

**РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное**  
**образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**  
**«Работа с переменными окружения в Python3»**

**Отчет по лабораторной работе № 2.18**  
**по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«19» декабря 2022г.

Подпись студента \_\_\_\_\_

Работа защищена «    » \_\_\_\_\_ 20\_\_ г.

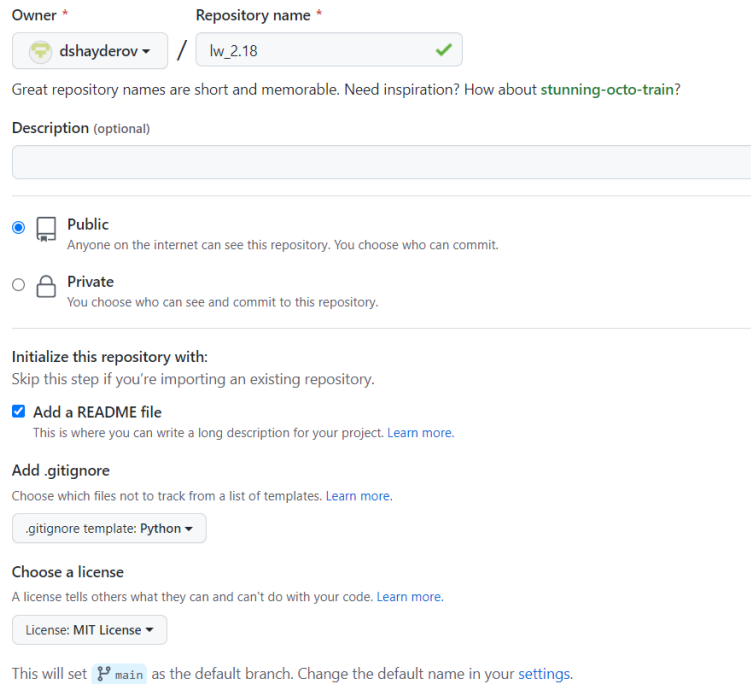
Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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

**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.



Owner <sup>\*</sup> dshayderov / Repository name <sup>\*</sup> lw\_2.18 ✓

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

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.](#)

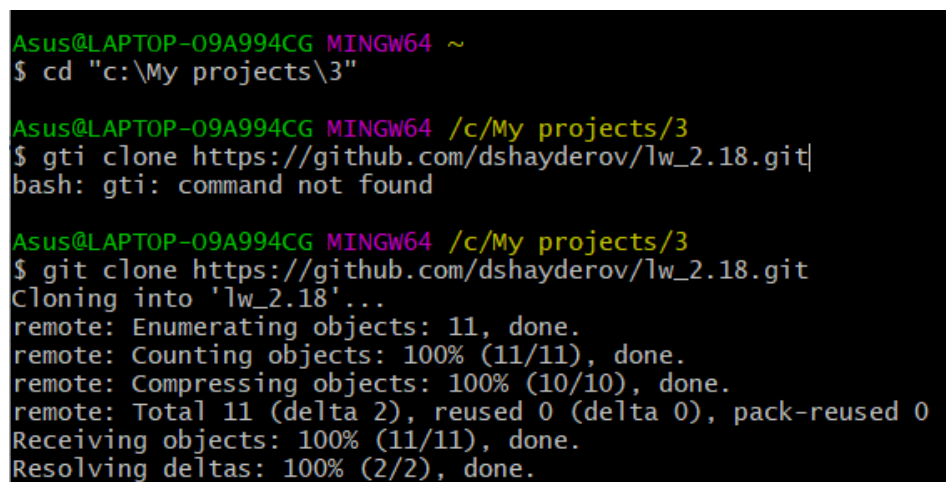
**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)  
.gitignore template: Python ▾

**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](#).

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.



```
Asus@LAPTOP-09A994CG MINGW64 ~
$ cd "c:\My projects\3"

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3
$ gti clone https://github.com/dshayderov/lw_2.18.git
bash: gti: command not found

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3
$ git clone https://github.com/dshayderov/lw_2.18.git
Cloning into 'lw_2.18'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3
$ cd lw_2.18

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3/lw_2.18 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/3/lw_2.18 (develop)
$

```

Рисунок 3 - Ветвление по модели git-flow

#### 4. Создание виртуального окружения.

```

C:\My projects\3\lw_2.18>python -m venv .venv

C:\My projects\3\lw_2.18>.venv\Scripts\activate

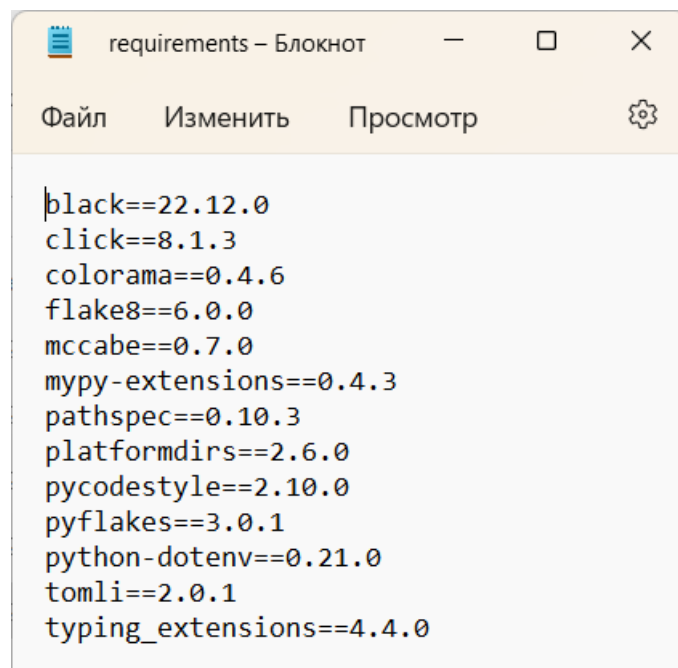
(.venv) C:\My projects\3\lw_2.18>pip install black, flake8
ERROR: Invalid requirement: 'black,'
WARNING: You are using pip version 21.2.3; however, version 22.3.1 is
available. You should consider upgrading via the 'C:\My projects\3\lw_2.18\venv
\Scripts\python.exe --upgrade' command.

(.venv) C:\My projects\3\lw_2.18>pip install black flake8
Collecting black
  Downloading black-22.12.0-cp39-cp39-win_amd64.whl (1.2 MB)
    | 1.2 MB 1.3 MB/s
Collecting flake8
  Using cached flake8-6.0.0-py2.py3-none-any.whl (57 kB)

```

Рисунок 4 - Виртуальное окружение

#### 5. Формирование файла requirements.txt.



```

black==22.12.0
click==8.1.3
colorama==0.4.6
flake8==6.0.0
mccabe==0.7.0
mypy_extensions==0.4.3
pathspec==0.10.3
platformdirs==2.6.0
pycodestyle==2.10.0
pyflakes==3.0.1
python-dotenv==0.21.0
tomli==2.0.1
typing_extensions==4.4.0

```

Рисунок 5 - Файл requirements.txt

#### 6. Создайте проект PyCharm в папке репозитория.

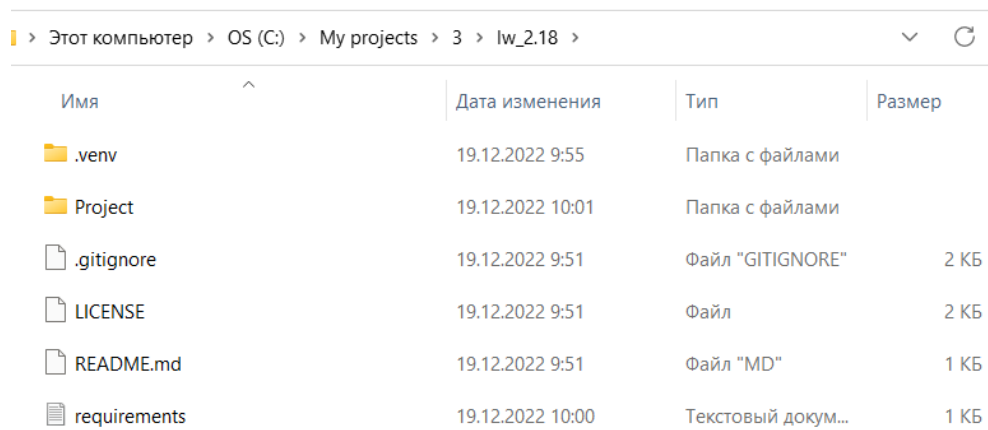


Рисунок 6 - Создание проекта

## 7. Проработать примеры лабораторной работы.

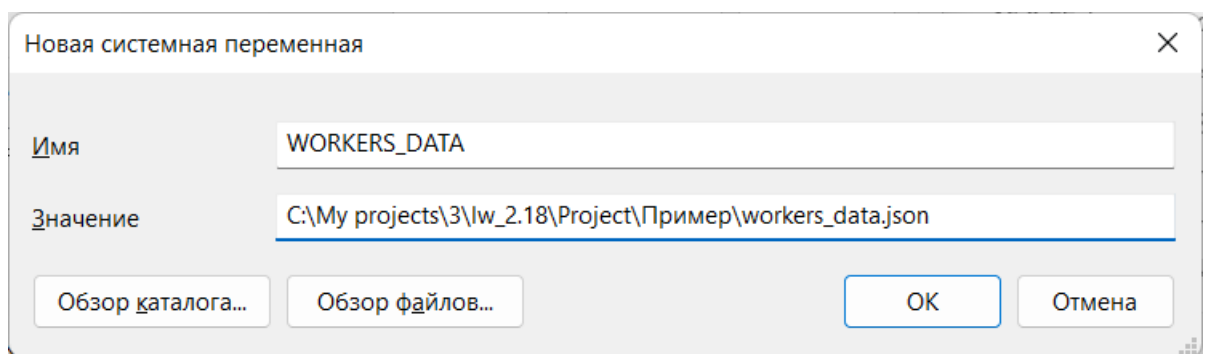


Рисунок 7 - Создание системной переменной WORKERS\_DATA

```
(.venv) C:\My projects\3\lw_2.18\Project\Пример>python primer.py add --name="Сидоров Сидор" --post="Главный инженер"
--year=2012

(.venv) C:\My projects\3\lw_2.18\Project\Пример>python primer.py display
+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Сидоров Сидор    |    Главный инженер   |    2012    |
+-----+-----+-----+-----+

(.venv) C:\My projects\3\lw_2.18\Project\Пример>
```

Рисунок 8 - Результат выполнения примера

## 8. Выполнить индивидуальные задания.

### Задание 1

Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

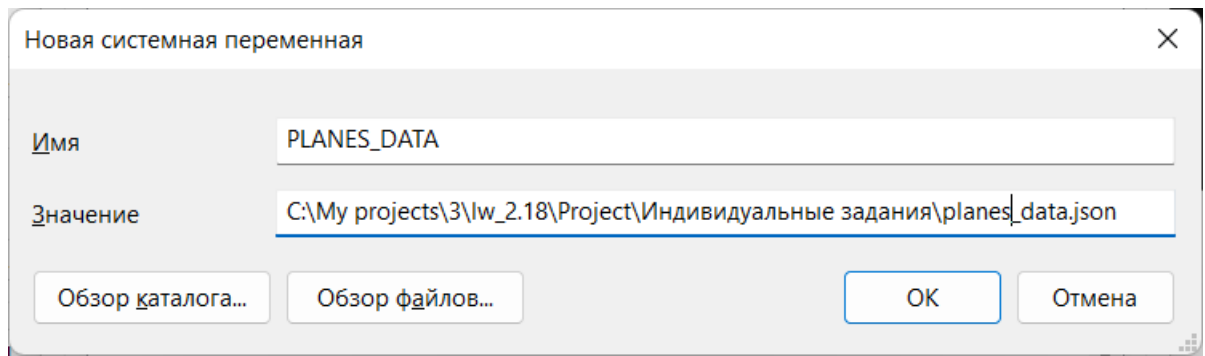


Рисунок 9 - Создание переменной окружения PLANES\_DATA

```
(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind1.py add --destination="Москва" --num=123
--typ="грузовой"

(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind1.py add --destination="Санкт-Петербург"
--num=456 --typ="пассажирский"

(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind1.py display
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Москва	123	грузовой
2	Санкт-Петербург	456	пассажирский

```
(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind1.py select --typ="грузовой"
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Москва	123	грузовой

Рисунок 10 - Результат выполнения задания 1

## Задание 2

Самостоятельно изучите работу с пакетом python-dotenv. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла .env.

```
(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind2.py add --destination="Омск" --num=789
--typ="пассажирский"

(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind2.py add --destination="Сочи" --num=136
--typ="грузовой"

(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind2.py display
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Москва	123	грузовой
2	Санкт-Петербург	456	пассажирский
3	Омск	789	пассажирский
4	Сочи	136	грузовой

```
(.venv) C:\My projects\3\lw_2.18\Project\Индивидуальные задания>python ind2.py select --typ="грузовой"
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Москва	123	грузовой
2	Сочи	136	грузовой

Рисунок 11 - Результат выполнения задания 2

## **Контрольные вопросы:**

### **1. Каково назначение переменных окружения?**

Переменная среды (переменная окружения) – это короткая ссылка на какой-либо объект в системе. С помощью таких сокращений, например, можно создавать универсальные пути для приложений, которые будут работать на любых ПК, независимо от имен пользователей и других параметров.

### **2. Какая информация может храниться в переменных окружения?**

Переменная окружения может хранить информацию о путях к исполняемым файлам, заданном по умолчанию текстовом редакторе, браузере, языковых параметрах (локали) системы или настройках раскладки клавиатуры.

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

Получить информацию о существующих переменных можно в свойствах системы. Для этого кликаем по ярлыку Компьютера на рабочем столе правой кнопкой мыши и выбираем соответствующий пункт.

Переходим в «Дополнительные параметры».

В открывшемся окне с вкладкой «Дополнительно» нажимаем 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, в свою очередь, дает возможность не указывать даже расширение файла, если оно прописано в ее значениях.

### **5. Как создать или изменить переменную окружения в Windows?**

Нажимаем кнопку Создать. Сделать это можно как в пользовательском разделе, так и в системном.

Вводим имя, например, desktop. Обратите внимание на то, чтобы такое название еще не было использовано (просмотрите списки).

В поле Значение указываем 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.

OLDPWD — предыдущая рабочая директория. Используется оболочкой для того, чтобы вернуться в предыдущий каталог при выполнении команды `cd -`.

HOME — домашняя директория текущего пользователя.

SHELL — путь к оболочке текущего пользователя (например, bash или zsh).

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 (например, различные типы файлов в выводе команды `ls` будут выделены разными цветами).

#### **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 (по умолчанию используется пробел).

PS1 — определяет внешний вид строки приглашения ввода новых команд.

PS2 — вторичная строка приглашения.



SHELLOPTS — параметры оболочки, которые можно устанавливать с помощью команды `set` .

UID — идентификатор текущего пользователя.

11. Как установить переменные оболочки в Linux?

Чтобы создать новую переменную оболочки с именем, например, `NEW_VAR` и значением `Ravesli.com` , просто введите:

```
$ NEW_VAR='Ravesli.com'
```

12. Как установить переменные окружения в Linux?

Команда `export` используется для задания переменных окружения.

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`, `PYTHONTRACEMALLOC`,  
`PYTHONPROFILEIMPORTTIME`, `PYTHONASYNCIODEBUG`,  
`PYTHONMALLOC`, `PYTHONMALLOCSTATS`,

PYTHONLEGACYWINDOWSFSENCODING,  
PYTHONLEGACYWINDOWSSTDIO, PYTHONCOERCECLOCALE,  
PYTHONDEVMODE,PYTHONUTF8, PYTHONWARNDEFAULTENCODING,  
PYTHONTHREADDEBUG, PYTHONDUMPREFS.

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

Для доступа к переменным среды в Python используется объект `os.environ`.

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

Для чтения значений переменных мы используем модуль `os`, а модуль `sys` — для прекращения работы приложения.

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

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

**Вывод:** были приобретены навыки по работе с переменными окружения с помощью языка программирования Python версии 3.x.