Data-Driven Programming
A Philosophy

https://github.com/mohawk2/presentations

Introduction

- Code gets in the way
- Let data do the talking
- Decisions are expensive

Introduction

- Why
- What
- How (worked examples)
- Resources
- Exercises

Why


NOW LET'S FORGET OUR
TROUBLES WITH A BIG BOWL
OF STRAWBERRY ICE CREAM.

Why

- Programming is hard
- Thinking gives you wrinkles
- If processing data, think about data
- Do less work (especially less copy-paste)!
- Approach helps in coding interviews

What

- Unix philosophy (ish)
- GUIs (sort of)
- Defer decisions to minimise error cost

Wikipedia definition

- “pattern matching and resulting processing”

Paragon definition

Suppose you wanted to create a cross tab query. In databases without native support for such constructs, you would often do something like the below (NOTE: example below is SQL Server syntax).

```
--Naive repetitive painful way
    SELECT proj_id, SUM(CASE WHEN proj_fund = 'Fund A' THEN amt ELSE NULL
END)
        As [totFund A],
        SUM(CASE WHEN proj_fund = 'Fund B' THEN amt ELSE NULL END)
            As [totFund B],
    ...ad nauseum, SUM(amt) As totalcost
    FROM projectfunding
    GROUP BY proj_id
```

Now if you were to pay attention to the repetitious pattern in the above, you may realize you can more quickly and more flexibly rewrite the above like the below by querying a fund list.

How

- Think about capturing branches into data
- Spot repetition
- Tests vs “real code”

BDD

```
testdir.makefile(".feature", simple="""
        Feature: Homepage functionality
            Scenario: Homepage
                When the user requests list home
                Then response 0 status code is "200"
                And response 0 element "#main_title" contains 'Welcome'
    """)
```

PDL::LinearAlgebra tests

```
runtest($x, 'mgschurx', \@mgschur_exp, [sequence(2,2),1,1,1,1,sub {1},3]);
runtest($x, 'mgschurx', \@mgschur_exp, [sequence(2,2),0,0,1,1,sub {1},1,1,0]);
runtest($x, 'mgschurx', \@mgschur_exp, [sequence(2,2),0,0,2,2,undef]);
runtest($x, 'mqr', pdl([-0.49738411,-0.86753043],[-0.86753043,0.49738411]));
runtest($wide->t, 'mqr', pdl([-0.523069,-0.5023351],[-0.0364932,-0.793903],
[-0.851508,0.34260173]));
runtest($x, 'mrq', pdl([0.27614707,-0.3309725],[0,-1.0396634]));
runtest($wide, 'mrq', pdl([0,0.68317233,-0.45724782],[0,0,-1.0587256]), [1]);
runtest($wide->t, 'mrq', pdl([-0.603012,-0.619496],[-0.684055,-0.226644],
[0,-0.728010]));
runtest($x, 'mql', pdl([0.99913307,-0.041630545],
[-0.041630545,-0.99913307]));
runtest($wide, 'mql', pdl([0.274721,-0.961523],[-0.961523,-0.274721]));
```

- Discussion: GUIs?

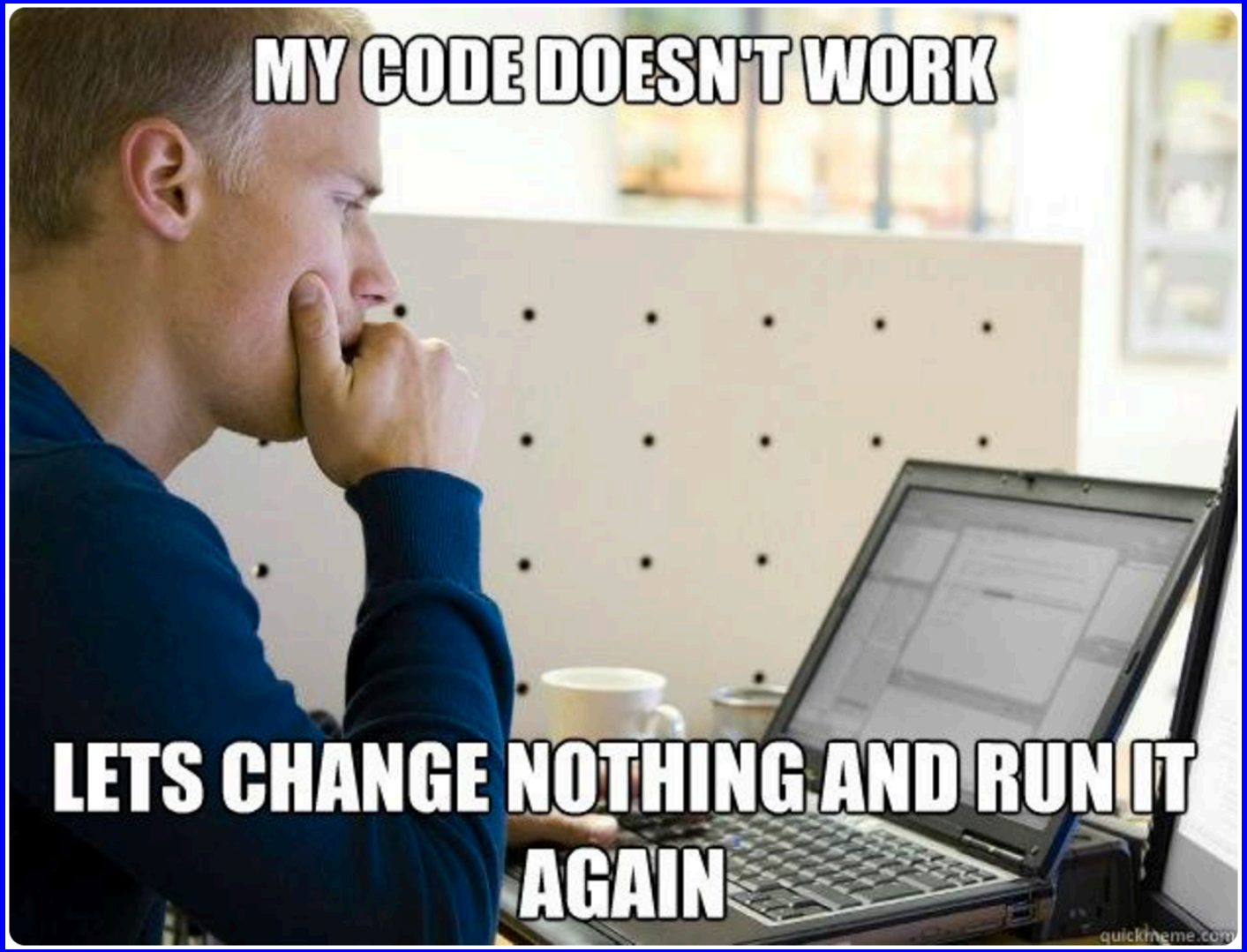
Summary

- Data-driven programming philosophy/approach
- What, why, how
- Examples

Resources

- https://en.wikipedia.org/wiki/Data-driven_programming
- https://stackoverflow.com/questions/1065584/what-is-data-driven-programming
- https://www.paragoncorporation.com/ArticleDetail.aspx?ArticleID=31
- https://pypi.org/project/pytest-bdd-web/
- https://github.com/mohawk2/presentations

Questions?


MY CODE DOESN'T WORK
LETS CHANGE NOTHING AND RUN IT AGAIN
quickmeme.com

Exercises

- Fibonacci plus tests
- Light frequency to colour plus tests
- Tax calculation plus tests