

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 №4
дисциплины «Анализ данных»
Вариант 29

Выполнил:
Саенко Андрей Максимович
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., канд. технических
наук, 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:

Проработаны примеры лабораторной работы:

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

Создана переменная окружения:

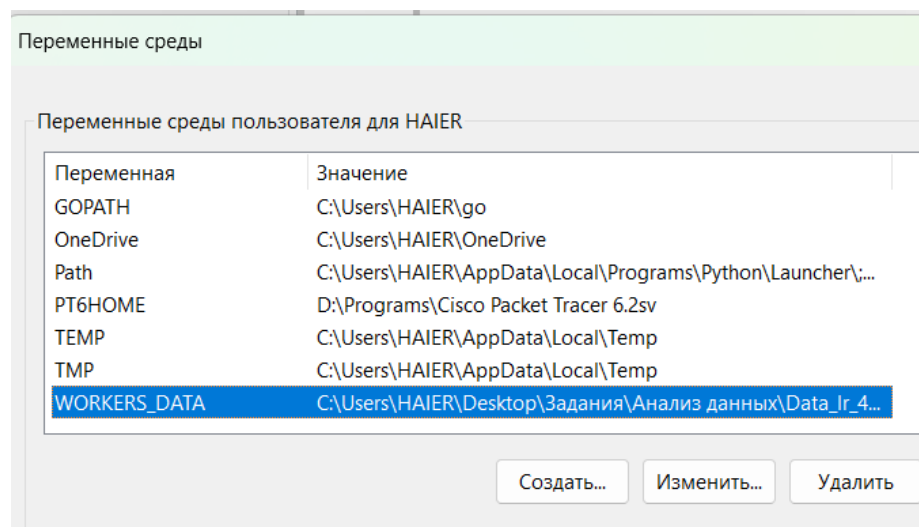


Рисунок 1 – Создана переменная среды

Код программы:

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

# Для примера 1 лабораторной работы 2.17 добавьте возможность получения имени
# файла данных, используя соответствующую переменную окружения.

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.
    """
    if staff:
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-" * 8)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
                "№", "Ф.И.О.", "Должность", "Год"
            )
        )
        print(line)
        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):
    today = date.today()

    result = []
    for employee in staff:
        if today.year - employee.get("year", today.year) >= period:
            result.append(employee)

    return result

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

def load_workers(file_name):
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d", "--data", action="store", required=False, help="The data file name"
    )

    parser = argparse.ArgumentParser("workers")

```

```

parser.add_argument("--version", action="version", version="% (prog)s 0.1.0")

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

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",
)

_ = subparsers.add_parser(
    "display", parents=[file_parser], help="Display all workers"
)

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",
)

args = parser.parse_args()

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)

is_dirty = False
if os.path.exists(data_file):
    workers = load_workers(data_file)
else:
    workers = []

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

```

```

elif args.command == "display":
    display_workers(workers)

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

if is_dirty:
    save_workers(data_file, workers)

if __name__ == "__main__":
    os.environ.setdefault("WORKERS_DATA", "C:/Users/HAIER/Desktop/Задания/"+
        "Анализ данных/Data_lr_4/data/workers.json")
    main()

```

Результат работы программы:

```

PS C:\Users\HAIER> python example_1.py display
Список работников пуст.
PS C:\Users\HAIER> python example_1.py add --name="Сидоров Сидор" --post="Главный инженер" --year=2012
PS C:\Users\HAIER> python example_1.py display

```

№	Ф.И.О.	Должность	Год
1	Сидоров Сидор	Главный инженер	2012

Рисунок 2 – Результат работы программы. Указывать место хранения файла json не требуется

Выполнены задания:

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

Вариант 29

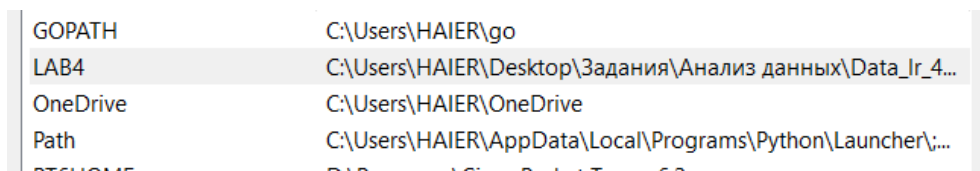


Рисунок 3 – Созданная переменная окружения

Код программы:

```

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

# Для своего варианта лабораторной работы 2.17 добавьте

```

```
# возможность получения имени файла данных, используя
# соответствующую переменную окружения.
```

```
import argparse
import json
import sys
import os.path
```

```
# Вариант 29
```

```
def add_route(staff, start, end, number):
```

```
    """
    Добавить маршрут
    """
    staff.append({
        'name_start': start,
        'name_end': end,
        'number': number
    })
    return staff
```

```
def list(routes):
```

```
    """
    Вывести список маршрутов
    """
    if routes:
        line = '+-{}--{}--{}--{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 8
        )
        print(line)

        print('| {:^4} | {:^30} | {:^30} | {:^8} |'.format(
            "№",
            "Начальный пункт",
            "Конечный пункт",
            "Номер"
        ))
    )
    print(line)

    for idx, route in enumerate(routes, 1):
        print('| {:>4} | {:<30} | {:<30} | {:>8} |'.format(
            idx,
            route.get('name_start', ""),
            route.get('name_end', ""),
            route.get('number', 0)
        ))
    )
    print(line)
else:
```

```

    print("Список маршрутов пуст.")

def save_routes(file_name, staff):
    """
    Сохранить все маршруты в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

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

def select_routes(routes, command):
    """
    Вывести выбранные маршруты
    """
    station = command
    count = 0

    for route in routes:
        if (station.lower() == route["name_start"].lower() or
            station == route["name_end"].lower()):

            count += 1
            print('{:>4}: {}-{}, номер маршрута: {}'.format(count,
                route["name_start"], route["name_end"], route["number"]))

    if count == 0:
        print("Маршрут не найден.")

def main(command_line=None):
    """
    Основная функция
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-f",
        "--filename",
        required=False,
        action="store",
        help="The data file name"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("routes")
    parser.add_argument(
        "--version",

```

```

        action="version",
        version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления маршрута.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new route"
    )
    add.add_argument(
        "-s",
        "--start",
        action="store",
        required=True,
        help="Start position on route"
    )
    add.add_argument(
        "-e",
        "--end",
        action="store",
        help="End position on route"
    )
    add.add_argument(
        "-n",
        "--number",
        action="store",
        type=int,
        required=True,
        help="Number of route"
    )
    # Создать субпарсер для отображения всех маршрутов.
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all routes"
    )
    # Создать субпарсер для выбора маршрутов.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the routes"
    )
    select.add_argument(
        "-t",
        "--station",
        action="store",
        type=str,
        required=True,
        help="Routes with this station"
    )
    # Выполнить разбор аргументов командной строки.

```



```

args = parser.parse_args(command_line)
os.environ.setdefault("LAB4", "C:/Users/HAIER/Desktop/Задания/" +
    "Анализ данных/Data_lr_4/data/routes.json")
file_name = args.filename
if not file_name:
    file_name = os.environ.get("LAB4")
if not file_name:
    print("Data file name not set in env variable", file=sys.stderr)
    sys.exit(1)

# Загрузить все маршруты из файла, если файл существует.
is_dirty = False
if os.path.exists(file_name):
    routes = load_routes(file_name)
else:
    routes = []
# Добавить маршрут.
if args.command == "add":
    if(routes is None):
        routes = []
    routes = add_route(
        routes,
        args.start,
        args.end,
        args.number
    )
    is_dirty = True
# Отобразить все маршруты.
elif args.command == "display":
    list(routes)

# Выбрать требуемые маршруты.
elif args.command == "select":
    select_routes(routes, args.station)
    print(args.station)
# Сохранить данные в файл, если список маршрутов был изменен.
if is_dirty:
    save_routes(file_name, routes)

if __name__ == '__main__':
    main()
    Результат работы программы:

```

```
PS C:\Users\HAIER> python ind_1.py add --start="Суздаль" --end="Воронеж" --number=1
PS C:\Users\HAIER> python ind_1.py display
```

№	Начальный пункт	Конечный пункт	Номер
1	Москва	Воронеж	1
2	Норильск	Воронеж	2
3	Москва	Ярославль	3
4	Нью-Йорк	Омск	5
5	Суздаль	Воронеж	1

Рисунок 4 – Вводить имя файла с данными больше не требуется

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

Код программы:

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

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

import argparse
import json
import sys
import os.path

from dotenv import dotenv_values

# Вариант 29
def add_route(staff, start, end, number):
    """
    Добавить маршрут
    """
    staff.append({
        'name_start': start,
        'name_end': end,
        'number': number
    })
    return staff

def list(routes):
    """
    Вывести список маршрутов
    """
    if routes:
        line = '+-{}+-{}+-{}+-{}-+'.format(
```

```

        '_' * 4,
        '_' * 30,
        '_' * 30,
        '_' * 8
    )
    print(line)

    print('| {:^4} | {:^30} | {:^30} | {:^8} |'.format(
        "№",
        "Начальный пункт",
        "Конечный пункт",
        "Номер"
    )
    )
    print(line)

    for idx, route in enumerate(routes, 1):
        print('| {:>4} | {:<30} | {:<30} | {:>8} |'.format(
            idx,
            route.get('name_start', ''),
            route.get('name_end', ''),
            route.get('number', 0)
        )
        )
        print(line)
    else:
        print("Список маршрутов пуст.")

def save_routes(file_name, staff):
    """
    Сохранить все маршруты в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

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

def select_routes(routes, command):
    """
    Вывести выбранные маршруты
    """
    station = command
    count = 0

    for route in routes:

```

```

        if (station.lower() == route["name_start"].lower() or
            station == route["name_end"].lower()):

            count += 1
            print('{:>4}: {}-{}, номер маршрута: {}'.format(count,
                route["name_start"], route["name_end"], route["number"]))

    if count == 0:
        print("Маршрут не найден.")

def main(command_line=None):
    """
    Основная функция
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-f",
        "--filename",
        required=False,
        action="store",
        help="The data file name"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("routes")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления маршрута.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new route"
    )
    add.add_argument(
        "-s",
        "--start",
        action="store",
        required=True,
        help="Start position on route"
    )
    add.add_argument(
        "-e",
        "--end",
        action="store",
        help="End position on route"
    )
    add.add_argument(
        "-n",
        "--number",

```

```

        action="store",
        type=int,
        required=True,
        help="Number of route"
    )
    # Создать субпарсер для отображения всех маршрутов.
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all routes"
    )
    # Создать субпарсер для выбора маршрутов.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the routes"
    )
    select.add_argument(
        "-t",
        "--station",
        action="store",
        type=str,
        required=True,
        help="Routes with this station"
    )
    # Выполнить разбор аргументов командной строки.
    args = parser.parse_args(command_line)
    file_name = args.filename
    if not file_name:
        file_name = dotenv_values(".env")["LAB4"]
    if not file_name:
        print("Data file name not set in env variable", file=sys.stderr)
        sys.exit(1)

    # Загрузить все маршруты из файла, если файл существует.
    is_dirty = False
    if os.path.exists(file_name):
        routes = load_routes(file_name)
    else:
        routes = []
    # Добавить маршрут.
    if args.command == "add":
        if(routes is None):
            routes = []
        routes = add_route(
            routes,
            args.start,
            args.end,
            args.number
        )
        is_dirty = True
    # Отобразить все маршруты.

```

```

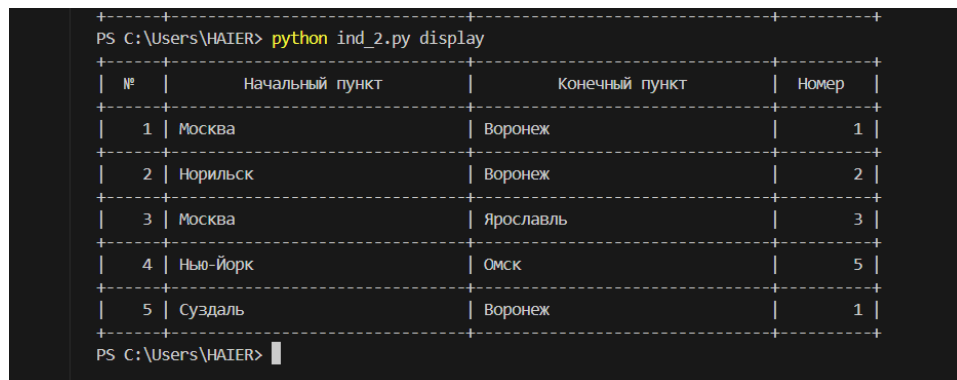
elif args.command == "display":
    list(routes)

# Выбрать требуемые маршруты.
elif args.command == "select":
    select_routes(routes, args.station)
    print(args.station)
# Сохранить данные в файл, если список маршрутов был изменен.
if is_dirty:
    save_routes(file_name, routes)

if __name__ == '__main__':
    main()

```

Результат работы программы:



```

PS C:\Users\HAIER> python ind_2.py display

```

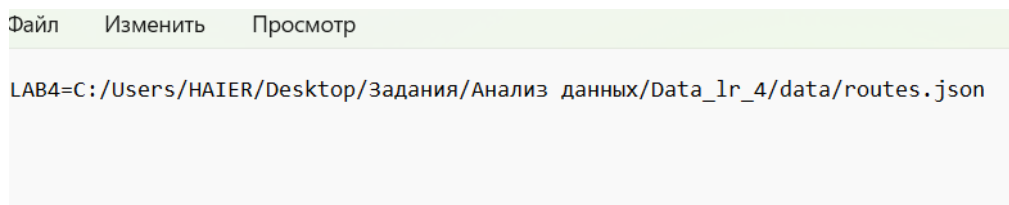
№	Начальный пункт	Конечный пункт	Номер
1	Москва	Воронеж	1
2	Норильск	Воронеж	2
3	Москва	Ярославль	3
4	Нью-Йорк	Омск	5
5	Суздаль	Воронеж	1

```

PS C:\Users\HAIER>

```

Рисунок 5 – Вводить имя файла с данными больше не требуется



Файл Изменить Просмотр

LAB4=C:/Users/HAIER/Desktop/Задания/Анализ данных/Data_lr_4/data/routes.json

Рисунок 6 – Содержимое файла .env

Ответы на 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?

Системные переменные окружения служат для хранения статичных, редко изменяющихся значений (таких, как пути к программам, системным каталогам, глобальные макроподстановки).

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

Доступ к переменным окружения в ОС Windows можно получить с помощью панели управления.

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 расширение файла.

5. Как создать или изменить переменную окружения в Windows?

Создать или изменить переменную окружения можно в окне «Переменные среды».

6. Что представляют собой переменные окружения в ОС Linux?

Набор именованных значений.

7. В чем отличие переменных окружения от переменных оболочки?

Переменные окружения доступны в масштабах всей системы.

8. Как вывести значение переменной окружения в Linux?

Можно использовать команду `printenv`.

9. Какие переменные окружения Linux Вам известны?

USER, PWD, HOME, SHELL, EDITOR, LOGNAME, PATH, LANG, MAIL, LS_COLORS

10. Какие переменные оболочки Linux Вам известны?

BASHOPTS, BASH_VERSION, COLUMNS, HISTSIZE, PS2, UID.

11. Как установить переменные оболочки в Linux?

`VAR_NAME='значение'`

12. Как установить переменные окружения в Linux?

Установить переменные окружения в Linux можно с помощью команды `export`.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Переменные окружения Linux необходимо делать постоянными, чтобы они сохранялись после завершения сеанса.

14. Для чего используется переменная окружения PYTHONHOME ?

Переменная окружения PYTHONHOME используется, чтобы переменная сохранялась после закрытия сеанса оболочки.

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?

Используются следующие переменные: PYTHONSTARTUP, PYTHONOPTIMIZE, PYTHONBREAKPOINT, PYTHONDEBUG, PYTHONINSPECT, PYTHONUNBUFFERED, PYTHONVERBOSE, PYTHONCASEOK, PYTHONDONTWRITEBYTECODE, PYTHONPYCACHEPREFIX, PYTHONHASHSEED.

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

Чтение переменных окружения осуществляется с помощью методов библиотеки os или dotenv.

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

Это можно сделать с помощью условного оператора

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

Для присвоения значения переменной среды используется функция os.environ.setdefault()

Выводы: в ходе выполнения работы были получены навыки работы с переменными среды.

