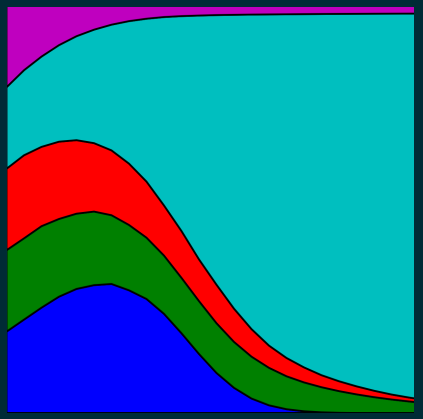
Writing tests for research software

@NikoletaGlyn



Software
Sustainability
Institute

TESTING

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

main.py

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return 2 * fib(n - 1)
```

n	2	3	4	...	16	17	18
F_n	1	2	3	...	987	1597	2584
F_{n-1}	1	1	2	...	610	987	1597
$\frac{F_n}{F_{n-1}}$	1.000	2.000	1.500	...	1.618	1.618	1.618

n	2	3	4	...	16	17	18
F_n	1	2	3	...	987	1597	2584
F_{n-1}	1	1	2	...	610	987	1597
$\frac{F_n}{F_{n-1}}$	1.000	2.000	1.500	...	1.618	1.618	1.618

$$\phi \simeq 1.61803...$$

```
.
|-- main.py
|-- golden.py
```

golden.py

```
import main

for n in range(10, 100000):
    golden_ratio = fib(n) / fib(n - 1)
    print(golden_ratio)
```

golden.py

```
import main

for n in range(10, 100000):
    golden_ratio = fib(n) / fib(n - 1)
    print(golden_ratio)
```

```
2.0
2.0
2.0
2.0
2.0
2.0
2.0
2.0
2.0
...
```

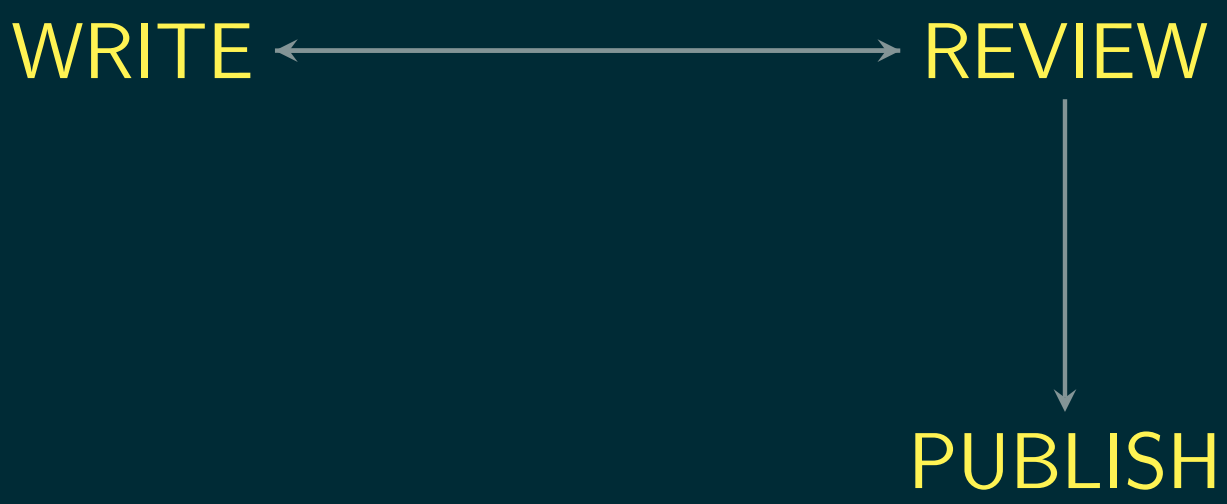
golden.py

```
import main

for n in range(10, 100000):
    golden_ratio = fib(n) / fib(n - 1)
    print(golden_ratio)
```

```
2.0
2.0
2.0
2.0
2.0
2.0
2.0
2.0
2.0
...
```

Glynatsi 2017, “SOLVES THE FIBONACCI MYSTERY”


WRITE
REVIEW
PUBLISH

20% OF GENETIC RESEARCH IS WRONG

Gene name errors are widespread in the scientific literature by Mark Ziemann, Yotam Eren and Assam El-Osta

INDUSTRY

INDUSTRY

AMAZON

```
.
|-- main.py
|-- golden.py
|-- test_main.py
```

test_main.py

```
import unittest
import main

class TestExample(unittest.TestCase):

    def test_initial(self):
        self.assertEqual(fib(0), 0)
        self.assertEqual(fib(1), 1)

    def test_fib(self):
        self.assertEqual(fib(2), 1)
        self.assertEqual(fib(3), 2)
```

test_main.py

```
import unittest
import main

class TestExample(unittest.TestCase):

    def test_initial(self):
        self.assertEqual(fib(0), 0)
        self.assertEqual(fib(1), 1)

    def test_fib(self):
        self.assertEqual(fib(2), 1)
        self.assertEqual(fib(3), 2)
```

```
python -m unittest test_main.py
```

test_main.py

```
import unittest
import main

class TestExample(unittest.TestCase):

    def test_initial(self):
        self.assertEqual(fib(0), 0)
        self.assertEqual(fib(1), 1)

    def test_fib(self):
        self.assertEqual(fib(2), 1)
        self.assertEqual(fib(3), 2)
```

```
python -m unittest test_main.py
```

```
self.assertEqual(fib(2), 1)
AssertionError: 2 != 1

----------------------------
Ran 2 tests in 0.000s

FAILED (failures=1)
```

main.py

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return 2 * fib(n - 1)
```

main.py

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n - 1) + fib(n - 2)
```

main.py

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n - 1) + fib(n - 2)
```

```
python -m unittest test_main.py
------------------------------
Ran 2 tests in 0.000s

OK
```

main.py

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n - 1) + fib(n - 2)
```

```
python -m unittest test_main.py
------------------------------
Ran 2 tests in 0.000s

OK
```

Glynatsi 2017, “TRYING TO RECLAIM REPUTATION”

Doc Testing

main.py

```
import unittest

def fib(n):
    """Returns the n th fibonacci number.

    For example:

        >>> fib(5)
        5
        >>> fib(6)
        8
        >>> fib(5) + fib(6)
        13
        >>> fib(7)
        10
   """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
```

```
python -m doctest main.py

Failed example:
    fib(7)
Expected:
    10
Got:
    13
**********************************
1 items had failures:
   1 of   4 in main.fib
***Test Failed*** 1 failures.
```

main.py

```
import unittest

def fib(n):
    """Returns the n th fibonacci number.

    For example:

        >>> fib(5)
        5
        >>> fib(6)
        8
        >>> fib(5) + fib(6)
        13
        >>> fib(7)
        13
   """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
```

Property Based Testing

```python
from hypothesis import given
from hypothesis.strategies import integers

class TestFib(unittest.TestCase):
    @given(k=integers(min_value=2))
    def test_fib(self, k):
        self.assertTrue(fib(k), fib(k - 1) + fib(k - 2))
```

https://github.com/HypothesisWorks

@DRMacIver

It's impossible to conduct research without software, say 7 out of 10 UK researchers

Simon Hettrick

uk/blog/2016-09-12-its-impossible-conduct-research-without-out-10-uk-researchers

USE

92%

IMPOSSIBLE

69%

DEVELOP

56%

TRAINING

79%

USE

92%

IMPOSSIBLE

69%

DEVELOP

56%

TRAINING

79%

Axelrod : https://github.com/Axelrod-Python/Axelrod

Arcas: https://github.com/Nikoleta-v3/Arcas

Ciw: https://github.com/CiwPython/Ciw

Pandas: https://github.com/pandas-dev/pandas

Skleanr: http://scikit-learn.org/stable/

Numpy: https://github.com/numpy/numpy

cryptography: https://github.com/pyca/cryptography

fastnumbers: https://github.com/SethMMorton/fastnumbers

yacluster: https://github.com/KrzysiekJ/yacluster

binaryornot: https://github.com/audreyr/binaryornot

@NikoletaGlyn
https://github.com/Nikoleta-v3
@SoftwateSaved
@PhoenixCUni