Clean Coding for Machine Learning Projects

Toronto Machine Learning Summit, Toronto
November 20, 2019

guild.ai

Agenda

- Introductions
- On “Clean Code”
- Live Refactoring Exercise
- Group Exercises

Introductions

About Me

- Programming languages: Python, C, C++, Erlang, Java, ML, Visual Basic
- Founder, Guild AI - ML Engineering Toolkit
- Founder and co-organizer, Chicago ML
- Previously CloudBees
  - Lead architect for CloudBees Jenkins Enterprise (primary product)
  - Director of PaaS operations
- Co-founder, Stax Networks (Java PaaS, acquired by CloudBees in 2010)
- Co-led BPM practice Capgemini North America

About You

Name

Where From

Short Background on ML and what you'd like to get out the workshop

On “Clean Code”

Motivation

Do no harm

Reproducibility

Agility

“Production Standard”

Data Science Code vs Traditional Software

	Data Science Code	*Tradition Software*
Expected Application	Experimental	Long running
Destination	Reports, Notebooks	Production systems
Collaborators	Few	Many
Expected Evolvability	Low	High
Size / Complexity	Low - Med	Low - Very High

Data Science Code vs Traditional Software

70% - 80% of my code is thrown away.

- A data scientist

Notebooks

“5 Reasons Why Jupyter Notebooks Suck” * source

Almost impossible to enable good code versioning

No IDE integration, no linting, no code-style correction

Very hard to test

Non-linear workflow of Jupyter

Jupyter is bad for running long asynchronous tasks

* Does not represent the opinions of TMLS workshop facilitator or its organizers :)

Goals for Code

Must work (functional)

Must solve a customer problem (suitable)

Must fit into existing system (deployable)

Must be understandable by other programmers (intentional)

Must support change (evolvable) *

Credit: Uncle Bob Martin from “The Clean Coder” with augmentation (*)

Test Driven Development

Write a minimal failing test

Write just enough code to make the test pass

Repeat ad nauseam

If you're happy slamming some code together that more or less works and you're happy never looking at the result again, TDD is not for you. TDD rests on a charmingly naïve geekoid assumption that if you write better code, you'll be more successful. TDD helps you to pay attention to the right issues at the right time so you can make your designs cleaner, you can refine your designs as you learn.

- Kent Beck from *Test Driven Development: By Example*

Test Driven Development

Code without tests is bad code. It doesn't matter how well written it is; it doesn't matter how pretty or object-oriented or well encapsulated it is.

With tests, we can change the behavior of our code quickly and verifiably. Without them, we really don't know if our code is getting better or worse.

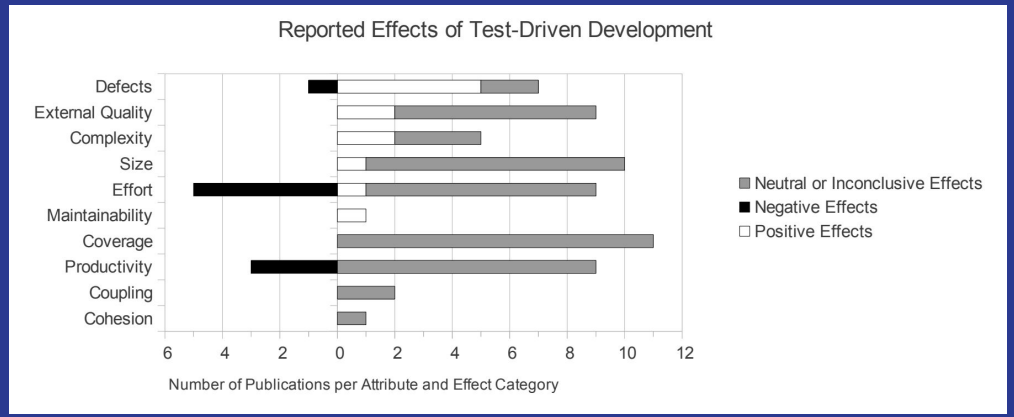
- Michael Feathers from *Working Effectively with Legacy Code*

Test Driven Development

[On the Effectiveness of Unit Test Automation at Microsoft]

[Assessing Test-Driven Development at IBM]

[Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies]



Source: Effects of Test-Driven Development

Garrett on TDD

As a movement/religion, meh

In practice - just see how far you can go without a test regime

Keep it narrative - think story, drama (example)

Test as needed, not on principle

Narrative Testing Using Python Doctest

doctest API docs

Test examples from Guild AI

How to run tests defined in a text file

“My day refactoring Python code”

Product Realities

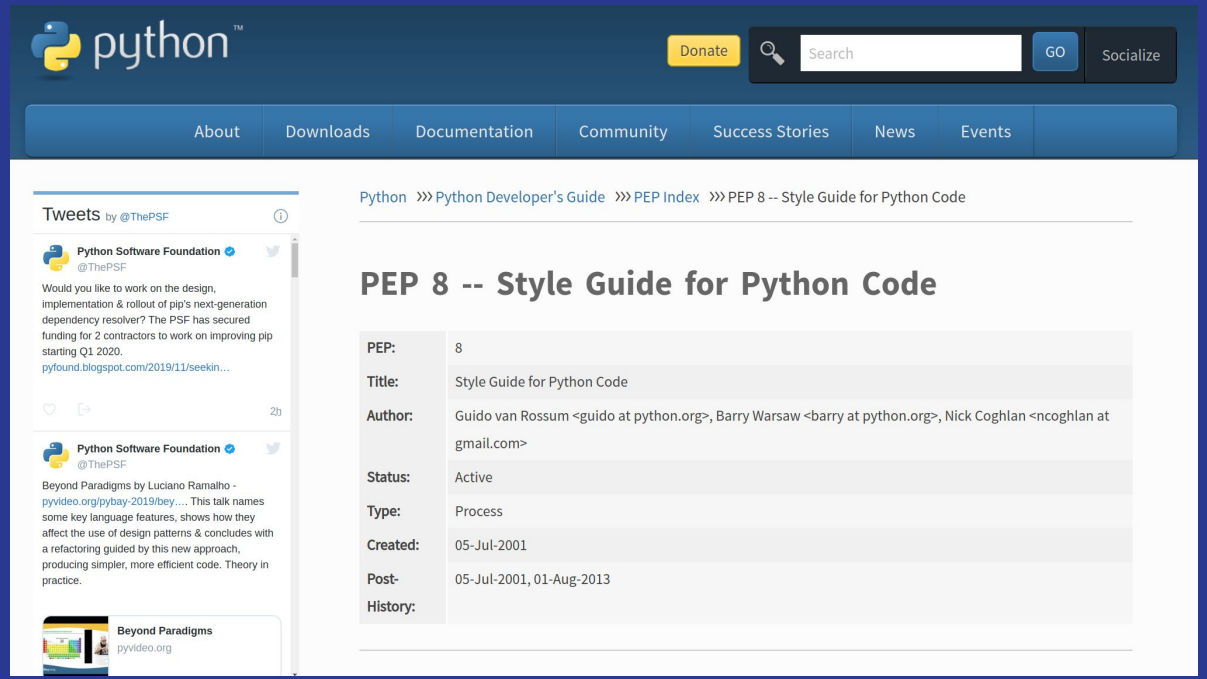
Adopting Erlang, a story

W. Edwards Deming

Deming to Production Manager: *Why is quality important to you?*

Manager: *Less rework.*

PEP 8


python™
Donate
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Tweets by @ThePSF
Python Software Foundation
@ThePSF
Would you like to work on the design, implementation & rollout of pip's next-generation dependency resolver? The PSF has secured funding for 2 contractors to work on improving pip starting Q1 2020.
pyfound.blogspot.com/2019/11/seekin…
2h
Python Software Foundation
@ThePSF
Beyond Paradigms by Luciano Ramalho - pyvideo.org/pybay-2019/bey…. This talk names some key language features, shows how they affect the use of design patterns & concludes with a refactoring guided by this new approach, producing simpler, more efficient code. Theory in practice.
Beyond Paradigms
pyvideo.org
Python >>> Python Developer's Guide >>> PEP Index >>> PEP 8 -- Style Guide for Python Code
PEP 8 -- Style Guide for Python Code
PEP: 8
Title: Style Guide for Python Code
Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status: Active
Type: Process
Created: 05-Jul-2001
Post-History: 05-Jul-2001, 01-Aug-2013

Python Linters

Linter	*Category*	*Description*
Pylint	Logical & Stylistic	Checks for errors, tries to enforce a coding standard, looks for code smells
PyFlakes	Stylistic	Analyzes programs and detects various errors
pycodestyle	Stylistic	Checks against some of the style conventions in PEP 8
pydocstyle	Stylistic	Checks compliance with Python docstring conventions
Bandit	Logical	Analyzes code to find common security issues
MyPy	Logical	Checks for optionally-enforced static types

Source: Python Code Quality: Tools & Best Practices

Garrett on Software

Make it Obvious

Comments (Generally) Considered Harmful

LOC Per Function: Proxy for Quality

Test With Stories

Functions

- Name
- Arguments
- Return Value
- Side Effects
- Exceptions

David Beazely on OO in Python

Garrett to Dave Beazley: *How do you deal with 'staticmethod' cruft in Python class definitions?*

David Beazely: *I don't do any of that. I've pretty much adopted a functional programming style for everything.*

Garrett on OO in Python

Don't

Use classes for "struct" only

Only use `self` in `__init__` and only for single value assignment

```
class DataSet:

    def __init__(self, name, path,
                 val_split=0.2):
        self.name = name
        self.path = path
        self.val_split = val_split
        self.data = None

# DataSet API (no methods)
def init_dataset(name, path):
    dataset = DataSet(name, path)
    load_data(dataset)
    return dataset
```

Garrett on OO in Python

Old Style:

https://github.com/guildai/guildai/blob/5c324a2d11e36e7f8ecab7f4630b667b4ee1a11c/guild/op_legacy.py

New Style:

https://github.com/guildai/guildai/blob/5c324a2d11e36e7f8ecab7f4630b667b4ee1a11c/guild/op.py

Refactoring Example

Refactoring Example GitHub Repo

https://github.com/gar1t/2019-TMLS-workshop

Group Refactoring Exercise

Code Refactoring Considerations

Is the intent clear?

Does the intent make sense?

How does the code fit into a deployment scenario?

What artifacts are generated?

How can the results be verified?

What might evolve? Does the code support that?

Is the code lint free?

Can results be recreated and verified automatically?

Followup

Workshop Repository

“ML Engineering” Slack Workspace?

“Live Coding” online sessions?

Project templates?

Other ideas??