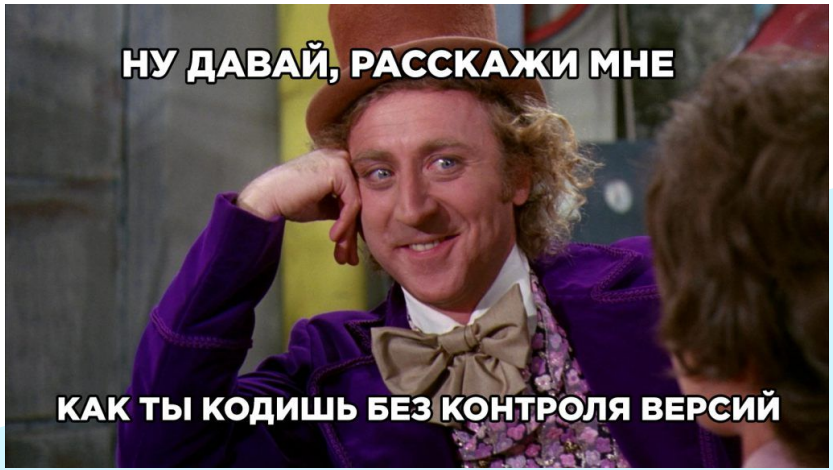
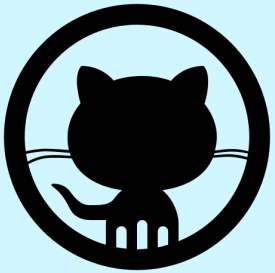
Git





Преимущества использования Git в индивидуальной и командной разработке

Индивидуальная работа:

- ➔ Гарантия доступности рабочей версии проекта;
- ➔ Безболезненный перенос проекта между устройствами;
- ➔ Переключение между разными версиями проекта.

Командная работа:

- ➔ Меньше опасность нежелательных последствий изменений, сделанных разными участниками команды;
- ➔ Удобное объединение результатов работы разных людей в единую версию.

Материалы

- https://try.github.io/ -- полезные ресурсы, в том числе и интерактивные.
- https://guides.github.com/ -- сборник мини-статей.
- https://chris.beams.io/posts/git-commit/ -- как правильно писать commit message.

Основные команды

- branch new_branch -- создаёт новую ветку new_branch
- checkout new_branch -- переключается на ветку new_branch
- pull -- стягивает изменения с origin (github или gitlab репозитория)
- add -- добавляет изменения к будущему коммиту
- commit -- делает коммит, то есть фиксирует произведенные (и добавленные) изменения
- stash -- откатывает сделанные изменения к последнему коммиту (позже их можно будет восстановить)
- push -- отправляет изменения в origin

Git: commit message

“Нормально, если на commit message потрачено больше времени, чем на код” -- совет от Senior Developer из Google

commit message -- это текстовое пояснение к коммиту.

Как делать НЕ нужно:

- ➔ писать бессмыслицу “lvnsdlfvw”;
- ➔ делать коммиты как можно реже, копить много изменений;
- ➔ писать о том, что вы сделали (“улучшил такой-то класс”). Вместо этого напишите, как изменения повлияли на проект.

Командная работа на Git. Как это делают в большинстве крупных компаний

● В master-ветке находится гарантированно рабочая версия. Обновления не должны ничего ухудшить.

● Существует отдельная ветка development для слияния работы разных людей. Версия этой ветки -- кандидат для вливания в master.

● В одной ветке работает только один человек.

● На каждую задачу отдельная ветка.

● По завершению задачи заводится merge request в ветку development.

● code review (просмотр и комментирование чужого кода) делают все желающие.

● merge request принимается только после закрытия всех вопросов, поднятых в code review.

Советы по организации пространства в репозитории

- data/ (каталог с данными, часто заносится в .gitignore)
- src/ (python-модули, основной каталог с кодом)
- notebooks/ (каталог с рабочими ноутбуками, заносится в .gitignore)
- reports/ (каталог с отчетами -- минималистичными ноутбуками)

● .gitignore (файл, указывающий git, какие каталоги и файлы следует игнорировать)

● README.md (общая документация по репозиторию)

● requirements.txt (файл, в котором перечислены зависимости, опционально)

● setup.py (файл, нужные для установки проекта, опционально)

Спасибо за внимание!