МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по практическому занятию №1

по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-22-1

Гончаров Даниил Ростиславович

«02» октября 2023 г.

Подпись студента_____

Работа защищена « »_____20__г.

Преподаватель Воронкин Р. А. _____
(подпись)

Ставрополь 2023

Цель работы: приобретение начальных навыков при работе с Git и GitHub, создание репозитория.

Ход работы:

1. Ответы на контрольные вопросы:

1) Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

2) В чем недостатки локальных и централизованных СКВ?

Самый очевидный минус для централизованных СКВ — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё. Также в локальных СКВ нет возможности взаимодействия с другими разработчиками.

3) К какой СКВ относится Git?

Git относится к распределенным системам контроля версий (РСКВ). В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) — они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

4) В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени. Git не хранит и не обрабатывает данные таким

способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git переосмысливает практически все аспекты контроля версий, которые были скопированы из предыдущего поколения большинством других систем. Это делает Git больше похожим на миниатюрную файловую систему с удивительно мощными утилитами, надстроенными над ней, нежели просто на СКВ.

5) Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

6) В каких состояниях могут находится файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7) Что такое профиль пользователя в GitHub?

Профиль - это ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8) Какие бывают репозитории в GitHub?

Репозитории бывают общедоступные и приватные. К общедоступным есть доступ у любого пользователя. К приватным же только у его разработчиков.

9) Укажите основные этапы модели работы с GitHub.

Чтобы перенести изменения с вашей копии в исходный репозиторий проекта, вам нужно сделать запрос на извлечение. Если вы хотите внести небольшие изменения в свою копию, вы можете использовать веб-интерфейс

GitHub. Однако такой подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10) Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, введите команду, чтобы отобразить текущую версию вашего Git. Если она сработала, давайте добавим в настройки Git ваше имя, фамилию и адрес электронной почты, связанный с вашей учетной записью GitHub.

11) Опишите этапы создания репозитория в GitHub.

Перейти на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

- Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториев, которые вы создавали.
- Описание (Description). Можно оставить пустым.
- Public/private. Выбрать открытый или приватный репозиторий.
- Выбрать .gitignore и LICENSE.
- После заполнения этих полей нажать кнопку Create repository.

12) Какие типы лицензий поддерживаются GitHub при создании репозитория?

Apache License 2.0, General Public License v3.0, MIT License, BSD 2, BSD 3, Boost Software License 1.0, Creative Commons Zero v1.0 Universal, The Unlicense, Mozilla Public License 2.0 и др.

13) Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите git clone и введите адрес: Клонирование репозитория необходимо для того, чтобы у пользователя было локальное хранилище проекта.

14) Как проверить состояние локального репозитория Git?

Перейдите в каталог хранилища и посмотрите на содержимое. Локальный репозиторий включает в себя все те же файлы, ветки и историю коммитов, как и удаленный репозиторий. Введите команду git status, чтобы проверить состояние вашего репозитория.

15) Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды git add; фиксации (коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push ?

Git фиксирует любые изменения в локльном репозитории и сообщает об этом пользователю с помощью терминала.

16) У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии.

Пользователи копирует данный репозиторий на свой компьютер и после каждого шага отправляют данные на удаленный репозиторий используя коммиты с подробными описаниями своих шагов, чтобы синхронизироваться со своим оппонентом.

17) GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

GitFlic - это первый российский облачный сервис для разработки и обслуживания исходного кода программ. На GitFlic можно размещать как open-source проекты, так и приватные. Сервис обеспечивает хранение данных в сертифицированных российских дата-центрах. Они соответствуют требованиям надежности уровня Tier 3 по классификации Uptime Institute. Так как сервис новый, то основные пользователи привыкли пользоваться уже привычным GitHub.

18) Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git?

GitHub Desktop - официальный клиент GitHub от разработчиков сервиса. С его помощью можно создавать репозитории, управлять запросами на включение кода, редактировать файлы и сравнивать изменения во встроенном редакторе кода. Для более удобного и быстрого изменения предусмотрена функция быстрого перехода в редактор по умолчанию, который можно выбрать в настройках. Также можно выбрать предпочитаемый терминал из установленных на машине. Файлы автоматически подтягиваются из директории репозитория, но работает и перетаскивание. В остальном GitHub Desktop довольно минималистичный и предоставляет только базовый набор инструментов для работы с удалёнными

репозиториями. В нём нет подробных графиков и визуального отображения истории веток. Также клиент работает только с GitHub, что заметно ограничивает сценарии его использования.

2. Выполнение практического занятия.

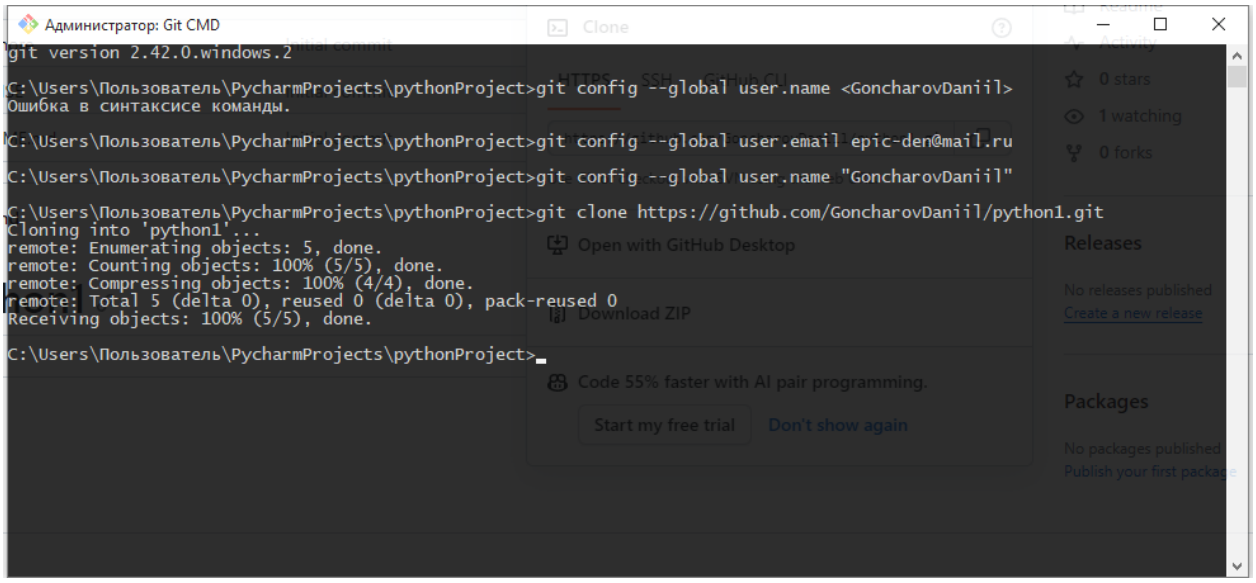

Рисунок 1. Установил Git

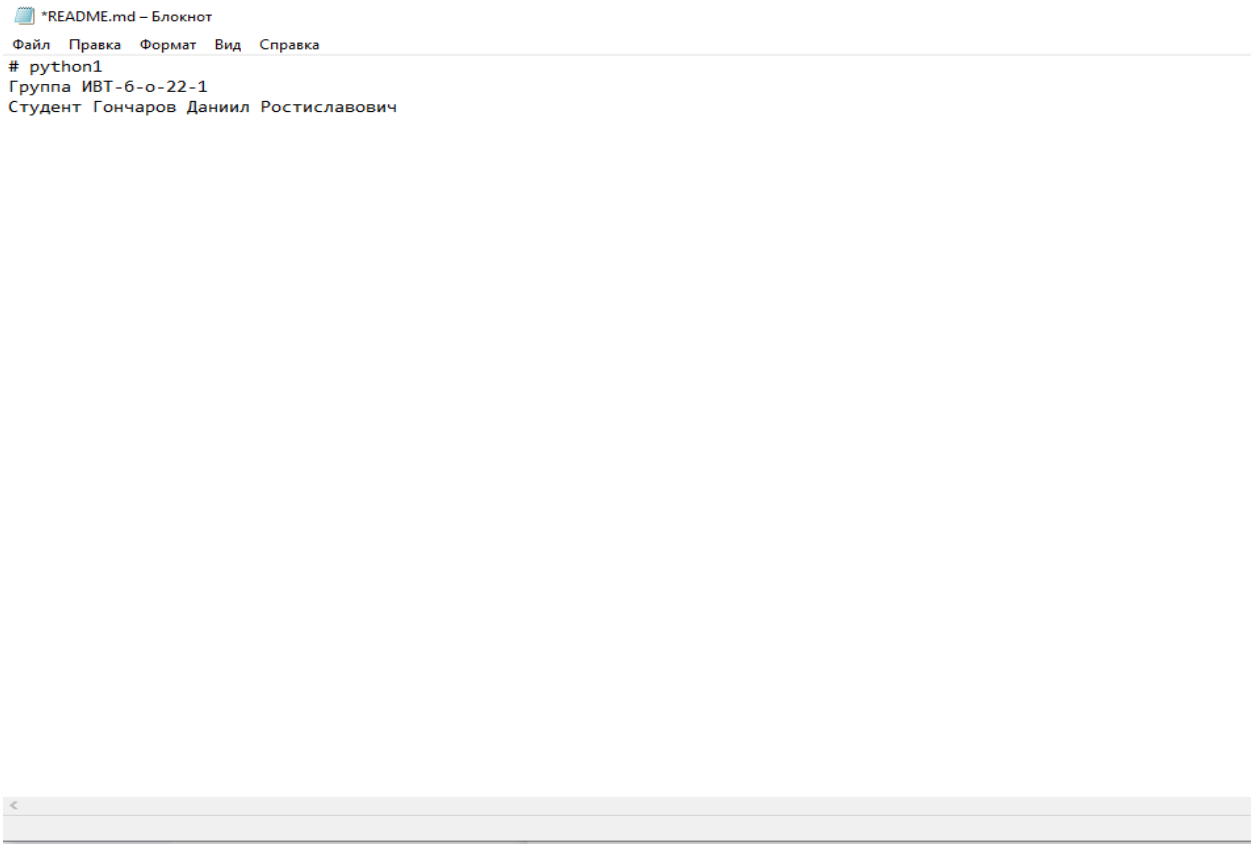

Рисунок 2. Добавил в Readme информацию о студенте, выполнявшем работу

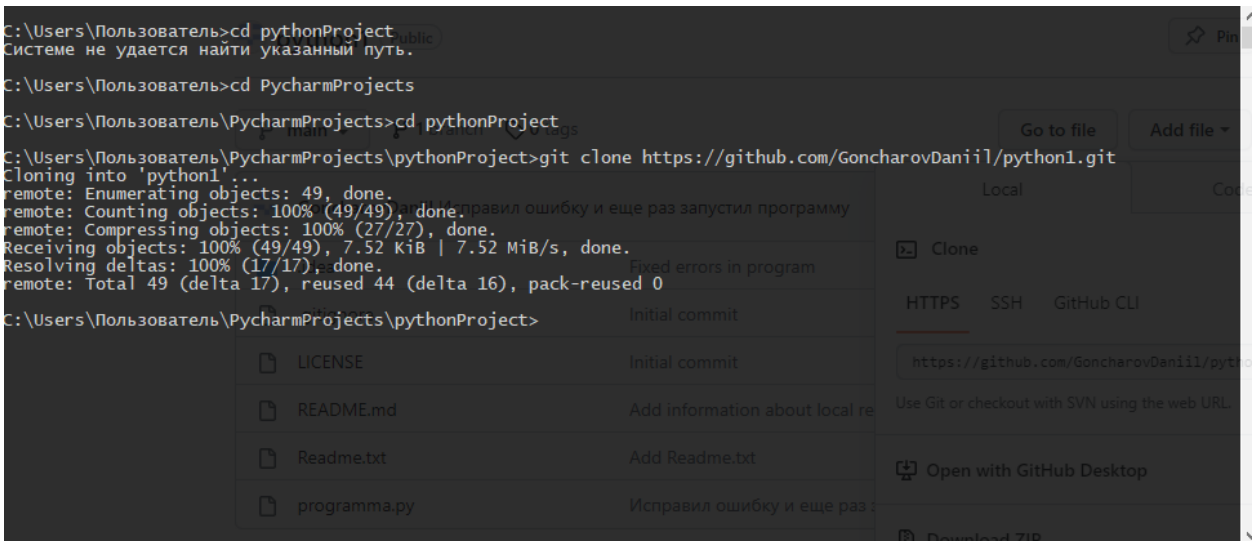

Рисунок 3. Клонировал репозиторий на компьютер

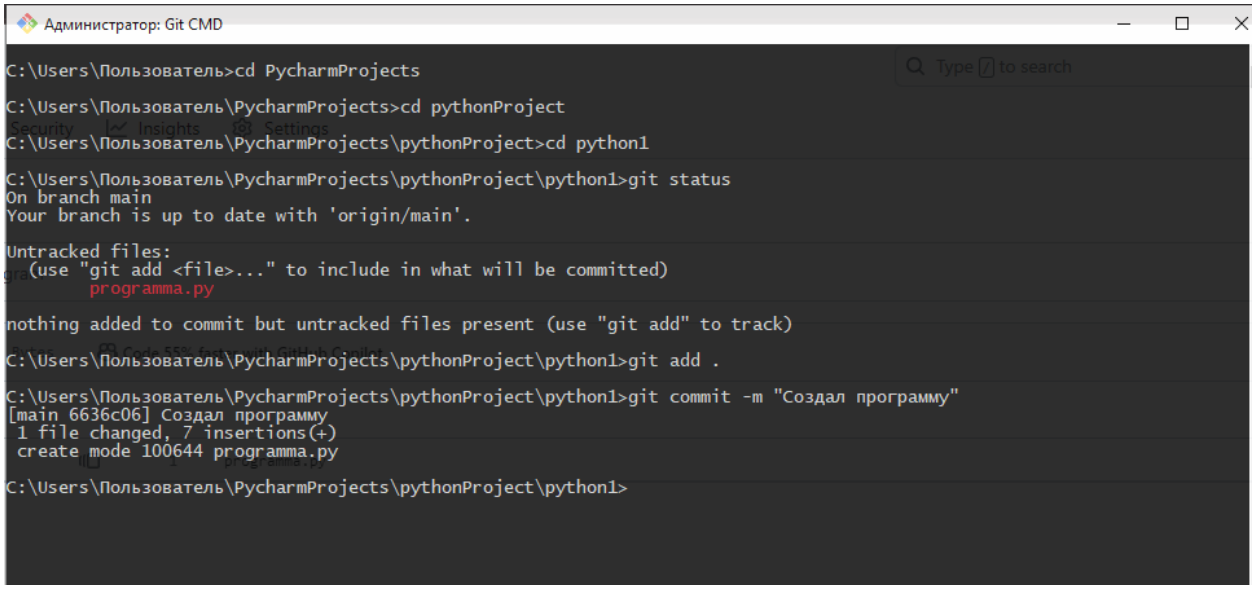

Рисунок 4. Создал программу на языке Python

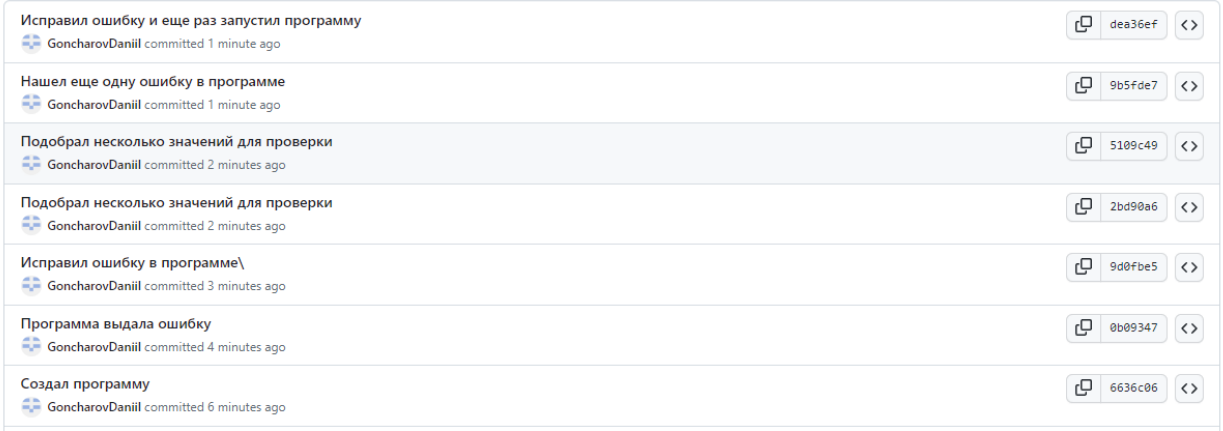

Рисунок 5. Выполнил программу с фиксированием изменений

```
C:\Users\Пользователь\PycharmProjects\pythonProject\python1>git push
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 12 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (19/19), 2.03 KiB | 2.03 MiB/s, done.
Total 19 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 1 local object.
To https://github.com/GoncharovDaniil/python1.git
   46ea4ce..dea36ef  main -> main

C:\Users\Пользователь\PycharmProjects\pythonProject\python1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\Пользователь\PycharmProjects\pythonProject\python1>
```

Рисунок 6. Отправил изменения в локальном репозиторий на удаленный

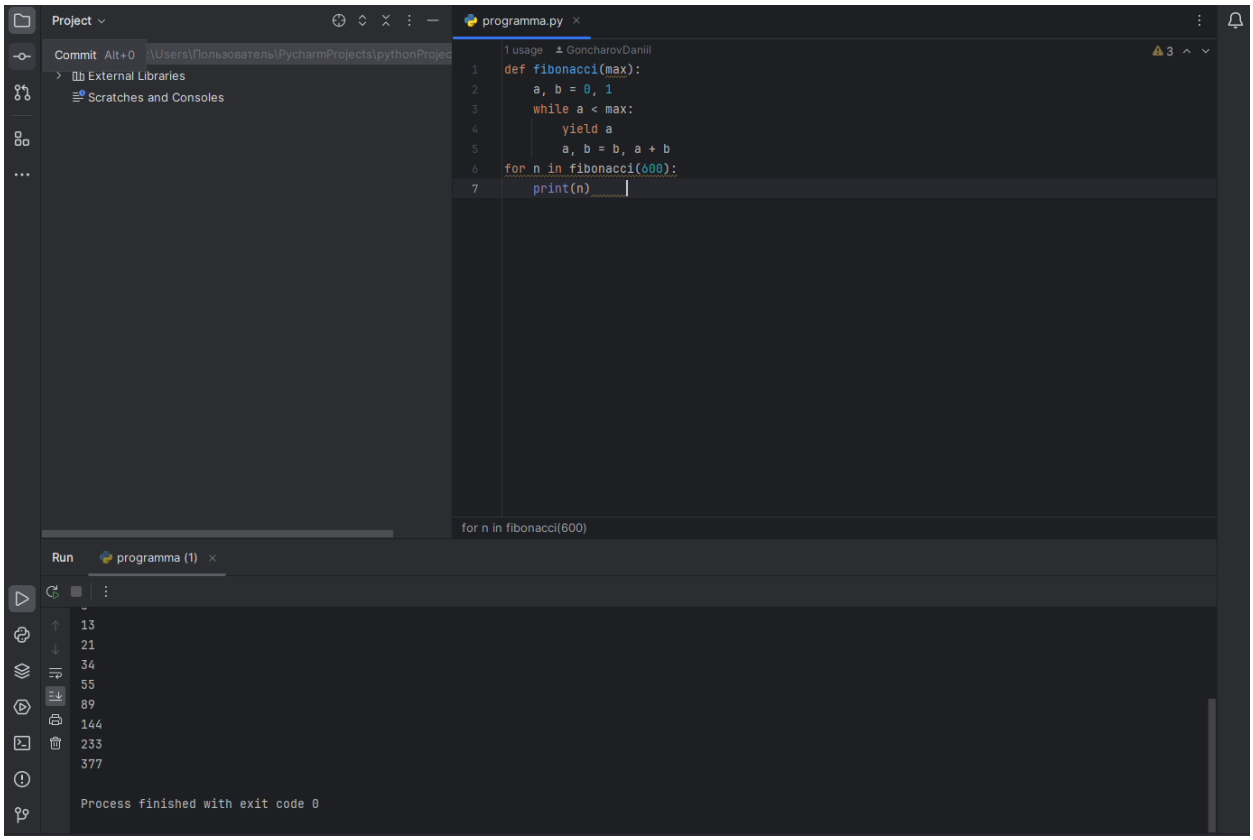

Рисунок 7. Программа на языке Python

Вывод: в ходе выполнения программы приобрел базовые навыки при работе с Git, научился создавать клонировать репозитории.