

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

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

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.18

Тема: «Работа с
переменными окружения в Python3»

Выполнил студент группы

ИВТ-б-о-21-1

Горшков В.И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

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

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

The screenshot shows a GitHub repository interface. At the top, the repository name is 'VitaliyPitsa Update .gitignore' with a commit hash '0f5c87b' and '3 commits'. Below this is a table of commits:

File	Commit Message	Time
.gitignore	Update .gitignore	now
LICENSE	Initial commit	2 minutes ago
README.md	Update README.md	1 minute ago

Below the table is the content of the README.md file:

```
lab13
```

Горшков Виталий ИВТ-б-о-21-1

Рисунок 1.1 – Созданный репозиторий

The screenshot shows a code editor window titled '.gitignore - Блокнот'. The file contains the following content:

```
##
# 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

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
```

Рисунок 1.2 – Дополнил правила в .gitignore

```

c:\Users\Admin\Desktop\git\Python13-2.18>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
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/Admin/Desktop/git/Python13-2.18/.git/hooks]

c:\Users\Admin\Desktop\git\Python13-2.18>

```

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

2. Создадим переменную среды для необходимую для выполнения лабораторной работы. Нажимаем ПКМ по значку «Этот компьютер» на рабочем столе и переходим во вкладку свойства.



Рисунок 2 – Вкладка свойства

Далее выбираем дополнительные параметры. В ней переходим во вкладку переменные среды.

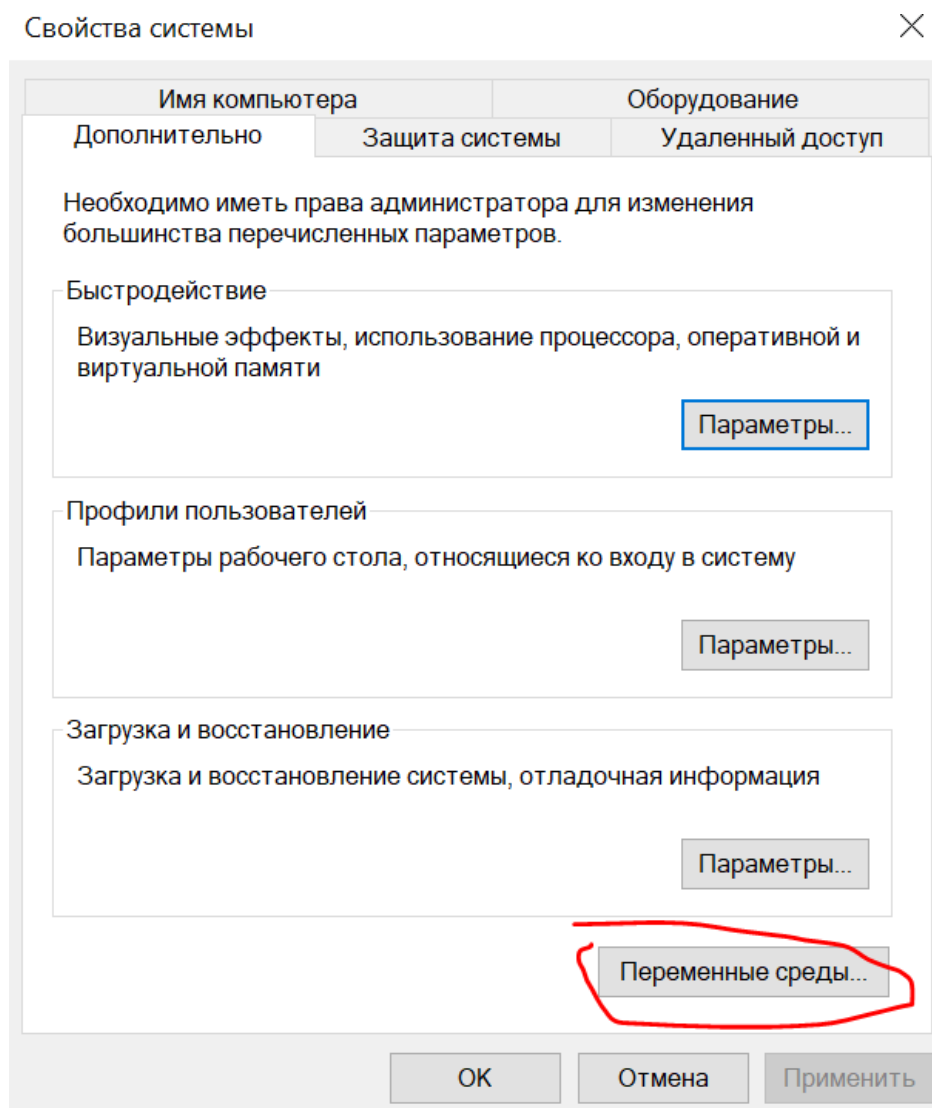


Рисунок 3 – Дополнительные параметры системы

Далее нажимаем «Создать». И указываем путь где будет храниться наша переменная.

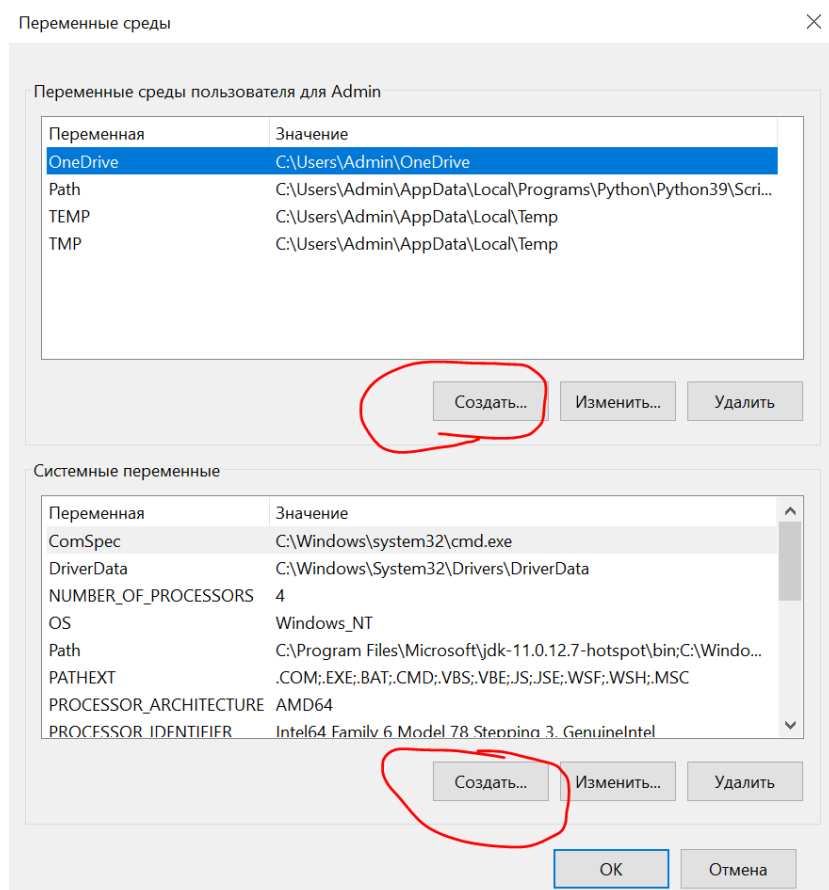


Рисунок 4 – Переменные среды

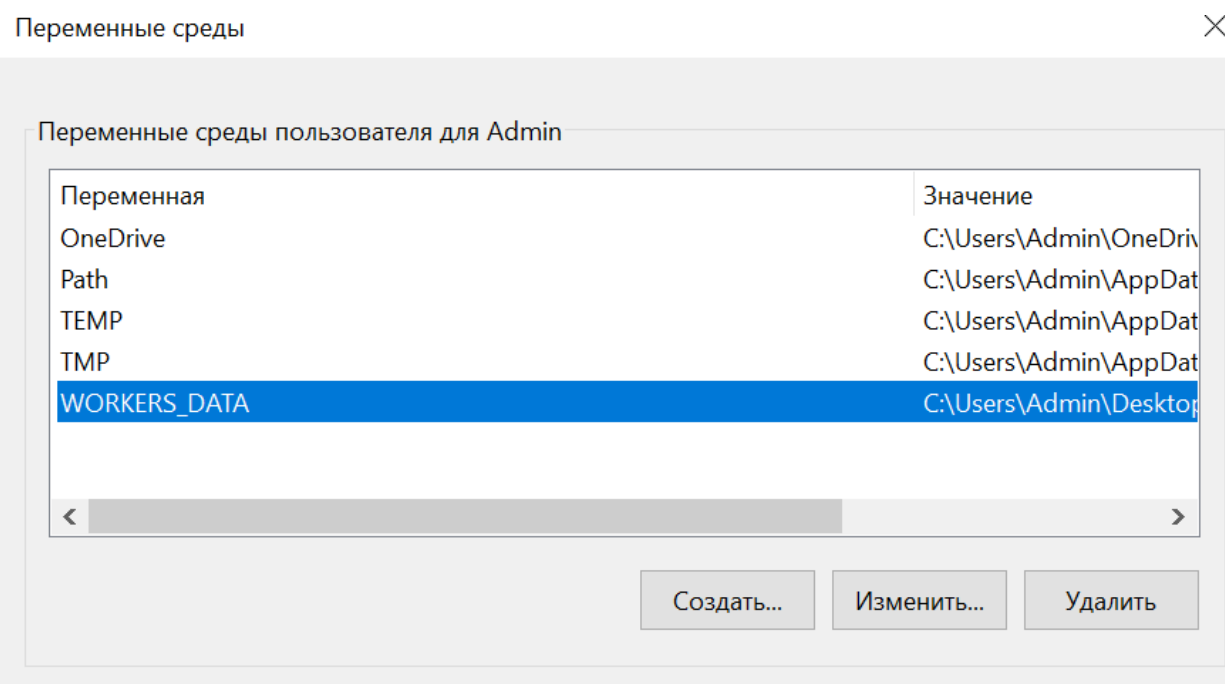


Рисунок 5 – Созданная переменная окружения

```
c:\Users\Admin\Desktop\git\Python13-2.18\Primers>python primer.py display
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Баранов ВИ | Сварщик | 2012 |
| 2 | Краманюк ДИ | Старший Сварщик | 2014 |
| 3 | Сидоров Сидор | Главный инженер | 2012 |
+-----+-----+-----+-----+

c:\Users\Admin\Desktop\git\Python13-2.18\Primers>python primer.py add --name="Меладзе Бармут" --post="Главный слесарь" --year=2011

c:\Users\Admin\Desktop\git\Python13-2.18\Primers>python primer.py display
+-----+-----+-----+-----+
| No | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Баранов ВИ | Сварщик | 2012 |
| 2 | Краманюк ДИ | Старший Сварщик | 2014 |
| 3 | Сидоров Сидор | Главный инженер | 2012 |
| 4 | Меладзе Бармут | Главный слесарь | 2011 |
+-----+-----+-----+-----+

c:\Users\Admin\Desktop\git\Python13-2.18\Primers>
```

Рисунок 6 – Результат выполнения примера лабораторной работы

Задание 1. Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

Для начала создадим переменную окружения.

Переменная	Значение
OneDrive	C:\Users\Витали
Path	C:\Users\Витали
PT6HOME	D:\Cisco Packet
TEMP	C:\Users\Витали
TMP	C:\Users\Витали
TRAINS_DATA	C:\Users\Витали

Рисунок 7 – Созданная переменная окружения

```

180         required=True,
181         help="The required select"
182     )
183
184     # Выполнить разбор аргументов командной строки.
185     args = parser.parse_args(command_line)
186
187     data_file = args.data
188     if not data_file:
189         data_file = os.environ.get("STUDENTS_DATA")
190     if not data_file:
191         print("The data file name is absent", file=sys.stderr)
192         sys.exit(1)
193
194     # Загрузить всех студентов из файла, если файл существует.
195     is_dirty = False
196     if os.path.exists(data_file):
197         students = load_students(data_file)
198     else:
199         students = []
200
201     # Добавить студента.
202     if args.command == "add":
203         students = add_student(

```

Рисунок 8 – Дополнение кода программы из лабораторной работы 2.17

Задание 2. Самостоятельно изучите работу с пакетом `python-dotenv`. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла `.env`.


```

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

data_file = args.data
dotenv_path = os.path.join(os.path.dirname(__file__), ".env")
if os.path.exists(dotenv_path):
    load_dotenv(dotenv_path)
if not data_file:
    data_file = os.getenv("STUDENTS_DATA")
if not data_file:
    print("The data file name is absent", file=sys.stderr)
    sys.exit(1)

# Загрузить всех студентов из файла, если файл существует.
is_dirty = False
if os.path.exists(data_file):
    students = load_students(data_file)
else:
    students = []

# Добавить студента.
if args.command == "add":
    students = add_student(

```

Рисунок 10 – Дополнил код программы из индивидуального задания №1

Вывод: в результате выполнения лабораторной работы были получены теоретические сведения и практические навыки для работы с переменными окружениями с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы:

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 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 с помощью 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()`.