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 №2.18
дисциплины «Анализ данных»

Выполнил:
Магдаев Даламбек Магомедович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., 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.

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Проработал пример лабораторной работы:

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py -h
positional arguments:
  {add,display,select}
    add                 Add a new worker
    display             Display all workers
    select              Select the workers

options:
  --version             show program's version number and exit
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py display -h
usage: workers display [-h] [-d DATA]

options:
  -h, --help            show this help message and exit
  -d DATA, --data DATA  The data file name
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py display
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Магдаев                        | Студент              |     2022 |
+------+--------------------------------+----------------------+----------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py add -n Иванов -p Директор -y 2000
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py display -d data.json
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Магдаев                        | Студент              |     2022 |
+------+--------------------------------+----------------------+----------+
|    2 | Иванов                         | Директор             |     2000 |
+------+--------------------------------+----------------------+----------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python prim.py select -P 5
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Иванов                         | Директор             |     2000 |
+------+--------------------------------+----------------------+----------+
```

Рисунок 1. Ввод, вывод и выбор работников в консоли

3. Выполнил индивидуальное задание №1: Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

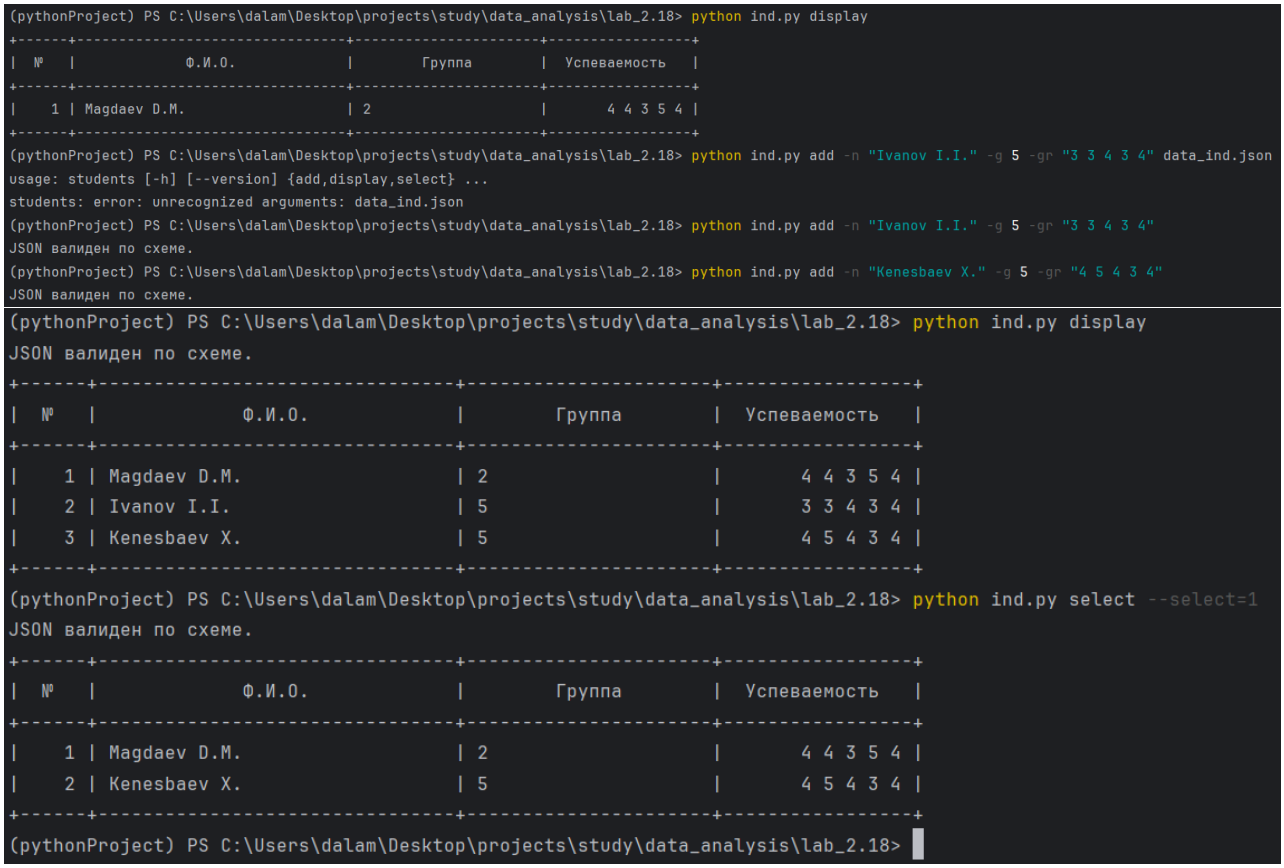

Рисунок 2. Страницы руководства и результат работы программы

Код индивидуального задания №1:

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

import json
import argparse
import os.path
import sys
from jsonschema import validate, ValidationError


def add_student(students, name, group, grade):
    students.append(
        {
            'name': name,
            'group': group,
            'grade': grade,
        }
    )
    return students


def show_list(students):
    # Заголовок таблицы.
    if students:

        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
```

```
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', ''),
                    student.get('grade', 0)
                )
            )
        print(line)
    else:
        print("Список студентов пуст.")


def show_selected(students):
    # Проверить сведения студентов из списка.
    result = []
    for student in students:
        grade = [int(x) for x in (student.get('grade', '').split())]
        if sum(grade) / max(len(grade), 1) >= 4.0:
            result.append(student)
    return result


def help():
    print("Список команд:\n")
    print("add - добавить студента;")
    print("display - вывести список студентов;")
    print("select - запросить студентов с баллом выше 4.0;")
    print("save - сохранить список студентов;")
    print("load - загрузить список студентов;")
    print("exit - завершить работу с программой.")


def save_students(file_name, students):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(students, fout, ensure_ascii=False, indent=4)


def load_students(file_name):
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
```

```
                "name": {"type": "string"},
                "group": {"type": "integer"},
                "grade": {"type": "string"},
            },
            "required": [
                "name",
                "group",
                "grade",
            ],
        },
    }
    with open(file_name, "r") as file_in:
        data = json.load(file_in)  # Прочитать данные из файла

    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")
    return data


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",
        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(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The student's name"
    )

    add.add_argument(
```

```
    "-g",
    "--group",
    type=int,
    action="store",
    help="The student's group"
)

add.add_argument(
    "-gr",
    "--grade",
    action="store",
    required=True,
    help="The student's grade"
)

# 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"
)

select.add_argument(
    "-s",
    "--select",
    action="store",
    required=True,
    help="The required select"
)

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

data_file = args.data
if not data_file:
    data_file = os.environ.get("INDIVIDUAL")
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.name,
```

```
            args.group,
            args.grade
        )
        is_dirty = True

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

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

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


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

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

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python ind_hard.py display
JSON валиден по схеме.
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Magdaev D.M.                   | 1                    |       4 4 3 5 4 |
|    2 | Ivanov I.I.                    | 4                    |       4 5 4 3 4 |
|    3 | Kenesbaev X.                   | 5                    |       3 3 4 3 4 |
+------+--------------------------------+----------------------+-----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python ind_hard.py add -n "Shams V." -g 8 -gr "5 5 5 5 5"
JSON валиден по схеме.
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python ind_hard.py display
JSON валиден по схеме.
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Magdaev D.M.                   | 1                    |       4 4 3 5 4 |
|    2 | Ivanov I.I.                    | 4                    |       4 5 4 3 4 |
|    3 | Kenesbaev X.                   | 5                    |       3 3 4 3 4 |
|    4 | Shams V.                       | 8                    |       5 5 5 5 5 |
+------+--------------------------------+----------------------+-----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.18> python ind_hard.py select --select=1
JSON валиден по схеме.
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Magdaev D.M.                   | 1                    |       4 4 3 5 4 |
|    2 | Ivanov I.I.                    | 4                    |       4 5 4 3 4 |
|    3 | Shams V.                       | 8                    |       5 5 5 5 5 |
+------+--------------------------------+----------------------+-----------------+
```

Рисунок 3. Результат работы программы

Код индивидуального задания №2:

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

import json
import argparse
from dotenv import load_dotenv
import os.path
import sys
from jsonschema import validate, ValidationError


def add_student(students, name, group, grade):
    students.append(
        {
            'name': name,
            'group': group,
            'grade': grade,
        }
    )
    return students


def show_list(students):
    # Заголовок таблицы.
    if students:

        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', ''),
                    student.get('grade', 0)
                )
            )
        print(line)
    else:
        print("Список студентов пуст.")
```

```
def show_selected(students):
    # Проверить сведения студентов из списка.
    result = []
    for student in students:
        grade = [int(x) for x in (student.get('grade', '').split())]
        if sum(grade) / max(len(grade), 1) >= 4.0:
            result.append(student)
    return result


def help():
    print("Список команд:\n")
    print("add - добавить студента;")
    print("display - вывести список студентов;")
    print("select - запросить студентов с баллом выше 4.0;")
    print("save - сохранить список студентов;")
    print("load - загрузить список студентов;")
    print("exit - завершить работу с программой.")


def save_students(file_name, students):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(students, fout, ensure_ascii=False, indent=4)


def load_students(file_name):
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name": {"type": "string"},
                "group": {"type": "integer"},
                "grade": {"type": "string"},
            },
            "required": [
                "name",
                "group",
                "grade",
            ],
        },
    }
    with open(file_name, "r") as file_in:
        data = json.load(file_in)  # Прочитать данные из файла

    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")
    return data


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",
    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(
    "-n",
    "--name",
    action="store",
    required=True,
    help="The student's name"
)

add.add_argument(
    "-g",
    "--group",
    type=int,
    action="store",
    help="The student's group"
)

add.add_argument(
    "-gr",
    "--grade",
    action="store",
    required=True,
    help="The student's grade"
)

# 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"
    )

    select.add_argument(
        "-s",
        "--select",
        action="store",
        required=True,
        help="The required select"
    )

    # 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("INDIVIDUAL_HARD")
    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.name,
            args.group,
            args.grade
        )
        is_dirty = True

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

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

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


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

Ответы на контрольные вопросы:

1) Каково назначение переменных окружения?

Ответ: 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.

Ответ: 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?

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

4) Каково назначение переменных PATH и PATHEXT?

Ответ: 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?

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

6) Что представляют собой переменные окружения в ОС Linux?

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

7) В чем отличие переменных окружения от переменных оболочки?

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

Ответ: 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?

Ответ: Наиболее часто используемая команда для вывода переменных окружения – printenv.

9) Какие переменные окружения Linux Вам известны? USER − текущий пользователь.

Ответ: 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?

Ответ: BASHOPTS – список задействованных параметров оболочки, разделенных двоеточием;

BASH_VERSION – версия запущенной оболочки bash;

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

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.

UID – идентификатор текущего пользователя.

11) Как установить переменные оболочки в Linux?

Ответ: $ NEW_VAR='значение'

12) Как установить переменные окружения в Linux?

Ответ: 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?

Ответ: Чтобы переменная сохранялась после закрытия сеанса оболочки.

14) Для чего используется переменная окружения PYTHONHOME?

Ответ: Переменная среды PYTHONHOME изменяет расположение стандартных библиотек Python.

15) Для чего используется переменная окружения PYTHONPATH?

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

16) Как осуществляется чтение переменных окружения в программах на языке программирования Python?

Ответ: value = os.environ.get('MY_ENV_VARIABLE')

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

Ответ: if os.environ[key_value]:

18) Как присвоить значение переменной окружения в программах на языке программирования Python?

Ответ: Для присвоения значения любой переменной среды используется функция os.environ.setdefault(«Переменная», «Значение»).

Вывод: в ходе выполнения лабораторной работы, приобретены навыки

построения приложений с переменными окружения с помощью языка программирования Python версии 3.x.