

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Программирование на Python»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22,
11.03.02 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», заочная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. технических
наук, доцент кафедры
инфокоммуникаций

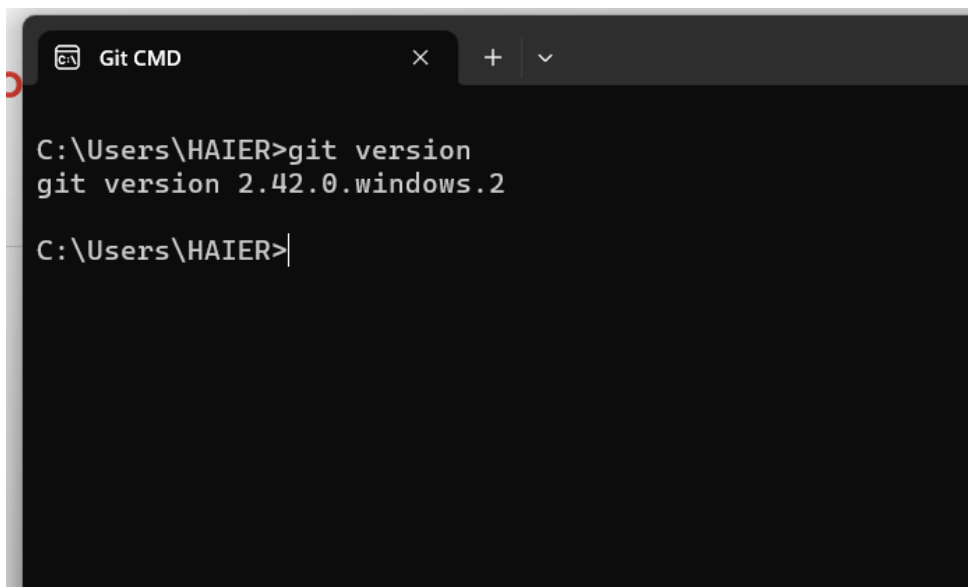
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: Исследование основных возможностей Git и GitHub

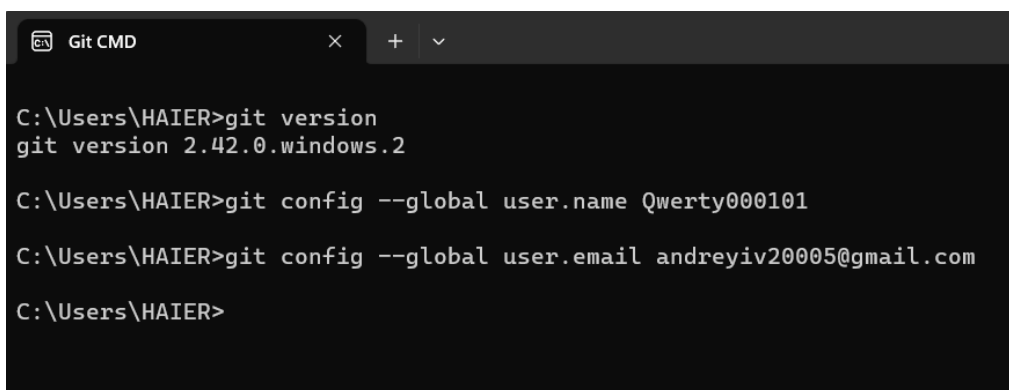
Цель: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Порядок выполнения работы:



```
Git CMD
C:\Users\HAIER>git version
git version 2.42.0.windows.2
C:\Users\HAIER>
```

Рисунок 1 – Установка Git



```
Git CMD
C:\Users\HAIER>git version
git version 2.42.0.windows.2
C:\Users\HAIER>git config --global user.name Qwerty000101
C:\Users\HAIER>git config --global user.email andreyiv20005@gmail.com
C:\Users\HAIER>
```

Рисунок 2 – Настройка Git

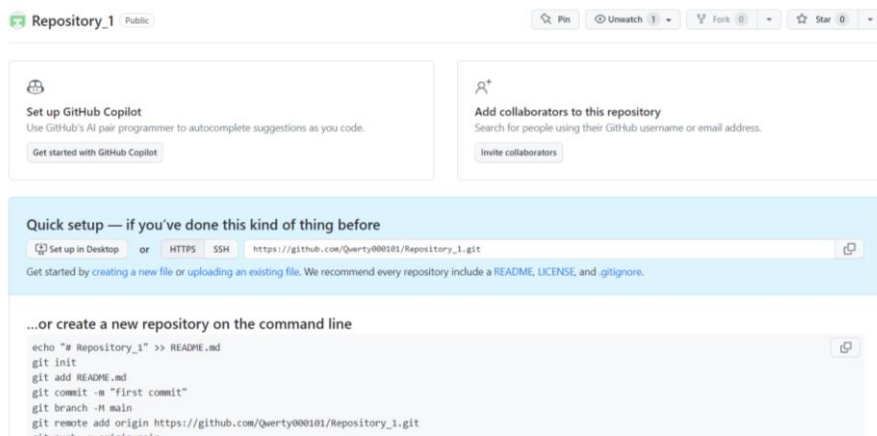


Рисунок 3 – Создание репозитория на GitHub

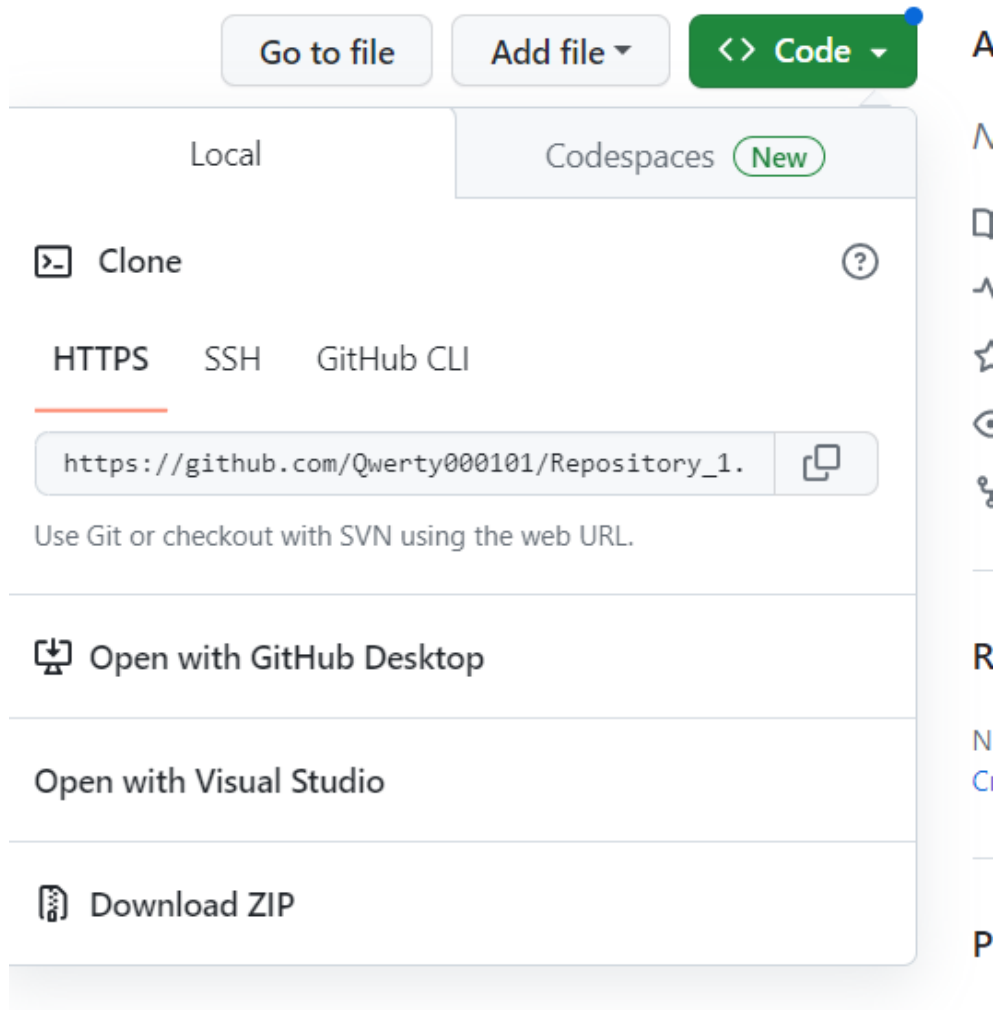


Рисунок 4 – Клонирование репозитория

```
Git CMD
C:\Users\HAIER>cd C:\Users\HAIER\Desktop\Задания\Программирование на Python
C:\Users\HAIER\Desktop\Задания\Программирование на Python>git clone https://github.com/Qwerty000101/Repository_1.git
Cloning into 'Repository_1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
C:\Users\HAIER\Desktop\Задания\Программирование на Python>
```

Рисунок 5 – Клонирование репозитория

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>
```

Рисунок 6 – Проверка состояния репозитория

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>
```

Рисунок 7 – Проверка состояния репозитория после внесения изменений

```
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git add README.md
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git add .
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git commit -m "Add information about local repository in readme file"
[main c3fb65a] Add information about local repository in readme file
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>
```

Рисунок 8 – Добавление файлов в систему контроля версий

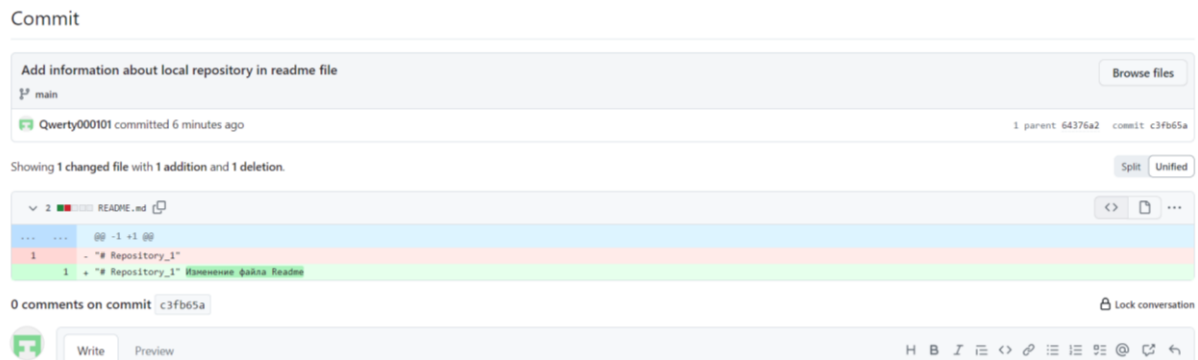
```

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 327.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Qwerty000101/Repository_1.git
 64376a2..c3fb65a  main -> main

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>

```

Рисунок 9 – Использование команды Push



```

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>git pull
Already up to date.

C:\Users\HAIER\Desktop\Задания\Программирование на Python\Repository_1>

```

Рисунок 9 – Использование команды Pull

Задание 1. Изучить теоретический материал работы.


Задание 2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).


Required fields are marked with an asterisk ().*

Owner * Qwerty000101 / Repository name * PythonLab1
✓ PythonLab1 is available.

Great repository names are short and memorable. Need inspiration? How about [redesigned-octo-waffle](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.


Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).


 You are creating a public repository in your personal account.

Рисунок 10 – Создание репозитория на GitHub

Задание 3. Выполните клонирование созданного репозитория на рабочий компьютер.

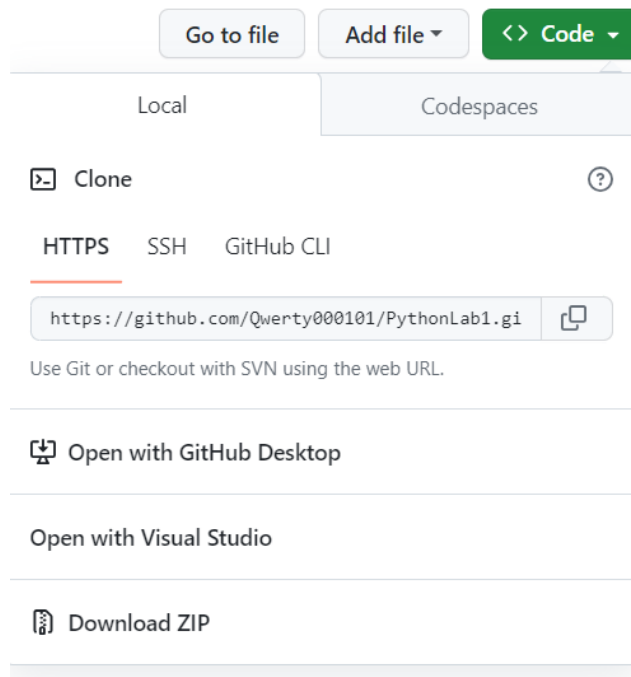


Рисунок 11 – Клонирование репозитория

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python>git clone https://github.com/Qwerty000101/PythonLab1.git
Cloning into 'PythonLab1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\HAIER\Desktop\Задания\Программирование на Python>
```

Рисунок 12 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

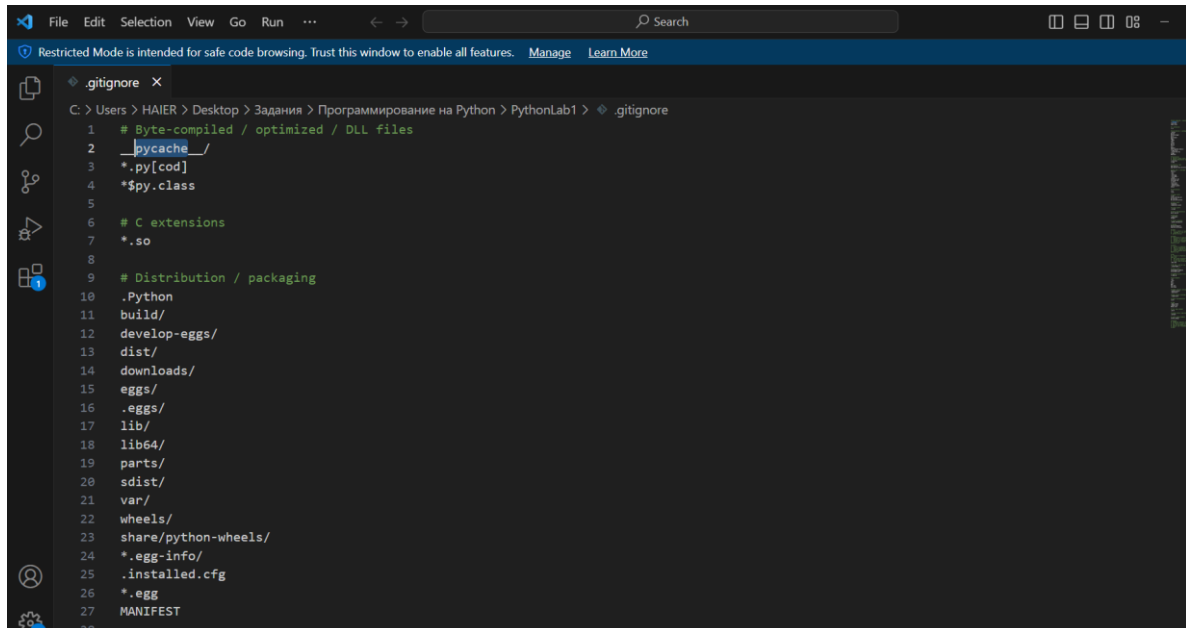


Рисунок 13 – Файл .gitignore

5. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу. Дополнение файла README.md

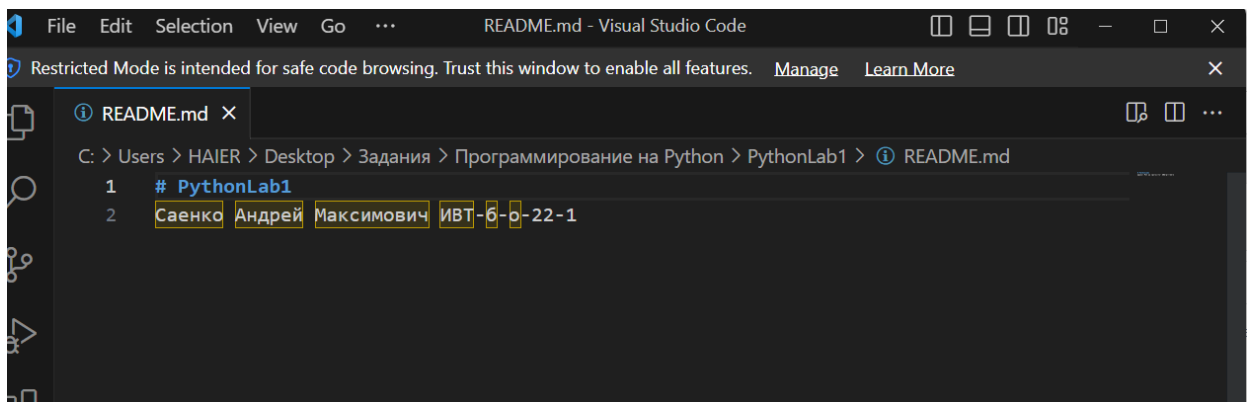
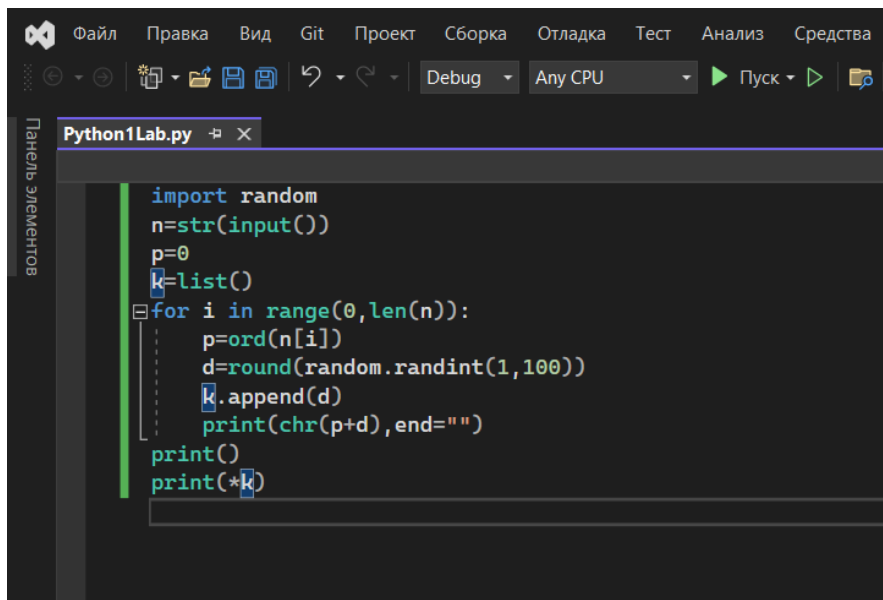


Рисунок 14 – Дополнение файла README.md

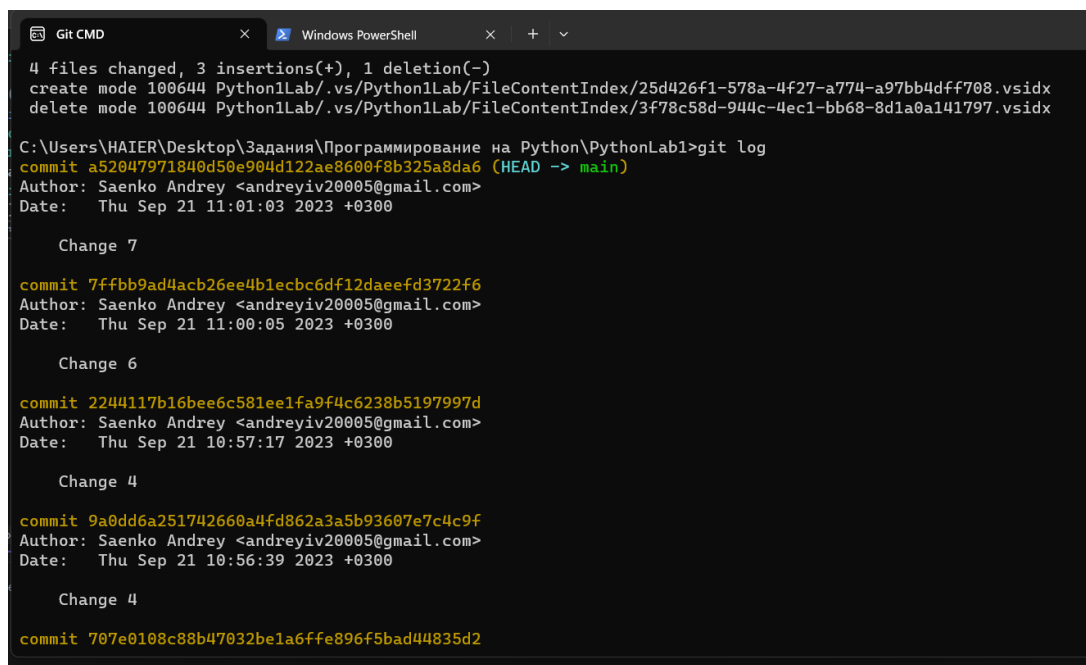
6. Напишите небольшую программу на выбранном Вами языке программирования. Фиксируйте изменения при написании программы в локальном репозитории. Должно быть сделано не менее 7 коммитов.



The image shows a screenshot of a code editor window titled 'Python1Lab.py'. The code is written in Python and implements a Caesar cipher. It imports the 'random' module, takes a string input from the user, and iterates over each character. For each character, it generates a random shift value between 1 and 100, shifts the character by that value, and prints the result. The final result is printed as a single string.

```
import random
n=str(input())
p=0
k=list()
for i in range(0,len(n)):
    p=ord(n[i])
    d=round(random.randint(1,100))
    k.append(d)
    print(chr(p+d),end=" ")
print()
print(*k)
```

Рисунок 15 – Код программы



The image shows a screenshot of a Windows PowerShell terminal window with the title 'Git CMD'. It displays the output of the 'git log' command. The output shows a list of commits, including the current commit (HEAD -> main) and several previous commits. The commits are listed in reverse chronological order, showing the history of the repository.

```
4 files changed, 3 insertions(+), 1 deletion(-)
create mode 100644 Python1Lab/.vs/Python1Lab/FileContentIndex/25d426f1-578a-4f27-a774-a97bb4dfff708.vsidx
delete mode 100644 Python1Lab/.vs/Python1Lab/FileContentIndex/3f78c58d-944c-4ec1-bb68-8d1a0a141797.vsidx

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab1>git log
commit a52047971840d50e904d122ae8600f8b325a8da6 (HEAD -> main)
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 11:01:03 2023 +0300

    Change 7

commit 7ffbb9ad4acb26ee4b1ecbc6df12daeeef3722f6
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 11:00:05 2023 +0300

    Change 6

commit 2244117b16bee6c581ee1fa9f4c6238b5197997d
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 10:57:17 2023 +0300

    Change 4

commit 9a0dd6a251742660a4fd862a3a5b93607e7c4c9f
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 10:56:39 2023 +0300

    Change 4

commit 707e0108c88b47032be1a6ffe896f5bad44835d2
```

Рисунок 16 – Результат ввода команды git log

```
Change 3

commit 78e043a36c1a705020f5b866f6cd2b8fd97fdcc6
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 10:55:02 2023 +0300

Change 2

commit f2723df899dce2557ab335e532f35e3a01d86aa8
Author: Saenko Andrey <andreyiv20005@gmail.com>
Date: Thu Sep 21 10:52:55 2023 +0300

Change 1

commit c1353a11c005ec289e0b7511c46e5dcfe0d7b2fa (origin/main, origin/HEAD)
Author: Qwerty000101 <93665628+Qwerty000101@users.noreply.github.com>
Date: Wed Sep 13 15:37:08 2023 +0300

Initial commit

C:\Users\NAIER\Desktop\Задания\Программирование на Python\PythonLab1>
```

Рисунок 17 – Результат ввода команды git log

7. Добавьте файл README и зафиксируйте сделанные изменения.

« Программирование на Python > PythonLab1

Поиск в: PythonLab1

Имя	Дата изменения	Тип	Размер
.git	21.09.2023 11:01	Папка с файлами	
Python1Lab	21.09.2023 11:00	Папка с файлами	
.gitignore	13.09.2023 15:39	Исходный файл G...	4 КБ
LICENSE	13.09.2023 15:39	Файл	2 КБ
README.md	13.09.2023 15:42	Исходный файл ...	1 КБ

Тип: Исходный файл Markdown
Размер: 79 байт
Дата изменения: 13.09.2023 15:42

Рисунок 18 – Файл README.md

Ответы на контрольные вопросы

1. Что такое СКВ и каково её назначение?

СКВ (система контроля версий) - это система, регистрирующая изменения в одном или нескольких файлах, что позволяет при необходимости вернуться к определённым версиям этих файлов.

2. В чём недостатки локальных и централизованных СКВ?

Используя локальную СКВ, пользователь может забыть, в какой директории он находится, и случайно изменить не тот файл или скопировать не те файлы.

Централизованные СКВ имеют единую точку отказа, предоставленную централизованным сервером. Если сервер выйдет из строя на некоторое время, то в течение этого времени никто не сможет использовать контроль версий и

обмениваться ими. Если будет повреждён жесткий диск, на котором хранится центральная БД, вся история проекта будет потеряна.

3. К какой СКВ относится Git?

Git относится к распределённым системам контроля версий.

4. В чём концептуальное отличие Git от других СКВ?

Основное отличие Git от другой СКВ - это подход к работе с данными. Большинство других систем хранят информацию в виде списка изменений в файлах. Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. При сохранении проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на снимок. Git не запоминает файлы, которые остались без изменений, а просто создаёт ссылку на предыдущую версию.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение, а в дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме, что не позволяет изменить содержимое файла или директории так, чтобы Git об этом не узнал.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

Есть три состояния, в которых могут находиться файлы в Git: зафиксированное (файл уже сохранён в локальной базе), изменённое (файл поменялся, но не был зафиксирован), подготовленные файлы (изменённые файлы, отмеченные для включения в следующий коммит).

7. Что такое профиль пользователя в GitHub?

Профиль - это публичная страница пользователя.

8. Какие бывают репозитории на GitHub?

Публичные репозитории – репозитории, доступные всем пользователям, которые могут просматривать и клонировать их.

Приватные репозитории доступны только авторизованным пользователям, которые имеют права на просмотр и клонирование.

9. Укажите основные этапы модели работы с GitHub.

Создание репозитория, клонирование, изменение, отправка изменений на сервер.

10. Как осуществляется первоначальная настройка Git после установки?

После загрузки Git проводится установка имени пользователя и адреса электронной почты, которые будут использоваться для идентификации автора коммитов.

11. Опишите этапы создания репозитория на GitHub.

1. Нажатие кнопки рядом с аватаром.

2. Ввод имени репозитория, создание описания, выбор типа, создание файлов .gitignore и license.

3. Создание репозитория.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

MIT, Apache, GNU GPL и Creative Commons

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

Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. После этого нужно ввести в командную строку команду `git clone` и адрес репозитория, после чего репозиторий будет скопирован в текущую выбранную папку.

14. Как проверить состояние локального репозитория Git?

Для проверки состояния локального репозитория Git, нужно использовать команду `git status`.

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/изменённого файла под версию

контроль с помощью команды `git add`; фиксации (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`?

После добавления/изменения файла в локальный репозиторий Git состояние репозитория изменится на «изменено».

После добавления нового/измененного файла под версионный контроль с помощью команды `git add`, состояние репозитория изменится на «готово к коммиту».

После фиксации (коммита) изменений с помощью команды `git commit`, состояние репозитория изменится на «зафиксировано».

После отправки изменений на сервер с помощью команды `git push`, состояние локального репозитория не изменится, но соответствующие изменения будут отображаться в репозитории на GitHub.

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. Примечание: описание необходимо начать с команды `git clone`.

Нужно клонировать репозиторий с GitHub на компьютеры с помощью команды `git clone` (адрес репозитория). После этого при внесении изменений нужно будет использовать команды `git push` и `git pull` для выгрузки и загрузки изменений.

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

Существует сервис GitLab, предназначенный для работы с Git-репозиториями.

Отличительные черты GitHub:

1. Доступен для широкого круга пользователей как бесплатный сервис с открытым исходным кодом;
2. Обеспечивает распределенное хранилище для кода;
3. Оказывает помощь во время жизненного цикла разработки;
4. Обеспечивает возможность совместно использовать код;
5. Дает возможность для отслеживания ошибок;
6. Поддерживает пространство вики и разные инструменты «социального кодирования».

Отличительные черты GitLab:

1. Есть бесплатная и платная версия для предприятий;
2. Предложение модели разработки с открытым ядром, плюс решение с открытым исходным кодом (написан на Ruby);
3. Обеспечивает полную совместимость с технологиями непрерывной интеграции;
4. Предлагает полную прозрачность для разработчиков для контекста и быстрых итераций по мере необходимости.

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? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

GitHub Desktop - клиент для работы с Git и GitHub. Он позволяет выполнить все операции, описанные в лабораторной работе, в том числе добавление/изменение файлов, коммиты и отправку изменений на сервер. Для выполнения этих операций пользователю нужно всего лишь выбрать нужный файл или изменение и нажать соответствующую кнопку.

Выводы

Были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.