Домашка №1.1

- Создать локальный репозиторий со своим проектом с помощью DS-шаблона (e.g. cookiecutter или любой другой)
- Настроить gitignore
- Склонировать код в удаленный репозиторий (Github)
- Отработать github-flow:
    - Добавить новый код и закоммитить его в удаленный репозиторий
    - Создать новую ветку и выполнить в нее коммит
    - Создать еще одну ветку и выполнить коммит в нее
    - Выполнить слияние первой ветви и main с помощью fast-forward
    - Выполнить слияние второй ветви и main с помощью merge
    - Создать конфликт и разрешить
    - Настроить добавление кода в ветвь main только через pull request(merge request)
    - Добавить новый код **в мастер** через механизм pull request(merge request)

Домашка №1.2

- В проект из ДЗ № 1 установить линтер и тайп-чеке (выбрать на свое усмотрение, flake8 и mypy можно взять по умолчанию)
- Добавьте произвольный питонячий скрипт, на котором будет происходить отладка инструментов
- Установить в конфигах кастомные правила:
    - Длина строки не должна превышать 99 символов
    - В `__init__.py` файлах нужно игнорировать ошибку `imported but unused` (F401 flake8)
    - Тайп чекер должен игнорировать `__pycache__` папки
    - mypy должен работать с параметров `strict=True`
- Настроить pre-commit hook, который на каждом коммита проверяет
    - Стиль кода (flake8 или подобный)
    - Корректности типов (mypy или подобный)
- Добавить в `README.md` пункт с установкой окружения (создание окружения, установка зависимостей и пр. необходимые шаги); Другим вариантом может быть Makefile или bash сценарий, который выполняет все необходимые шаги.
- Не забывайте пользоваться ветками *(за пуш в мастер будет операция вычитания баллов)*

Алгоритм проверки домашки:

- Проверяющий клонирует репозиторий с проектом и создает окружение по инструкции из README.md (или с помощью запуска bash/Makefile сценария)
- Проверяющий в произвольном питонячем скрипте пытается испортить типы аргументов функции и код-стайл
- Проверяющий делает коммит; домашка считается корректно выполненной, если коммит не создался и всплыла ошибка линтера и тайп-чекера

Важные уточнения:

- Итогом вашей работы всегда должен быть коммит в мастере через механизм PR, который прошел все проверки линтеров.
- Каждая лаба – логическое продолжение предыдущей, все изменения должны подливаться в осноной репозиторий в мастер.
- Код стоит поддерживать под unix (macos, linux), адаптировать под windows не нужно
- Шаблон, который собирается в первой лабе призван унифицировать работу с проектом. Следуйте тем правилам, которые в нем описаны.