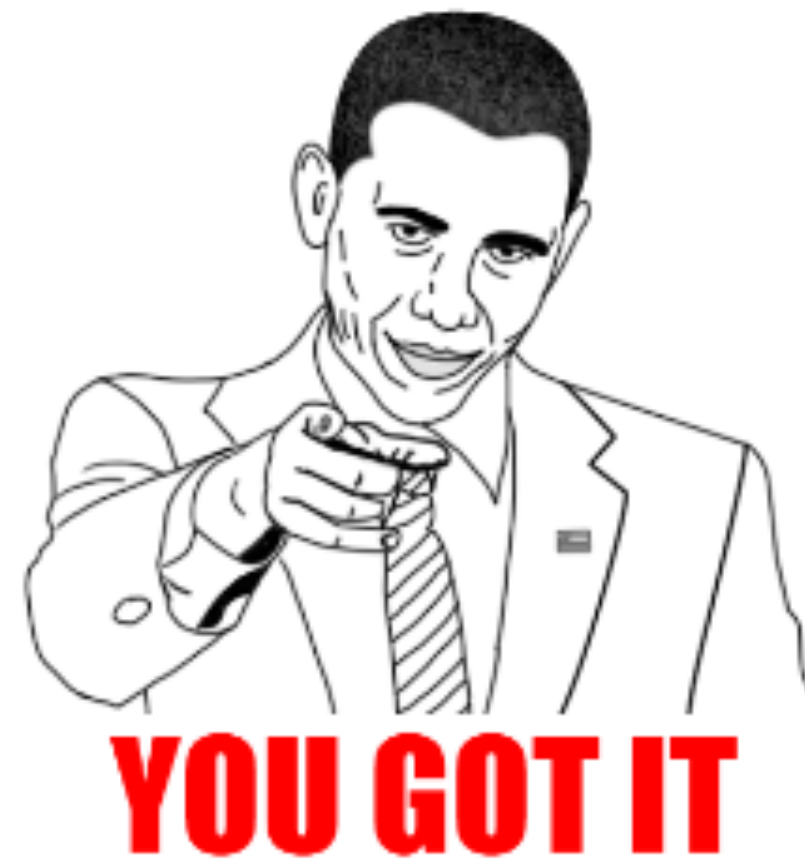
- Project in progress, no Automation?
 1. start with End To End style smoke integration tests
 2. review and test plan
 3. cover with “One feature”/“Unit” style independent tests

Implementing Automation

- Project just started and Auto-POC is approved OR ready features are automated?
 1. Test plan new features
 2. Cover with “One feature”/“Unit” style independent tests
 3. Add End to End integration smoke tests

Wasn't it Easy? :)

Of course, “Automation is Easy” is a semi-truth
but the devil is not so black as he is painted;)



How to start?

- Choose language
- Learn language (interactive tutorials, koans, exercism.io/checkio.org, learnxinyminutes.com, books, etc.)
- Choose **Easy Tools**
- Find a mentor (friend, dev on your project, chat, etc...)
- Go :)

Choose language?

- Have project?
 - choose language of project's developers
- Have no project but want to find work fast?
 - choose one of the most popular language
- Have no project but want to code in your style, and it does not matter how long will you seek for the job?
 - choose language that fits you

Learn language?

Google :p

interactive tutorials + koans + exercism.io +
docs.seleniumhq.org + google.com
=
“You can do this in any language.”

Easy Tools?

- Just a few examples...
 - Python: Selene
 - C#: NSelene
 - Java: Selenide
 - Ruby: Capybara


```
tasks = ss("#todo-list>li")
```

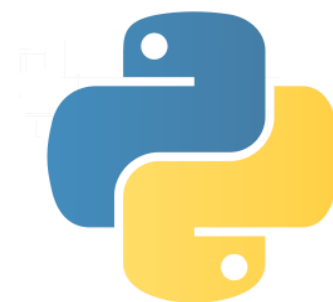
```
...
```

```
def toggle(task_text):
```

```
    tasks.element_by(have.exact_text(task_text)).element(".toggle").click()
```

```
def should_be(*task_texts):
```

```
    tasks.filtered_by(be.visible).should(have.exact_texts(*task_texts))
```



```
public class Tasks {  
    public static ElementsCollection tasks = $$("#todo-list>li");  
  
    ...  
  
    public static void toggle(String taskText) {  
        tasks.findBy(exactText(taskText)).find(".toggle").click();  
    }  
  
    public static void shouldBe(String... taskTexts) {  
        tasks.filterBy(visible).shouldHave(exactTexts(taskTexts));  
    }  
}
```



```
public static class Tasks
{
    public static SCollection List = SS("#todo-list>li");

    ...

    public static void Toggle (string taskText)
    {
        List.FindBy(Have.ExactText(taskText)).Find(".toggle").Click();
    }

    public static void ShouldBe(params string[] names)
    {
        List.FilterBy(Be.Visible).Should(Have.Texts(names));
    }
}
```



```
module Tasks
  extend Capybara::DSL

  def self.tasks
    all "#todo-list>li"
  end

  ...

  def self.toggle task_text

    (tasks.find_by {|task| task.text == task_text}).find(".toggle").click
  end

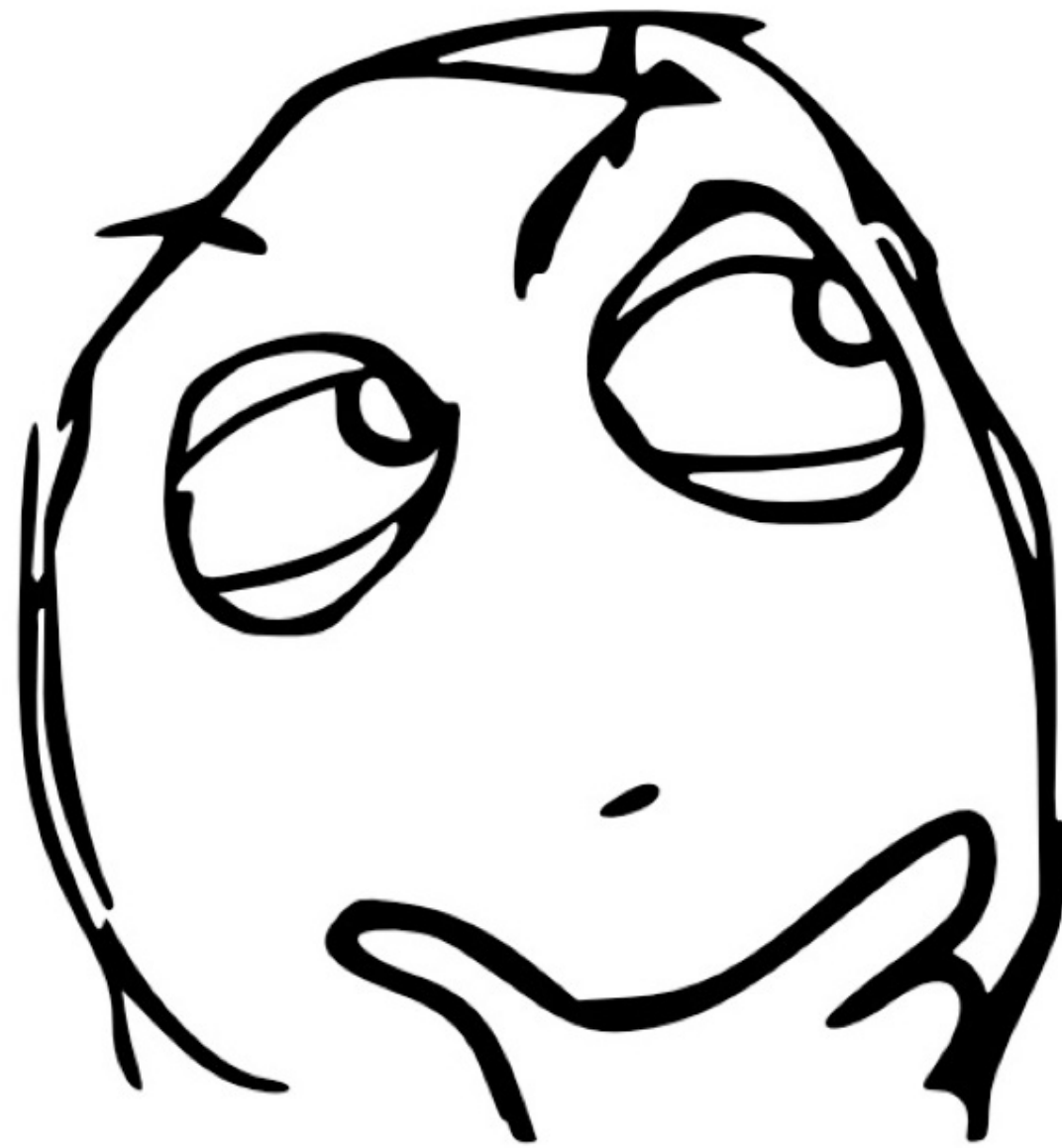
  def self.should_be *task_texts

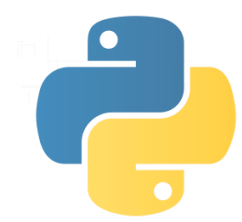
    tasks.texts.should == task_texts
  end
end
```



Easy Tool =

?

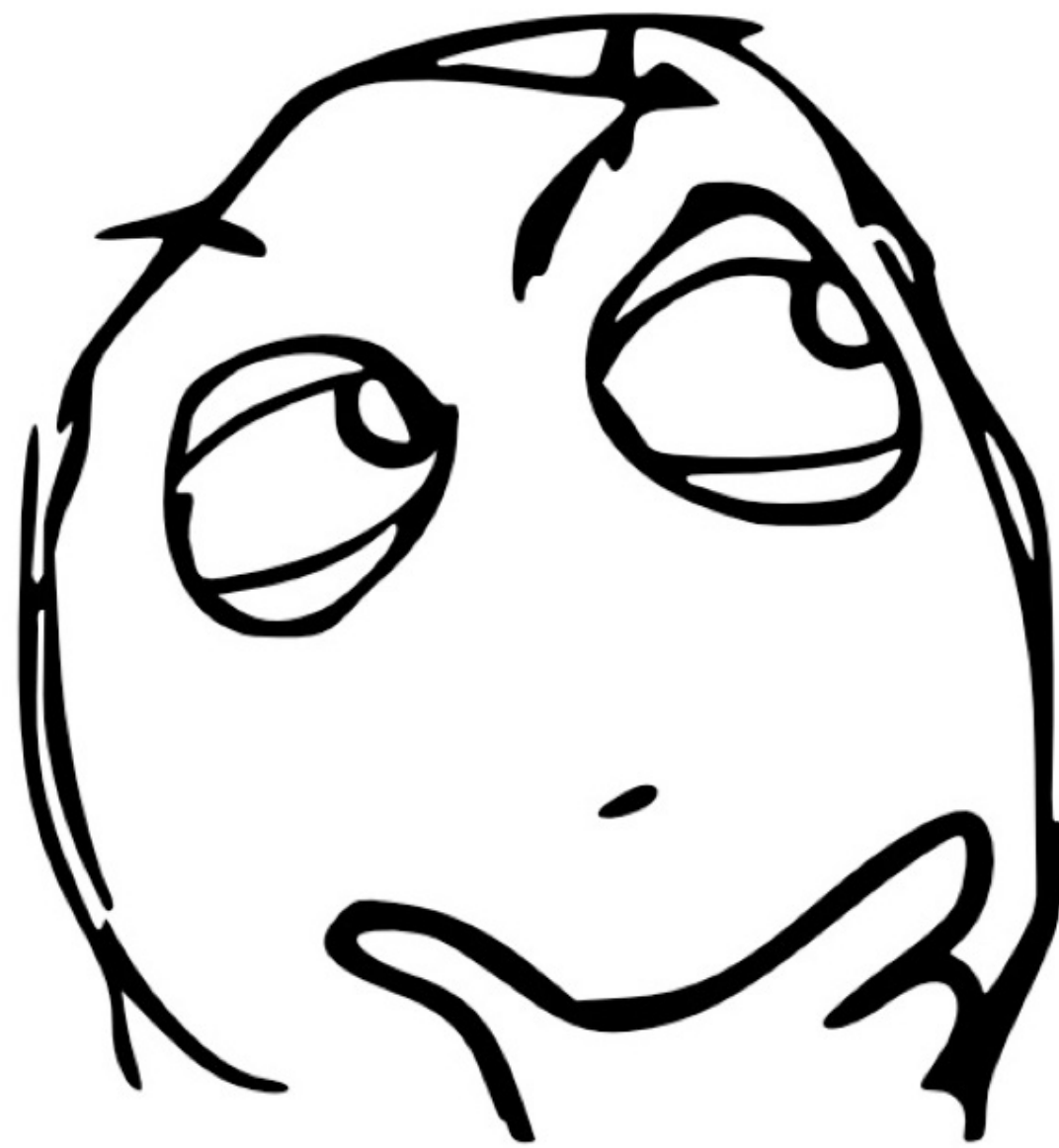


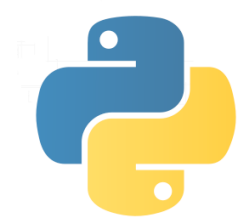


Selene

=

?



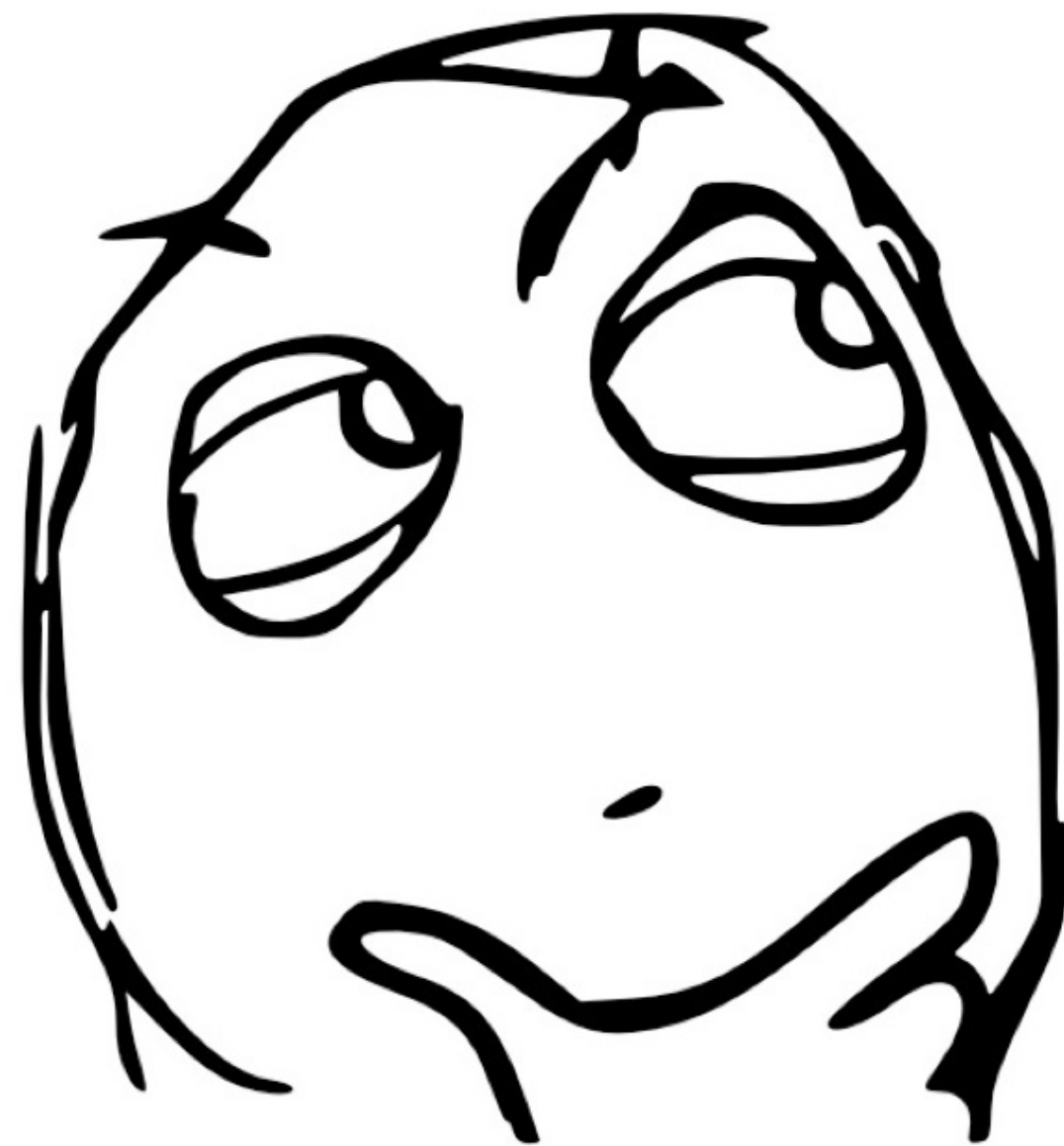


Selene

=

...
web automation tool

...

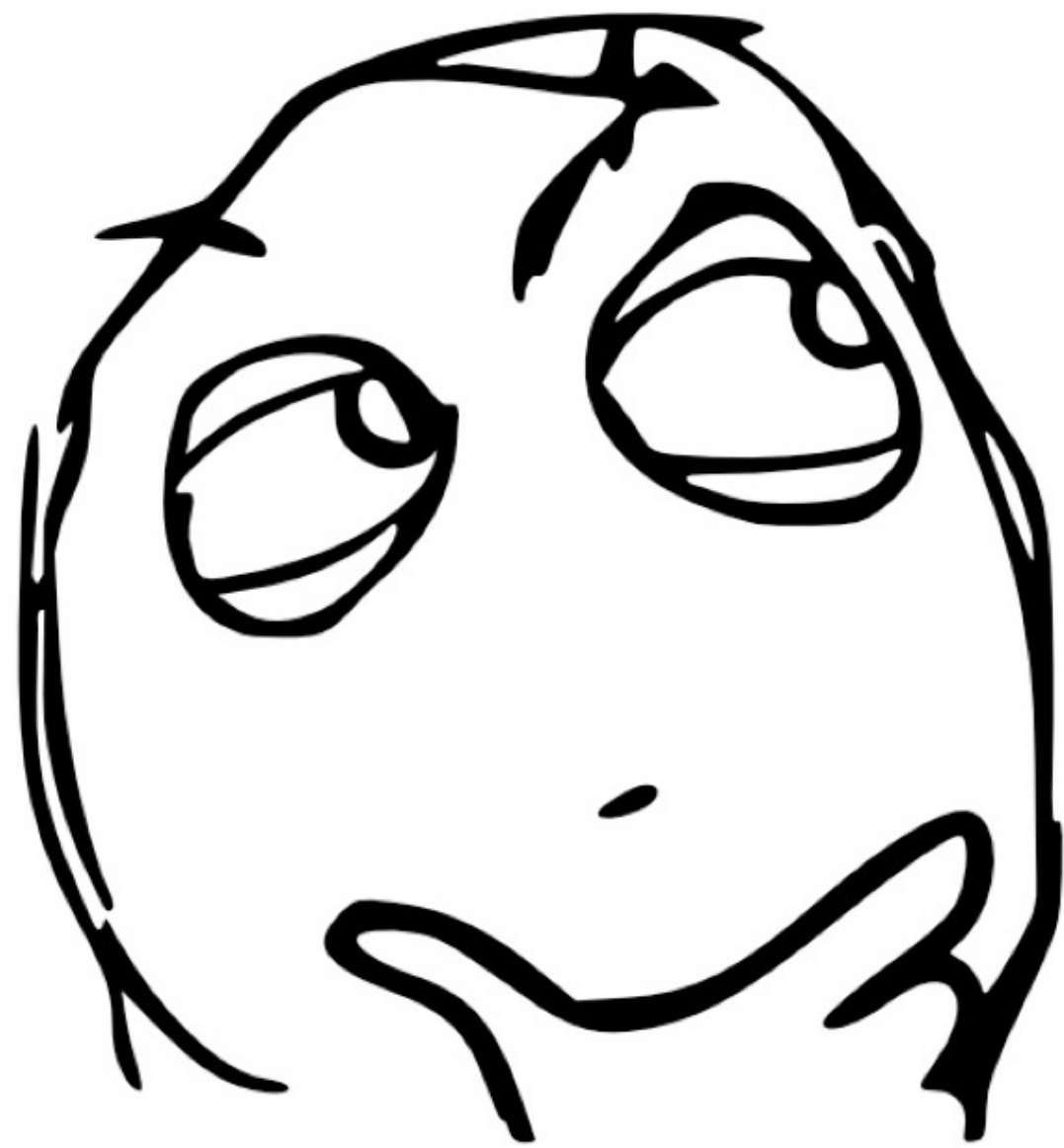




Selene =

...
web automation tool

selenium wrapper



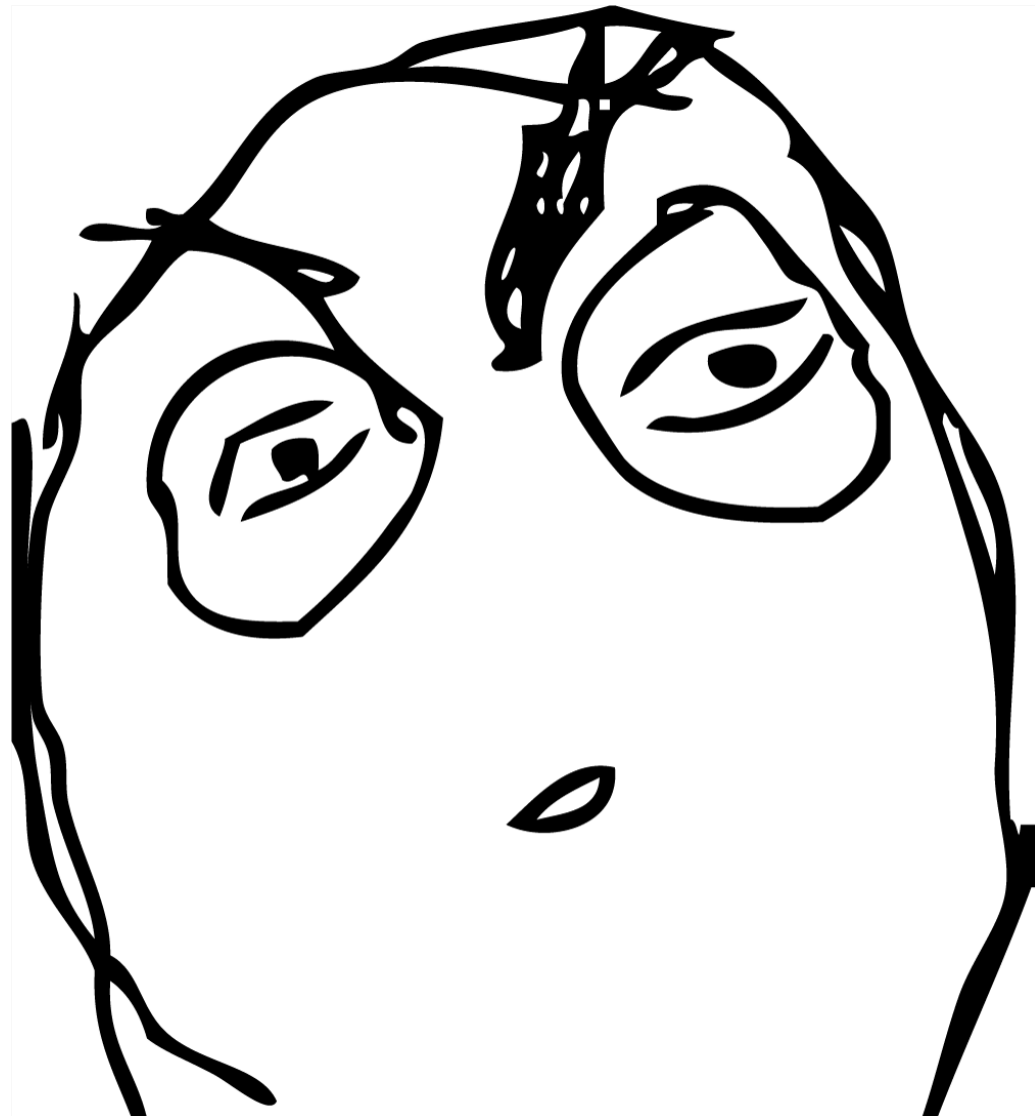


Selene =

...

~~web automation tool~~

~~selenium wrapper~~

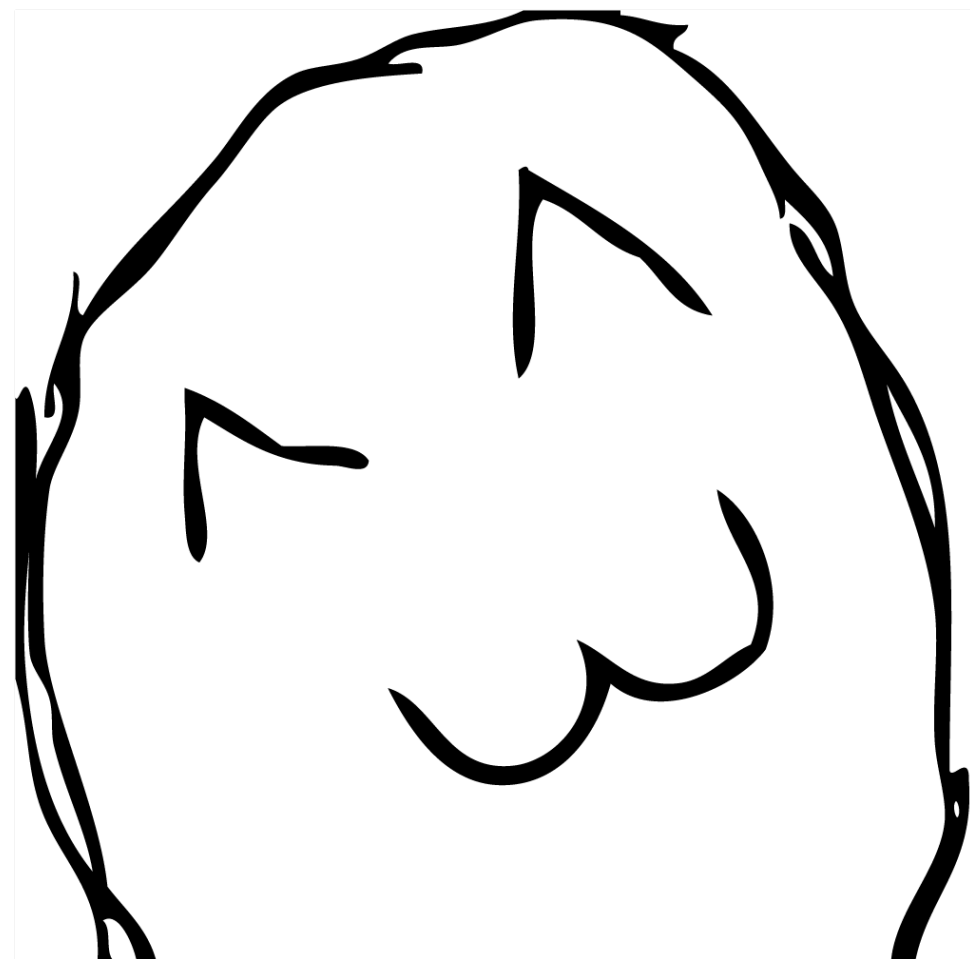


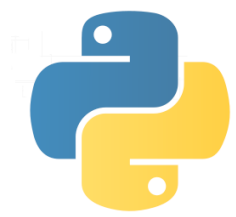


Selene

=

Effective
web **test** automation tool



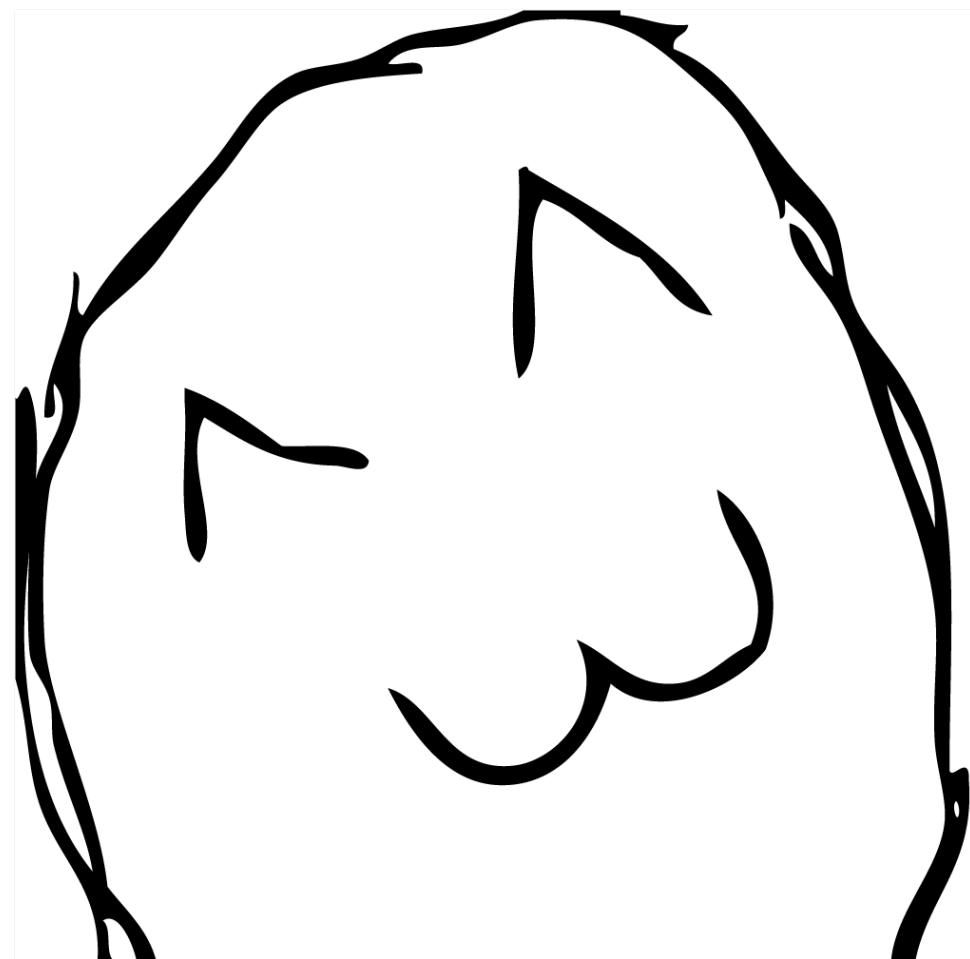


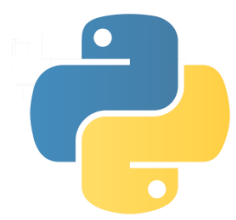
Selene

=

Effective
web **test** automation tool

being also
selenium wrapper

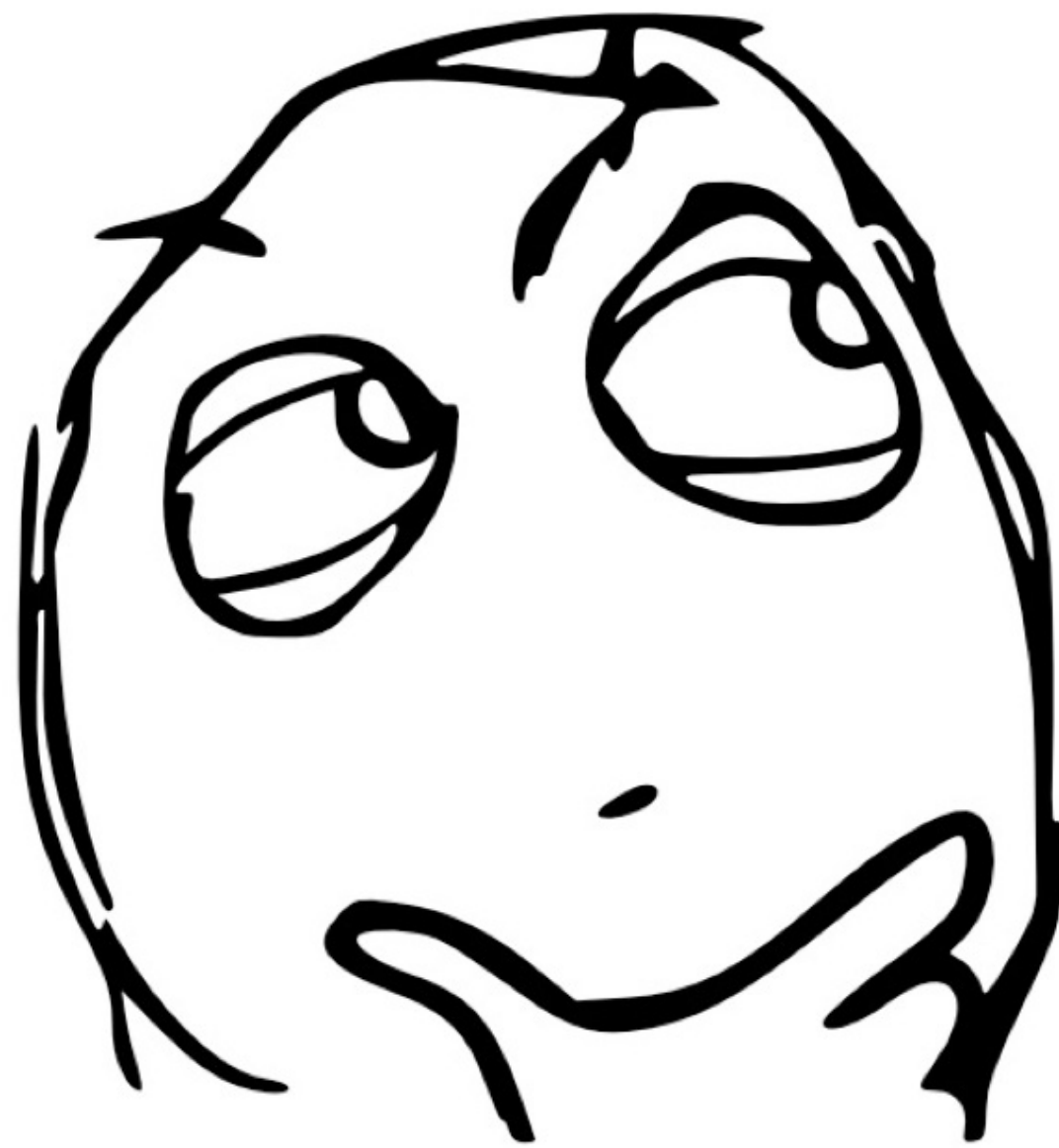




Selene

=

Effective
web test automation tool



=

?



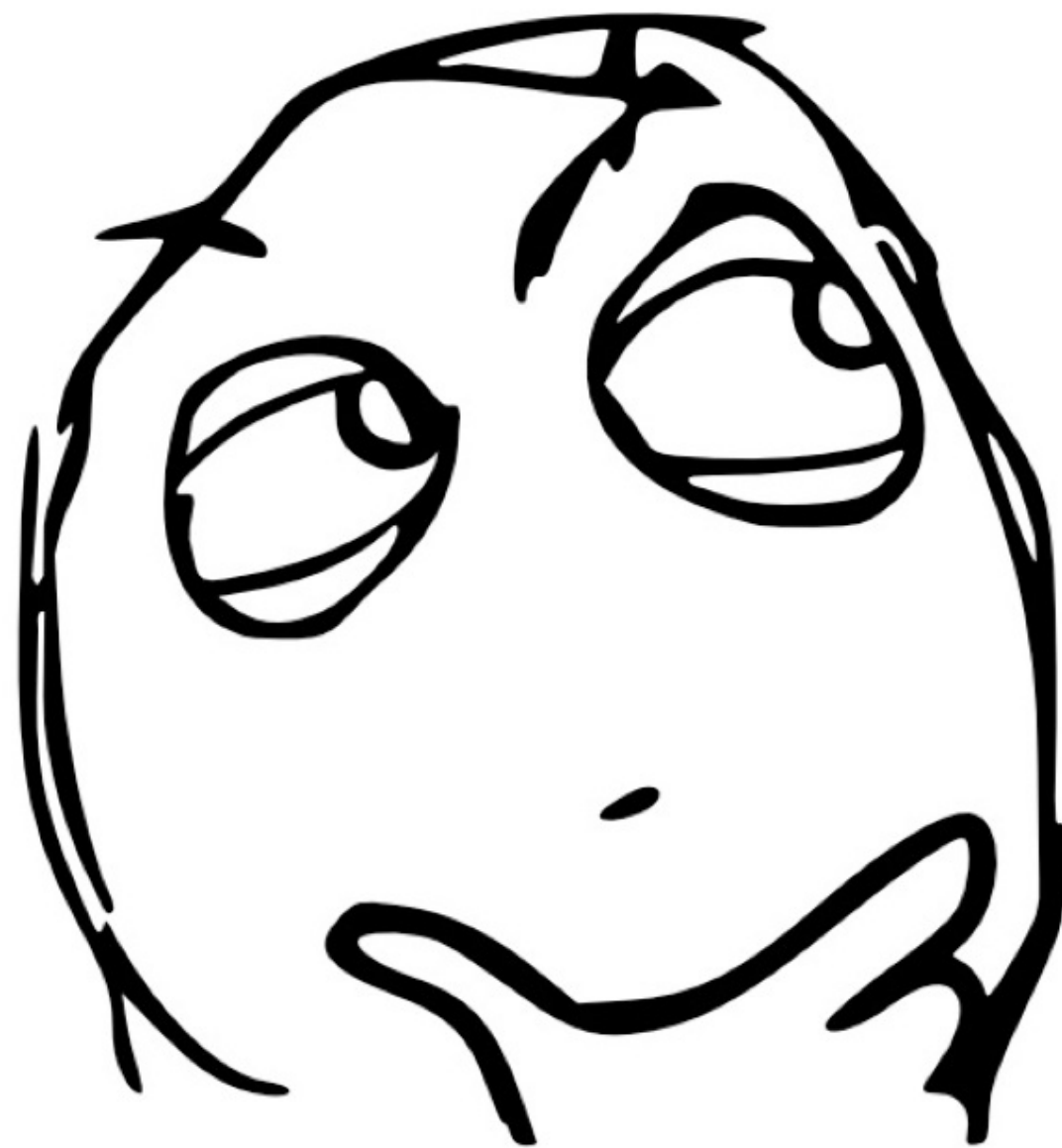
Selene

=

Effective
web test automation tool

=

tool to automate
web **UI tests logic**



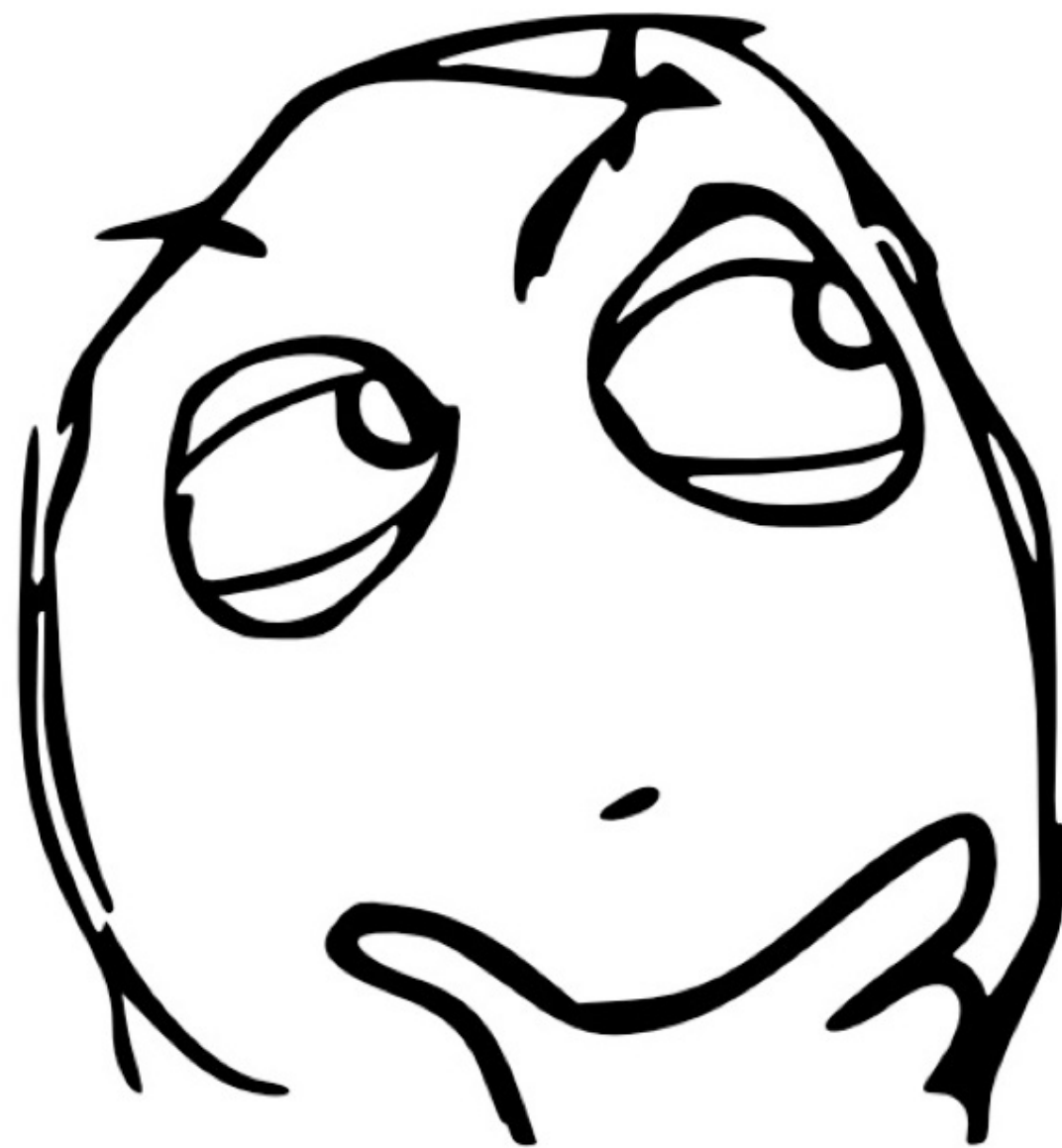


Selene

=

Effective
web test automation tool

=



tool to automate
web **UI tests logic**
not browser

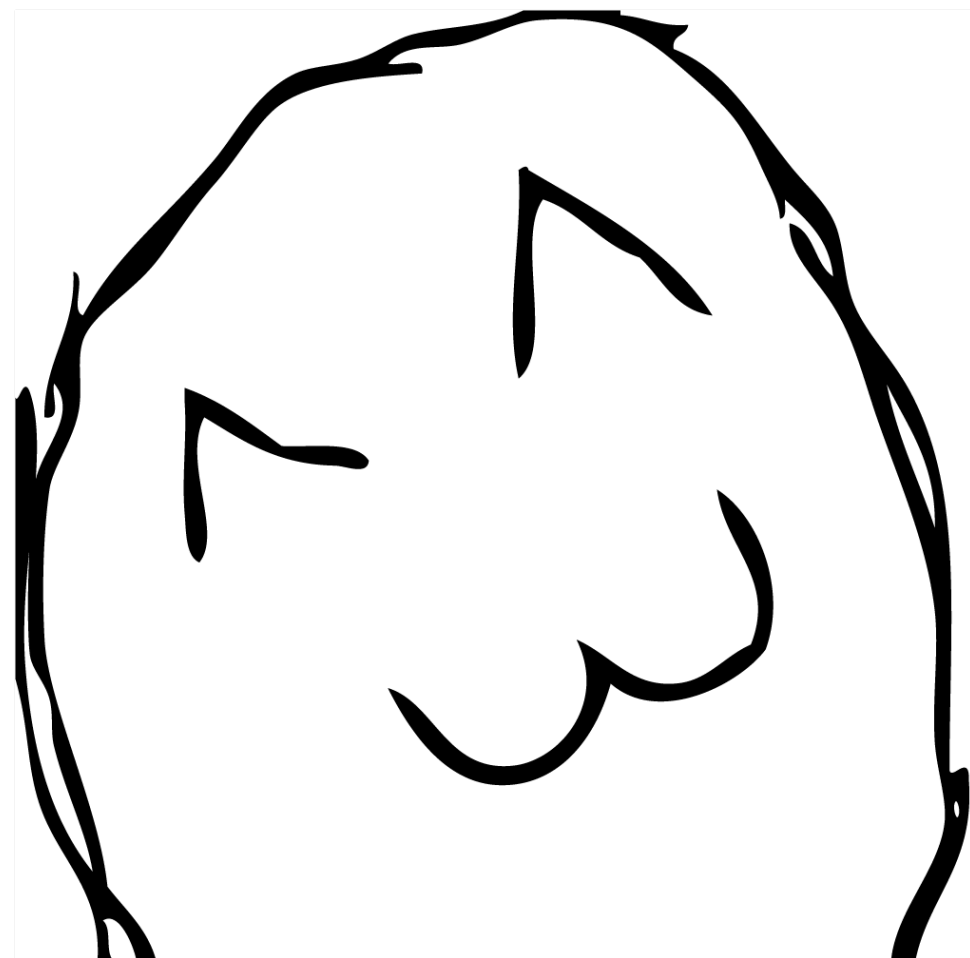


Selene

=

Effective
web test automation tool

=



tool to automate
web **UI tests logic**

not browser

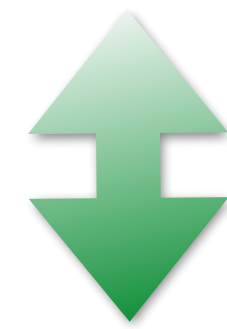
(it should be already automated;)



Selene

=

Effective
web test automation tool



concise API

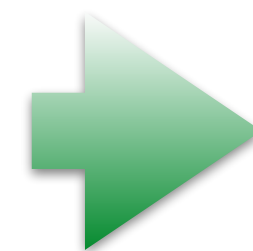
=

...

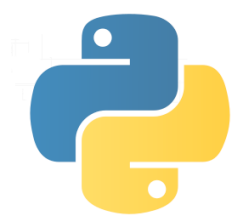
...

...

...



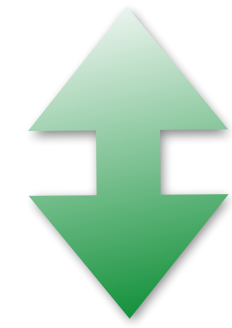
tool to automate
web **UI tests logic**
not browser



Selene

=

Effective
web test automation tool



concise API

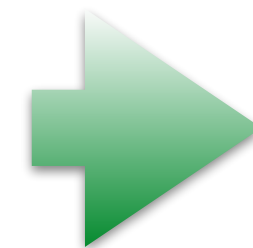
=

waiting search

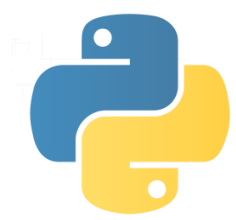
...

...

...



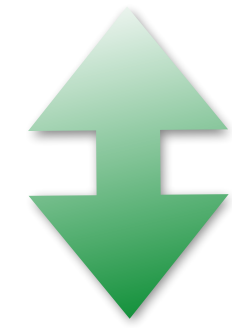
tool to automate
web **UI tests logic**
not browser



Selene

=

Effective
web test automation tool



concise API

waiting search

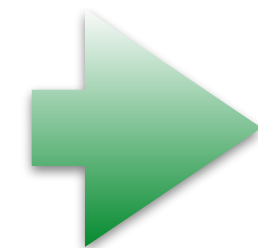
waiting asserts

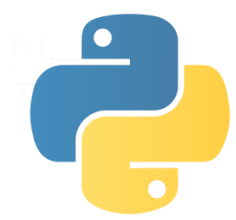
...

...

=

tool to automate
web **UI tests logic**
not browser

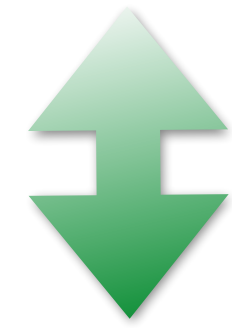




Selene

=

Effective
web test automation tool



concise API

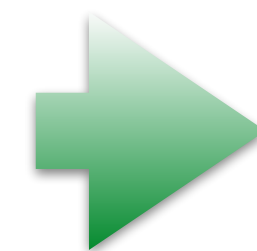
waiting search

waiting asserts

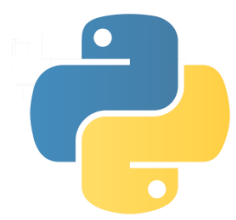
dynamic elements

...

=



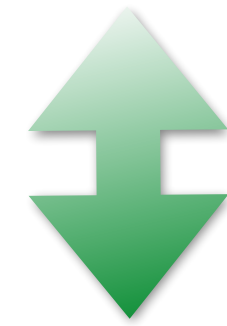
tool to automate
web **UI tests logic**
not browser



Selene

=

Effective
web test automation tool



concise API

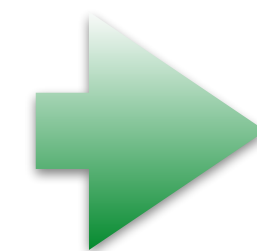
waiting search

waiting asserts

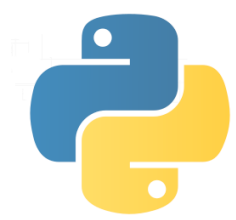
dynamic elements

informative errors

=



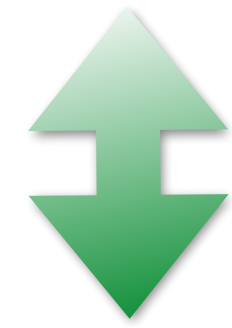
tool to automate
web **UI tests logic**
not browser



Selene

=

Effective
web test automation tool



concise API

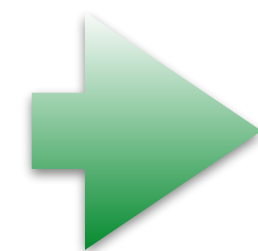
waiting search

waiting asserts

dynamic elements

informative errors


=



tool to automate
web **UI tests logic**
not browser

“UI Tests Logic” Automation with **Selene**

todos

	<i>What needs to be done?</i>
<input type="checkbox"/>	a
<input checked="" type="checkbox"/>	b
<input type="checkbox"/>	c

2 items left

All Active Completed

Clear completed

Double-click to edit a todo

UI Tests Logic

```
class TestTodoMVC(object):  
    def test_filter_active_tasks(self):  
  
        # visit page  
  
        # add "a"  
        # add "b"  
        # add "c"  
        # tasks should be "a", "b", "c"  
  
        # toggle "b"  
  
        # filter active  
        # tasks should be "a", "c"
```


UI Tests Logic

```
# visit page
```

```
# add "a"
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
# add "a"
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Concise API

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Automatic Driver Management

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Concise API

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c" search element "short-cut"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Concise API

```
visit("https://todomvc4tasj.herokuapp.com/")

s("#new-todo").set("a").press_enter()
# add "b"      ↑
# add "c"      ↑ default conversion to "by css" locator
# tasks should be "a", "b", "c"

# toggle "b"

# filter active
# tasks should be "a", "c"
```

UI Tests Logic

Concise API

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

↑ *with implicit clear()*

UI Tests Logic

Concise API

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

↑ *chainable methods*

UI Tests Logic

Dynamic Elements

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

search actually starts here

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Waiting Search

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
# add "b"
```

```
# add "c"
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

with implicit waiting for visibility

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
s("#new-todo").set("b").press_enter()
```

```
s("#new-todo").set("c").press_enter()
```

```
# tasks should be "a", "b", "c"
```

```
# toggle "b"
```

```
# filter active
```

```
# tasks should be "a", "c"
```

UI Tests Logic

Waiting Asserts

```
visit("https://todomvc4tasj.herokuapp.com/")

s("#new-todo").set("a").press_enter()
s("#new-todo").set("b").press_enter()
s("#new-todo").set("c").press_enter()
ss("#todo-list li").should(have.exact_texts("a", "b", "c"))

# toggle "b"
# filter active
# tasks should be "a", "c"
```

↑
aka "explicit waits"

UI Tests Logic


Waiting Asserts

```
visit("https://todomvc4tasj.herokuapp.com/")

s("#new-todo").set("a").press_enter()
s("#new-todo").set("b").press_enter()
s("#new-todo").set("c").press_enter()
ss("#todo-list li").should(have.texts("a", "b", "c"))

# toggle "b"

# filter active
# tasks should be "a", "c"
```



handy conditions

UI Tests Logic

Informative errors

```
visit("https://todomvc4tasj.herokuapp.com/")

s("#new-todo").set("a").press_enter()
s("#new-todo").set("b").press_enter()
s("#new-todo").set("c").press_enter()
ss("#todo-list li").should(have.texts("a.", "b.", "c."))
```

```
E TimeoutException: Message:
E     failed while waiting 4 seconds
E     to assert ExactTexts
E     for all_by('css selector', '#todo-list li')
E
E     reason: ConditionMismatchException: condition did not match
E     expected: ('a.', 'b.', 'c.')
E     actual: ['a', 'b', 'c']
E     screenshot: /Users/ayia/.selene/screenshots/1484695102265/screen_1484695102266.png
```

UI Tests Logic

Concise API & Waiting Search

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
s("#new-todo").set("b").press_enter()
```

```
s("#new-todo").set("c").press_enter()
```

```
ss("#todo-list li").should(have.texts("a", "b", "c"))
```

```
ss("#todo-list li").element_by(have.text("b")).element(".toggle").click()
```

```
# filter active
```

```
# tasks should be "a", "c"
```

*laconic inner collection
search by text*

inner element search

instead of bulky xpath locators

UI Tests Logic

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
s("#new-todo").set("b").press_enter()
```

```
s("#new-todo").set("c").press_enter()
```

```
ss("#todo-list li").should(have.texts("a", "b", "c"))
```

```
ss("#todo-list li").element_by(have.text("b")).element(".toggle").click()
```

```
s(by.link_text("Active")).click()
```

```
# tasks↑ should be "a", "c"
```

custom locators

UI Tests Logic

Concise API & Waiting Asserts

```
visit("https://todomvc4tasj.herokuapp.com/")
```

```
s("#new-todo").set("a").press_enter()
```

```
s("#new-todo").set("b").press_enter()
```

```
s("#new-todo").set("c").press_enter()
```

```
ss("#todo-list li").should(have.texts("a", "b", "c"))
```

```
ss("#todo-list li").element_by(have.text("b")).element(".toggle").click()
```

```
s(by.link_text("Active")).click()
```

```
tasks.filtered_by(be.visible).should(have.texts("a", "c"))
```



filtering collection

UI Tests Logic

```
visit("https://todomvc4tasj.herokuapp.com/")

s("#new-todo").set("a").press_enter()
s("#new-todo").set("b").press_enter()
s("#new-todo").set("c").press_enter()
ss("#todo-list li").should_have(texts("a", "b", "c"))

ss("#todo-list li").element_by(have.text("b")).element(".toggle").click()

s(by.link_text("Active")).click()
tasks.filtered_by(be.visible).should(have.texts("a", "c"))
```

Page steps for even
more readable code?

```
#tasks.py
```

```
def visit():  
    tools.visit("https://todomvc4tasj.herokuapp.com/")  
  
def add(*task_texts):  
    for text in task_texts:  
        s("#new-todo").set(text).press_enter()  
  
def filter_active():  
    s(by.link_text("Active")).click()  
  
def filter_completed():  
    s(by.link_text("Completed")).click()
```

```
#tasks.py
```

```
tasks = ss("#todo-list>li")
```

```
...
```

```
def toggle(task_text):  
    tasks.element_by(have.text(task_text)).element(".toggle").click()
```

```
def should_be(*task_texts):  
    tasks.filtered_by(be.visible).should(have.texts(*task_texts))
```

Dynamic Elements

```
tasks = ss("#todo-list>li")
```

```
...  ↑
```

possible because search does not start here

*in fact, **ss** creates “lazy elements proxy” aka “elements lazy finder”;*)

```
class TestTodoMVC(object):  
  
    def test_filter_tasks(self):  
  
        tasks.visit()  
  
        tasks.add("a", "b", "c")  
        tasks.should_be("a", "b", "c")  
  
        tasks.toggle("b")  
  
        tasks.filter_active()  
        tasks.should_be("a", "c")  
  
        tasks.filter_completed()  
        tasks.should_be("b")
```


Customisation

```
config.browser_name = "chrome"  
...
```

or

```
config.browser_name = Browser.CHROME  
...
```

will use "firefox" by default

```
config.app_host = "http://mydomain.com"  
...  
visit("/subpage")
```

Default Timeouts Behaviour

```
s("#new-todo").should(be.enabled)
```

will wait until 4 seconds

Custom

```
s("#new-todo").should(be.enabled, timeout=10)
```

or

```
config.timeout = 10  
...  
s("#new-todo").should(be.enabled)
```

No Tool for *your* language?

No Tool for your language?

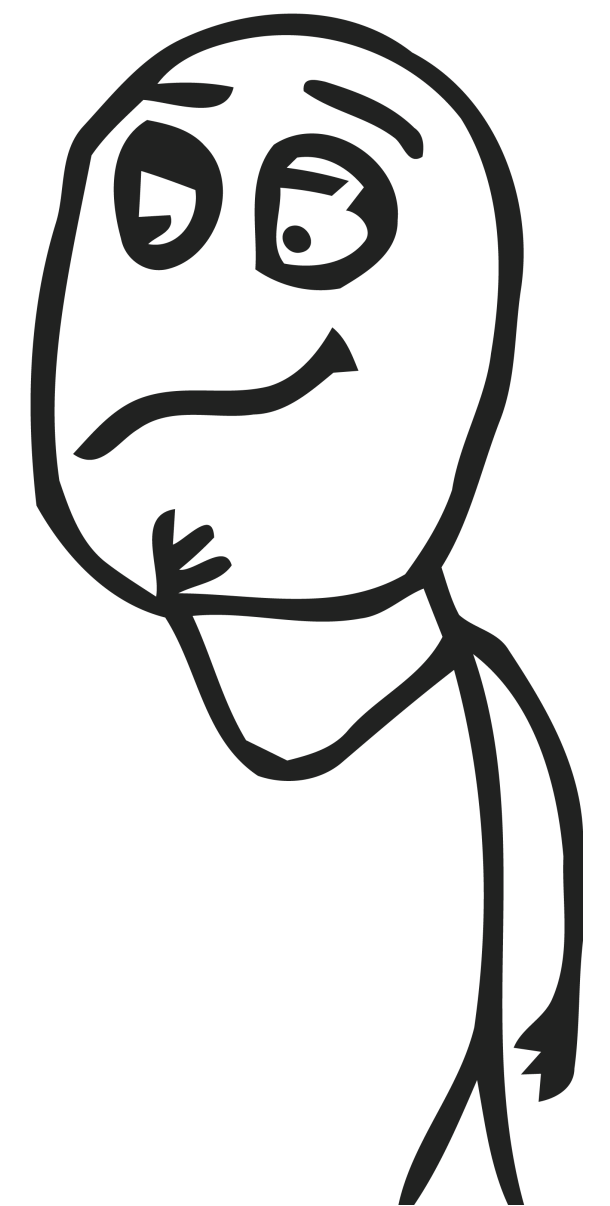
- Have project?
 - ask project's developers to write it for you
- Have no project and lack of knowledge?
 - switch to language that has easy tools :)
- Brave?
 - implement it by your own ;)



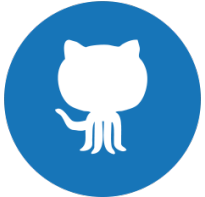


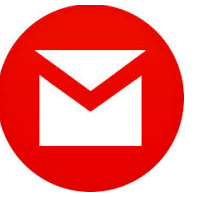

Remember...

- Automation **is NOT** a separate Role
- Automation **IS A TOOL** for Test Engineer to do his work effectively

Q&A



Thank you!

yashaka @     

github.com/automician

automician.com

seleniumcourses.com



AUTOMICIAN

@yashaka

01.2017