

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 №21
дисциплины «Основы программной инженерии»

Выполнил:
Матвеев Александр Иванович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

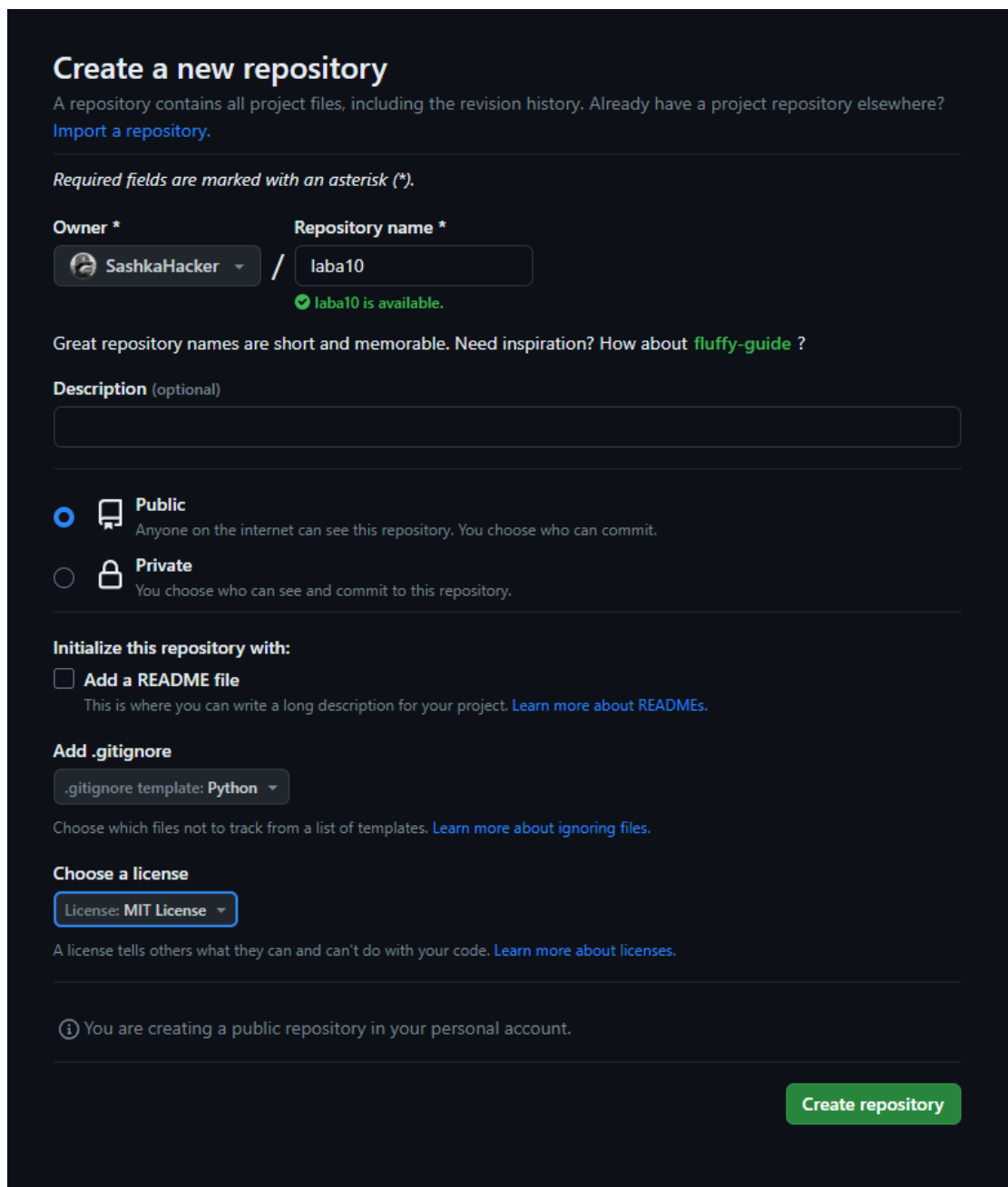
Ставрополь, 2024 г.

Тема: Работа с переменными окружения в Python3.

Цель работы: Приобретение навыков по работе с переменными окружения с помощью языка программирования Python версии 3.x.

Ход работы.

1. Создание нового репозитория с лицензией MIT.



Create a new repository

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

Required fields are marked with an asterisk ().*

Owner * SashkaHacker / **Repository name *** laba10

✔ laba10 is available.

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

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.](#)

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

Create repository

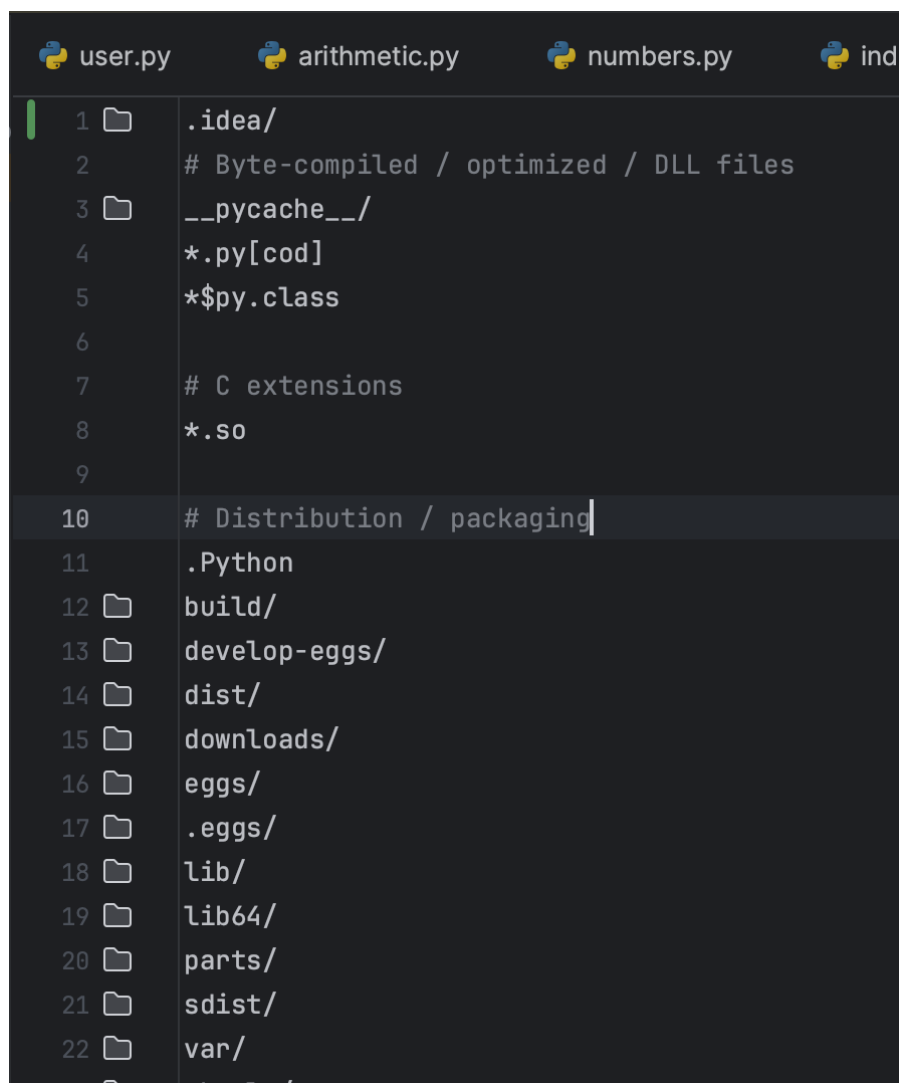
Рисунок 1 – Создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
[→ GitHub git clone https://github.com/SashkaHacker/laba16.git
Cloning into 'laba16'...
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.
```

Рисунок 2 – Клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.



The image shows a code editor with a dark theme. At the top, there are four tabs: 'user.py', 'arithmetic.py', 'numbers.py', and 'ind'. The active tab is 'user.py'. The editor displays the content of a '.gitignore' file. The file is numbered from 1 to 22. The content is as follows:

```
1  .idea/
2  # Byte-compiled / optimized / DLL files
3  __pycache__/
4  *.py[cod]
5  *$py.class
6
7  # C extensions
8  *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
```

Рисунок 3 – Файл .gitignore

4. Проработка примера из лабораторной работы.

```
WORKERS_DATA=data.json

C:\Users\Sashka\Documents\GitHub\laba21\examples>python ex1.py display
```

№	Ф.И.О.	Должность	Год
1	Иванов Иван Иванович	Инженер	2015
2	Петров Петр Петрович	Разработчик	2010
3	Сидорова Мария Ивановна	Аналитик	2012
4	Кузнецов Сергей Петрович	Тестировщик	2018
5	Смирнова Ольга Николаевна	Менеджер проектов	2014

Рисунок 4 – Результат работы примера №1

5. Выполнение индивидуального задания №1. Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

```
(venv) → laba21 git:(develop) ✖ python3 individual1.py display
```

№	Фамилия	Имя	Номер телефона	Дата рождения
1	Иванов	Иван Иванович	89031234567	01:01:2005
2	Петров	Петр Петрович	89032345678	15:05:2008
3	Сидоров	Сидор Сидорович	89033456789	23:03:2010
4	Васильев	Василий Васильевич	89034567890	30:08:2012
5	Алексеев	Алексей Алексеевич	89035678901	17:11:2014

Рисунок 5 – Результат работы индивидуального задания №1

6. Выполнение индивидуального задания №2. Самостоятельно изучите работу с пакетом python-dotenv. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла .env.

```
(venv) → lab21 git:(develop) ✖ python3 individual2.py display
```

№	Фамилия	Имя	Номер телефона	Дата рождения
1	Иванов	Иван Иванович	89031234567	01:01:2005
2	Петров	Петр Петрович	89032345678	15:05:2008
3	Сидоров	Сидор Сидорович	89033456789	23:03:2010
4	Васильев	Василий Васильевич	89034567890	30:08:2012
5	Алексеев	Алексей Алексеевич	89035678901	17:11:2014

Рисунок 6 – Результат работы индивидуального задания №2

Контрольные вопросы:

1. Переменные окружения используются для хранения конфигурационной информации для процессов, запущенных в операционной системе.
2. В переменных окружения могут храниться пути к файлам, параметры сети, настройки локализации и другие технические данные.
3. В ОС Windows доступ к переменным окружения можно получить через системный диалог "Системные свойства" или используя команды `echo %ИМЯ_ПЕРЕМЕННОЙ%` в Command Prompt и `$Env:ИМЯ_ПЕРЕМЕННОЙ` в PowerShell.
4. Переменная `PATH` содержит список каталогов, в которых операционная система ищет исполняемые файлы, а `PATHEXT` определяет расширения файлов, которые считаются исполняемыми.
5. Создать или изменить переменную окружения в Windows можно через "Системные свойства" -> "Дополнительно" -> "Переменные среды" или с помощью команды `setx` в Command Prompt.
6. В ОС Linux переменные окружения представляют собой глобальные настройки, которые влияют на работу оболочки и запущенных в ней программ.
7. Переменные окружения доступны для всех процессов в системе, в то время как переменные оболочки существуют только в рамках текущей сессии оболочки.
8. Вывести значение переменной окружения в Linux можно с помощью команды `echo $ИМЯ_ПЕРЕМЕННОЙ`.
9. Известные переменные окружения Linux включают `HOME`, `PATH`, `LANG`, `SHELL` и другие.
10. Известные переменные оболочки Linux включают `PS1` (приглашение командной строки), `PWD` (текущий рабочий каталог) и `OLDPWD` (предыдущий рабочий каталог).
11. Установить переменные оболочки в Linux можно с помощью команды `export` в оболочке.

12. Установить переменные окружения в Linux можно также с помощью команды `export` или добавив их в файлы, такие как `~/.bashrc` или `~/.profile`.

13. Сделать переменные окружения Linux постоянными необходимо для сохранения этих переменных между перезапусками системы или новыми сессиями оболочки.

14. Переменная окружения `PYTHONHOME` указывает на каталог установки Python и используется для определения местоположения стандартных библиотек.

15. Переменная окружения `PYTHONPATH` используется для добавления дополнительных каталогов, в которых Python будет искать модули и пакеты.

16. Другие переменные окружения для управления работой интерпретатора Python включают `PYTHONIOENCODING` для кодировки ввода/вывода и `PYTHONUNBUFFERED` для управления буферизацией вывода.

17. Чтение переменных окружения в программах на Python осуществляется с помощью модуля `os`, например, через `os.environ` или `os.getenv()`.

18. Проверить, установлено ли значение переменной окружения в Python, можно с помощью `os.getenv()` и проверки возвращаемого значения на `None`.