Things I've Learned from Being Thrown Into Python and Django

Ryan Scott

Introduction

- Software Engineer @ Percipient Networks
- Former Node.js and Salesforce developer
- Several years using Python and Django for fun
- 2 months using Python and Django professionally

Problem:

How do you quickly find the documentation of an object?

Python Standard Library: `help()`

- Print the help text for a module, function, class, method, keyword, etc.
- This is why we write documentation!
- Stay in the terminal, no need to use Google
- Also check out `dir()`

Python Standard Library: help()

```
>>> import requests
>>> help(requests)
Help on package requests:
...
DESCRIPTION
    requests HTTP library
    ~~~~~~~~~~~~~~~~~~~~~

    Requests is an HTTP library, written in Python, for human beings. Basic GET
    usage:

       >>> import requests
       >>> r = requests.get('https://www.python.org')
       >>> r.status_code
       200
       >>> 'Python is a programming language' in r.content
       True
...
```

Problem:

How do you easily initialize a dictionary?

```
>>> colors = ['yellow', 'blue', 'red', 'blue',
              'blue', 'yellow', 'green', 'red', 'yellow']

>>> color_count = {}
>>> for color in colors:
...     if color not in color_count:
...         color_count[color] = 1
...     else:
...         color_count[color] += 1
>>> color_count
{'blue': 3, 'green': 1, 'red': 2, 'yellow': 3}
```

Python Standard Library: `defaultdict`

- Solves the common problem of having to initialize a dictionary before using it
- Only saves a couple lines of code, but this can be huge for readability in complex loops!

Python Standard Library: `defaultdict`

```
>>> colors = ['yellow', 'blue', 'red', 'blue',
              'blue', 'yellow', 'green', 'red', 'yellow']

>>> from collections import defaultdict
>>> color_count = defaultdict(int)
>>> for color in colors:
...     color_count[color] += 1
>>> color_count
defaultdict(<type 'int'>, {'blue': 3, 'green': 1, 'red': 2, 'yellow': 3})
```

Problem:

How do you efficiently iterate over a massive or infinite list?

Generator functions

- Easily iterate through a massive or infinite list
- Avoid pulling the whole list into memory

Generators functions

```
>>> def fib(max):
...     a, b = 0, 1
...     for i in xrange(max):
...         yield a
...         a, b = b, a + b

>>> for num in fib(10):
...     print num
# Outputs the first 10 Fibonacci numbers
```

Problem:

How do you make it easy to onboard developers??

and

How do you simplify repetitive tasks?

Fabric

- Command-line tool for executing local and remote shell commands
- Don't Repeat Yourself for development tasks
- Make development easy! (Especially when using a virtual machine)

Fabric example #1

Reset your development database:

```
@task
def db_reset(db_name="dashboard"):
    """Delete and recreate the database."""
    drop(db_name)
    create()
    createcachetable()
    migrate()
    load_fixture_data()
    createinitialrevisions()
```

```
$ fab dev db_reset
# Database data has been reset
```

Fabric example #2

```python
@task
def runserver():
    """Run the Django runserver command which autoreloads on file changes."""
    with common.setup_venv(), common.no_prefix():
        run('python manage.py runserver_plus')
```

```
$ fab dev runserver
# Calls 'python manage.py runserver_plus' on dev environment

$ fab staging runserver
# Calls 'python manage.py runserver' on staging environment
```

Problem:

How do you improve developer experience in the shell and when debugging?

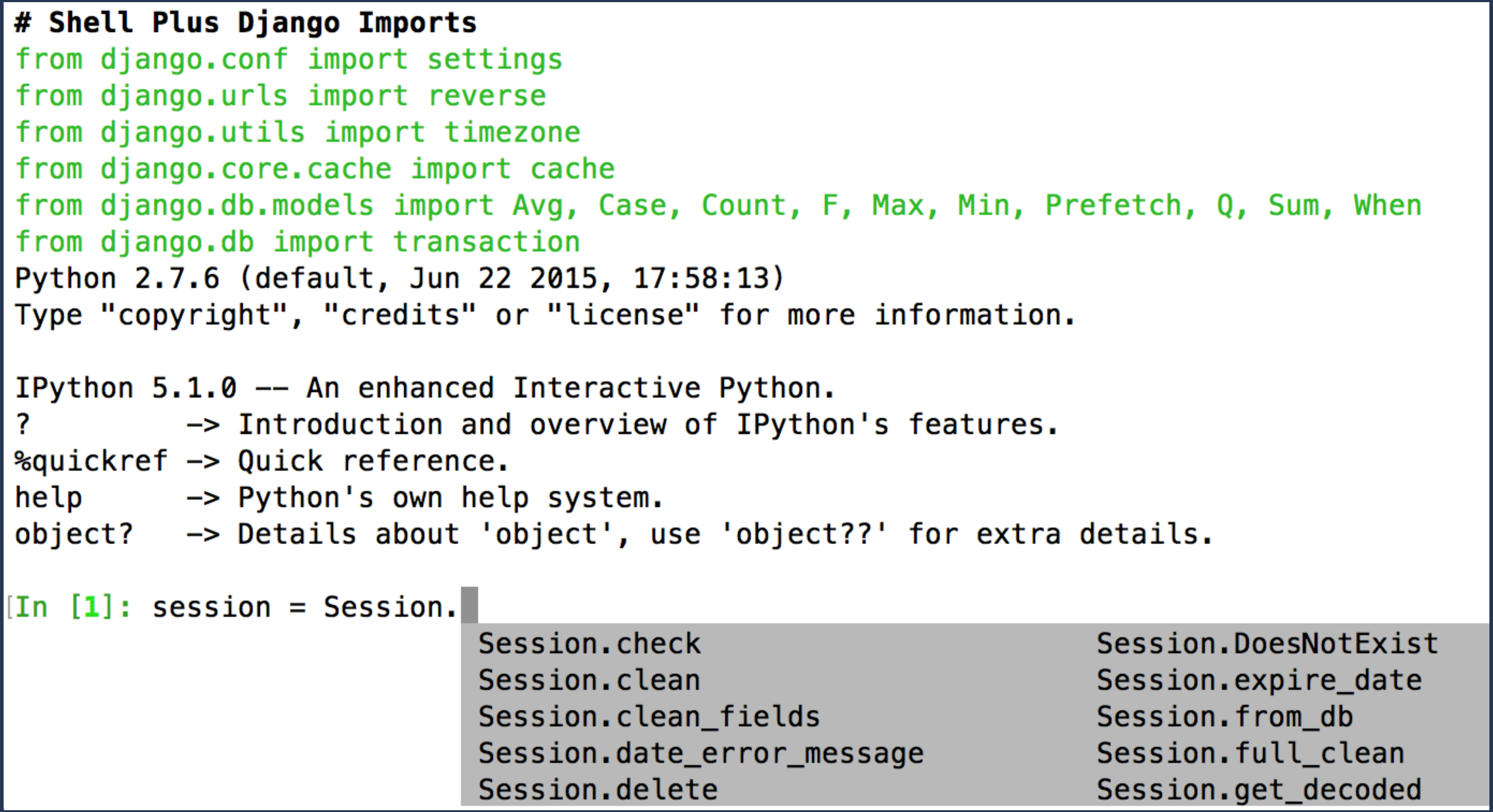
Django Extensions

- Collection of custom extensions for Django
- https://django-extensions.readthedocs.io/
- My favorites:
  - `shell_plus`
  - `runserver_plus`

Django Extensions: `shell_plus`

- Django shell improvement
- Autoloads the app's database models
- Save time, Don't Repeat Yourself!

Django Extensions:
`shell_plus`

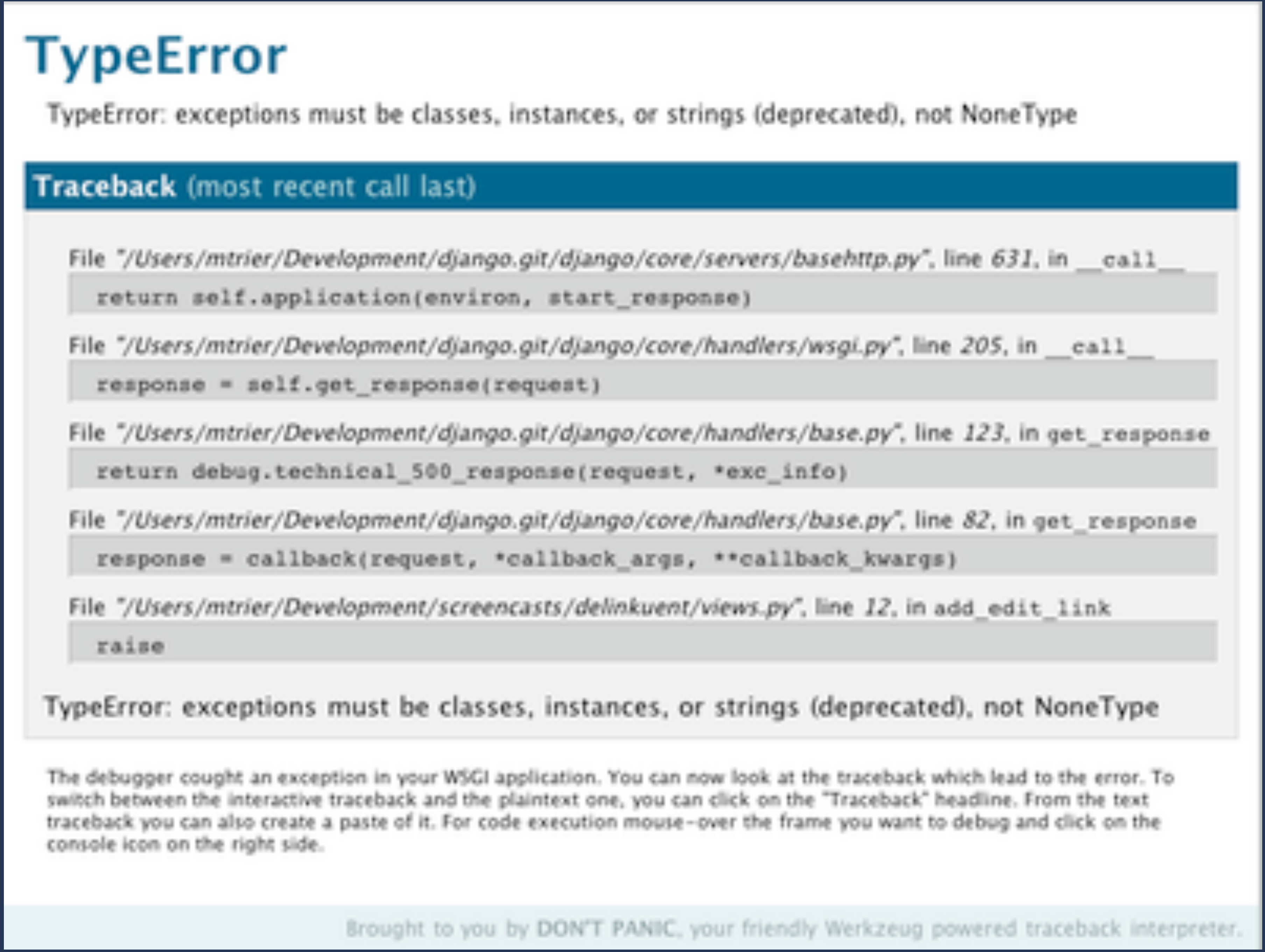


```
# Shell Plus Django Imports
from django.conf import settings
from django.urls import reverse
from django.utils import timezone
from django.core.cache import cache
from django.db.models import Avg, Case, Count, F, Max, Min, Prefetch, Q, Sum, When
from django.db import transaction
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: session = Session.
                           Session.check                      Session.DoesNotExist
                           Session.clean                      Session.expire_date
                           Session.clean_fields               Session.from_db
                           Session.date_error_message         Session.full_clean
                           Session.delete                     Session.get_decoded
```

Django Extensions: `runserver_plus`

- Django runserver **plus** the Werkzeug debugger
- Improved traceback page during exceptions
- Adds an interactive debugging console to evaluate code on the server

Django Extensions: `runserver_plus`



TypeError

TypeError: exceptions must be classes, instances, or strings (deprecated), not NoneType

Traceback (most recent call last)

```
File "/Users/mtrier/Development/django.git/django/core/servers/basehttp.py", line 631, in __call__
  return self.application(environ, start_response)
File "/Users/mtrier/Development/django.git/django/core/handlers/wsgi.py", line 205, in __call__
  response = self.get_response(request)
File "/Users/mtrier/Development/django.git/django/core/handlers/base.py", line 123, in get_response
  return debug.technical_500_response(request, *exc_info)
File "/Users/mtrier/Development/django.git/django/core/handlers/base.py", line 82, in get_response
  response = callback(request, *callback_args, **callback_kwargs)
File "/Users/mtrier/Development/screencasts/delinkuent/views.py", line 12, in add_edit_link
  raise
TypeError: exceptions must be classes, instances, or strings (deprecated), not NoneType
```

The debugger cought an exception in your WSGI application. You can now look at the traceback which lead to the error. To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

Thank you

github.com/percipient/talks

ryan@strongarm.io