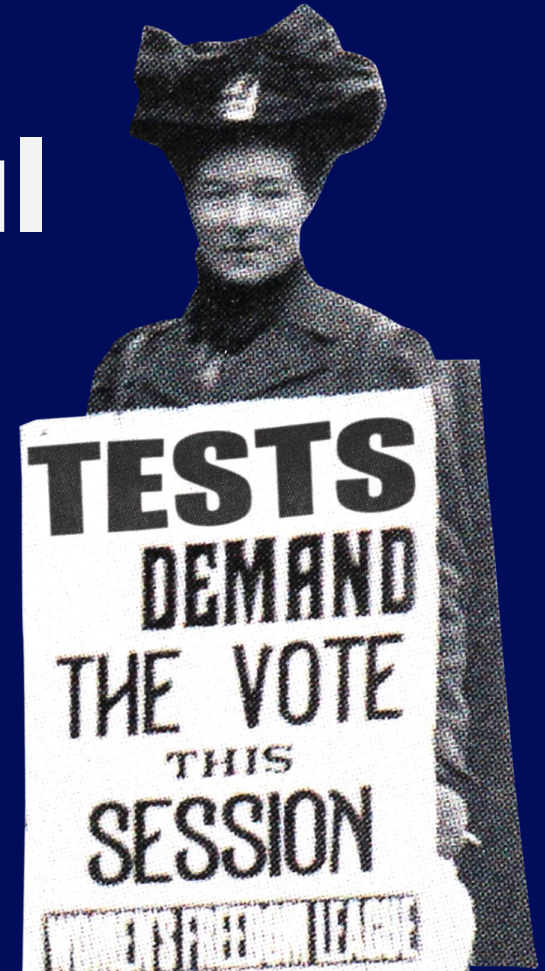


# Testing as an Equal 1st class citizen (to coding)

*ACCU 6th April 2022*



# Jon Jagger

Director of Software

[jon@merkely.com](mailto:jon@merkely.com)

merkely





**equal rights  
for tests!**

**VOTES  
FOR  
TESTS**

**WOMEN'S FREEDOM LEAGUE**

**TESTS  
THE VOTE  
THIS  
SESSION**

**WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.**

**TESTS  
THE VOTE  
THIS  
SESSION**

**WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.**

**LEGISLATION  
TYRANNY**

**WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.**



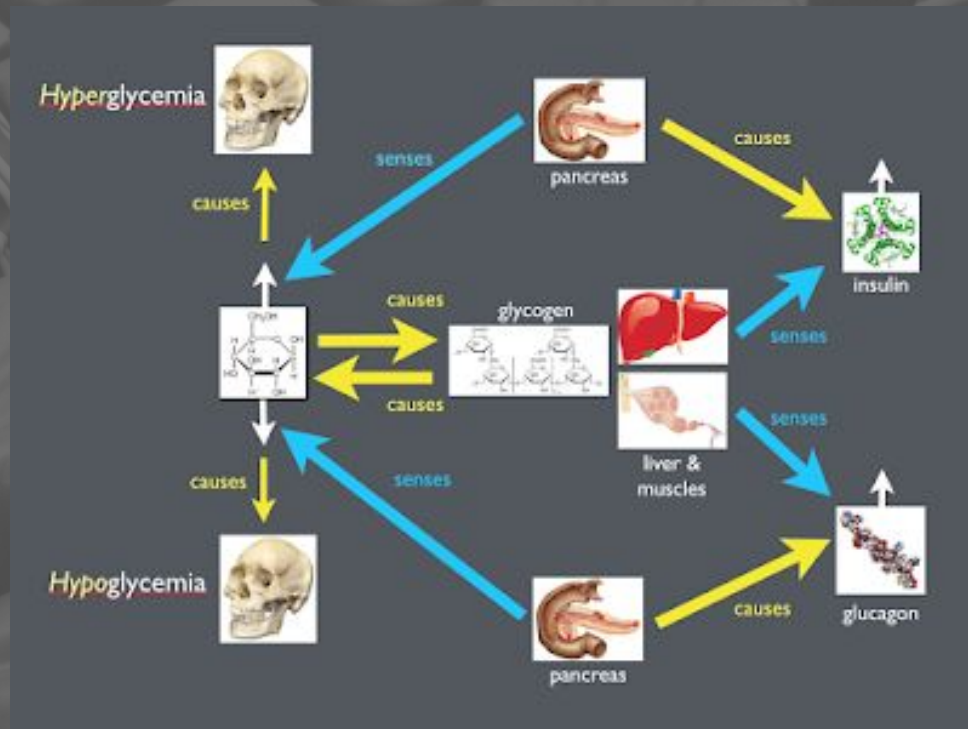
**never believe any  
assertion containing  
the words never or  
always**

# blood glucose



The value of your blood glucose can go up as well as down.  
Your health is at risk if you do not keep up a balanced diet.

# The Glucose Cycle



# The Equilibrium Law

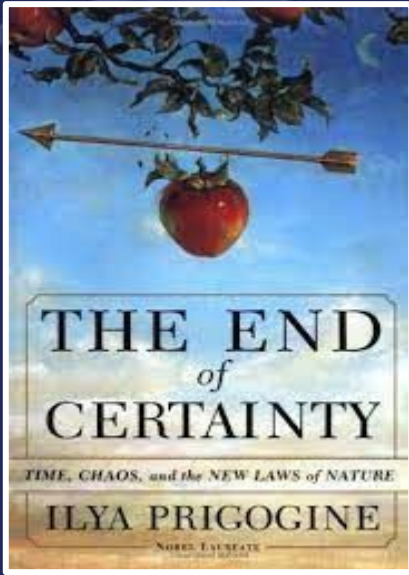
**stable systems  
tend to *oppose*  
their own *proper*  
function**



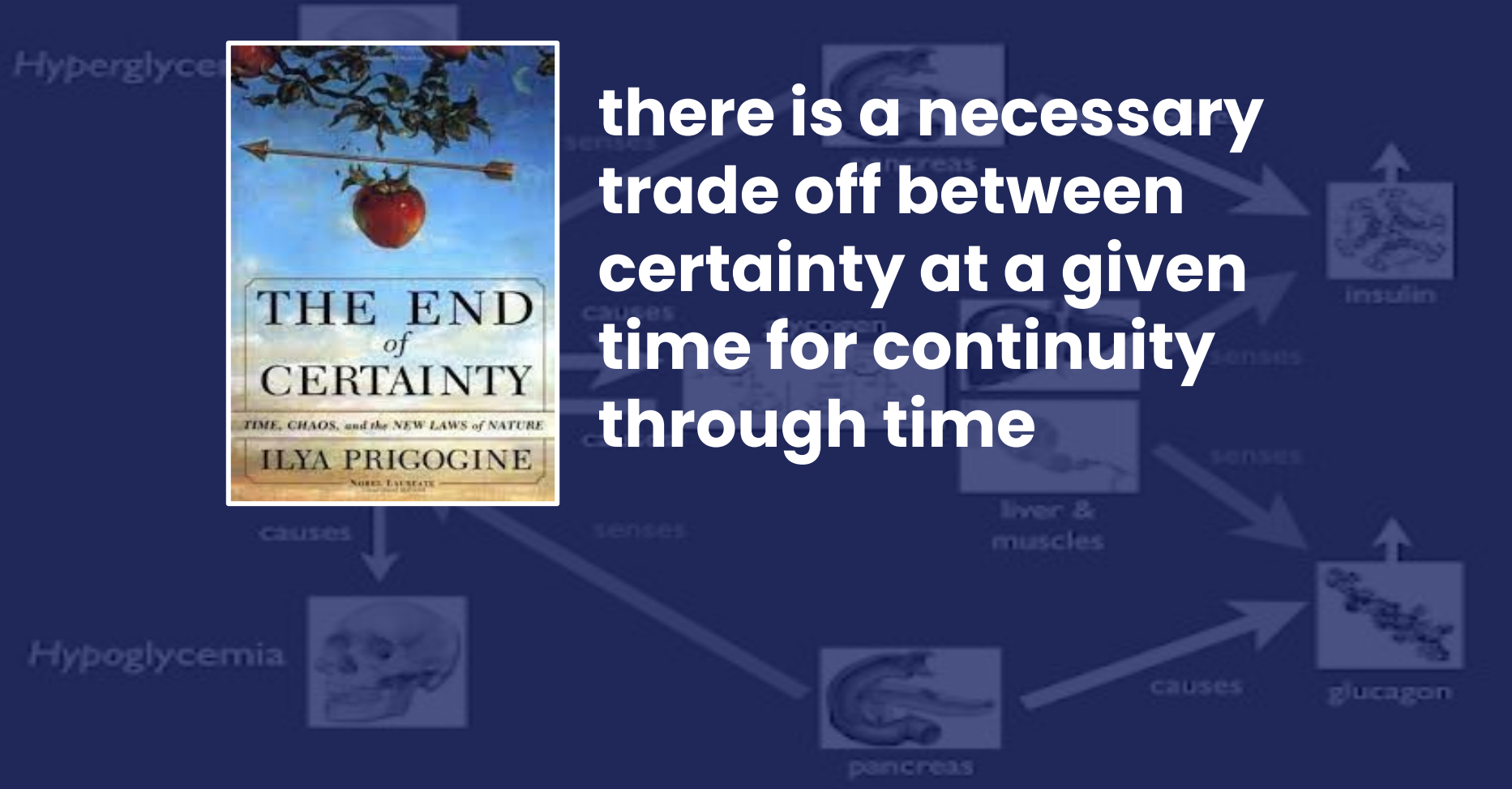


**dynamic thinking**  
**static thinking**



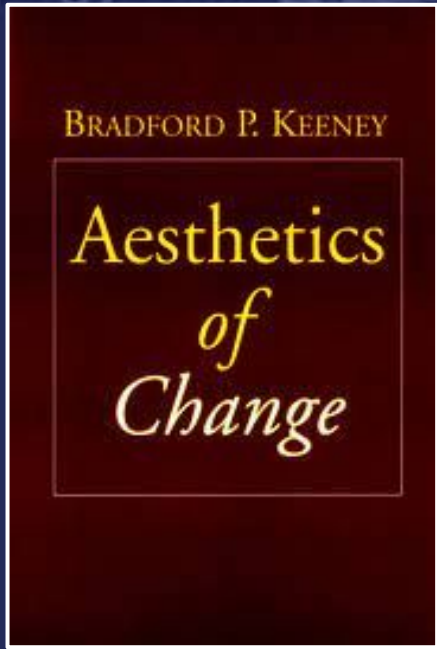


there is a necessary trade off between certainty at a given time for continuity through time



**evolution is always  
co-evolution**





all change can be understood as the effort to maintain some constancy, and all constancy as maintained through change

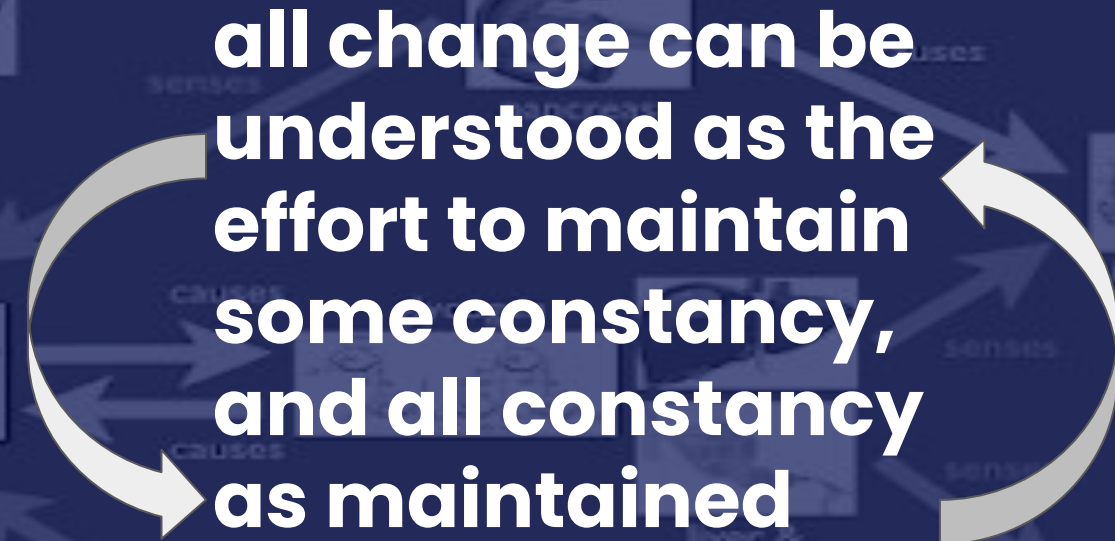
Hyperglycemia

Hypoglycemia

insulin

glucagon

pancreas



Hyperglycemia



causes



causes



Hypoglycemia

...

activity

stability

activity

stability

activity

stability

...



pancreas

causes



insulin

senses

senses

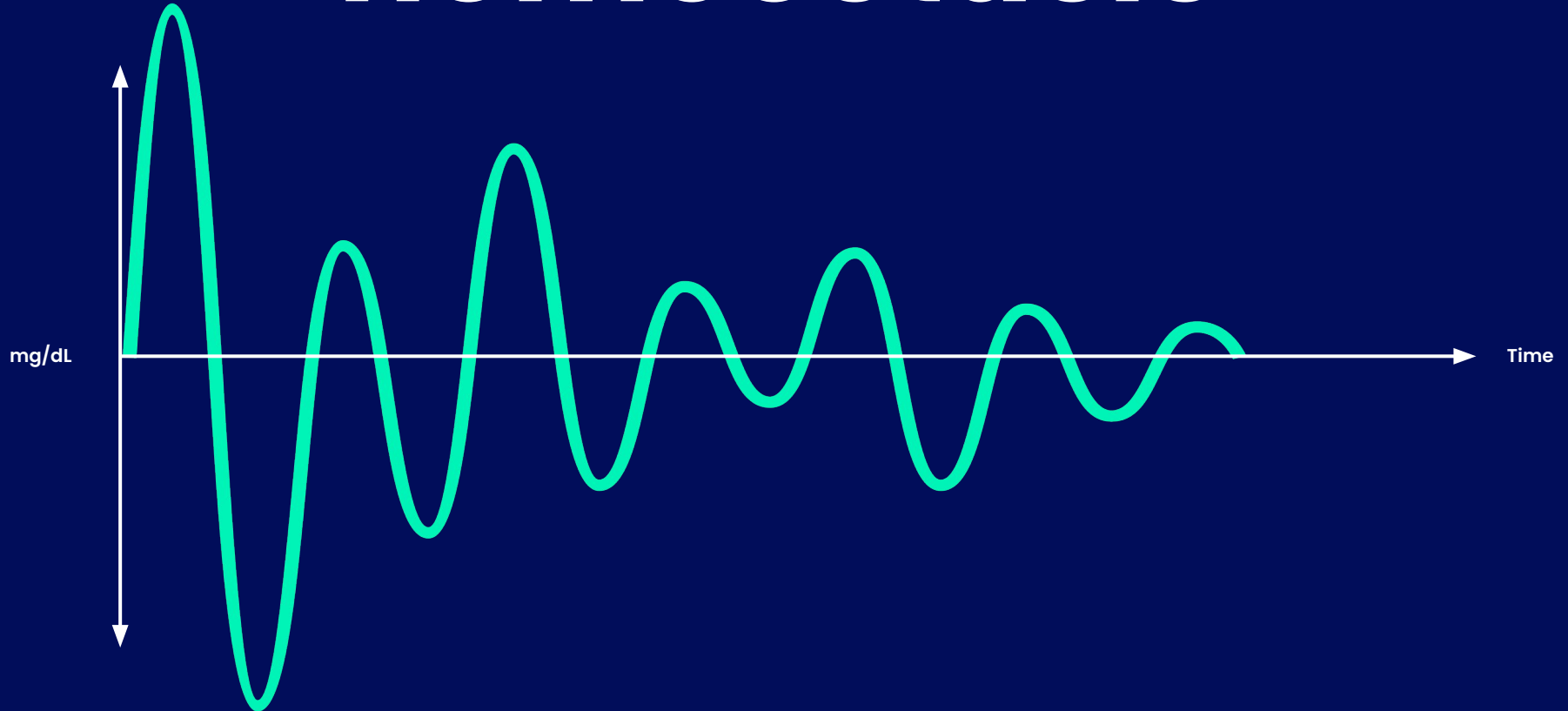


glucagon

causes



# homeostasis





**noun**

**v**

**verb**

# 100-proof





**testing and coding  
opposing each other in a  
stable  
co-evolving  
system**

**VOTES  
FOR  
TESTS**

WOMEN'S FREEDOM LEAGUE

**TESTS  
DEMAND  
THE VOTE  
THIS  
SESSION**

WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST  
STRAND

**TESTS  
DEMAND  
THE VOTE  
THIS  
SESSION**

WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST  
STRAND

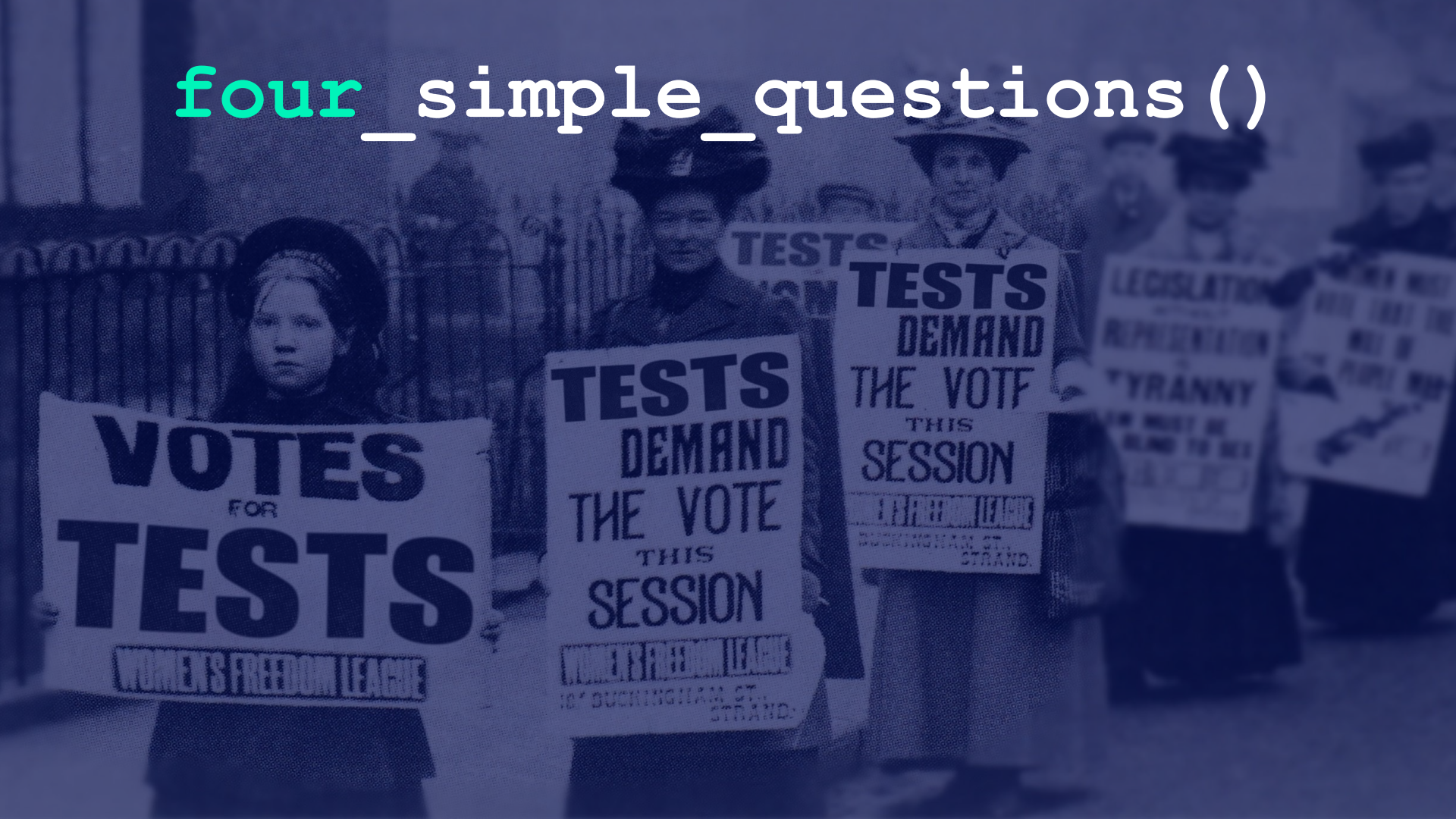
LEGISLATION  
REFUSION  
TYRANNY

WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST  
STRAND

and now for  
***something  
completely  
different?***



# four\_simple\_questions()



**Q1**

**what is a function's  
coverage?**

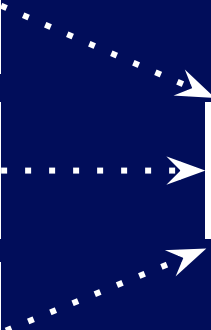
**eg, simple branch coverage**

**test\_f1()**

**test\_f2()**

**test\_f3()**

**f()**

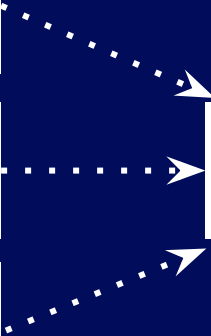
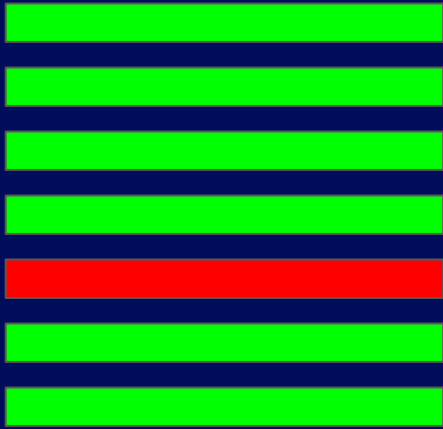


**test\_f1()**

**test\_f2()**

**test\_f3()**

**f()**

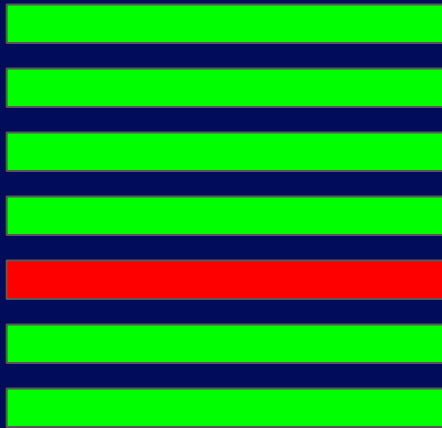


**test\_f1()**

**test\_f2()**

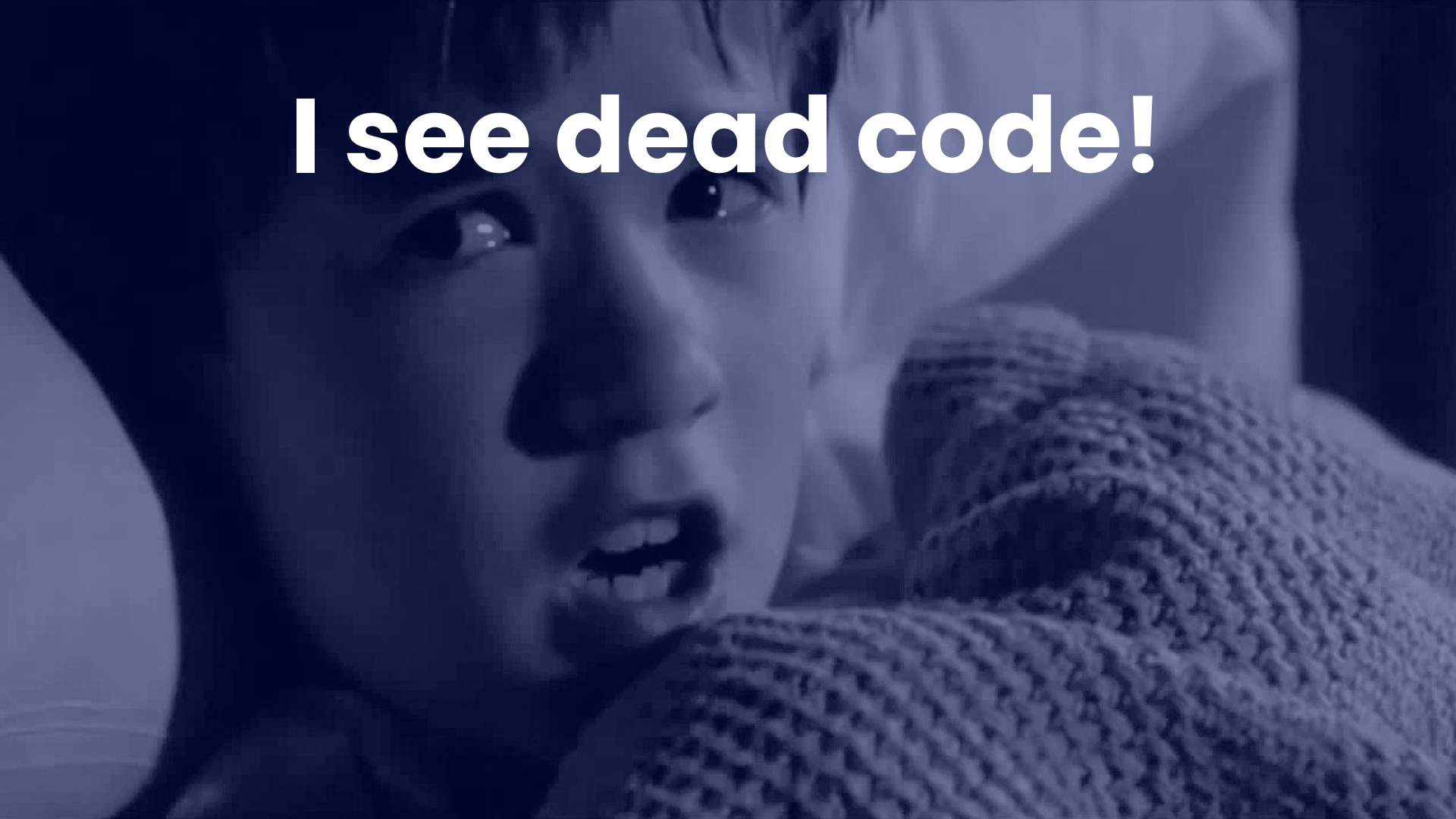
**test\_f3()**

**f()**



**suppose this  
statement  
is dead code**

**I see dead code!**







**lost \$450m in a  
few hours on  
1st August 2012**

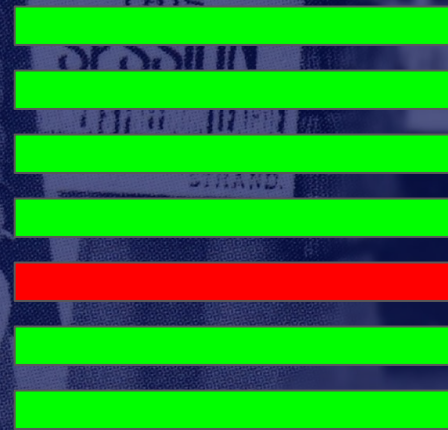
# what am I thinking now?

`test_f1()`

`test_f2()`

`test_f3()`

`f()`





**equal rights  
for tests!**

**VOTES  
FOR  
TESTS**

**WOMEN'S FREEDOM LEAGUE**

**TESTS  
THE VOTE  
THIS  
SESSION**

**WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.**

**TESTS  
THE VOTE  
THIS  
SESSION**

**LEGISLATION  
TYRANNY**

**WOMEN'S FREEDOM LEAGUE**

if you use a tool in the code,  
consider using the tool in the test-code

`test_f1()`



`test_f2()`



`test_f3()`



`f()`



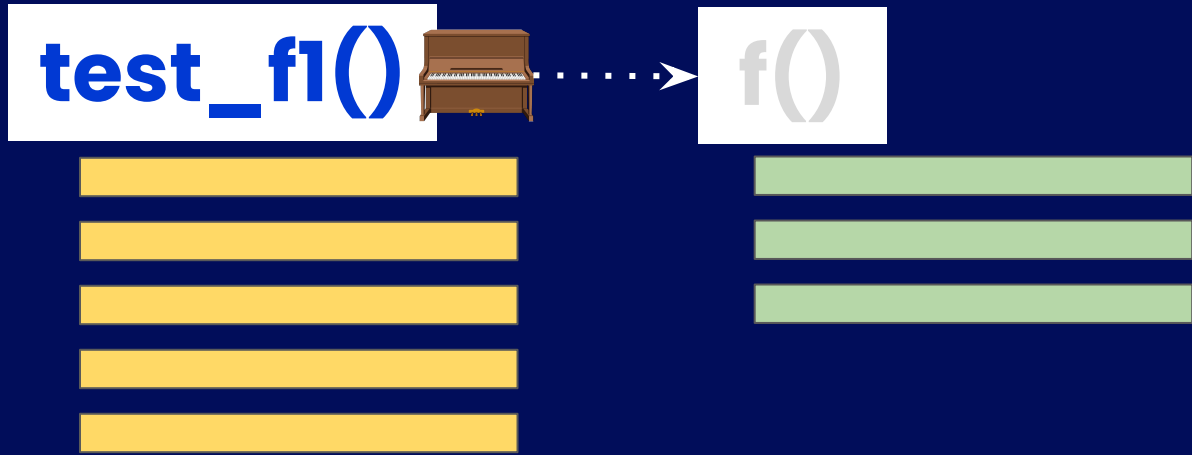
TESTS  
DEMAND  
THE VOTE

VOTE  
FOR  
TESTS  
WOMEN'S FREEDOM LEAGUE

LEGISLATION  
TYRANNY

107 DUCKINGHAM ST.  
STRAND.

# what is the coverage of a test function?



often I **do** get full coverage

`test_f1()`



often I **don't** get full coverage

`test_f1()`



`asserter()`



sometimes I get **no** coverage

**test\_f1()**



**test\_f1()**





**full coverage is  
important mostly  
so I can see when I  
don't have it**

unless I detect and  
delete dead code  
I **will** accumulate  
dead code

# **Q2** what is a function's complexity?

**eg, the number of different paths**



3

# what am I thinking now?

f()

3





# equal rights for tests!

**VOTES**  
FOR  
**TESTS**

WOMEN'S FREEDOM LEAGUE

**TESTS**  
THE VOTE  
THIS  
SESSION

WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

**TESTS**  
THE VOTE  
THIS  
SESSION

LEGISLATION  
TYRANNY

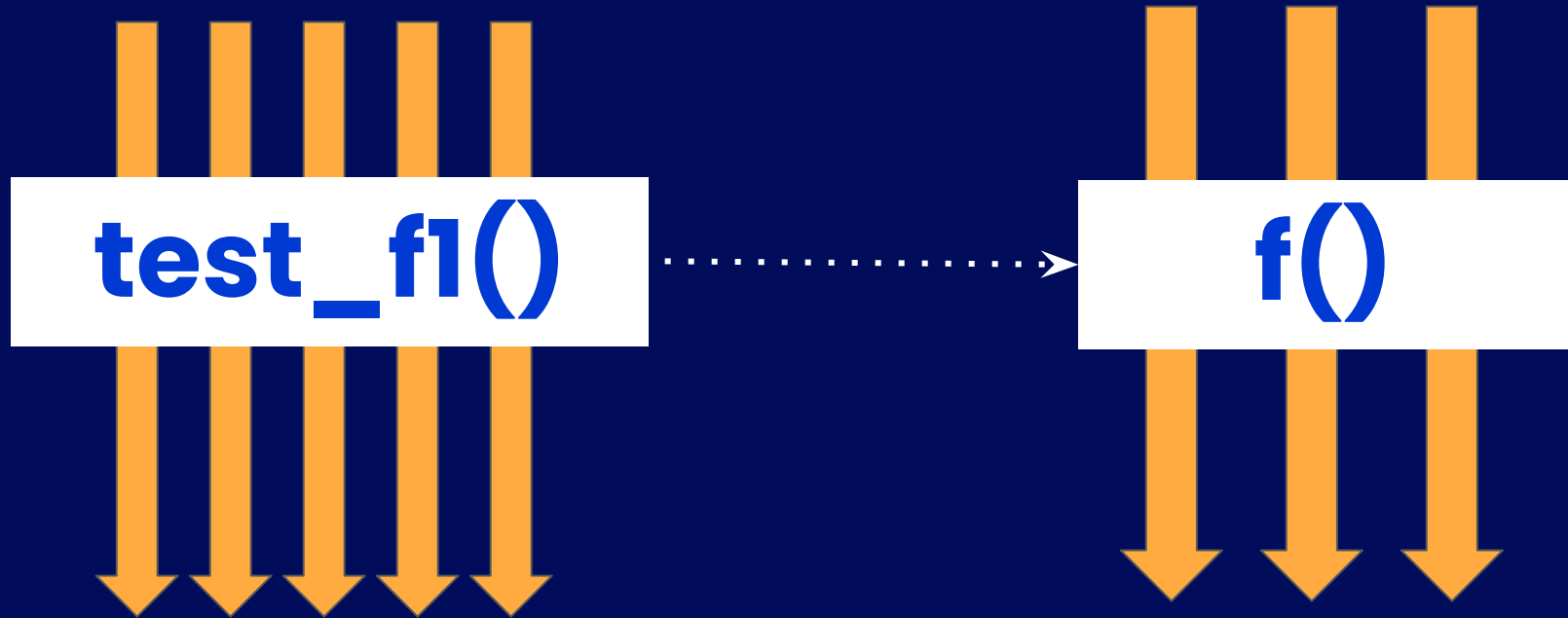
WOMEN WILL  
NOT BE  
TREATED AS  
PROPERTY



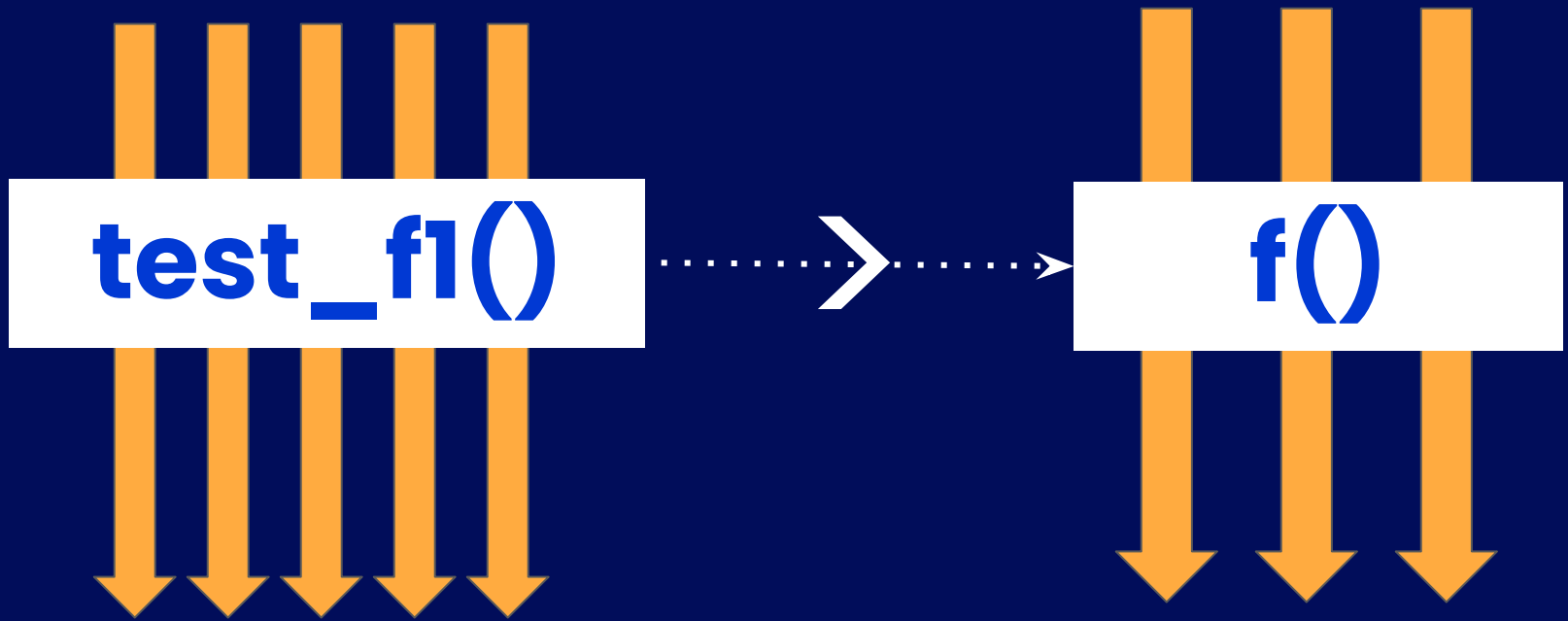
**test\_f1()**

5

# can you see a problem?



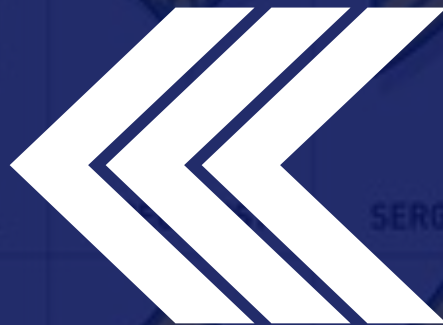




**test\_f1()**

**specific**

**f()**



**Q3**

**what is a function's  
behaviour?**

**eg, what does it print?**

# what am I thinking now?



what am I thinking now?

equal rights  
for tests!

**VOTES**  
FOR  
**TESTS**  
WOMEN'S FREEDOM LEAGUE

**TESTS**  
THE VOTE  
THIS  
SESSION  
107 DUCKINGHAM ST.  
STRAND.

**TESTS**  
THE VOTE  
THIS  
SESSION  
STRAND.

LEGISLATION  
TYRANNY

WOMEN'S FREEDOM LEAGUE

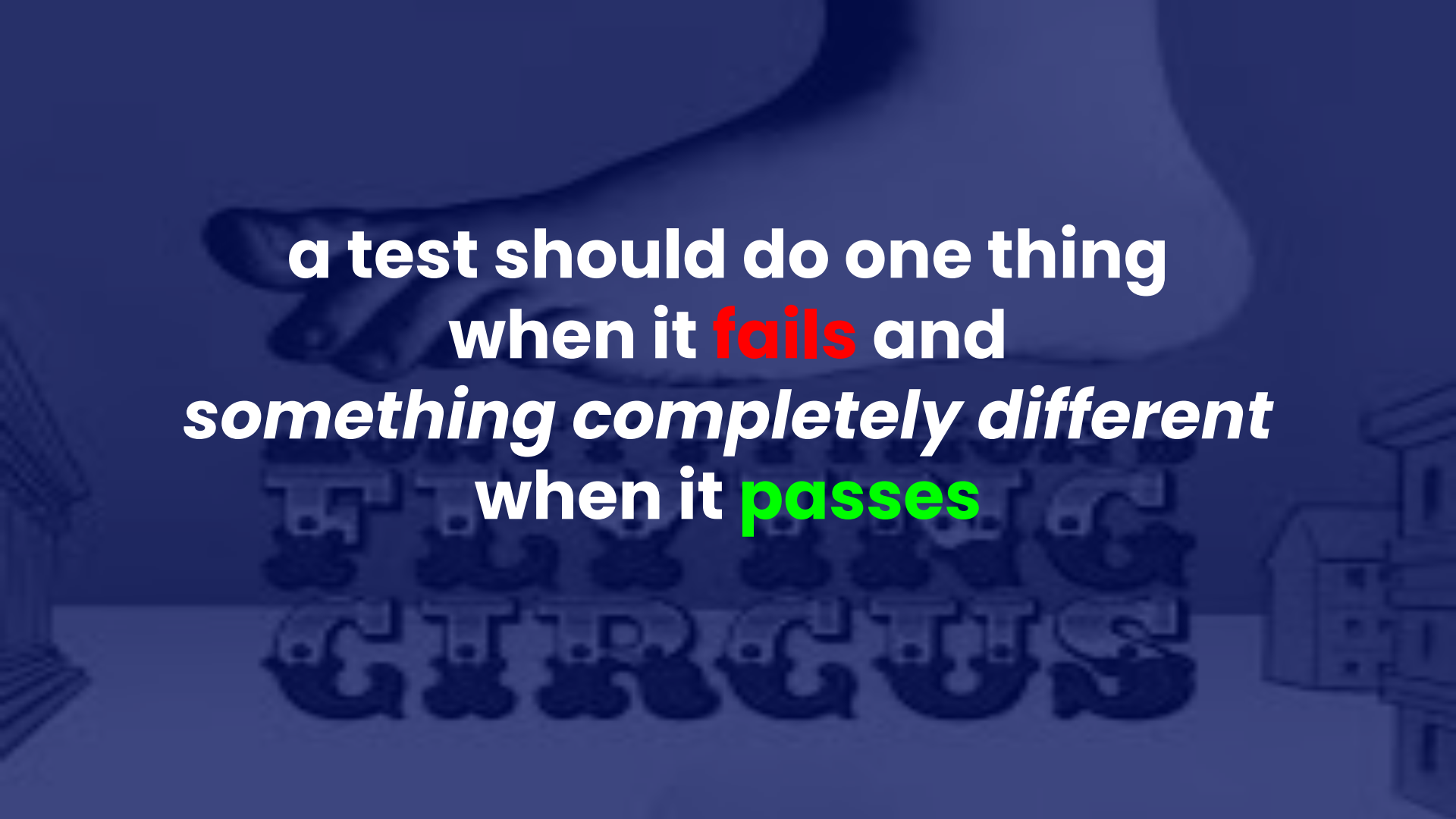
# hands up...

if you've ever thought  
a **failing** test  
was **passing**

# hands up...

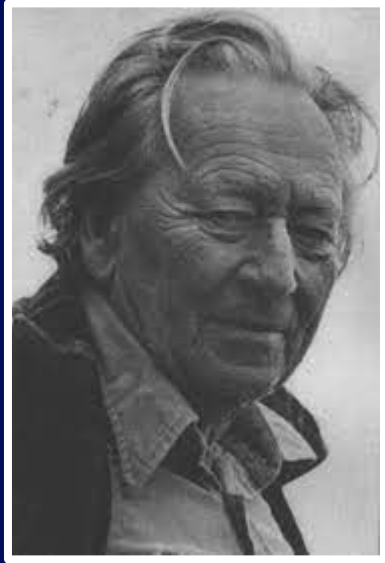
if you've ever thought  
a **passing** test  
was **failing**



A hand is shown pointing towards the text. The background is a solid blue color with faint, semi-transparent outlines of keyboard keys. The text is centered and reads: 

**a test should do one thing  
when it **fails** and  
*something completely different*  
when it **passes****





**Gregory Bateson**

**a difference  
that makes  
a difference**

when a test **fails** you want  
helpful diagnostics printed

when a test **passes** you want  
~nothing printed

test\_fails\_to\_get\_sha\_when\_image\_not\_retrieved\_from\_registry

docker = <pytest\_mock.plugin.MockerFixture object at 0xffff9007a190>

```
def test_fails_to_get_sha_when_image_not_retrieved_from_registry(mock):
```

```
    """
```

```
    Note: It is possible to get the sha for an image
    that has not yet been pushed to a registry
    """
```

```
    stub = DockerImagesGetAttrStub([])
```

```
    mocked_docker_image = mock.patch('fingerprinters.docker_fingerprinter.docker.from_env', return_value=stub)
```

```
    fingerprinter = DockerFingerprinter()
```

```
    image_name = 'acme/road-runner:3.7'
```

```
    with raises(ChangeError) as exc:
```

```
        fingerprinter.sha(f"{DOCKER_PROTOCOL}{image_name}")
```

```
    start = f"cannot determine digest for image: {image_name}"
```

```
    diagnostic = str(exc.value)
```

```
    assert diagnostic.startswith(start), diagnostic
```

```
AssertionError: Cannot determine digest for image: acme/road-runner:3.7 Check the image name is correct and push it to a registry.
```

```
    assert False
```

```
+ where False = <built-in method startswith of str object at 0xffff900fecf0>('cannot determine digest for image: acme/road-runner:3.7 Check the image name is correct and push it to a registry.')
```

```
+ where <built-in method startswith of str object at 0xffff900fecf0> = 'Cannot determine digest for image: acme/road-runner:3.7 Check the image name is correct. Check the image has been pushed to a registry.'.startswith
```

tests/unit/test\_bitbucket\_control\_pull\_request.py::test\_bitbucket\_not\_compliant\_raises PASSED  
tests/unit/test\_git.py::test\_list\_commits\_between\_master\_and\_production PASSED  
tests/unit/test\_merkelypipe.py::test\_defaults\_to\_Merkelypipe\_dot\_json\_in\_data\_dir PASSED  
tests/unit/test\_required\_env\_var.py::test\_value\_raises\_when\_not\_set\_in\_os PASSED  
tests/unit/test\_doc\_env\_vars.py::test\_doc\_example\_for\_all\_env\_vars PASSED  
tests/unit/test\_urls.py::test\_url\_for\_project PASSED  
tests/unit/test\_log\_deployment.py::test\_docker\_image PASSED  
tests/unit/test\_fingerprinter.py::test\_fingerprint\_\_file\_at\_root PASSED  
tests/unit/test\_fingerprinter.py::test\_builder\_\_raises\_when\_name\_is\_unknown PASSED  
tests/unit/test\_env\_var.py::test\_string\_is\_empty\_when\_set\_to\_empty\_string PASSED  
tests/unit/test\_approve\_deployment.py::test\_raises\_when\_src\_repo\_root\_does\_not\_exist PASSED  
tests/unit/test\_host\_env\_var.py::test\_value\_defaults\_to\_not\_set\_from\_env\_var PASSED  
tests/unit/test\_log\_test.py::test\_junit\_output\_with\_source\_dir\_specified\_with\_env\_var PASSED  
tests/unit/test\_sha256\_fingerprinter.py::test\_sha256\_protocol\_\_no\_artifact\_name\_raises PASSED  
tests/unit/test\_file\_fingerprinter.py::test\_no\_existing\_file\_\_same\_error\_message PASSED  
tests/unit/test\_log\_artifact.py::test\_log\_artifact\_\_file PASSED  
tests/unit/test\_dir\_fingerprinter.py::test\_dir\_fingerprint\_\_different\_non\_empty\_dir\_properties PASSED  
tests/unit/test\_docker\_fingerprinter.py::test\_docker\_fingerprint\_\_when\_image\_does\_not\_exist PASSED  
tests/unit/test\_merkelypipe.py::test\_MERKELYPIPE\_AND\_MERKELYPIPE\_LINE\_are\_used\_for\_all\_commands PASSED  
tests/unit/test\_http\_retry.py::test\_retry\_\_http\_status\_is\_not\_503 PASSED  
tests/unit/test\_mock\_file\_fingerprinter.py::test\_mock\_file\_fingerprinter\_\_unexpected\_artifact\_sha\_arg PASSED  
tests/unit/test\_host\_env\_var.py::test\_value\_is\_required PASSED  
tests/unit/test\_doc\_ref.py::test\_ref\_targets PASSED  
tests/unit/test\_mock\_file\_fingerprinter.py::test\_prints\_exception\_info\_when\_exception\_trips PASSED  
tests/unit/test\_http\_retry.py::test\_get\_stops\_retrying\_when\_non\_503\_and\_returns\_response PASSED  
tests/unit/test\_runner.py::test\_raises\_when\_merkely\_command\_is\_unknown PASSED  
tests/unit/test\_command\_builder.py::test\_command\_named\_raises\_when\_unknown PASSED  
tests/unit/test\_fingerprint\_env\_var.py::test\_docker\_protocol\_properties PASSED  
tests/unit/test\_required\_env\_var.py::test\_is\_required PASSED  
tests/unit/test\_compound\_env\_var.py::test\_ci\_env\_var\_value\_class PASSED  
tests/unit/test\_fingerprinter.py::test\_fingerprint\_\_file\_in\_directory PASSED  
tests/unit/test\_sha256\_fingerprinter.py::test\_sha256\_protocol\_\_valid\_sha\_and\_non\_empty\_artifact PASSED  
tests/unit/test\_mock\_docker\_fingerprinter.py::test\_raises\_when\_unexpected\_artifact\_sha\_arg PASSED  
tests/unit/test\_scoped\_dir\_copier.py::test\_raises\_when\_source\_dir\_does\_not\_exist PASSED

**Q4**

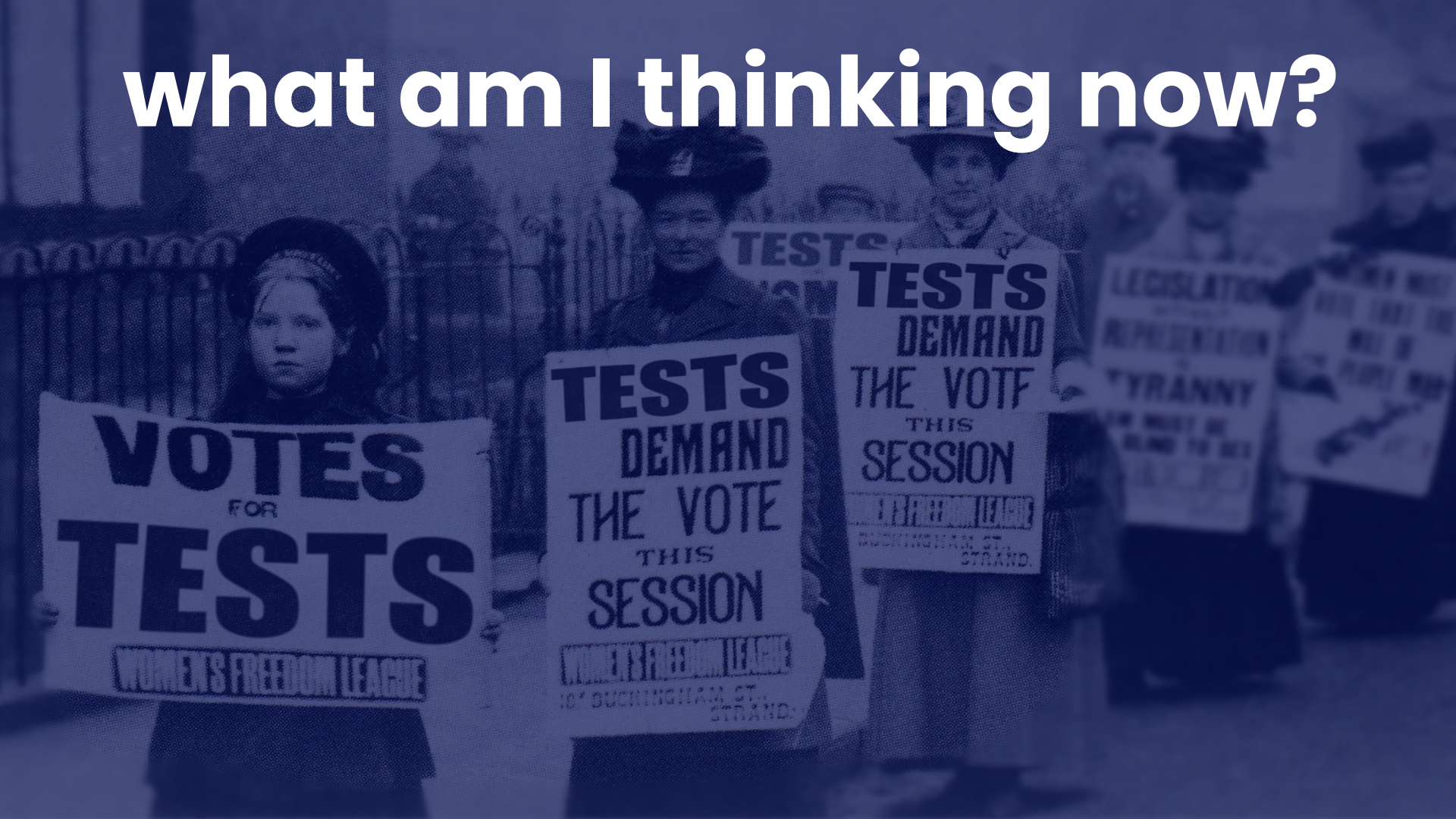
**what is a function's  
name?**

# naming a function




- member function?
- return value assignment?
- argument expressions?
- side effects?
- ...
- ...

# what am I thinking now?



what am I thinking now?

equal rights  
for tests!



VOTES  
FOR  
TESTS  
WOMEN'S FREEDOM LEAGUE

TESTS  
FOR TESTS  
THE VOTE  
THIS  
SESSION  
WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

TESTS  
FOR TESTS  
THE VOTE  
THIS  
SESSION  
WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

LEGISLATION  
TYRANNY

WOMEN WILL NOT  
BE DECEIVED



**equal rights means  
equal opportunities,  
not equal outcomes**

**VOTES  
FOR  
TESTS**  
WOMEN'S FREEDOM LEAGUE

**TESTS  
DEMAND  
THE VOTE  
THIS  
SESSION**  
WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

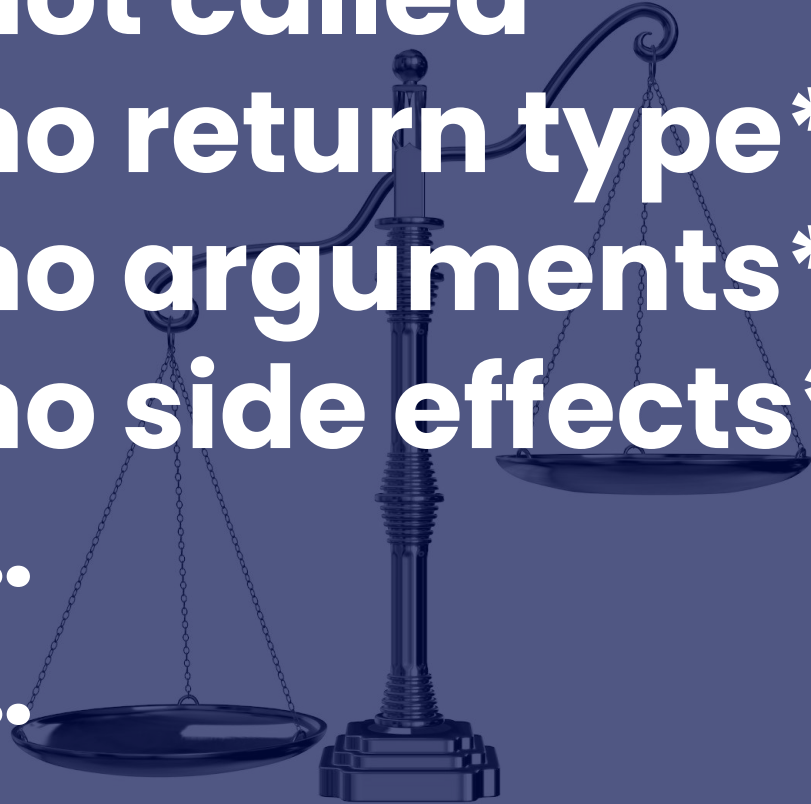
**TESTS  
DEMAND  
THE VOTE  
THIS  
SESSION**  
WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

LEGISLATION  
TYRANNY

WOMEN'S FREEDOM LEAGUE  
107 DUCKINGHAM ST.  
STRAND.

**naming a test function  
is different because a  
test function is different**

- **not called\***
- **no return type\***
- **no arguments\***
- **no side effects\***
- ...
- ...
- ...



```
7
8 ▶ def test_MERKELY_OWNER_and_MERKELY_PIPE_LINE_are_used_for_all_commands_except_declare_pipeli
9   os_env = {
10     "MERKELY_COMMAND": "log_artifact",
11     "MERKELY_OWNER": "acme",
12     "MERKELY_PIPELINE": "road-runner",
13   }
14   external = External(env=os_env)
15   cls = Command.named('declare_pipeline')
16   json = cls(external).merkelypipe
17   assert json["owner"] == "acme"
18   assert json["name"] == "road-runner"
19
20
21 ▶ def test_MERKELY_OWNER_and_MERKELY_PIPE_override_entries_in_Merkelypipe_even_for_declare_pipe
22   # declare_pipeline requires a Merkelypipe.json, all other commands don't.
23   # Even for declare_pipeline, allow env-vars to override the "owner"
24   # and "name" entries so that when anyone is doing the loan-calculator demo
25   # they only need to set the values once in the env-vars for all commands.
26   os_env = {
27     "MERKELY_COMMAND": "declare_pipeline",
28     "MERKELY_OWNER": "Acme",
```



**too much context for  
a single identifier?**



**where should  
context live?**



# in a doc-string?

```
def test():
```

```
    """
```

```
    when the query finds no matches  
    update_one_ledger() returns None  
    and does not update the ledger
```

```
    """
```

# a uid for the name?

```
def test_d878c500():  
    """  
    when the query finds no matches  
    update_one_ledger() returns None  
    and does not update the ledger  
    """
```

```
$ date +"%s%N" | sha256sum | awk '{print substr($1,1,6)}'  
d878c5  
$
```



# pytest --ds ledger

```
def test_d878c500():  
    """  
    when the query finds no matches  
    update_one_ledger() returns None  
    and does not update the ledger  
    """
```

```
def pytest_addoption(parser):
    parser.addoption(
        "--ds", action="append", help="Only run tests with matching docstrings"
    )

def pytest_collection_modifyitems(session, config, items):
    filter_strings = config.getoption('--ds')
    if filter_strings is None:
        return

    def doc_string_match(item):
        ds = item.function.__doc__
        if ds is None:
            print(f"No docstring for {item.function.__name__}")
            return False
        else:
            return any(fs for fs in filter_strings if fs in ds)

    keep = [item for item in items if doc_string_match(item)]
    items.clear()
    items.extend(keep)
```

**confstest.py (top level)**

# t fixture?

```
def test_d878c500(t):
```

```
    """
```

```
    when the query finds no matches  
    update_one_ledger() returns None  
    and does not update the ledger
```

```
    """
```

```
    db = documentdb.Database(t.id)  
    coll = db.collection(name=t.id, use_ledger=True)  
    coll.create_document(t.id, {"x": 42}, now())  
    assert coll._ledger_count(t.id) == 1
```

```
@pytest.fixture
```

```
def t(request):
```

```
    test_name = request.node.name  
    parts = test_name.split('_')  
    return T(parts[1][0:8])
```

```
class T:
```

```
    def __init__(self, uid):  
        self.id = uid
```

**t.id == "4dd3b712"**

# summary

- the equilibrium law
- thinking dynamically & thinking statically
- coding & testing co-evolving
- use tools/techniques on code and test-code?
- equal rights for tests!

**have fun  
be nice to each other  
and the planet**

