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 ЛАБОРАТОРНАЯ РАБОТЕ №4
дисциплины «Анализ данных»
Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических наук, доцент, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Работа с переменными окружения в Python3

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

Ход работы

Пример 1. *Для примера 1 лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответстсвующую переменную окружения.*

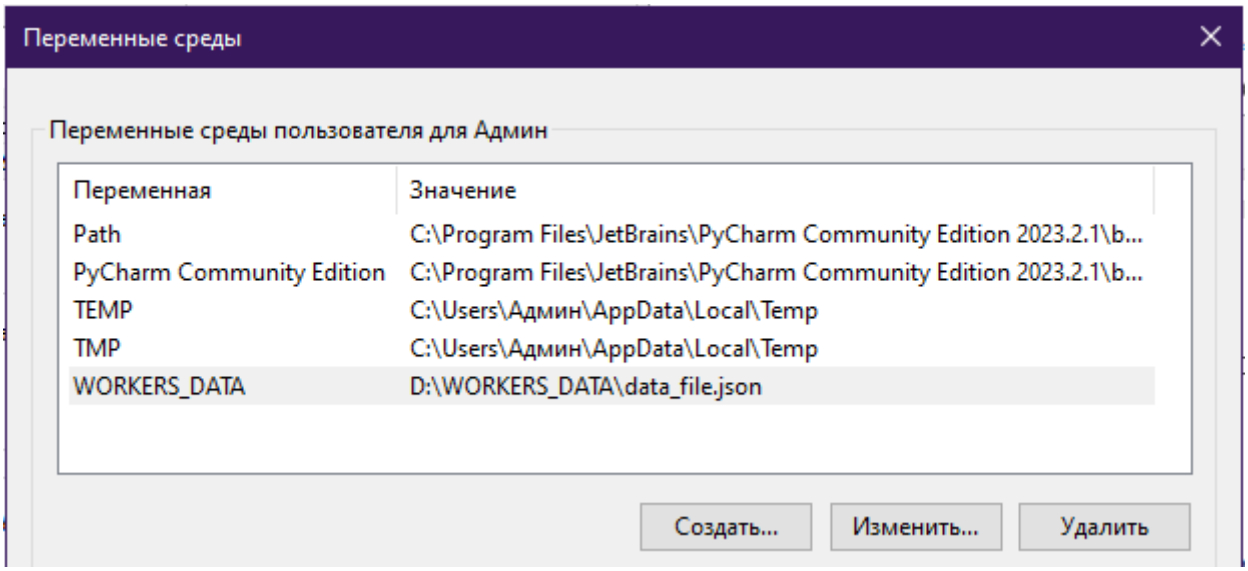

Рисунок 1. Переменная окружения WORKERS_DATA

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import os
import sys
from datetime import date


def add_worker(staff, name, post, year):
    """
    Добавить данные о работнике.
    """
    staff.append(
        {
        "name": name,
        "post": post,
        "year": year
        }
    )

    return staff


def display_workers(staff):
    """
    Отобразить список работников.
    """
```

```
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
            print(line)

    else:
        print("Список работников пуст.")


def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()

    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)

    # Возвратить список выбранных работников.
        return result


def save_workers(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)
```

```python
def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d",
        "--data",
        action="store",
        required=False,
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления работника.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new worker"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The worker's name"
    )
    add.add_argument(
        "-p",
        "--post",
        action="store",
        help="The worker's post"
    )
    add.add_argument(
        "-y",
        "--year",
        action="store",
        type=int,
        required=True,
        help="The year of hiring"
    )

    # Создать субпарсер для отображения всех работников.
    _ = subparsers.add_parser(
```

```
        "display",
        parents=[file_parser],
        help="Display all workers"
    )

    # Создать субпарсер для выбора работников.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the workers"
    )
    select.add_argument(
        "-P",
        "--period",
        action="store",
        type=int,
        required=True,
        help="The required period"
    )

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

    # Получить имя файла.
    data_file = args.data
    if not data_file:
        data_file = os.environ.get("WORKERS_DATA")
    if not data_file:
        print("The data file name is absent", file=sys.stderr)
        sys.exit(1)

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

    # Добавить работника.
    if args.command == "add":
        workers = add_worker(
            workers,
            args.name,
            args.post,
            args.year
        )
        is_dirty = True

    # Отобразить всех работников.
    elif args.command == "display":
        display_workers(workers)

    # Выбрать требуемых рааботников.
    elif args.command == "select":
        selected = select_workers(workers, args.period)
        display_workers(selected)

    # Сохранить данные в файл, если список работников был изменен.
    if is_dirty:
        save_workers(data_file, workers)
```

```python
if __name__ == "__main__":
    main()
```

```
D:\Rep\DA_4\Project>python example.py add --name="Сидоров Сидор" --post="Главный инженер" --year=2012

D:\Rep\DA_4\Project>python example.py display
+-----+--------------------------------+----------------------+----------+
|  №  |            Ф.И.О.              |      Должность       |   Год    |
+-----+--------------------------------+----------------------+----------+
|   1 | Журавлёв А.В                   | Менеджер             |     2005 |
+-----+--------------------------------+----------------------+----------+
|   2 | Михайлова С.А                  | Дизайнер             |     2014 |
+-----+--------------------------------+----------------------+----------+
|   3 | Третьяков К.Б                  | Директор             |     2002 |
+-----+--------------------------------+----------------------+----------+
|   4 | Сидоров Сидор                  | Главный инженер      |     2012 |
+-----+--------------------------------+----------------------+----------+
```

Рисунок 2. Результат работы программы

Индивидуальное задание:

Задание 1

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

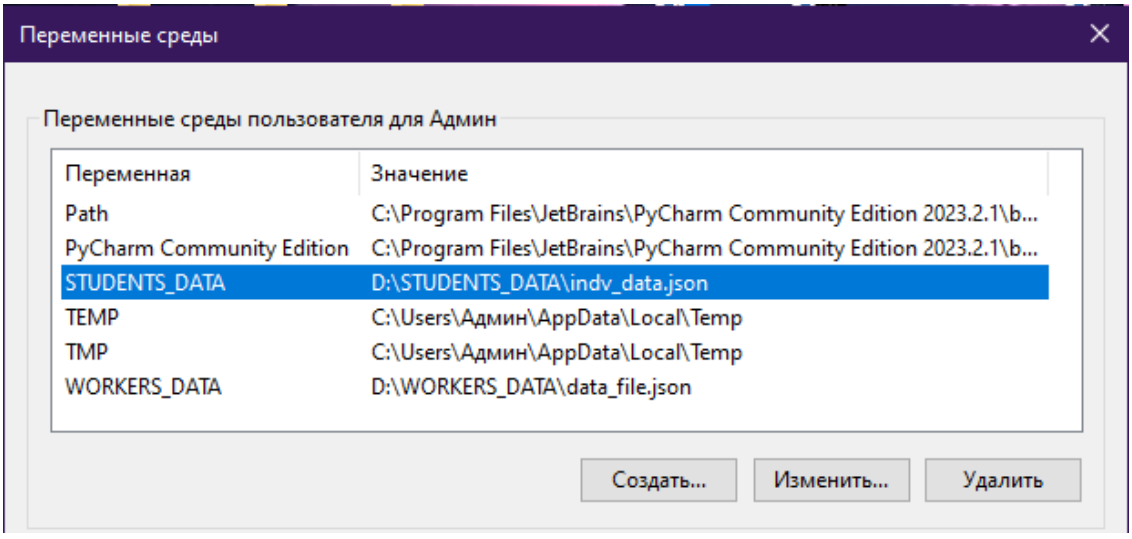

Рисунок 3. Переменная окружения STUDENTS_DATA

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import os.path
import sys


def add_student(staff, surname, group_number, grades):
    """
    Добавить данные о студенте.
    """
    staff.append(
```

```
        {
            "surname": surname,
            "group_number": group_number,
            "grades": grades
        }
    )

    return staff


def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('surname', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades')[0])
                )
            )
        print(line)

    else:
        print("Список студентов пуст.")


def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(int(grade) >= 4 for grade in student['grades'][0]):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result


def save_students(file_name, staff):
```

```
    """
    Сохранить всех учеников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_students(file_name):
    """
    Загрузить всех учеников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d",
        "--data",
        action="store",
        required=False,
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("students")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления студента.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new student"
    )
    add.add_argument(
        "-sn",
        "--surname",
        action="store",
        type=str,
        required=True,
        help="The student's surname"
    )
    add.add_argument(
        "-gn",
        "--group_number",
        action="store",
        type=int,
        help="The student's group"
    )
    add.add_argument(
        "-g",
```

```
        "--grades",
        action="store",
        nargs='+',
        type=list,
        required=True,
        help="grades"
    )

    # Создать субпарсер для отображения всех студентов.
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all students"
    )

    # Создать субпарсер для выбора студентов.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the students"
    )

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

    # Получить имя файла.
    data_file = args.data
    if not data_file:
        data_file = os.environ.get("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(
            students,
            args.surname,
            args.group_number,
            args.grades
        )
        is_dirty = True

    # Отобразить всех студентов.
    elif args.command == "display":
        display_students(students)

    # Выбрать требуемых студентов.
    elif args.command == "select":
        selected = select_students(students)
        display_students(selected)

    # Сохранить данные в файл, если список студентов был изменен.
    if is_dirty:
        save_students(data_file, students)
```

```
if __name__ == "__main__":
    main()
```

```
(myenv) D:\Rep\DA_4\Project>python indv1.py add --surname="Михайлова С.А" -gn=2 --grades="33434"

(myenv) D:\Rep\DA_4\Project>python indv1.py add --surname="Третьякова К.Б" -gn=3 --grades="44454"

(myenv) D:\Rep\DA_4\Project>python indv1.py add --surname="Журавлёв А.В" -gn=1 --grades="55545"
```

Рисунок 4. Добавление новых студентов

```
(myenv) D:\Rep\DA_4\Project>python indv1.py display
+------+--------------------------------+----------------------+----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | Сидоров С.А                    | 2                    |  3, 4, 3, 3, 3 |
|    2 | Михайлова С.А                  | 2                    |  3, 3, 4, 3, 4 |
|    3 | Третьякова К.Б                 | 3                    |  4, 4, 4, 5, 4 |
|    4 | Журавлёв А.В                   | 1                    |  5, 5, 5, 4, 5 |
+------+--------------------------------+----------------------+----------------+
```

Рисунок 5. Отображение информации обо всех студентов

```
(myenv) D:\Rep\DA_4\Project>python indv1.py select
+------+--------------------------------+----------------------+----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | Третьякова К.Б                 | 3                    |  4, 4, 4, 5, 4 |
|    2 | Журавлёв А.В                   | 1                    |  5, 5, 5, 4, 5 |
+------+--------------------------------+----------------------+----------------+
```

Рисунок 6. Выбор студентов, имеющих оценки 4 и 5

Задание 2

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

```
.env
1   STUDENTS_DATA = D:\STUDENTS_DATA\indv_data.json
```

Рисунок 7. Файл .env

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from dotenv import load_dotenv
import argparse
import json
import os.path
import sys


def add_student(staff, surname, group_number, grades):
    """
```

```
    Добавить данные о студенте.
    """
    staff.append(
        {
            "surname": surname,
            "group_number": group_number,
            "grades": grades
        }
    )

    return staff


def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('surname', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades')[0])
                )
            )
        print(line)

    else:
        print("Список студентов пуст.")


def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(int(grade) >= 4 for grade in student['grades'][0]):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result
```

```python
def save_students(file_name, staff):
    """
    Сохранить всех учеников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_students(file_name):
    """
    Загрузить всех учеников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d",
        "--data",
        action="store",
        required=False,
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("students")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления студента.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new student"
    )
    add.add_argument(
        "-sn",
        "--surname",
        action="store",
        type=str,
        required=True,
        help="The student's surname"
    )
    add.add_argument(
        "-gn",
        "--group_number",
        action="store",
        type=int,
        help="The student's group"
```

```
    )
    add.add_argument(
        "-g",
        "--grades",
        action="store",
        nargs='+',
        type=list,
        required=True,
        help="grades"
    )

    # Создать субпарсер для отображения всех студентов.
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all students"
    )

    # Создать субпарсер для выбора студентов.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the students"
    )

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

    # Загрузка переменных окружения из файла .env
    load_dotenv()

    # Получить имя файла.
    data_file = args.data
    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(
            students,
            args.surname,
            args.group_number,
            args.grades
        )
        is_dirty = True

    # Отобразить всех студентов.
    elif args.command == "display":
        display_students(students)

    # Выбрать требуемых студентов.
    elif args.command == "select":
```

```
        selected = select_students(students)
        display_students(selected)

    # Сохранить данные в файл, если список студентов был изменен.
    if is_dirty:
        save_students(data_file, students)


if __name__ == "__main__":
    main()
```

```
(base) D:\Rep\DA_4\Project>python indv2.py display
+------+--------------------------------+----------------------+----------------+
|  №   |            Ф.И.О.              |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | Сидоров С.А                    | 2                    |  3, 4, 3, 3, 3 |
|    2 | Михайлова С.А                  | 2                    |  3, 3, 4, 3, 4 |
|    3 | Третьякова К.Б                 | 3                    |  4, 4, 4, 5, 4 |
|    4 | Журавлёв А.В                   | 1                    |  5, 5, 5, 4, 5 |
+------+--------------------------------+----------------------+----------------+
```

Рисунок 8. Результат работы программы

Контрольные вопросы:

1. Переменные окружения используются для хранения информации, которая может быть использована программами или операционной системой

2. В переменных окружения может храниться различная информация, такая как пути к исполняемым файлам, настройки программ, локализация и другие параметры

3. Для доступа к переменным окружения в ОС Windows можно воспользоваться командой "set" в командной строке

4. Переменная PATH хранит список путей к исполняемым файлам, а переменная PATHEXT определяет расширения файлов, которые могут быть выполнены как исполняемые

5. Для создания или изменения переменной окружения в Windows можно воспользоваться панелью управления или командной строкой

6. В Linux переменные окружения используются для хранения информации о системных параметрах и настройках среды

7. Переменные окружения содержат информацию о среде выполнения программ, в то время как переменные оболочки относятся к параметрам самой оболочки командной строки

8. В Linux значение переменной окружения можно вывести с помощью команды "echo $ИМЯ_ПЕРЕМЕННОЙ"

9. Некоторые из известных переменных окружения в Linux: PATH, HOME, USER, SHELL и другие

10. Некоторые известные переменные оболочки Linux: PS1, PS2, PS3, PS4, IFS и другие

11. Переменные оболочки в Linux можно установить в файле настроек оболочки (например, /.bashrc)

12. Переменные окружения в Linux можно установить в файле настроек среды (например, /.profile)

13. Делать переменные окружения Linux постоянными нужно для того, чтобы они были доступны при каждом запуске сессии пользователя

14. Переменная окружения PYTHONHOME используется для указания корневой директории установленной версии Python

15. Переменная окружения PYTHONPATH используется для указания директорий, в которых Python будет искать модули

16. Для управления работой интерпретатора Python также могут использоваться переменные окружения PYTHONSTARTUP, PYTHONIOENCODING и другие

17. В Python переменные окружения можно прочитать с помощью модуля os: например, os.getenv("ИМЯ_ПЕРЕМЕННОЙ")

18. Для проверки установленного значения переменной окружения в Python можно использовать функцию os.getenv("ИМЯ_ПЕРЕМЕННОЙ") и проверить результат на None

19. Для присвоения значения переменной окружения в Python можно использовать функцию os.putenv("ИМЯ_ПЕРЕМЕННОЙ", "ЗНАЧЕНИЕ")

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с переменными окружения с помощью языка программирования Python версии 3.x