Glorious Mistakes

25 years of Python Errors

Daniel Roy Greenfeld

Glorious Mistakes

25 years of Python Errors

Daniel Roy Greenfeld

Hello Lithuania!

Thank you!

PyCon LT 2024

Let's them a cheer!

PyCon LT 2024

Lithuania: First time here

Hello Lithuania!

My family left

100+ years ago

We're back!

We ❤ Lithuania

My mother loves Šaltibarščiai[1]

Hi Mom!

[1] shalt-eh-barsh-chay

Let's do this!

Glorious Mistakes

25 years of Python Errors

Daniel Roy Greenfeld

About these errors

Some are mine

Some I witnessed

Some I fixed

Some I'm fixing

Some are ongoing

1.
Missed opportunity of lifetime

1999

1999

Scheduled an interview

At a  company!

1999: Could have worked at Zope Corp

However,

this is 1999

is a weird, obscure language

1999: Real coders use Java Enterprise!

Big companies excited about Java

Unnecessary complexity was a virtue

XML everywhere!

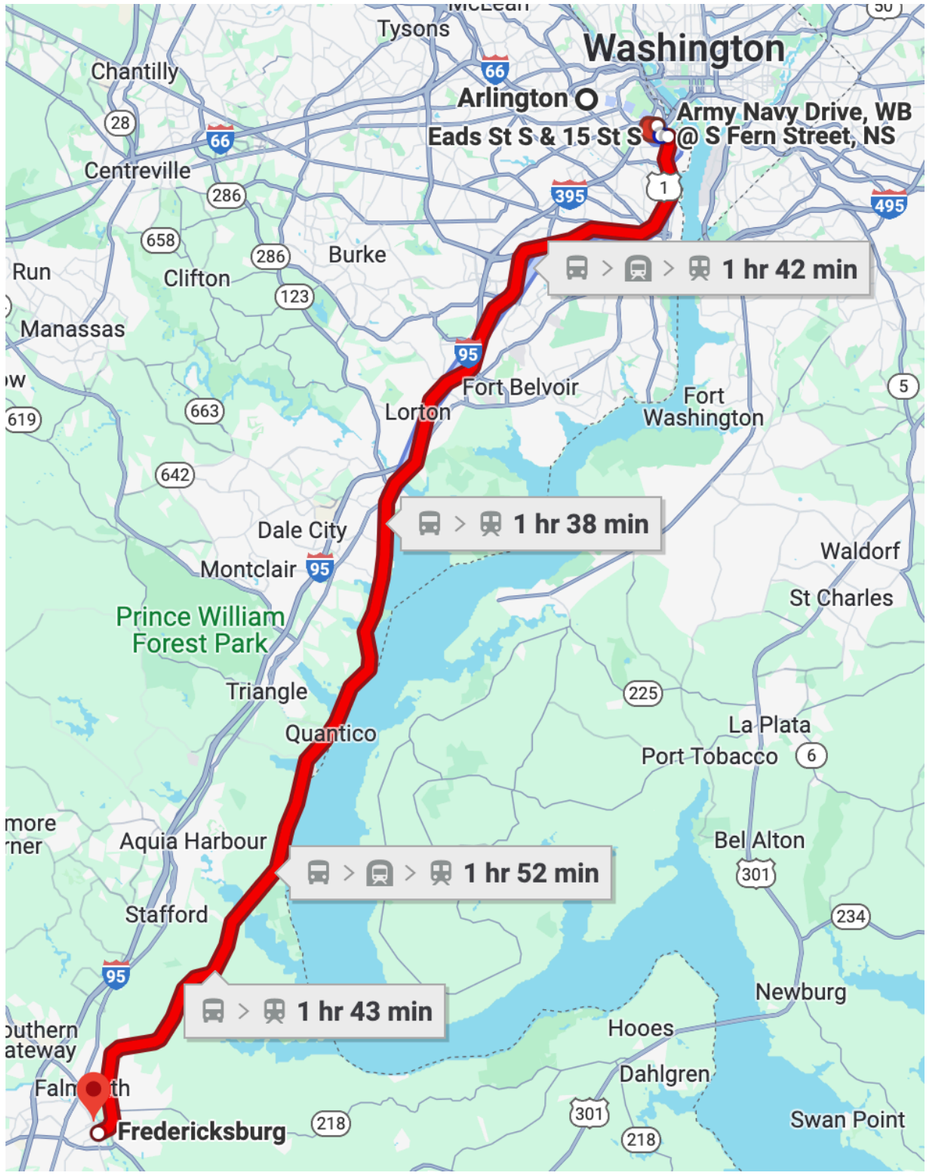
JAKARTA® EE

Senior dev:

"Python is a MISTAKE"

Senior dev:

"Stick with Java"


Washington
Tysons
Chantilly
Arlington
Army Navy Drive, WB
Eads St S & 15 St S
@ S Fern Street, NS
Centreville
Burke
Clifton
Manassas
Fort Belvoir
Lorton
Fort Washington
Dale City
Montclair
Prince William Forest Park
Triangle
Quantico
Waldorf
St Charles
La Plata
Port Tobacco
Aquia Harbour
Bel Alton
Stafford
Newburg
Hooes
Dahlgren
Swan Point
Fredericksburg
1 hr 42 min
1 hr 38 min
1 hr 52 min
1 hr 43 min

I cancelled the interview!


What did I miss
by cancelling an interview?

Zope was hiring most of the Python devs

Including Guido van Rossum in 2000!

At the start of my career I lost the chance to work with Guido van Rossum

At a time when Python was so new Guido (and others experts there) spent untold hours lifting up people like me.

1.
Missed opportunity of lifetime

Lessons Learned

1. Always go to the interview!
2. Don't listen to people who dismiss Python

daniel.feldroy.com/posts/2010-01-my-pre-history-with-plone

Second Mistake

2.

Unnecessary Sophistication

Late 1990s to 2008-ish

The Age of Zope

What is Zope?

What is Zope?

A consulting company

What is Zope?

A pioneering web framework

What is Zope?

A pioneering web framework

Invented URL routing

```
$ tree .
├── accounts.cfm      ========>
├── index.php
├── users.pl
```

```
/accounts/
/index/
/users/
```

What is Zope?

A pioneering web framework

Built-in admin

What is Zope?

A pioneering web framework

NoSQL Database

- A pickled, indexed Python object!
- Powered by B-Trees!

What is Zope?

A pioneering web framework

Even Typing!

Plus many more features

Zope dominated Python

Zope dominated Python

- Zope was synonymous with Python web

Zope dominated Python

- Zope was synonymous with Python web
- Early PyCon US had a Zope track

Zope dominated Python

What happened to Zope?

What happened to Zope?

Created in the era where complexity equaled value

Too complex:

Any problem can stop development

Zope Hiring Story at NASA

In 2008 at NASA my team rejected a candidate in part because he made the impossible claim to have learned, coded, and deployed a Zope-powered system within a month.

Example Zope Gotcha

Impossible claim: Complexity made quirks harder

```
from datetime.datetime import DateTime
from zope.schema import Datetime
from zope.schema import TextLine
```

daniel.feldroy.com/posts/2007-11-head-meets-desk

Example Zope Gotcha

Impossible claim: Complexity made quirks harder

```
from datetime.datetime import DateTime
from zope.schema import Datetime  # <-- Titlecase
from zope.schema import TextLine
```

daniel.feldroy.com/posts/2007-11-head-meets-desk

Example Zope Gotcha

Impossible claim: Complexity made quirks harder

```
from datetime.datetime import DateTime
from zope.schema import Datetime  # <-- Titlecase
from zope.schema import TextLine  # <-- Camelcase
```

daniel.feldroy.com/posts/2007-11-head-meets-desk

XML was in fashion: Making things public using Zope[3]

```
<browser:page name="captcha.wav"
  for="captchad.interfaces.ICaptchaContainer"
  class="captchad.browser.folder.CaptchaAudio"
  permission="zope.Public" />

<browser:page
  name="captcha.png"
  for="captchad.interfaces.ICaptchaContainer"
  class="captchad.browser.folder.CaptchaImage"
  permission="zope.Public" />
```

[3] https://daniel.feldroy.com/posts/2007-11-im-serving-out-image-and-audio-files

Zope

```
<browser:page name="captcha.wav"
  for="captchad.interfaces.ICaptchaContainer"
  class="captchad.browser.folder.CaptchaAudio"
  permission="zope.Public" />

<browser:page
  name="captcha.png"
  for="captchad.interfaces.ICaptchaContainer"
  class="captchad.browser.folder.CaptchaImage"
  permission="zope.Public" />
```

Django CBVs

```
class PublicCaptchaView(
        DetailView
    ):
    # Public by default


class PrivateCaptchaView(
        DetailView,
        LoginRequiredMixin
    ):
    # Private with mixin
```

What happened to Zope?

Easy things were hard, hard things were nigh impossible

Proficiency with Zope required:

A large investment of time

- Expedited via EXPENSIVE paid instruction

Unfortunately common Zope dialogue

Unfortunately common Zope dialogue

1. Coder: "This is really hard and the docs are incomplete"

Unfortunately common Zope dialogue

1. Coder: "This is really hard and the docs are incomplete"
2. Contributor: "Take my paid class to understand it"

Unfortunately common Zope dialogue

1. Coder: "This is really hard and the docs are incomplete"
2. Contributor: "Take my paid class to understand it"
3. Coder: "Can we not use XML so much?"

Unfortunately common Zope dialogue

1. Coder: "This is really hard and the docs are incomplete"
2. Contributor: "Take my paid class to understand it"
3. Coder: "Can we not use XML so much?"
4. Contributor: "Here's more complexity and XML!"

Unfortunately common Zope dialogue

1. Coder: "This is really hard and the docs are incomplete"
2. Contributor: "Take my paid class to understand it"
3. Coder: "Can we not use XML so much?"
4. Contributor: "Here's more complexity and XML!"
5. Coder: "Bah! I'm going to use something else"

The community drifted away

Django

New community members went right to Django

Lessons learned

1. Complexity as a business model is bad
2. Overcomplicating projects to sell training and consulting and ensure job security can backfire

Third Mistake

3.

Overly clever hacks

My obnoxiously clever hack

I defined my own numeric variables called `true` and `false`

```
# business/logic.py
true = 0
false = 1
```

Almost-but-not the breaks Python true/false contract

```
true = 0                    # falsy
False, [], ""   , {}        # falsy

false = 1                   # truthy
True, [1], "Hi", {1: 2} # truthy
```

Almost-but-not the breaks Python true/false contract

```
true = 0                        # falsy
false = 1                       # truthy

False, [], ""   , {}        # falsy
True, [1], "Hi", {1: 2} # truthy
```

Obnoxiously Clever!

```
from business import logic, true, false

def process_result(vendorResponse):
    result = None
    for res in vendorResponse.results:
        if res.status == false:
            result = logic(res)
        if result.is_valid == True
            break
    return result
```

My code confused everyone around (including me)

```
from business import logic, true, false

def process_result(vendorResponse):
    result = None
    for res in vendorResponse.results:
        if res.status == false:  # Truthy value
            result = logic(res)
        if result.is_valid == True # Truthy value
            break
    return result
```

My code confused everyone around (including me)

```
from business import logic, true, false

def process_result(vendorResponse):
    result = None
    for res in vendorResponse.results:
        if res.status == false: # no code highlighting!
            result = logic(res)
        if result.is_valid == True # Code highlighting!
            break
    return result
```

What I broke

by reversing booleans

What I broke by reversing booleans

Production

Repeatedly

Production

So bad

Production

It took me 17 years to admit this error

Production

A better approach

```
class VendorStatus(Enum):
    POSITIVE: int = 0
    NEGATIVE: int = 1
```

This mistake...

```
from business import logic, true, false

def process_result(vendorResponse):
    result = None
    for res in vendorResponse.results:
        if res.status == false: # WTF was I doing?
            result = logic(res)
        if result.is_valid == True
            break
    return result
```

Becomes a better approach

```
from business import logic, VendorStatus

def process_result(vendorResponse):
    result = None
    for res in vendor_response.results:
        if res.status == VendorStatus.NEGATIVE:
            result = logic(res)
        if result.is_valid == True
            break
    return result
```

A better approach

```
from business import logic, VendorStatus

def process_result(vendorResponse):
    result = None
    for res in vendor_response.results:
        if res.status == VendorStatus.NEGATIVE:
            result = logic(res)
        if result.is_valid == True
            break
    return result
```

What I learned:

What I learned:

1. Python code should be intuitive

What I learned:

1. Python code should be intuitive
2. Code shouldn't require three cups of coffee to be understood

What I learned:

1. Python code should be intuitive
2. Code shouldn't require three cups of coffee to be understood
3. The business logic might be complicated but your code shouldn't be

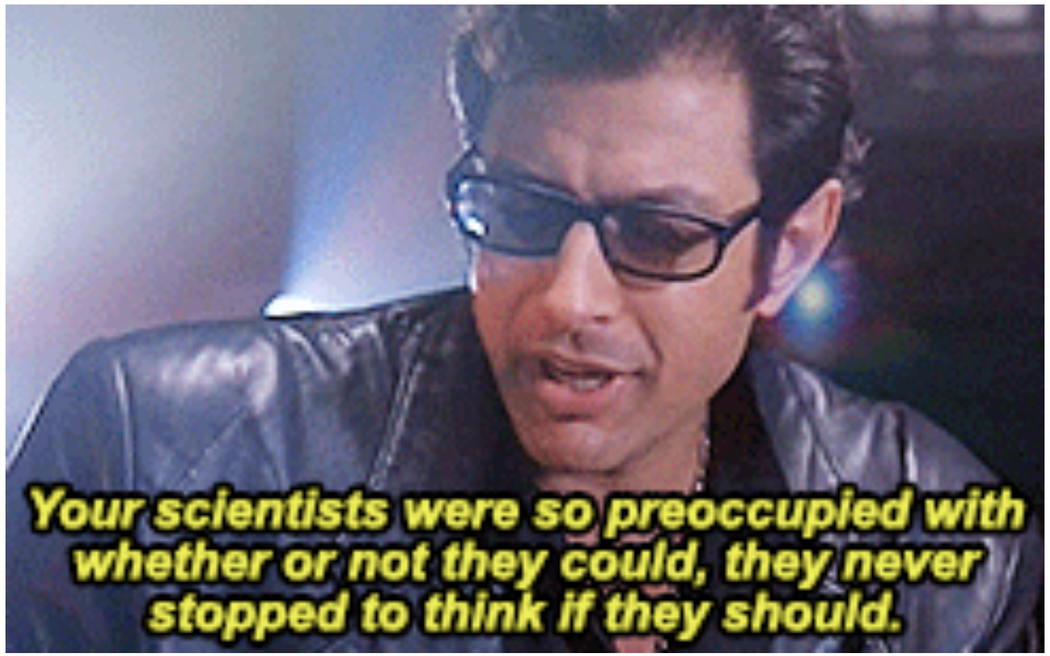
Fourth Mistake

4.

Django Cleverness

2012-era Django

Dr. Ian Malcolm, Mathematician/
Chaotician @ Jurassic Park



Cleverness of the era

Mistake: Database configuring installed apps

```
--  SQL to fetch which app to load
select app_path from apps where app_id = {environ['APP_ID']}";

# Dynamically loaded INSTALL_APPS
INSTALLED_APPS += [x['app_path'] for x in records]
```

Better approach:

Build your own plugin system

Mistake: Removing None

Null pointer exceptions are a billion dollar mistake.

-- Tony Hoare

Mistake: Removing None

(I'm not going to argue with the inventor of Null Pointers)

Mistake: Removing None

However, *replacing None in Python is questionable*

```
class NullPointerHandler:
    # 50+ lines of code goes here
```

Mistake: Removing None

`NullPointerHandler` decorates/wraps:

- Every Python stdlib that could return None

Mistake: Removing None

`NullPointerHandler` decorates/wraps:

- Every PyPI package that could return None

Mistake: Removing None

- OMG `NullPointerHandler` touches everything!

Mistake: Removing None

1. Edge case nightmares!
2. Curious bottlenecks

Better approach:

If coding Python, don't worry about Null Pointer Exceptions

Mistake: Cache-powered Django concrete models

Child SQL tables with all of the parent table data

Significant performance overhead

Mistake: Cache-powered Django concrete models

Child SQL tables with all of the parent table data

Significant performance overhead

- "Solution": Caching to handle data duplication

Mistake: Cache-powered Django concrete models

Child SQL tables with all of the parent table data

Significant performance overhead

- "Solution": Caching to handle data duplication
- But cache invalidation is hard

Mistake: Cache-powered concrete models

- Caching to handle data duplication
- But cache invalidation is hard

"There are only two hard things in Computer Science: cache invalidation and naming things."

— Phil Karlton

Better approach:

Avoid concrete models

Mistake: Hack MongoDB into Django

Django is designed for relational databases

MongoDB is a document datastore without relations

Mistake: Hack MongoDB into Django

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)
- Clever code addresses 80-95% of Django!

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)
- Clever code addresses 80-95% of Django!
- But..

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)
- Clever code addresses 80-95% of Django!
- But..
  - Often breaks most of the third-party package ecosystem

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)
- Clever code addresses 80-95% of Django!
- But..
  - Often breaks most of the third-party package ecosystem
  - Edge case insanity, 5-20% of Django remains

Mistake: Hack MongoDB into Django

- Easy to get started! (Django-NoSQL, Djongo, etc)
- Clever code addresses 80-95% of Django!
- But..
  - Often breaks most of the third-party package ecosystem
  - Edge case insanity, 5-20% of Django remains
  - Maintainance issues (Django-NoSQL, Djongo, etc)

Better approach:

Use MongoDB with something besides Django[m]

[m] https://daniel.feldroy.com/posts/when-to-use-mongodb-with-django

So much cleverness!

That was sarcasm

So much complexity

The reality

Many Django

Failed

Most of our 2012 work was underpaid Django rescue work

Most of our 2012 work was underpaid Django rescue work

Devs coded with too much cleverness

Most of our 2012 work was underpaid Django rescue work

Devs leave because stuck and funds mostly gone

Most of our 2012 work was underpaid Django rescue work

We get invited to fix the problems on tiny budgets

(money was paid to devs who left messes)

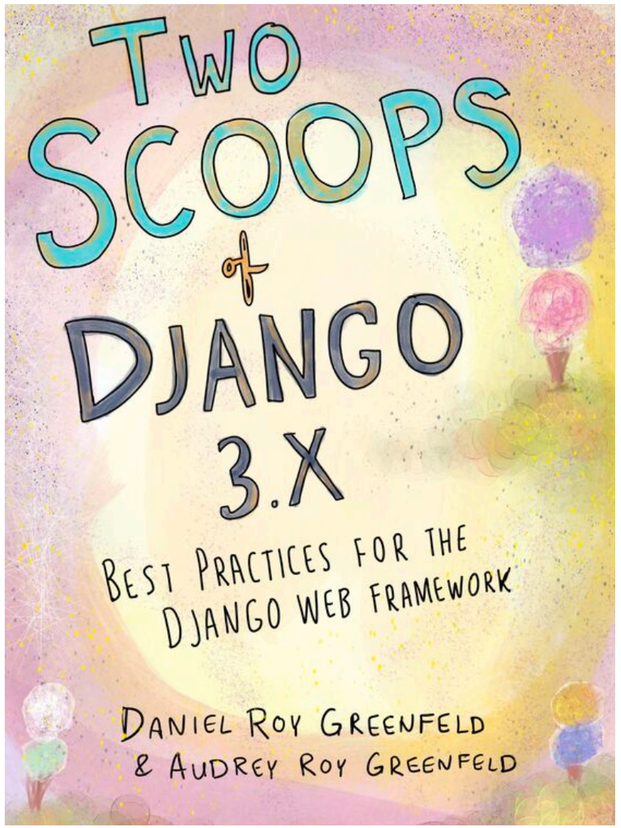
If only people stopped the clever coding!

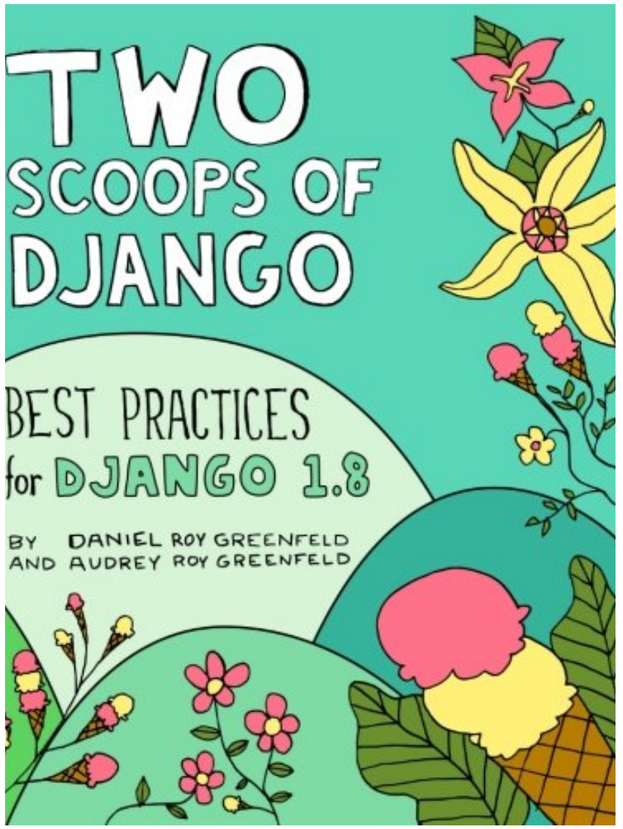
There needs to be a guide

to keep things reasonable

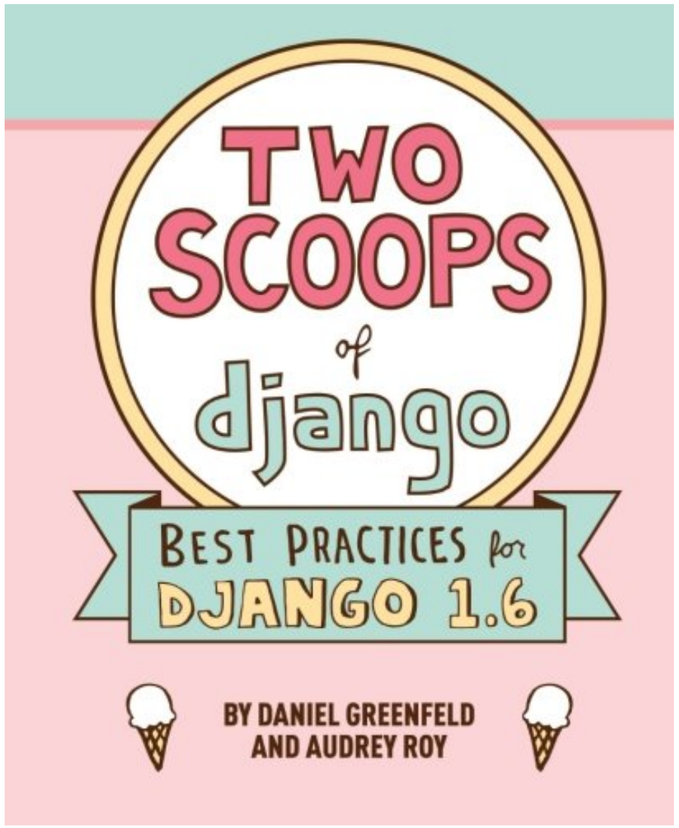
Let's talk it over ice cream!

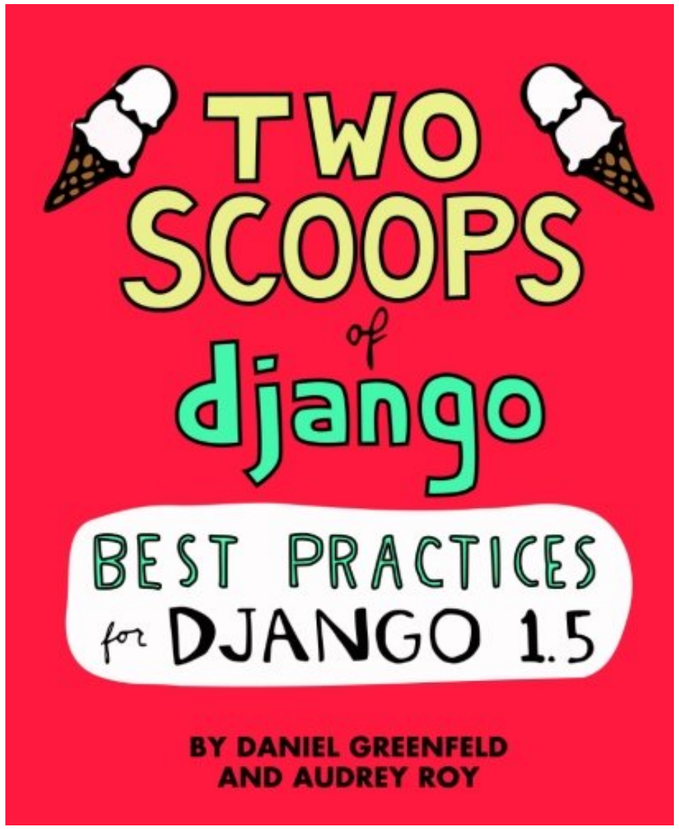
Two Scoops of Django





Two Scoops of Django





Lesson Learned

Lesson Learned

1. We can address problems by writing and coding

Lesson Learned

1. We can address problems by writing and coding
2. Encouraging simplicity over complexity

Lesson Learned

1. We can address problems by writing and coding
2. Encouraging simplicity over complexity
3. Business can be complex, your code shouldn't be

Let's move on...

5.

Modern Complexity

Python Types

Problem or not?

"I like types but they aren't always good"

David Seddon

For control of behaviors Types are bliss

- FastAPI
- Typer
- pydantic

```python
from pathlib import Path
import typer

def main(
        year: int,
        path: Path
    ):
    print(year)
    print(path.read_text())

if __name__ == "__main__":
    typer.run(main)
```

For control of behaviors Types are bliss

- FastAPI
- Typer
- pydantic

```python
from pathlib import Path
import typer

def main(
        year: int,
        path: Path
    ):
    print(year)
    print(path.read_text())

if __name__ == "__main__":
    typer.run(main)
```

For control of behaviors Types are bliss

- FastAPI
- Typer
- pydantic

```python
from pathlib import Path
import typer

def main(
        year: int,
        path: Path
    ):
    print(year)
    print(path.read_text())

if __name__ == "__main__":
    typer.run(main)
```

For control of behaviors Types are bliss

- FastAPI
- Typer
- pydantic

```python
from pathlib import Path
import typer

def main(
        year: int,
        path: Path
    ):
    print(year)
    print(path.read_text())

if __name__ == "__main__":
    typer.run(main)
```

Types can add significantly complexity

Types can add significantly complexity

- Yes, straightforward types improve clarity

Types can add significantly complexity

- Yes, straightforward types improve clarity
- It's the edge cases that hurt

Types can add significantly complexity

- Yes, straightforward types improve clarity
- It's the edge cases that hurt
- Django is an edge case for Python types

Types can add significantly complexity

- Yes, straightforward types improve clarity
- It's the edge cases that hurt
- Django is an edge case for Python types
  - Django-Stubs doesn't cover everything yet

Types are decent at catching bugs

But

Saying:

"Types catch bugs before they reach production so we can stop writing so many tests"

Is the same as:

"The ORM means our security is perfect" [+]

[+] Crypto-dudes at hackathon in 2014

ORM detail:

Helps protect against SQL injection

Does't block bogus information in valid SQL

Mypy is slow

Mypy is slow

(Yes, I know there is a daemon)

1. If no cache, 93 lines of code takes me 5s
2. What if I'm testing millions of lines of Python?

Upgrading mypy can be really hard

Pro-tip:

keep mypy up-to-date

and cleverness

I love protocols but admit for some this might be 'clever'

```python
from typing import Protocol

class Child(Protocol):
      name: str

      def play(self, game): ...

class Uma:
      name: int = 5

      def play(self, game):
        # Playtime goes here

def playground(child: Child) -> None: ...
```

```python
playground(Uma())

# error: Argument 1 to "playground" has
# incompatible type "Uma"; expected "Child"
    # note:  Following member(s) of "Uma" have conflicts:
    # note:      name: expected "str", got "int"
```

Remember Zope?

Zope had Types

Zope.interface preceded typing.Protocol[6]

```python
from zope.interface import Interface, Attribute

class IChild(Interface):

    name = Attribute("Name of person")

    def play(game):
        """Do some play"""
```

```python
from zope.interface import implementer

@implementer(IChild)
class Child:

    name = 'Uma'

    def play(self, game):
        return work.start()
```

[6] https://peps.python.org/pep-0544/

Question:

Are Python types the Zope of the 2020s?

Problems with Python types

Problems with Python types

- Adding types to everything forces in the edge cases

Problems with Python types

- Adding types to everything forces in the edge cases
- Cleverness required to get mypy to pass on some projects

Problems with Python types

- Adding types to everything forces in the edge cases
- Cleverness required to get mypy to pass on some projects
- Typechecking is slow

Are Python types the Zope of the 2020s?

I don't think so

Unlike

...typing in Python has escape hatches

```
from tying import Any

MyModel.name == generic_value  # type: ignore
```

Lessons learned:

1. Escape hatches are good
2. If types are blockers, you can get past the problem

"Desperate times call for desperate measures"

Final error

6.

Not using Python for Good

My work used for good

- People using it to land jobs and build dream projects
- Various community service efforts
  - Health research projects
  - Charities and non-profits

My work used for evil

My work used for evil

My work used for evil

- Oppression

My work used for evil

- Oppression
- Commercial ventures to spy on private citizens

My work used for evil

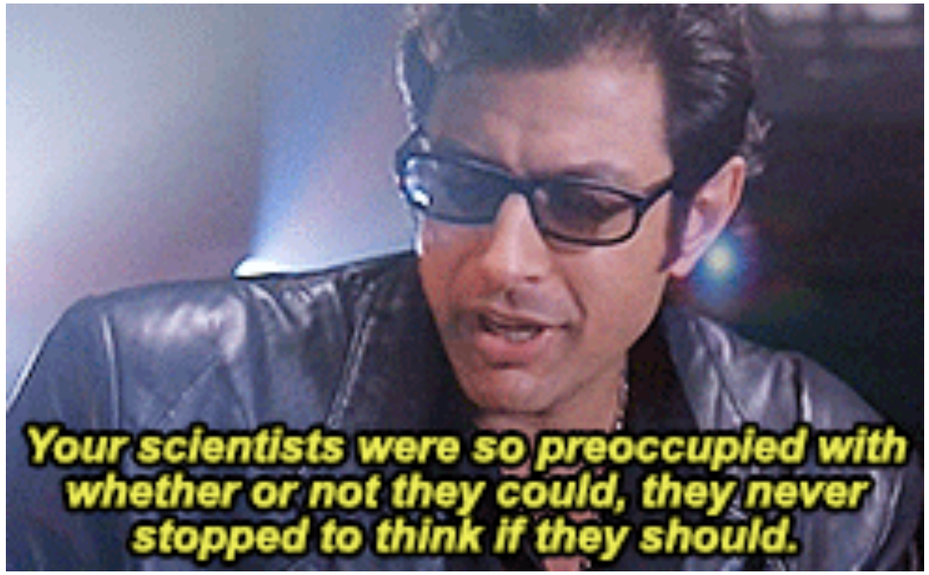
- Oppression
- Commercial ventures to spy on private citizens
- Promotion of stupidity: flat earth / no moon landing theories

My work used for evil

- Oppression
- Commercial ventures to spy on private citizens
- Promotion of stupidity: flat earth / no moon landing theories
- App to sell wildlife illegally

Ethics in Code

Dr. Ian Malcolm,
Mathematician/Chaotician
@ Jurassic Park



Questions we should ask of every employer/client

Who is helped?

Who is hurt?

Does this project:

- Promote falsehood over truth?
- Subvert election processes?
- Intrude on individual sovreignty?
- Threaten the planet?

In other words, does it help the future?

Let's try this on:

Climate Change

Climate change

- A series of incrementally hotter years
- Heat records broken annually
- Droughts in normally wet places

Where is this? Southern California?

No - This is Auburge, France in 2022

It doesn't have to be this way...

...and here's why

Lessons learned thanks to our books:

- Programming gives us an outsized impact on the world
- We can make a difference


PROGRAMMING
gives you
SUPER
POWERS

Programming Gives Us Super Powers

To improve the lives of ourselves and our families

Programming Gives Us Super Powers

To build new and innovative businesses

Programming Gives Us Super Powers

To share ideas and knowledge around the world


PROGRAMMING
gives you
SUPER
POWERS

The power
to save the planet

My impact on climate as an individual is tiny[7]

[7] I am responsible for myself and ~2 other people

My climate impact as a software developer working on decarbonization and electrification

Gigantic

I am responsible for approximately

60,000

people on renewable energy

You may ask

"isn't flying here contemptuous of my cause?"

If I can convince 1+ audience members to join my mission

Their impact might be similar or greater than my own

60,000

people on renewable power

This is going to be my legacy

Join me in electrifying the world!

- Solar
- Wind
- Batteries

climatebase.org

Jobs for all nations

octopus.energy/careers

KRAKENTECH

PART OF THE octopus energy GROUP

My employer, not available in Lithuania yet

What will be your legacy?

Making a billionaire another billion?

Or

Improving the world?

About Me

Daniel Roy Greenfeld

Software Artisan @ Kraken Tech

Coder, Author @ feldroy.com

Husband of Audrey Roy Greenfeld

Father of Uma

Superhero saving the planet

feldroy.com

Any Questions ?

feldroy.com

climatebase.org

Jobs for all nations

octopus.energy/careers

KRAKENTECH

PART OF THE octopus energy GROUP

My employer, not available in Lithuania yet