

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

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе № 1.1**

**«Исследование основных возможностей Git и GitHub»**

**по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

«10» сентября 2022 г.

Подпись студента \_\_\_\_\_

Работа защищена

« » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь, 2022

## Цель работы:

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

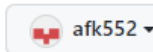
## Выполнение работы:

Создание репозитория на GitHub, рисунки 1 и 2.

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



Repository name \*



Great repository names are short and memorable. Need inspiration? How about [fluffy-succotash?](#)

Description (optional)

Лабораторная работа 1.1 Исследование основных возможностей Git и GitHub



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:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

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

.gitignore template: Python

Choose a license

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

License: MIT License

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.

Create repository

Рисунок 1 Страница создания репозитория GitHub

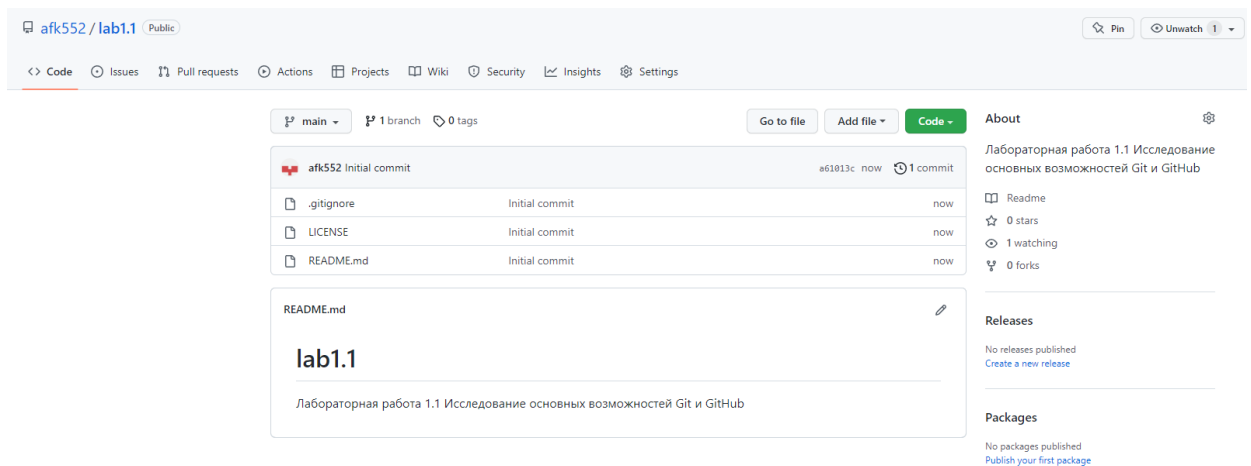


Рисунок 2 Созданный репозиторий

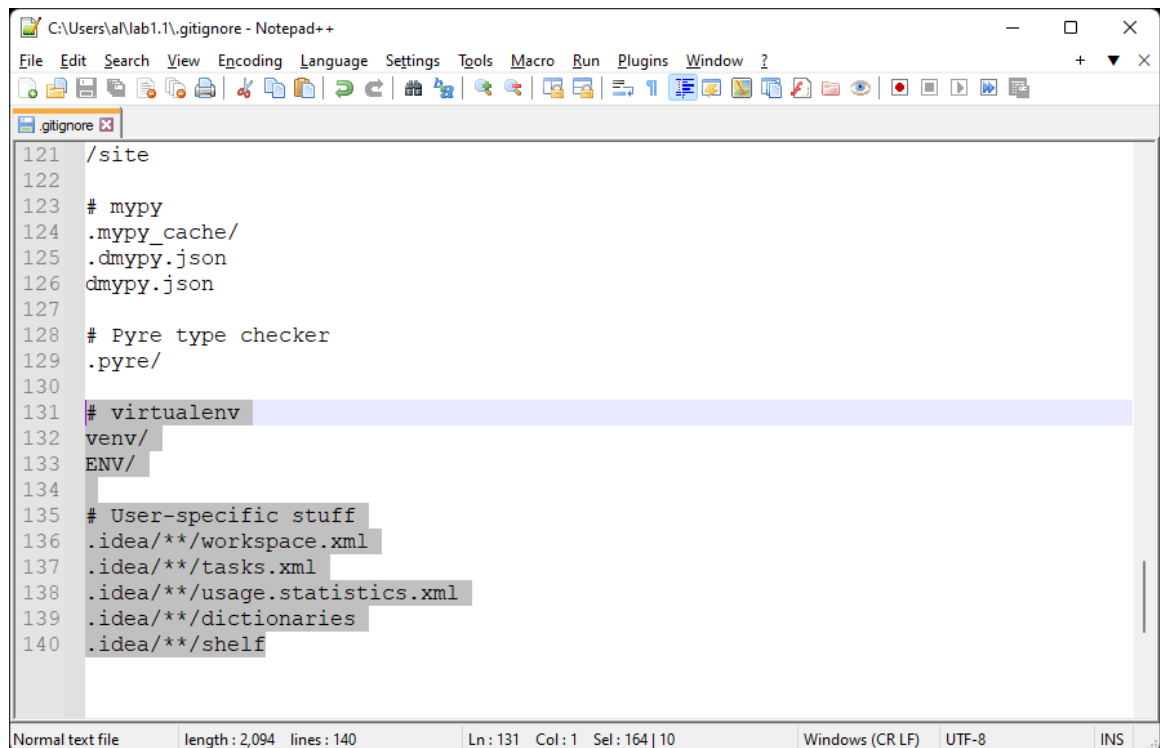
Клонирование созданного репозитория на рабочем компьютере, рисунок 3.

```
Command Prompt

C:\Users\al>git clone https://github.com/afk552/lab1.1
Cloning into 'lab1.1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
Receiving objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
C:\Users\al>
```

Рисунок 3 Окно командной строки

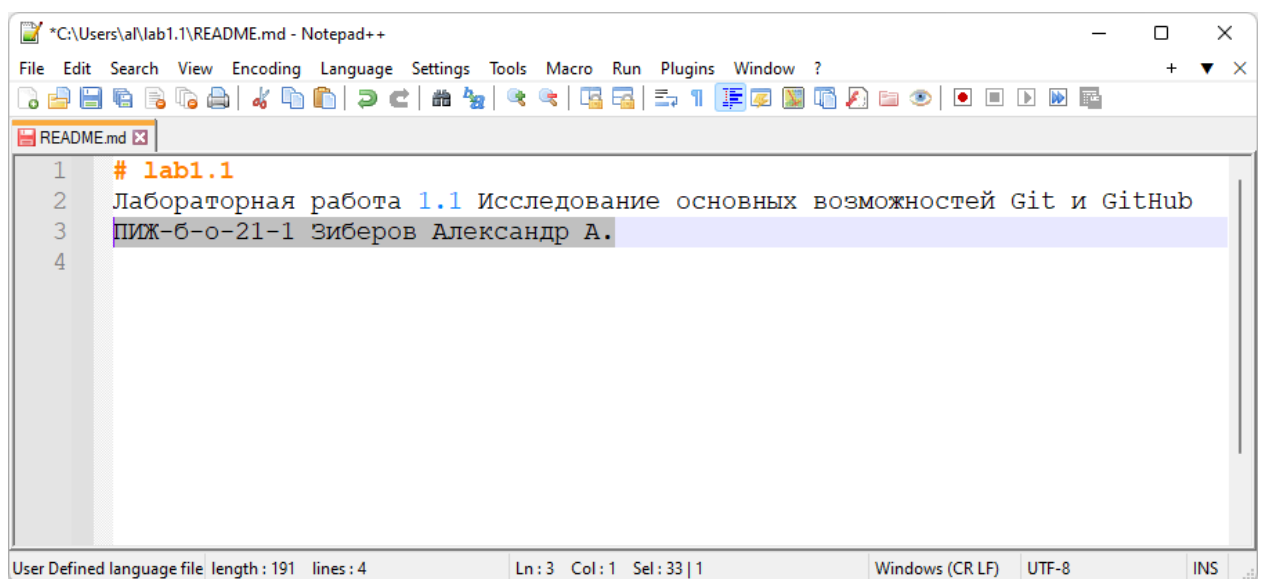
Дополнение файла .gitignore необходимыми правилами для выбранного языка и виртуальной среды, рисунок 4.



```
121 /site
122
123 # mypy
124 .mypy_cache/
125 .dmypy.json
126 dmypy.json
127
128 # Pyre type checker
129 .pyre/
130
131 # virtualenv
132 venv/
133 ENV/
134
135 # User-specific stuff
136 .idea/**/workspace.xml
137 .idea/**/tasks.xml
138 .idea/**/usage.statistics.xml
139 .idea/**/dictionaries
140 .idea/**/shelf
```

Рисунок 4 Окно Notepad++

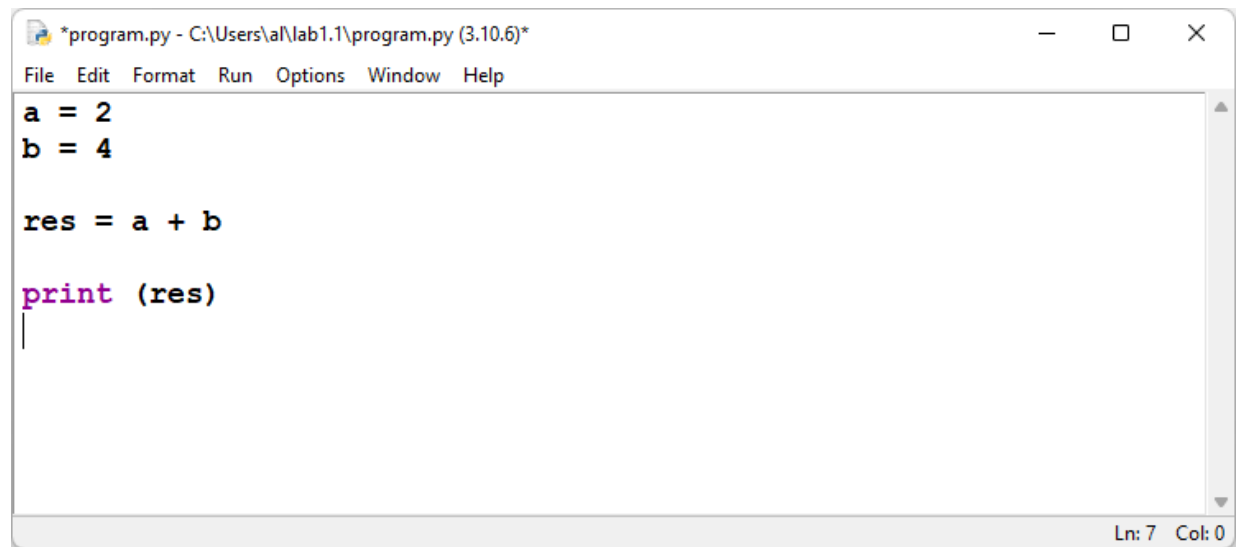
Добавление в файл README.md информации о группе и ФИО студента, выполняющего лабораторную работу, рисунок 5.



```
1 # lab1.1
2 Лабораторная работа 1.1 Исследование основных возможностей Git и GitHub
3 ПИЖ-б-о-21-1 Зиберов Александр А.
4
```

Рисунок 5 Окно Notepad++

Написание небольшой программы с фиксацией изменений и созданием коммитов (не менее 7), рисунки 6-19.



The screenshot shows a window titled "\*program.py - C:\Users\al\lab1.1\program.py (3.10.6)\*". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

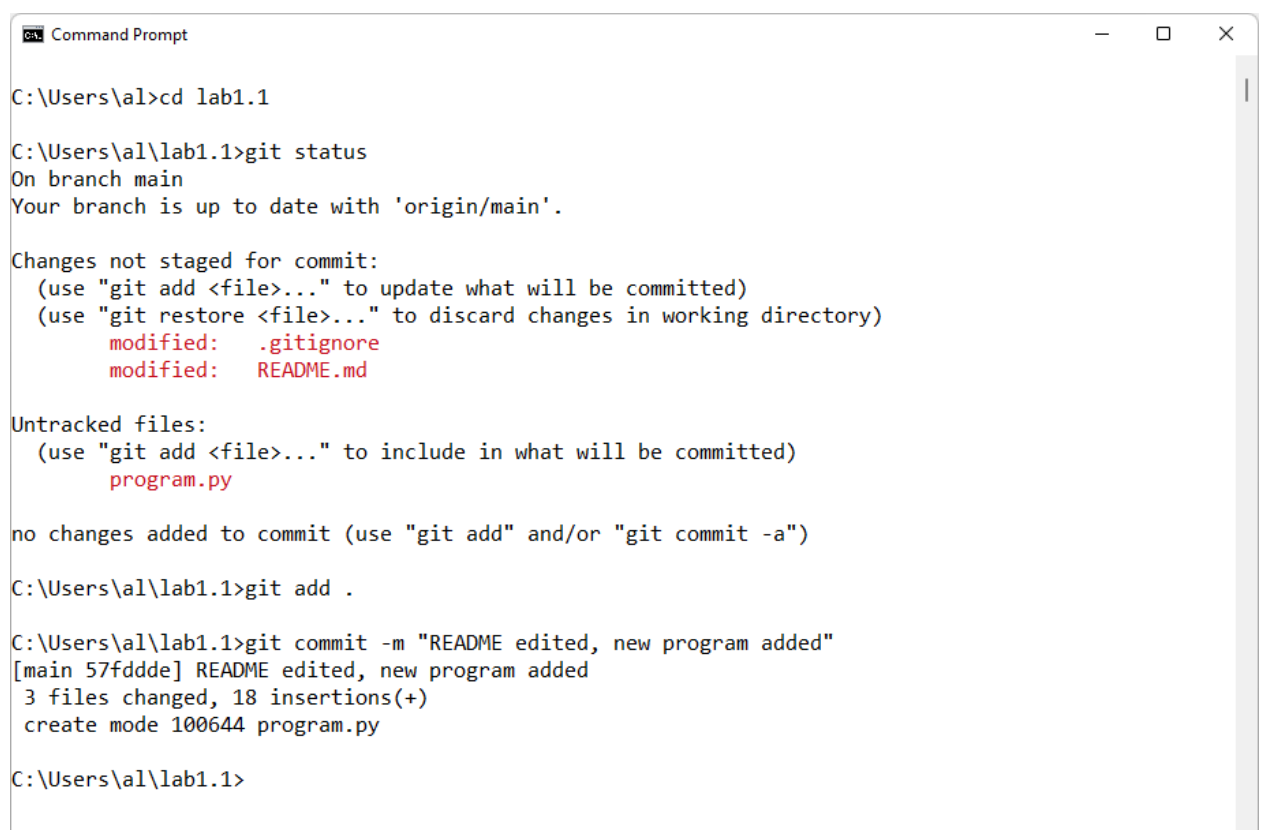
```
a = 2
b = 4

res = a + b

print (res)
```

The status bar at the bottom right indicates "Ln: 7 Col: 0".

Рисунок 6 Окно IDE



The screenshot shows a Command Prompt window with the following text:

```
C:\Users\al>cd lab1.1

C:\Users\al\lab1.1>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:   .gitignore
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        program.py

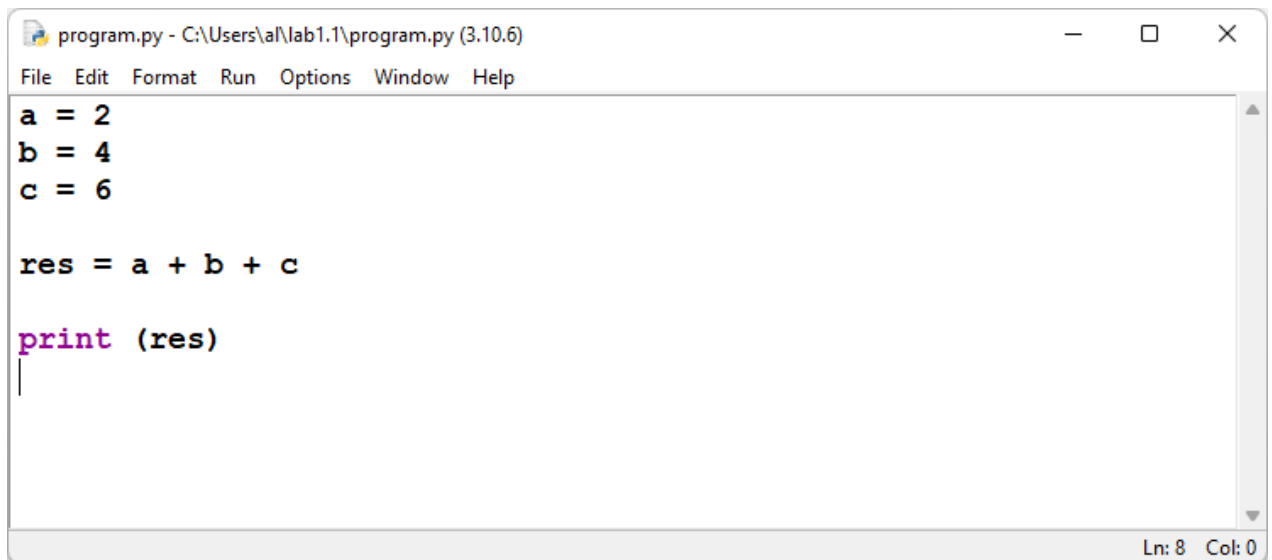
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "README edited, new program added"
[main 57fddde] README edited, new program added
3 files changed, 18 insertions(+)
create mode 100644 program.py

C:\Users\al\lab1.1>
```

Рисунок 7 Окно командной строки



```
program.py - C:\Users\al\lab1.1\program.py (3.10.6)
File Edit Format Run Options Window Help

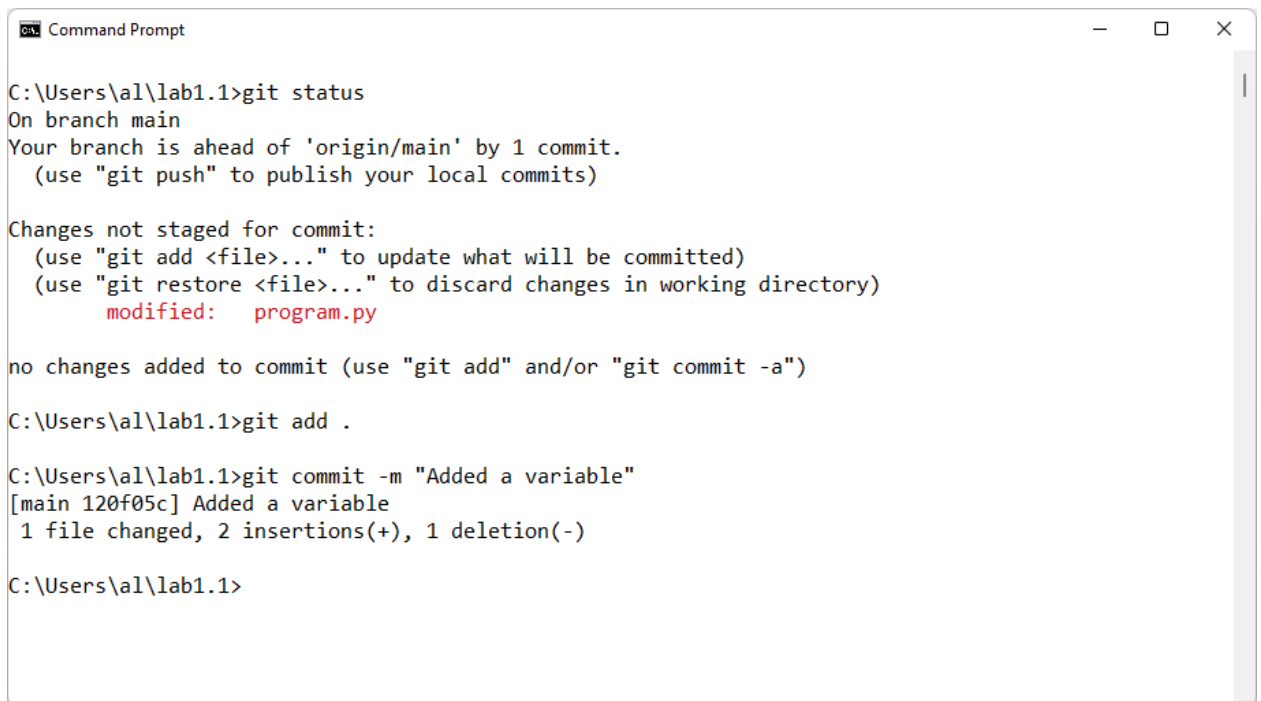
a = 2
b = 4
c = 6

res = a + b + c

print (res)
|

Ln: 8 Col: 0
```

Рисунок 8 Окно IDE



```
Command Prompt

C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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:   program.py

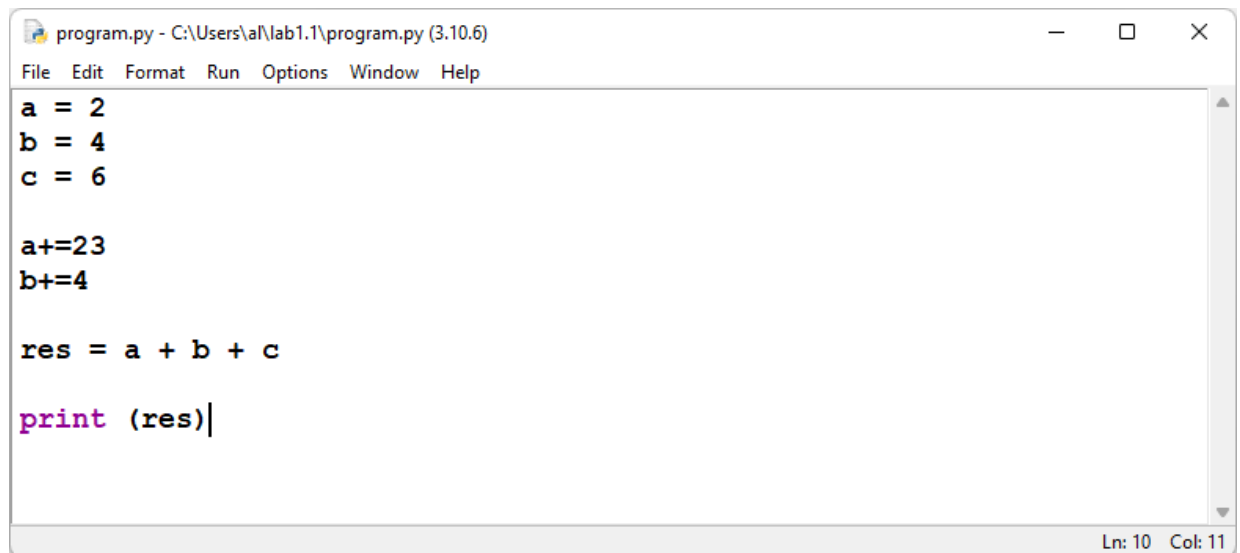
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Added a variable"
[main 120f05c] Added a variable
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\al\lab1.1>
```

Рисунок 9 Окно командной строки



The image shows a screenshot of an IDE window titled "program.py - C:\Users\al\lab1.1\program.py (3.10.6)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
a = 2
b = 4
c = 6

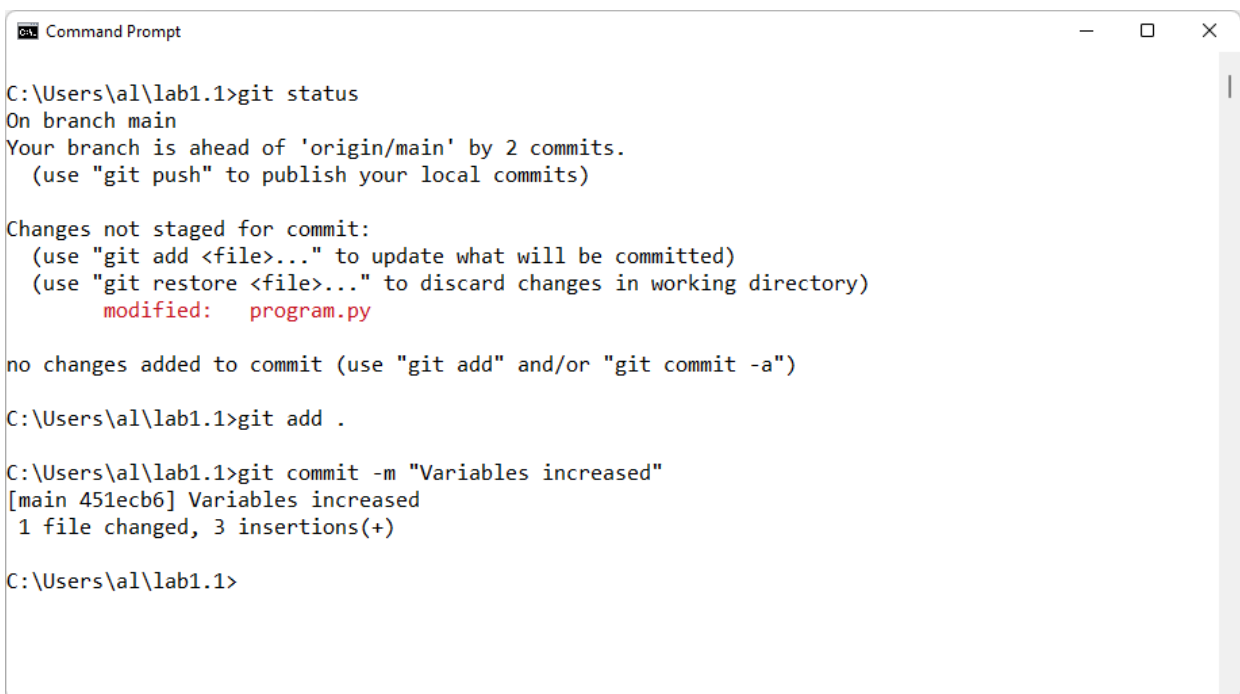
a+=23
b+=4

res = a + b + c

print (res)|
```

The status bar at the bottom right indicates "Ln: 10 Col: 11".

Рисунок 10 Окно IDE



The image shows a screenshot of a Command Prompt window titled "Command Prompt". The window displays the output of several git commands:

```
C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

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:   program.py

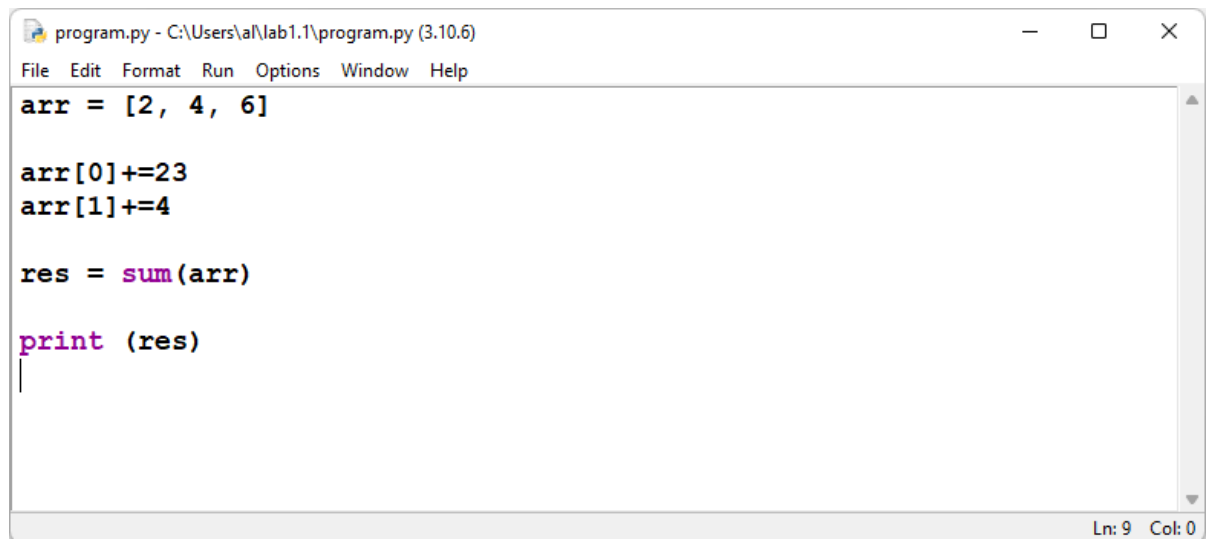
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Variables increased"
[main 451ecb6] Variables increased
 1 file changed, 3 insertions(+)
```

The prompt "C:\Users\al\lab1.1>" is shown at the bottom.

Рисунок 11 Окно командной строки



```
program.py - C:\Users\al\lab1.1\program.py (3.10.6)
File Edit Format Run Options Window Help

arr = [2, 4, 6]

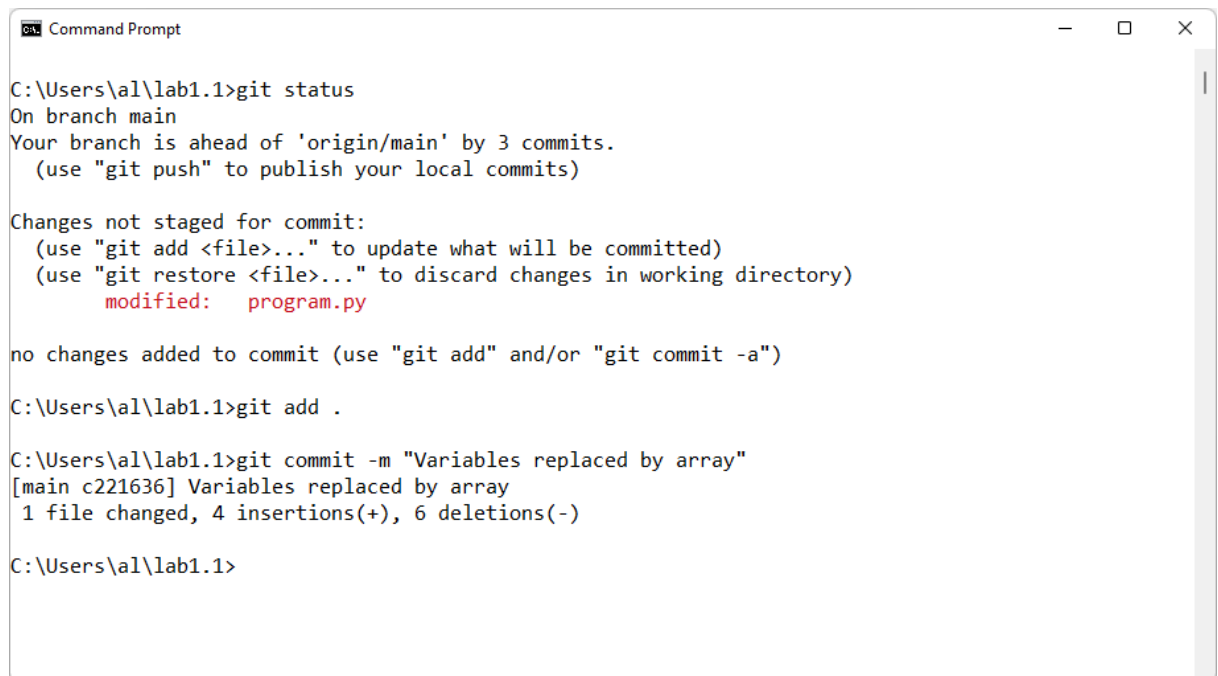
arr[0]+=23
arr[1]+=4

res = sum(arr)

print (res)
|

Ln: 9 Col: 0
```

Рисунок 12 Окно IDE



```
Command Prompt

C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

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:   program.py

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Variables replaced by array"
[main c221636] Variables replaced by array
 1 file changed, 4 insertions(+), 6 deletions(-)

C:\Users\al\lab1.1>
```

Рисунок 13 Окно командной строки





The image shows a screenshot of an IDE window titled "program.py - C:\Users\al\lab1.1\program.py (3.10.6)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
import random

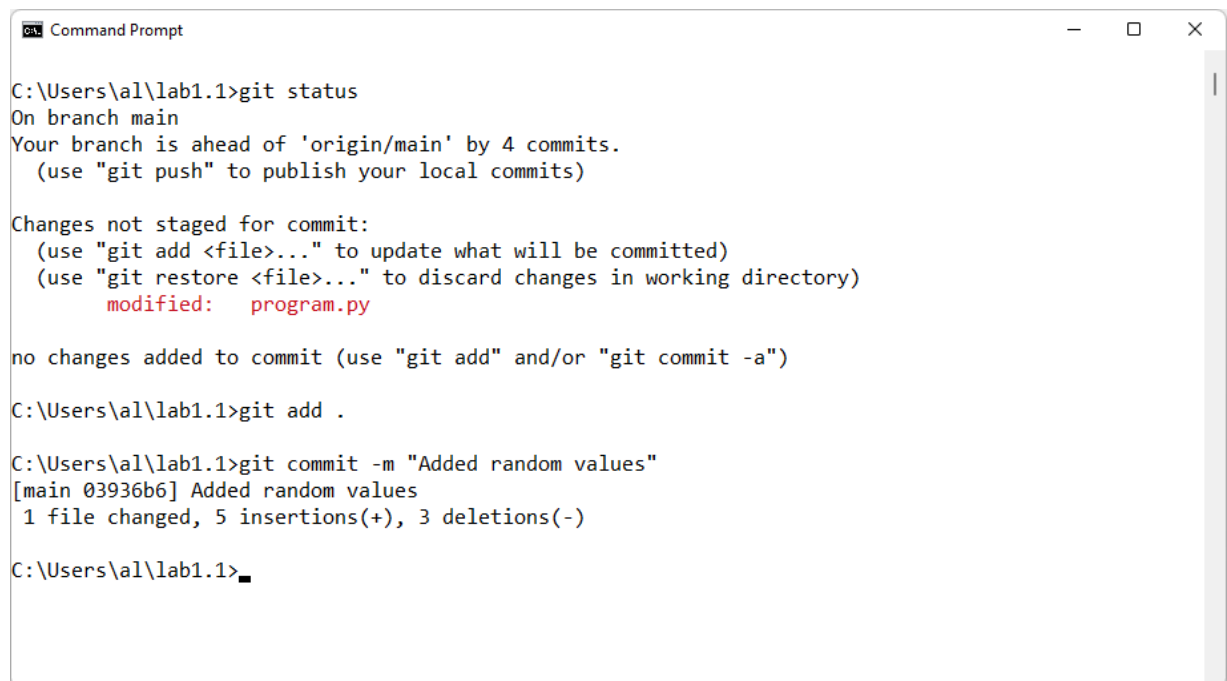
arr = []
for i in range (3):
    print (i)
    arr.append(random.randint(1,100))

res = sum(arr)

print (res)|
```

The status bar at the bottom right indicates "Ln: 10 Col: 11".

Рисунок 14 Окно IDE



The image shows a screenshot of a Command Prompt window titled "Command Prompt". The window displays the output of several git commands:

```
C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

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:   program.py

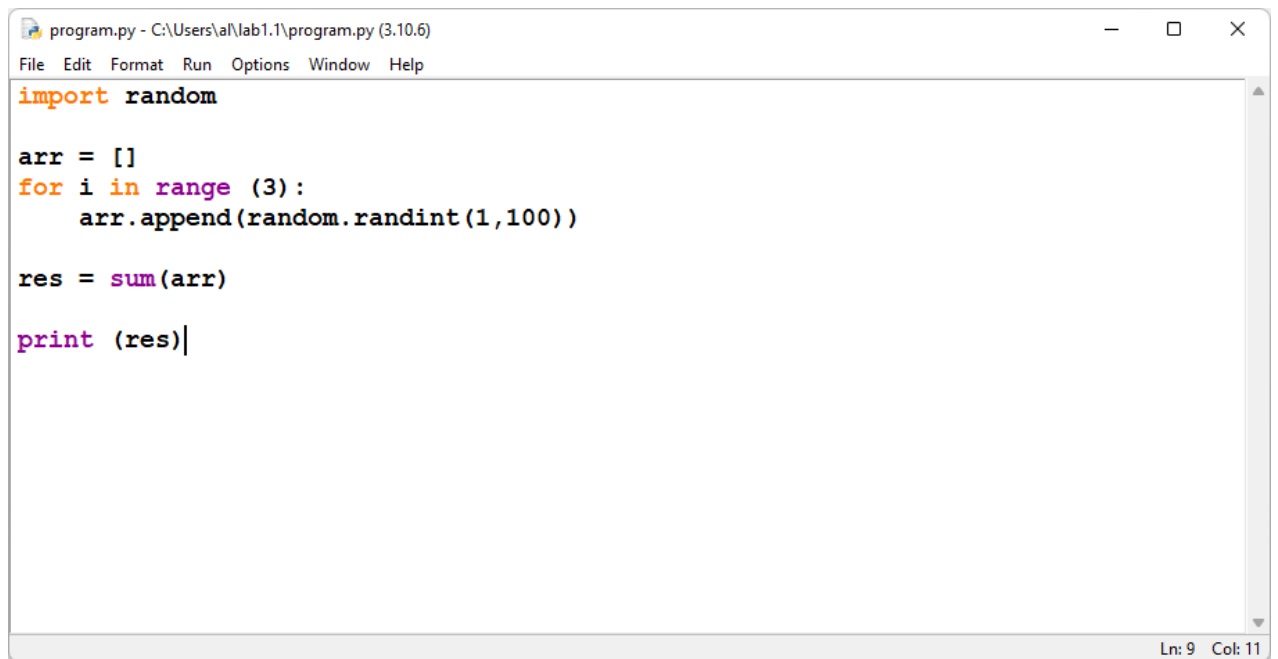
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Added random values"
[main 03936b6] Added random values
 1 file changed, 5 insertions(+), 3 deletions(-)

C:\Users\al\lab1.1>
```

Рисунок 15 Окно командной строки



The image shows a screenshot of an IDE window titled "program.py - C:\Users\al\lab1.1\program.py (3.10.6)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main area contains the following Python code:

```
import random

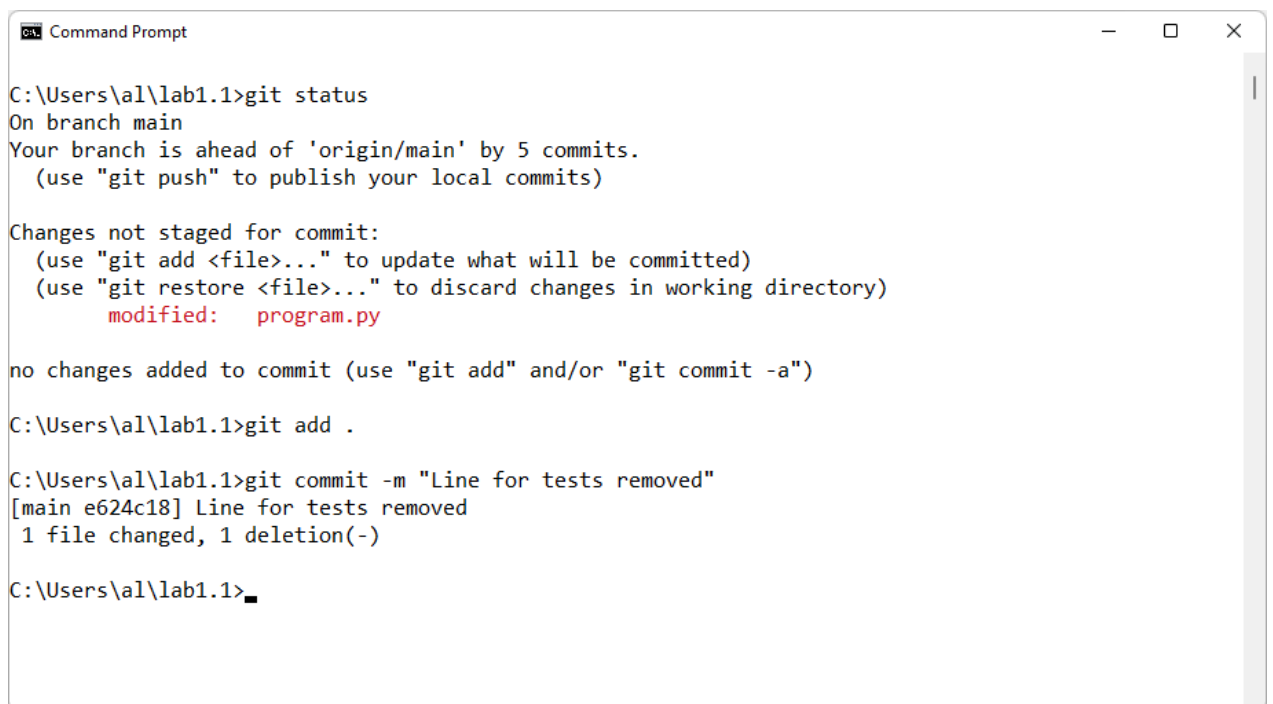
arr = []
for i in range (3):
    arr.append(random.randint(1,100))

res = sum(arr)

print (res)|
```

The status bar at the bottom right indicates "Ln: 9 Col: 11".

Рисунок 16 Окно IDE



The image shows a screenshot of a Command Prompt window titled "Command Prompt". The window contains the following text:

```
C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

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:   program.py

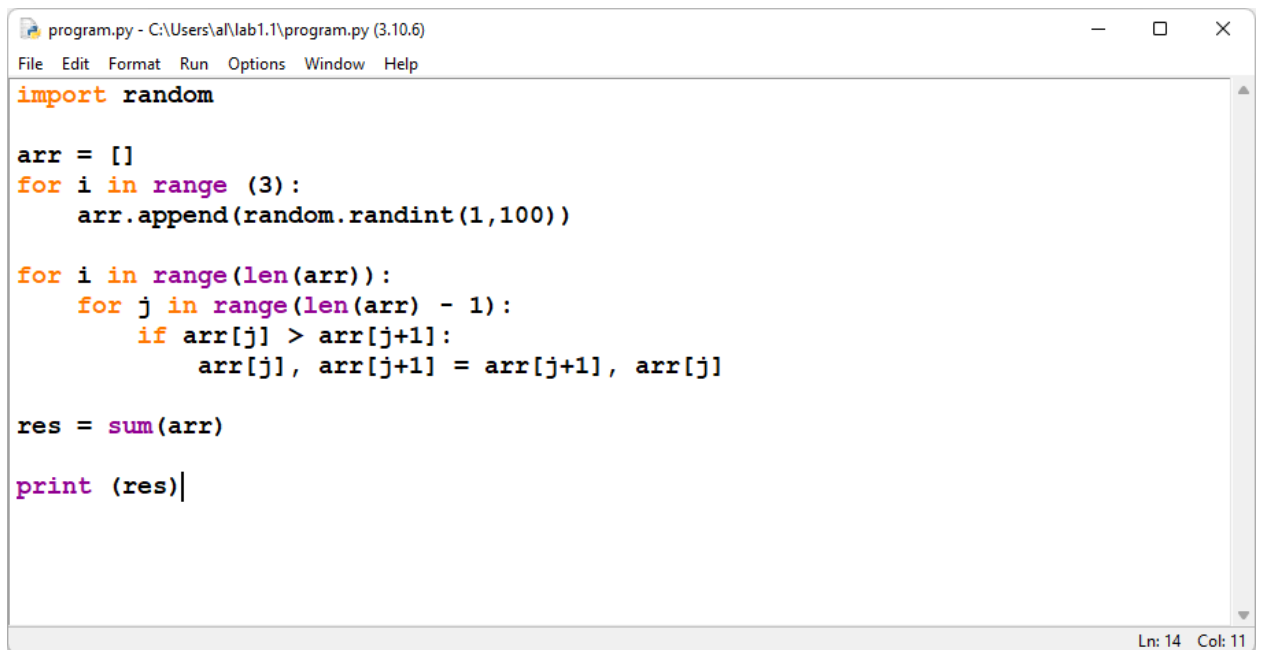
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Line for tests removed"
[main e624c18] Line for tests removed
 1 file changed, 1 deletion(-)

C:\Users\al\lab1.1>
```

Рисунок 17 Окно командной строки



The image shows a window titled "program.py - C:\Users\al\lab1.1\program.py (3.10.6)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
import random

arr = []
for i in range(3):
    arr.append(random.randint(1,100))

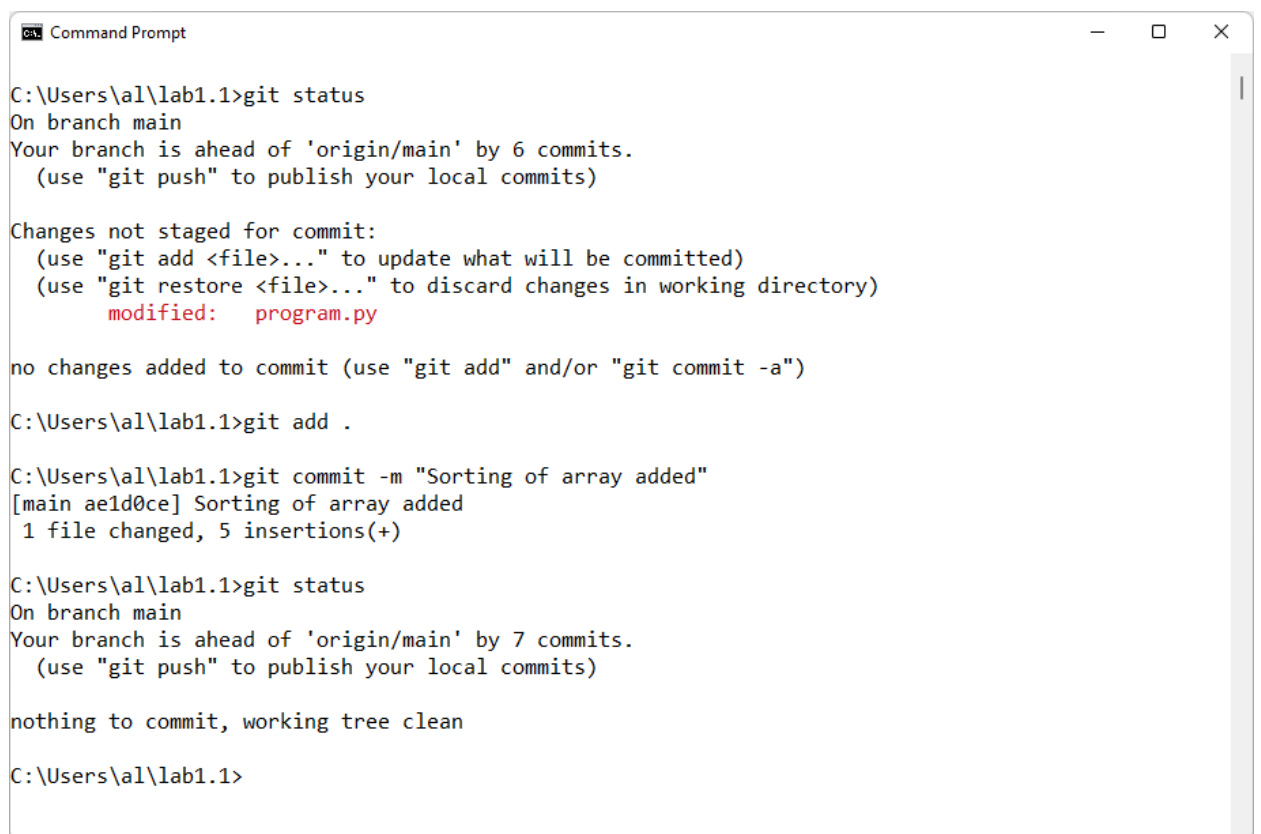
for i in range(len(arr)):
    for j in range(len(arr) - 1):
        if arr[j] > arr[j+1]:
            arr[j], arr[j+1] = arr[j+1], arr[j]

res = sum(arr)

print (res)|
```

The status bar at the bottom right indicates "Ln: 14 Col: 11".

Рисунок 18 Окно IDE



The image shows a "Command Prompt" window with the following text:

```
C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 6 commits.
  (use "git push" to publish your local commits)

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:   program.py

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\al\lab1.1>git add .

C:\Users\al\lab1.1>git commit -m "Sorting of array added"
[main ae1d0ce] Sorting of array added
 1 file changed, 5 insertions(+)

C:\Users\al\lab1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 7 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\al\lab1.1>
```

Рисунок 19 Окно командной строки

Редактирование разметки файла README.md и фиксация сделанных изменений, рисунки 20, 21.

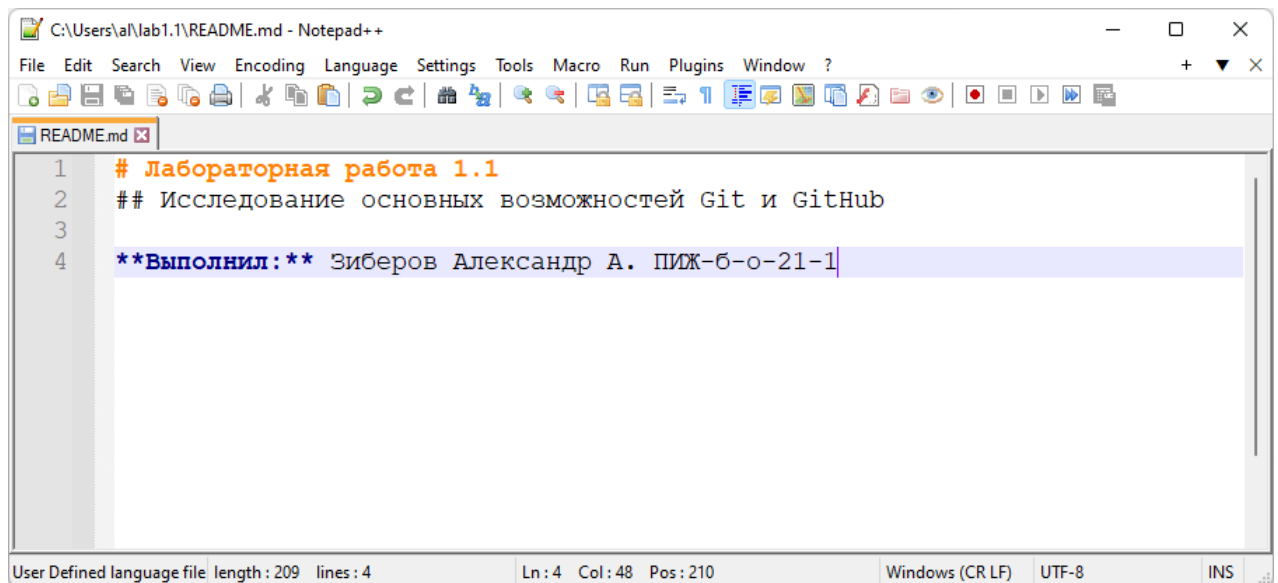


Рисунок 20 Окно Notepad++

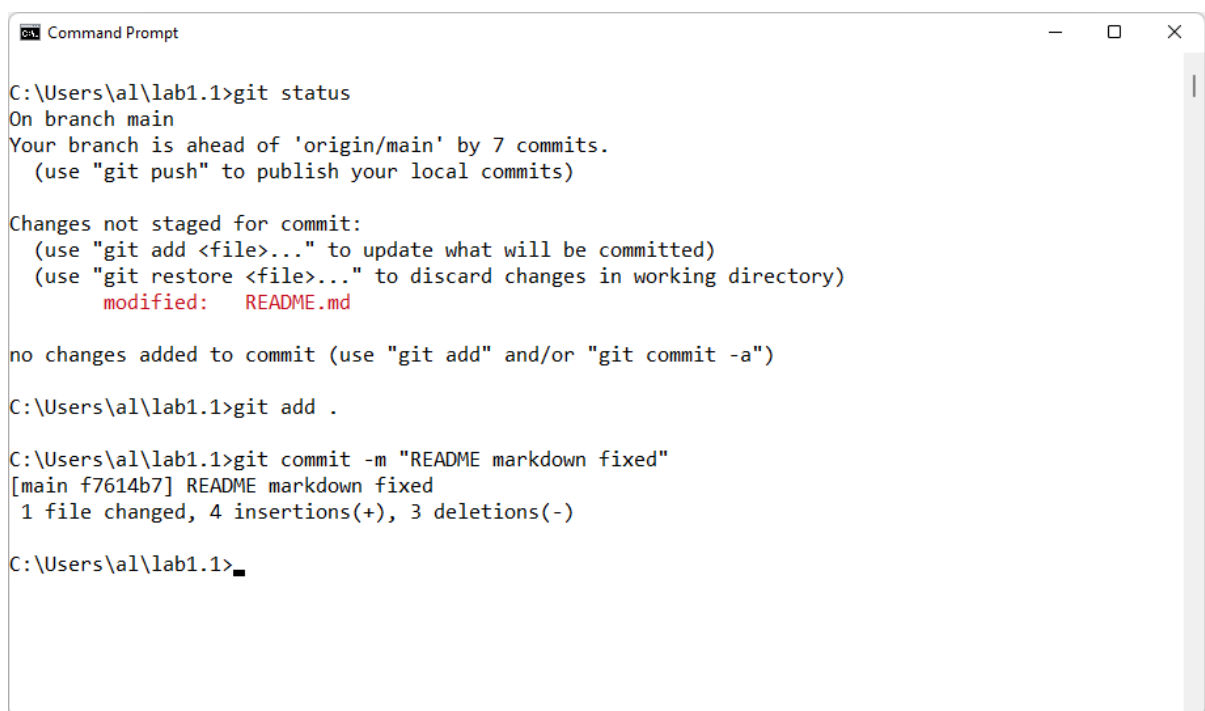


Рисунок 21 Окно командной строки

**Вывод:** При выполнении этой работы была изучена система контроля версий Git и работа с веб-сервисом GitHub.

## **Контрольные вопросы:**

### **1. Что такое СКВ и каково ее назначение?**

Система контроля версий — это система, которая регистрирует и отображает изменения в файлах, а также позволяет вернуться к старым версиям этих файлов (откатиться).

### **2. В чем недостатки локальных и централизованных СКВ?**

Недостатки локальных СКВ:

- большая вероятность запутаться в копиях (изменить файлы не той версии или заменить не той версией текущую)
- при выходе из строя носителя с файлами при отсутствии бэкапов можно потерять весь проект

Недостатки централизованных СКВ:

- при падении сервера никто не сможет воспользоваться СКВ
- при неисправности носителя с БД при отсутствии бэкапов можно потерять весь проект

### **3. К какой СКВ относится Git?**

Git относится к распределённым системам контроля версий. Распределенные СКВ копируют весь репозиторий, что является полной копией проекта и в случае выхода из строя одного сервера, эту копию можно развернуть на другом без потери данных.

### **4. В чем концептуальное отличие Git от других СКВ?**

Git, в отличие от распределенных СКВ, не хранит информацию об изменениях в каждом файле, а использует систему «снимков». После каждого коммита, система запоминает как выглядит каждый файл и сохраняет ссылку на этот снимок. Если файл не изменился, Git не запоминает снова этот файл, а создает ссылку на предыдущую его версию.

## **5. Как обеспечивается целостность хранимых данных в Git?**

Целостность хранимых данных в Git обеспечивается вычислением хэш-сумм вида SHA-1 (40 шестнадцатиричных символов). Эта функциональность встроена в Git на низком уровне. А также, Git сохраняет все объекты в свою базу данных не по имени, а по хеш-сумме содержимого объекта.

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

У файлов в Git есть три состояния:

- committed (зафиксированный): файл уже сохранен в локальном репозитории с изменением;
- modified (измененный): файл, который изменился, но изменения не были зафиксированы;
- staged (подготовленный): измененные файлы, отмеченные для следующего коммита.

## **7. Что такое профиль пользователя в GitHub?**

Профиль пользователя в GitHub — персональная страница программиста, на которой можно увидеть информацию о нём, а также публичные репозитории и его действия в других проектах. Работодатели могут посмотреть и принять во внимание профиль при приёме на работу.

## **8. Какие бывают репозитории в GitHub?**

Репозитории в профиле GitHub: public (публичный), private (частный).

Репозитории по месту хранения: локальный репозиторий (копия проекта на ПК), удаленный репозиторий (репозиторий на GitHub).

Также, можно сделать форк репозитория (origin) и проводить изменения в нём, не внося изменений в оригинальный (upstream).

## **9. Укажите основные этапы модели работы с GitHub.**

1. Клонирование репозитория локально;
2. Внесение изменений в локальном репозитории;
3. Commit и push в репозиторий GitHub;

## **10. Как осуществляется первоначальная настройка Git после установки?**

После установки и проверки работоспособности Git необходимо добавить переменные `user.name` и `user.email` в конфиг. Для этого нужно выполнить следующие команды: `git config --global user.name <YOUR_NAME>` и `git config --global user.email <EMAIL>`.

## **11. Опишите этапы создания репозитория в GitHub.**

1. Имя репозитория
2. Описание репозитория
3. Вид репозитория: публичный или приватный.
4. Включение/отключение создания файла README.md
5. Выбор файла .gitignore
6. Выбор лицензии

## **12. Какие типы лицензий поддерживаются GitHub при создании репозитория?**

Apache License 2.0, GNU General Public License v3.0, MIT License, BSD 2-Clause "Simplified" License, BSD 3-Clause "New" or "Revised" License, Boost Software License 1.0, Creative Commons Zero v1.0 Universal, Eclipse Public License 2.0, GNU Affero General Public License v3.0, GNU General Public License v2.0, GNU Lesser General Public License v2.1, Mozilla Public License 2.0, The Unlicense.

Лицензия MIT — лицензия открытого и свободного программного обеспечения. Является одной из самых ранних свободных лицензий. Она является разрешительной лицензией, то есть позволяет программистам

использовать лицензируемый код в закрытом программном обеспечении при условии, что текст лицензии предоставляется вместе с этим программным обеспечением.

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

Клонирование репозитория осуществляется при помощи Git командой `git clone <link>`. Клонировать репозиторий нужно для хранения локальной копии с возможностью осуществления и применения дальнейших изменений.

**14. Как проверить состояние локального репозитория Git?**

Командой `git status`.

**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` ?**

Добавление/изменение файла в локальном репозитории Git: `untracked files`.

Добавление нового/ измененного файла под версионный контроль с помощью команды `git add: staged`.

Фиксация (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push: up to date`.

**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`.

1. `git clone <rep. link>`
2. `git add .`
3. `git commit -m "Something new"`
4. `git push`
5. `git pull`

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

Аналоги GitHub: GitLab, BitBucket, Launchpad, GitFlic.

В GitLab разрешение предоставляется на основе ролей людей, в то время как в GitHub разработчики могут предоставлять доступ на чтение или запись к определенным репозиториям. GitLab предлагает бесплатные частные репозитории для проектов с открытым исходным кодом, а GitHub - нет. GitHub способен предоставить полную историю обновлений комментариев - GitLab не поддерживает это. GitLab предоставляет панель мониторинга для анализа времени, планирования и мониторинга.

**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.**

Для GitHub существует приложение с интерфейсом на основе Electron – GitHub Desktop. При первом запуске, приложение потребует входа в учетную запись. Из приложения можно создавать репозитории, клонировать или добавить локальную копию и все это делается в несколько кликов в верхнем

меню программы. Также, при внесении изменений, программа сразу их показывает в том же виде, как и на сайте GitHub.

1. Клонирование репозитория: File -> Clone Repository -> Clone.
2. Изменения автоматически отображаются, добавлять на контроль не нужно.
3. В панели под измененными файлами можно написать описание коммита и закоммитить изменения в ветку.
4. Для пуша в GitHub нужно нажать кнопку Push origin.