Лабораторная работа №1 (Модернизированная версия)

Тема: Организация рабочего окружения и работа с Git. Стратегии ветвления и автоматизация контроля качества. **Цель:** Приобрести практические навыки настройки среды разработки в Ubuntu, освоить базовые операции системы контроля версий Git, изучить стратегии ветвления Git Flow/GitHub Flow и настроить автоматизированный контроль качества с помощью pre-commit хуков. **Стек технологий:** Ubuntu, Python, Git, GitHub, IDE (VSCode), pre-commit. **Количество часов:** 4 академических часа. **Формат сдачи:** Ссылка на созданный GitHub-репозиторий, содержащий историю коммитов, созданные ветки, файл `README.md`, конфигурационный файл `.pre-commit-config.yaml` и скриншоты выполнения операций.

Теоретическая справка

- **Git** — распределенная система контроля версий, созданная Линусом Торвальдсом. Позволяет отслеживать изменения в коде, возвращаться к предыдущим версиям и организовать командную работу.
- **GitHub** — веб-сервис для хостинга IT-проектов и их совместной разработки с использованием Git.
- **Стратегии ветвления:**
  - **Git Flow:** Сложная модель с ветками `main`, `develop`, `feature/*`, `release/*`, `hotfix/*`. Подходит для проектов с плановыми релизами.
  - **GitHub Flow:** Упрощенная модель, где ветка `main` всегда стабильна, а новые функции разрабатываются в отдельных ветках с последующим созданием Pull Request.
- **Pre-commit hooks:** Автоматизированные скрипты, выполняемые перед коммитом для проверки качества кода (форматирование, линтинг, проверка синтаксиса).

Задание (Пошаговая инструкция для Ubuntu)

Часть 0: Предварительная настройка (Аутентификация по SSH)

1. **Сгенерируйте пару SSH-ключей:**

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

2. **Добавьте ключ в ssh-agent:**

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

3. **Добавьте публичный ключ в аккаунт GitHub.**
4. **Проверьте подключение:**

```
ssh -T git@github.com
```

Часть 1: Настройка Git и создание локального репозитория

1. **Установите Git и настройте глобальные параметры:**

```
sudo apt update && sudo apt install git
git config --global user.name "Ваше Имя"
git config --global user.email "your.email@example.com"
git config --global core.editor "code --wait"
```

2. **Создайте и откройте папку для проекта:**

```
mkdir python-git-lab1
cd python-git-lab1
code .
```

3. **Инициализируйте локальный Git-репозиторий:**

```
git init
```

Часть 2: Настройка pre-commit хуков

1. **Установите pre-commit:**

```
pip install pre-commit
```

2. **Создайте файл `.pre-commit-config.yaml` в корне проекта:**

```
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.4.0
    hooks:
      - id: trailing-whitespace
      - id: end-of-file-fixer
      - id: check-yaml
      - id: check-added-large-files
  - repo: https://github.com/psf/black
    rev: 23.3.0
    hooks:
      - id: black
        language_version: python3
```

3. **Активируйте pre-commit:**

```
pre-commit install
```

Часть 3: Основной workflow с использованием GitHub Flow

1. **Создайте файл `hello.py` и напишите простейшую программу на Python.**
2. **Добавьте файл в индекс и сделайте коммит:**

```
git add hello.py
git commit -m "Add initial hello.py script"
```

3. **Убедитесь, что pre-hook сработал (автоматическое форматирование кода).**
4. **Создайте ветку для новой функции:**

```
git checkout -b feature/greeting
```

5. **Создайте файл `greeting.py` с функцией приветствия.**
6. **Добавьте и закоммитьте изменения:**

```
git add greeting.py
git commit -m "Add greeting module with user input"
```

7. **Вернитесь на основную ветку и выполните слияние:**

```
git checkout main
git merge feature/greeting
```

Часть 4: Работа с удаленным репозиторием (GitHub)

1. **Создайте публичный репозиторий на GitHub с названием `python-git-lab1`.**
2. **Свяжите локальный репозиторий с удаленным:**

```
git remote add origin git@github.com:ВАШ_АККАУНТ/python-git-lab1.git
```

3. **Отправьте ветки на GitHub:**

```
git push -u origin main
git push origin feature/greeting
```

Часть 5: Имитация Git Flow

1. **Создайте ветку `develop` и отправьте её на GitHub:**

```
git checkout -b develop
git push origin develop
```

2. **Создайте ветку `feature/calculator` от `develop`:**

```
git checkout develop
git checkout -b feature/calculator
```

3. **Добавьте файл `calculator.py` с простой функцией калькулятора.**

4. **Закоммитьте изменения и отправьте ветку на GitHub.**
5. **Вернитесь в `develop` и выполните слияние:**

```
git checkout develop
git merge feature/calculator
git push origin develop
```

Часть 6: Финальные штрихи

1. **Создайте файл `README.md` с описанием проекта и инструкциями по запуску.**
2. **Добавьте скриншоты выполнения операций (например, результат работы pre-commit).**
3. **Отправьте все изменения на GitHub.**

Критерии оценки

На Удовлетворительно:

- [ ] SSH-ключ сгенерирован и добавлен в GitHub.
- [ ] Git сконфигурирован с указанием имени и email.
- [ ] Локальный репозиторий проинициализирован.
- [ ] Создано как минимум 2 осмысленных коммита в ветке `main`.
- [ ] Репозиторий на GitHub содержит файлы `hello.py` и `README.md`.

На Хорошо:

- **Все критерии на "Удовлетворительно" выполнены.**
- [ ] Настроен pre-commit и выполнено хотя бы одно автоматическое форматирование.
- [ ] Создана и merged ветка `feature/greeting`.
- [ ] Репозиторий на GitHub содержит файл `greeting.py`.

На Отлично:

- **Все критерии на "Хорошо" выполнены.**
- [ ] Реализована имитация Git Flow с ветками `develop` и `feature/calculator`.
- [ ] Файл `README.md` содержит описание проекта и инструкции по запуску.
- [ ] Pre-commit настроен и работает корректно (скриншоты приложены).

Рекомендуемая литература

1. **Официальная книга "Pro Git"** (доступна онлайн на git-scm.com).
2. **Vincent Driessen. "A Successful Git Branching Model"** (статья о Git Flow).
3. **Документация pre-commit** (доступна на pre-commit.com).