

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

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.18

Дисциплина: «Программирование на Python»

Тема: «Работа с переменными окружения в Python3»

Выполнил: студент 2 курса

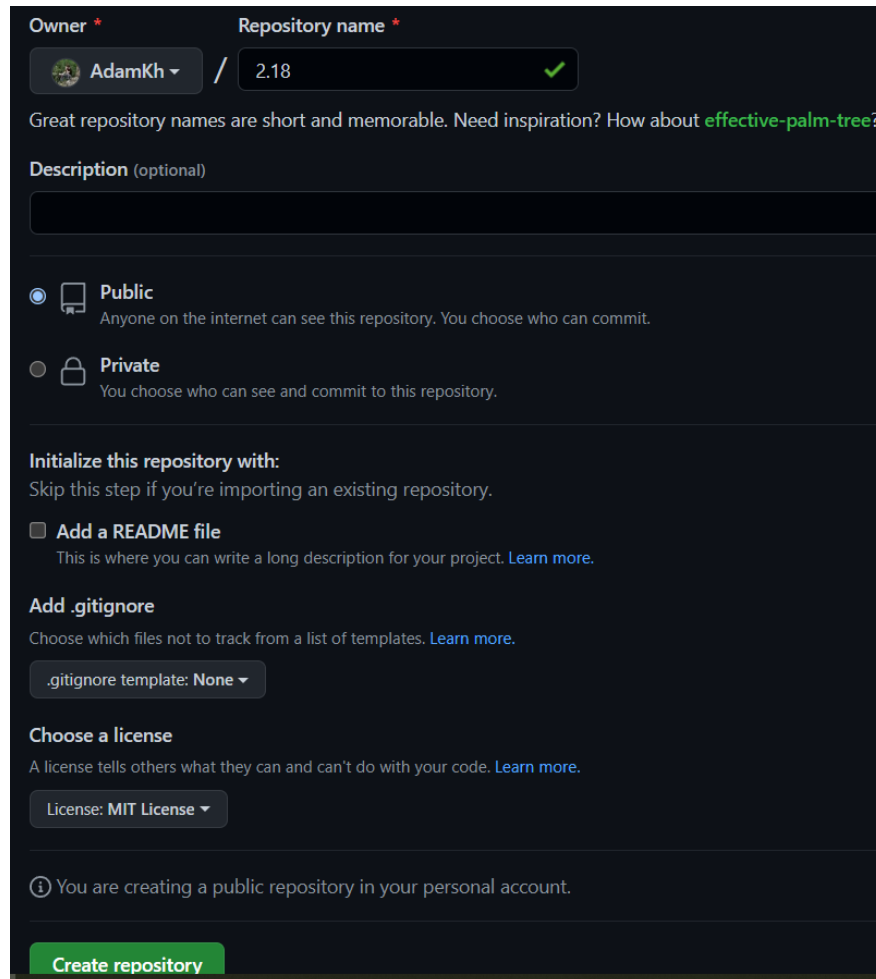
группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.6» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.



Owner * AdamKh / Repository name * 2.18 ✓

Great repository names are short and memorable. Need inspiration? How about [effective-palm-tree?](#)

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:
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: None ▾

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

License: MIT License ▾

i You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\Users\adamkh\Desktop\3sem\Python\2.17>git clone https://github.com/AdamKh/2.17.git
Cloning into '2.17'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 1.2 Клонирование репозитория

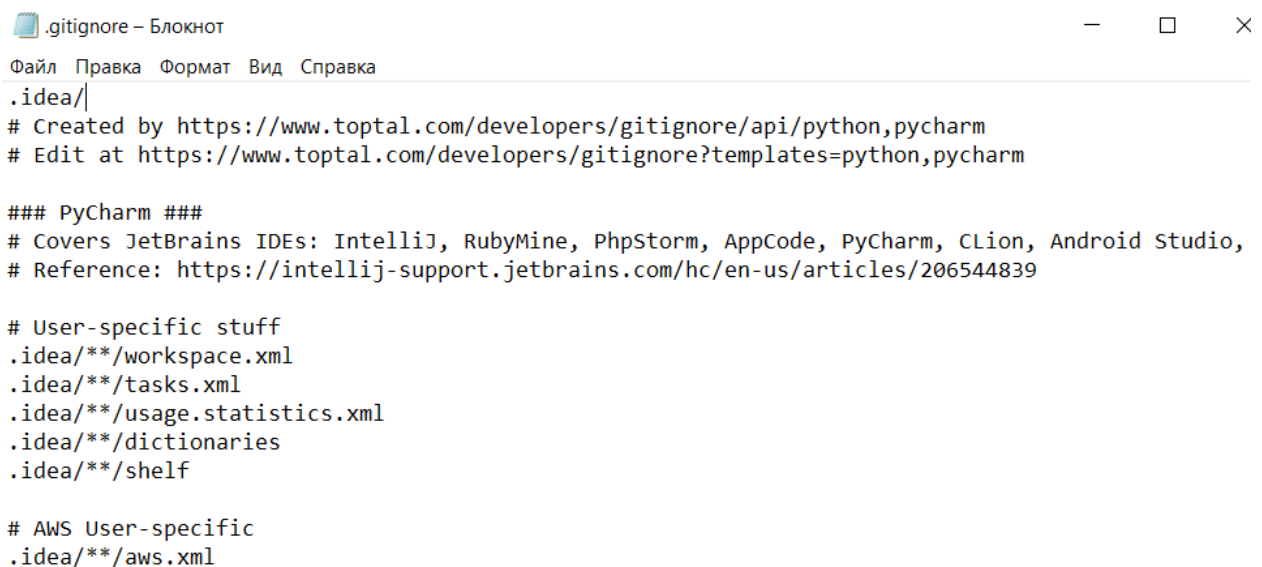
```

C:\Users\adamkh\Desktop\3sem\Python\2.18>git flow init
Initialized empty Git repository in C:/Users/adamkh/Desktop/3sem/Python/2.18/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/adamkh/Desktop/3sem/Python/2.18/.git/hooks]
C:\Users\adamkh\Desktop\3sem\Python\2.18>

```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления git-flow



```

.gitignore – Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio,
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

```

Рисунок 1.4 Изменение .gitignore

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

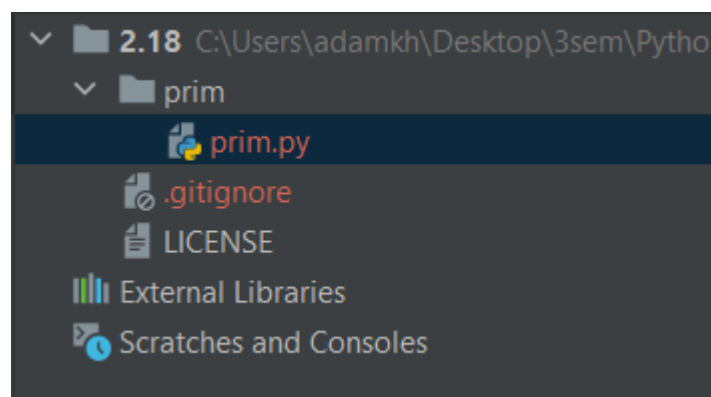


Рисунок 2.1 Создание проекта в PyCharm

О программе

Ваш компьютер защищен.

[Просмотреть сведения в разделе "Безопасность Windows"](#)

Характеристики устройства

Имя устройства	DESKTOP-6FFGPOM
Процессор	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
Оперативная память	8,00 ГБ (доступно: 7,43 ГБ)
Код устройства	5F08C24A-08E0-4443-8113-7F4C19C82081

[Сопутствующие параметры](#)

[Параметры BitLocker](#)

[Диспетчер устройств](#)

[Удаленный рабочий стол](#)

[Защита системы](#)

[Дополнительные параметры системы](#)

[Переименовать этот ПК \(для опытных пользователей\)](#)

Рисунок 2.2 Свойства «Этот компьютер»

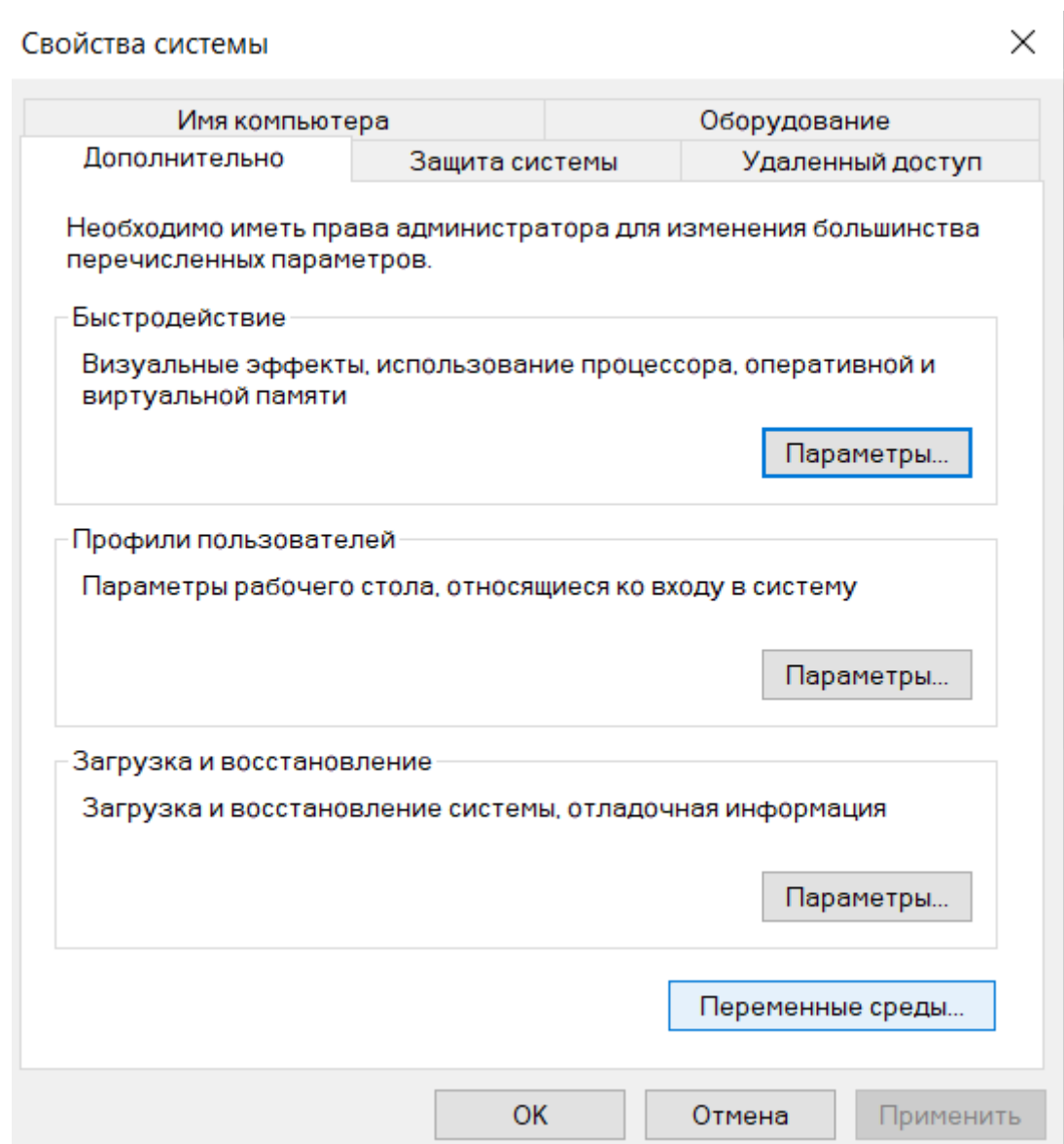


Рисунок 2.3 Доп. параметры с-мы

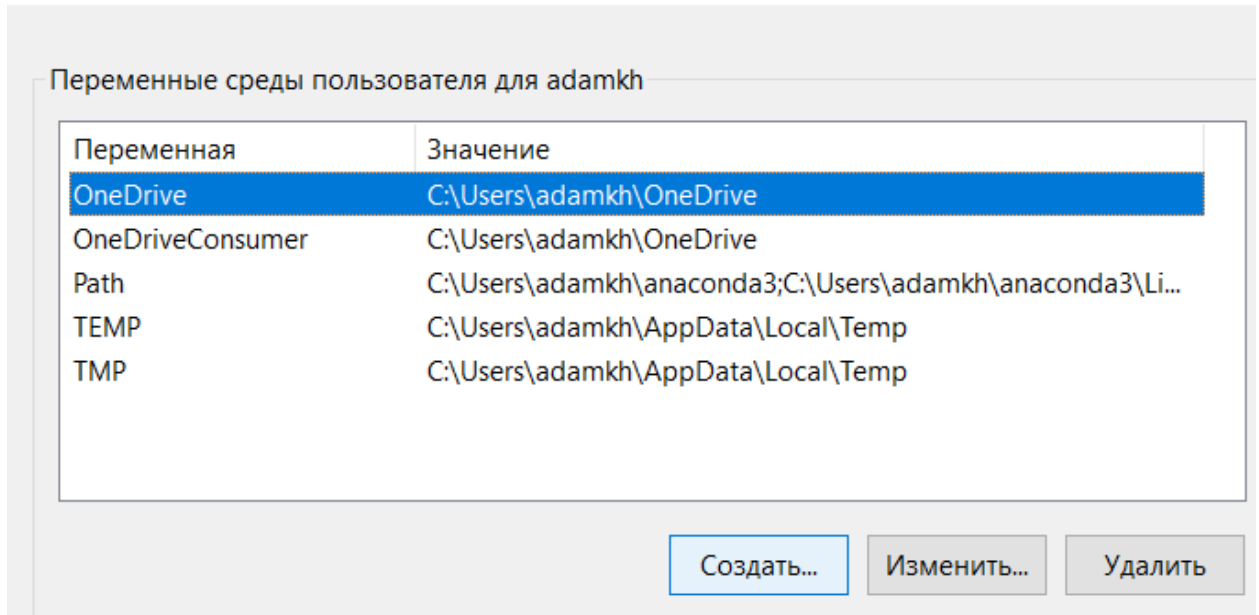


Рисунок 2.4 Создание новой переменной среды

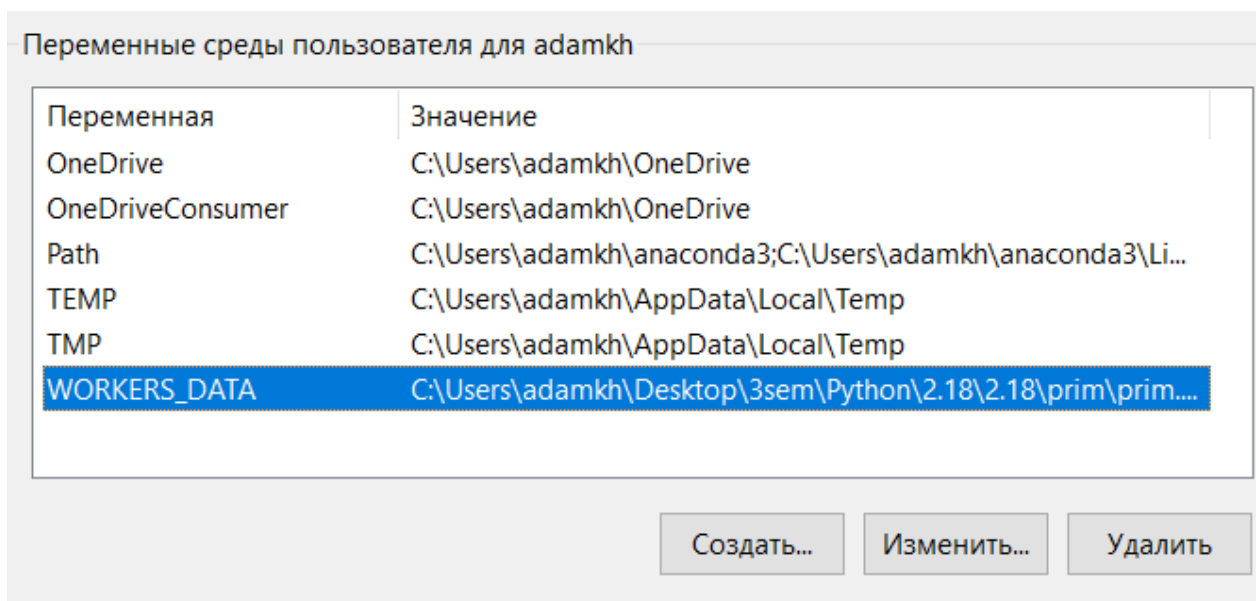


Рисунок 2.5 Созданная новая переменная среда

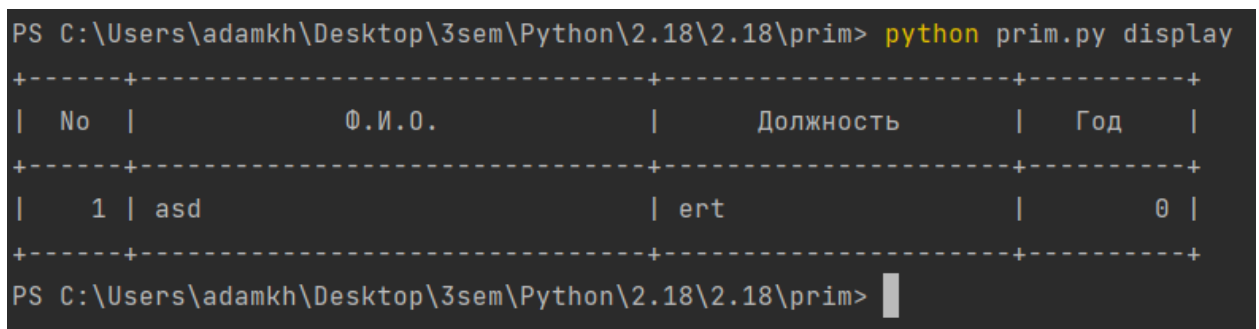


Рисунок 2.6 Вызов команды display без указания файла с данными
3. (15 вариант). Выполнил индивидуальные задания.

```

172     # Выполнить разбор аргументов командной строки.
173     args = parser.parse_args(command_line)
174
175     data_file = args.data
176     if not data_file:
177         data_file = os.environ.get("STUDENTS_DATA")
178     if not data_file:
179         print("The data file name is absent", file=sys.stderr)
180         sys.exit(1)
181
182     # Получить имя файла.
183     data_file = args.data

```

Рисунок 3.1 Изменения в коде программы

```

C:\Users\adamkh\Desktop\3sem\Python\2.18\2.18\ind>python ind1.py display
+-----+-----+-----+-----+
| № | Фамилия и имя | Знак Зодиака | Дата рождения |
+-----+-----+-----+-----+
| 1 | qwe asd | ert | 11/11/1111111 |
| 2 | Skalette Vito | Oven | 22/11/1943 |
| 3 | Bdsaaa ASdqw | Asdwq | 02/01/1995 |
+-----+-----+-----+-----+
C:\Users\adamkh\Desktop\3sem\Python\2.18\2.18\ind>

```

Рисунок 3.2 Вывод программы индивидуального задания №1

```

172     # Выполнить разбор аргументов командной строки.
173     args = parser.parse_args(command_line)
174
175     data_file = args.data
176     dotenv_path = os.path.join(os.path.dirname(__file__), ".env")
177     if os.path.exists(dotenv_path):
178         load_dotenv(dotenv_path)
179     if not data_file:
180         data_file = os.getenv("HUMANS_DATA")
181     if not data_file:
182         print("The data file name is absent", file=sys.stderr)
183         sys.exit(1)
184
185     # Загрузить всех людей из файла, если файл существует.

```

Рисунок 3.1 Изменения в коде программы

```

. .env
C: > Users > adamkh > Desкто
1 HUMANS_DATA

```

Рисунок 3.2 Содержимое файла .env

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.18\2.18\ind>python ind2.py display
```

№	Фамилия и имя		Знак Зодиака	Дата рождения
1	qwe	asd	ert	11/11/1111111
2	Skalette	Vito	Oven	22/11/1943
3	Bsdsaaa	ASdqw	Asdwq	02/01/1995

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.18\2.18\ind>
```

Рисунок 3.3 Вывод программы индивидуального задания №2

5. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\rep_2.6>git add .
C:\rep_2.6>git commit -m "added programs + modidied .gitignore"
[develop 2582c62] added programs + modidied .gitignore
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
C:\rep_2.6>
```

```
C:\rep_2.6>git push
Everything up-to-date
```

Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```
C:\rep_2.6>git merge develop
Updating 5d4b8d1..2582c62
Fast-forward
 .gitignore | 157 ++++++
--
 ind.py      | 105 ++++++
 prim.py     | 99  ++++++
 zadaniya.py | 21  ++++++
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\rep_2.6>git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 5.16 KiB | 2.58 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AdamKh/rep_2.6.git
5d4b8d1..2582c62 main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

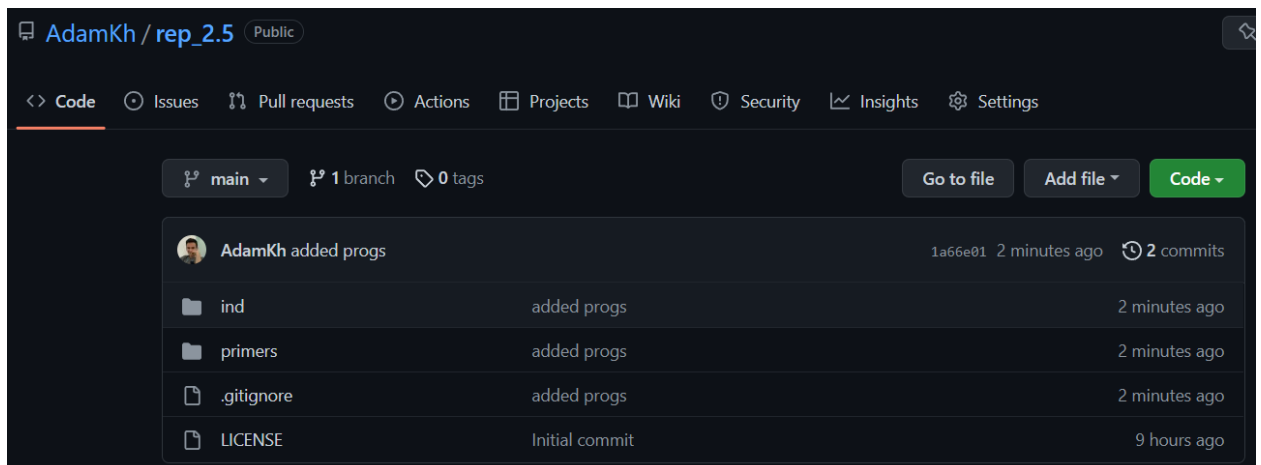


Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. Каково назначение переменных окружения?

Переменные окружения используются для передачи информации процессам, которые запущены в оболочке.

2. Какая информация может храниться в переменных окружения?

Переменные среды хранят информацию о среде операционной системы.

Эта информация включает такие сведения, как путь к операционной системе, количество процессоров, используемых операционной системой, и расположение временных папок.

3. Как получить доступ к переменным окружения в ОС Windows?

Нужно открыть окно свойства системы и нажать на кнопку “Переменные среды”.

4. Каково назначение переменных PATH и PATHEXT?

PATH позволяет запускать исполняемые файлы и скрипты, «лежащие» в определенных каталогах, без указания их точного местоположения.

PATHEXT дает возможность не указывать даже расширение файла, если оно прописано в ее значениях.

5. Как создать или изменить переменную окружения в Windows?

В окне “Переменные среды” нужно нажать на кнопку “Создать”, затем ввести имя переменной и путь.

6. Что представляют собой переменные окружения в ОС Linux?

Переменные окружения в Linux представляют собой набор именованных значений, используемых другими приложениями.

7. В чем отличие переменных окружения от переменных оболочки?

Переменные окружения (или «переменные среды») – это переменные, доступные в масштабах всей системы и наследуемые всеми дочерними процессами и оболочками.

Переменные оболочки — это переменные, которые применяются только к текущему экземпляру оболочки. Каждая оболочка, например, bash или zsh, имеет свой собственный набор внутренних переменных.

8. Как вывести значение переменной окружения в Linux?

Наиболее часто используемая команда для вывода переменных окружения – `printenv`.

9. Какие переменные окружения Linux Вам известны?

USER — текущий пользователь.

PWD – текущая директория.

HOME – домашняя директория текущего пользователя.

SHELL – путь к оболочке текущего пользователя.

EDITOR – заданный по умолчанию редактор. Этот редактор будет вызываться в ответ на команду `edit`.

LOGNAME – имя пользователя, используемое для входа в систему.

PATH – пути к каталогам, в которых будет производиться поиск вызываемых команд. При выполнении команды система будет проходить по данным каталогам в указанном порядке и выберет первый из них, в котором будет находиться исполняемый файл искомой команды.

LANG – текущие настройки языка и кодировки. TERM – тип текущего эмулятора терминала.

MAIL – место хранения почты текущего пользователя. LS_COLORS задает цвета, используемые для выделения объектов.

10. Какие переменные оболочки Linux Вам известны?

BASHOPTS – список задействованных параметров оболочки, разделенных двоеточием.

BASH_VERSION – версия запущенной оболочки bash.

COLUMNS – количество столбцов, которые используются для отображения выходных данных.

DIRSTACK – стек директорий, к которому можно применять команды pushd и popd.

HISTFILESIZE – максимальное количество строк для файла истории команд.

HISTSIZE – количество строк из файла истории команд, которые можно хранить в памяти.

HOSTNAME – имя текущего хоста.

IFS – внутренний разделитель поля в командной строке.

PS1 – определяет внешний вид строки приглашения ввода новых команд.

PS2 – вторичная строка приглашения.

SHELLOPTS – параметры оболочки, которые можно устанавливать спомощью команды set.

UID – идентификатор текущего пользователя.

11. Как установить переменные оболочки в Linux?

Чтобы создать новую переменную оболочки с именем, нужно ввести имя этой переменной потом знак равенства и указать значение новой переменной

12. Как установить переменные окружения в Linux?

Команда `export` используется для задания переменных окружения.

С помощью данной команды мы экспортируем указанную переменную, в результате чего она будет видна во всех вновь запускаемых дочерних командных оболочках.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Чтобы переменная сохранялась после закрытия сеанса оболочки.

14. Для чего используется переменная окружения PYTHONHOME?

Переменная среды PYTHONHOME изменяет расположение стандартных библиотек Python.

15. Для чего используется переменная окружения PYTHONPATH?

Переменная среды PYTHONPATH изменяет путь поиска по умолчанию для файлов модуля.

16. Какие еще переменные окружения используются для управления работой интерпретатора Python?

PYTHONSTARTUP PYTHONOPTIMIZE PYTHONBREAKPOINT
PYTHONDEBUG PYTHONINSPECT PYTHONUNBUFFERED
PYTHONVERBOSE PYTHONCASEOK PYTHONDONTWRITEBYTECODE
PYTHONPYCACHEPREFIX PYTHONHASHSEED PYTHONIOENCODING
PYTHONNOUSERSITE PYTHONUSERBASE PYTHONWARNINGS
PYTHONFAULTHANDLER

17. Как осуществляется чтение переменных окружения в программах на языке программирования Python?

Путём использования модуля `os`, при помощи которого программист может получить и изменить значения всех переменных среды.

18. Как проверить, установлено или нет значение переменной окружения в программах на языке программирования Python?

При помощи модуля `os` можно просмотреть все переменные окружения, у которых есть значение.

19. Как присвоить значение переменной окружения в программах на языке программирования Python?

Для присвоения значения любой переменной среды используется функция `setdefault()`.