Как я делаю свой фреймворк мутационного тестирования?

Евгений Блинов для Python Birthday Conf by Sber

Обо мне

Делаю разные питоновые штуки

Контакты

Гитхаб: github.com/pomponchik

Линкедин: linkedin.com/in/pomponchik

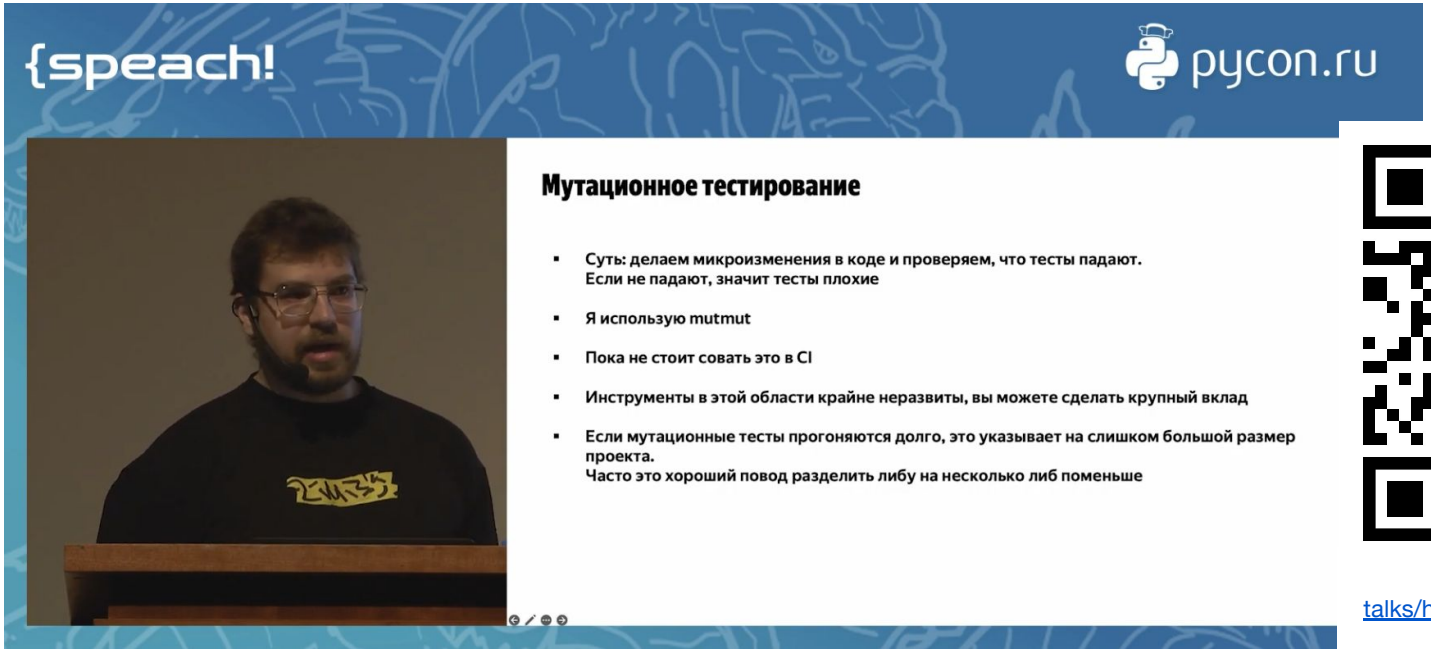
Личный сайт: pomponchik.org

Я + MT == 

Кто такой этот *мутационное тестирование*?

- Ломаем исходный код (например меняем + на -)
- Запускаем тесты
- Падают – хорошо
- Не падают – плохо
- Повторяем тысячи раз, получаем метрику: процент “убитых мутантов” (MSI)
- ИИ-агент ориентируется на метрику, и при необходимости дописывает недостающие тесты
- Проблема: эквивалентные мутанты

2024: PyCon



pomponchik.org/talks/how-to-test-libraries-in-python/

2026: бутылочное горлышко агентского кодинга

- Все используют LLM для генерации кода
- Тесты кажутся легкой целью
- Но программисты по факту все равно перечитывают весь код
- Тесты и их корректность – по-прежнему сложнейшая задача в программировании

> LLM пишет тесты

Мутационное тестирование – это решение 🔥🔥🔥

- Нам нужен хороший способ определять *автоматически*, хороши тесты или нет
- Покрытие – недостаточно точная метрика
- Мутационное тестирование выглядит *единственным способом* определить истинное качество тестов

ИИ *может* писать качественные тесты!

- ИИ-агент может запускать МТ на своих тестах и убеждаться в их качестве
- Можно строить агентские системы, которые покрывают качественными тестами ваш код
- Систему МТ можно использовать как источник обратной связи при обучении LLM с подкреплением

Ограничения

- Эта штука не выявляет баги, несоответствие спеке
- Она проверяет только то, что юнит-тесты *что-то* проверяют
- Не заменяет другие виды тестирования
- Все еще дорого по компьюту
- Должно применяться после всех других видов тестирования

Так думаю не только я

- Вообще идея мутационного тестирования старая, из 70-х
- Крупные компании типа Google и Meta внедряют в продакшен массовые проверки с помощью огромных мутационных систем
- Однако опенсорсный тулинг *для питона* все еще не очень
- В некоторых языках все обстоит гораздо лучше (например, JS, C#, Java)

Google

- На 2021 год: активно используется более чем 24 000 инженеров, проверяются все PRы
- Система встроена в продакшен
- Все крутится вокруг дешевых инкрементальных проверок

research.google/pubs/practical-mutation-testing-at-scale-a-view-from-google/

Meta (запрещено в РФ)

- Активно мутируют
- Интегрируют с LLM: нейронка как ломает код, так и пишет тесты

engineering.fb.com/2025/09/30/security/llms-are-the-key-to-mutation-testing-and-better-compliance/

Короче!

- Технологии созрели для массового внедрения
- Актуальность резко выросла после появления кодинг-агентов
- Нам всем нужно MT!

Поэтому я решил:

ПОРА

Я отсмотрел много вариантов и мне ни один не зашел

()

Мутационное тестирование, готовое к продакшену

- Что нужно для продакшена?
  - Простота и понятность!
  - Быстрый запуск, параллелизм между машинами
  - Легкий способ встраивать в CI, настраивать “куалити-гейты”
  - Возможность легко настроить под себя, дописать недостающие мутационные операторы
  - Широкий набор мутационных операторов “из коробки”
  - Интеграция с LLM с двух сторон

Что я решил сделать:

- Параллелизм из коробки:
  - Локально на тачке
  - В облаке
  - Не привязанный к pytest как в mutmut
- Гибкая плагинная система:
  - Возможность легко добавить мутации
  - Большой каталог доступных мутаций
  - LLM как источник мутаций
- Все максимально просто, готово к интеграции с существующими инструментами

Как я это делаю?

Мои принципы:

- Максимальная модульность
- Разбиение продукта на подлибы
- Вертикальная интеграция
- Собственная плагинная система

Пример компонента: metacode

- Многие тулзы используют машиночитаемые комментарии
- Мне это нужно, чтобы мьютить мутации
- Я заметил, что нет общего стандарта вокруг этого
- Предложил свое решение
- Пример максимальной модульности

github.com/mutating/metacode

Следите за анонсами!

Ссылки:

- Гитхаб: github.com/mutating
- Сайт: mutating.tech

За красивую тему для презентации отдельная благодарность компании Evrone