МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Дисциплина: «Языки программирования»

Отчет по лабораторной работе №18

Работа с переменными окружения в Python3

Выполнил студент группы ИТС-б-о-21-1

Усиков Данил

«__»_____20__г.

Подпись студента_____

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Роман Александрович

_____

(подпись)

Ставрополь, 2022

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

Ссылка на репозиторий https://github.com/danilusikov0913/YPlr8

Ход работы:

1. Пример

Для хранения имени файла данных будем использовать переменную окружения WORKERS_DATA .

При этом сохраним возможность передавать имя файла данных через именной параметр --data .

Иными словами, если при запуске программы в командной строке не задан параметр --data , то

имя файла данных должно быть взято из переменной окружения WORKERS_DATA

Напишем программу для решения поставленной задачи.

```
#!/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(
                "No",
                "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)


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)


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

Рисунок 1 – Код примера

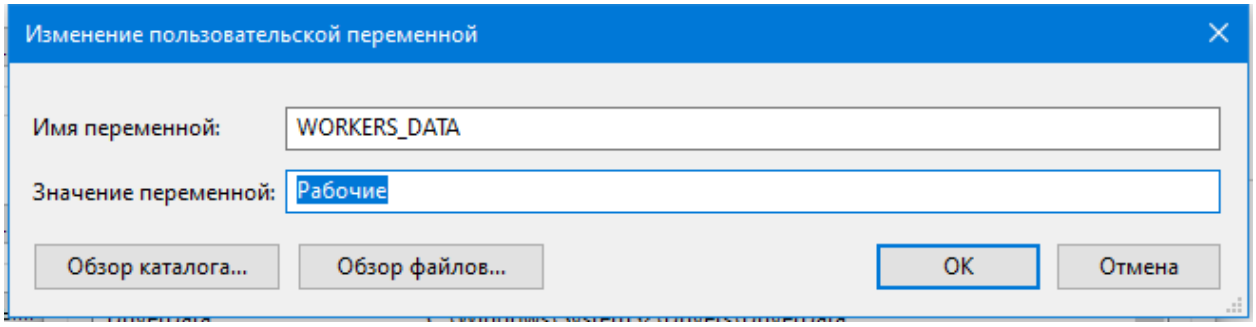

Рисунок 2 – Добавление переменной среды

```
C:\MonstR\8>python lr8prim.py add --name="Сидоров Сидор" --post="Главный инженер" --year=2012

C:\MonstR\8>python lr8prim.py display
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                  | Главный инженер      |     2012 |
+------+--------------------------------+----------------------+----------+

C:\MonstR\8>python lr8prim.py select --period=10
+------+--------------------------------+----------------------+----------+
|  No  |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                  | Главный инженер      |     2012 |
+------+--------------------------------+----------------------+----------+

C:\MonstR\8>
```

Рисунок 3 – Результат работы

2. Индивидуальное задание 1

Задание 1

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

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os
import sys


def add_shop(list_shop, name_shop, name_product, prise):
    """
    Добавить данные магазина.
    """
    list_shop.append(
        {
            "name_shop": name_shop,
            "name_product": name_product,
            "prise": prise
        }
    )
    return list_shop


def display_shop(list_shop):
    """
    Отобразить список магазинов.
    """
    if list_shop:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 6,
            '-' * 20,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^6} | {:^20} | {:^30} | {:^20} |'.format(
                "No",
```

```
                "Название магазина",
                "Название продукта",
                "Стоимость товара"
            )
        )
        print(line)
        for idx, listshop in enumerate(list_shop, 1):
            print(
                '| {:>6} | {:<20} | {:<30} | {:>20} |'.format(
                    idx,
                    listshop.get('name_shop', ''),
                    listshop.get('name_product', ''),
                    listshop.get('prise', 0)
                )
            )
        print(line)
    else:
        print("Список магазинов пуст.")


def select_product(list_shop, shop_sear):
    """
    Выбрать продукт.
    """
    search_shop = []
    for shop_sear_itme in list_shop:
        if shop_sear == shop_sear_itme['name_product']:
            search_shop.append(shop_sear_itme)
    return search_shop


def save_shop(file_name, list_shop):
    """
    Сохранить всех работников в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(list_shop, fout, ensure_ascii=False, indent=4)


def load_list_shop(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    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 shop"
    )
    add.add_argument(
        "-ns",
        "--name_shop",
        action="store",
        required=True,
        help="The shop's name"
    )
    add.add_argument(
        "-np",
        "--name_product",
        action="store",
        help="The product's name"
    )
    add.add_argument(
        "-ps",
        "--prise",
        action="store",
        type=int,
        required=True,
        help="Cost of goods"
    )
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all shops"
    )
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the product"
    )
    select.add_argument(
        "-SS",
        "--shop_sear",
        action="store",
        type=str,
        required=True,
        help="The name product"
    )
    args = parser.parse_args(command_line)

    # Получить имя файла.
    data_file = args.data
    if not data_file:
        data_file = os.environ.get("SHOPS_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):
        shop = load_list_shop(data_file)
    else:
        shop = []
    if args.command == "add":
        shop = add_shop(
            shop,
            args.name_shop,
            args.name_product,
```

```
            args.prise
        )
        is_dirty = True
    elif args.command == "display":
        display_shop(shop)

    elif args.command == "select":
        selected = select_product(shop, args.shop_sear)
        display_shop(selected)
    if is_dirty:
        save_shop(data_file, shop)


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

Рисунок 4 – Код индивидуального задания

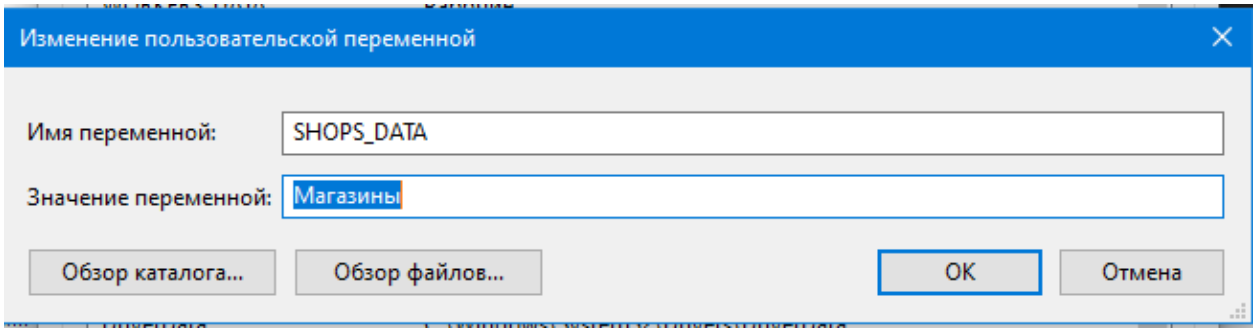

Рисунок 5 – Код индивидуального задания

```
C:\MonstR\8>python lr8iz.py add -ns="Магнит" -np="Молоко" -ps=72

C:\MonstR\8>python lr8iz.py display
+--------+----------------------+----------------------------------+----------------------+
|   No   |  Название магазина   |       Название продукта          |   Стоимость товара   |
+--------+----------------------+----------------------------------+----------------------+
|      1 | Магнит               | Молоко                           |                   72 |
+--------+----------------------+----------------------------------+----------------------+

C:\MonstR\8>python lr8iz.py select -SS="Молоко"
+--------+----------------------+----------------------------------+----------------------+
|   No   |  Название магазина   |       Название продукта          |   Стоимость товара   |
+--------+----------------------+----------------------------------+----------------------+
|      1 | Магнит               | Молоко                           |                   72 |
+--------+----------------------+----------------------------------+----------------------+

C:\MonstR\8>python lr8iz.py select -SS="Сыр"
Список магазинов пуст.

C:\MonstR\8>
```

Рисунок 6 – Результат работы

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?

Компьютер, свойства, дополнительные параметры и среды, дополнительно, переменные среды

4. Каково назначение переменных PATH и PATHEXT?

«PATH» позволяет запускать исполняемые файлы и скрипты, «лежащие» в определенных

PATHEXT, в свою очередь, дает возможность не указывать даже расширение файла, если оно прописано в ее значениях каталогах, без указания их точного местоположения.

5. Как создать или изменить переменную окружения в Windows?

Компьютер, свойства, дополнительные параметры и среды, дополнительно, переменные среды, создать или изменить

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 и оболочки всегда присутствуют в сеансах оболочки и могут быть очень полезны. Они позволяют родительским процессам устанавливать детали конфигурации для своих дочерних процессов и являются способом установки определенных параметров без использования отдельных файлов.

8. Как вывести значение переменной окружения в Linux?

9. Какие переменные окружения Linux Вам известны?

10. Какие переменные оболочки Linux Вам известны?

11. Как установить переменные оболочки в Linux?

12. Как установить переменные окружения в Linux?

13. Для чего необходимо делать переменные окружения Linux постоянными?

14. Для чего используется переменная окружения PYTHONHOME ?

Переменная среды PYTHONHOME изменяет расположение стандартных библиотек Python. По умолчанию библиотеки ищутся в prefix/lib/pythonversion и exec_prefix/lib/pythonversion , где prefix и exec_prefix - это каталоги, зависящие от установки, оба каталога по умолчанию - /usr/local.

Когда для PYTHONHOME задан один каталог, его значение заменяет prefix и exec_prefix. Чтобы указать для них разные значения, установите для PYTHONHOME значение prefix:exec_prefix

15. Для чего используется переменная окружения PYTHONPATH ?

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

Формат такой же, как для оболочки PATH : один или несколько путей к каталогам, разделенных os.pathsep (например, двоеточие в Unix или точка с запятой в Windows). Несуществующие каталоги игнорируются. $ unset NEW_VAR

Помимо обычных каталогов, отдельные записи PYTHONPATH могут относиться к zip-файлам, содержащим чистые модули Python в исходной или скомпилированной форме. Модули расширения нельзя импортировать из zip-файлов.

Путь поиска по умолчанию зависит от установки Python, но обычно начинается с префикса /lib/pythonversion . Он всегда добавляется к PYTHONPATH .

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

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

Для начала потребуется импортировать модуль os, чтобы считывать переменные. Для доступа к переменным среды в Python используется объект os.environ . С его помощью программист может получить и изменить значения всех переменных среды. Далее мы рассмотрим различные способы чтения, проверки и присвоения значения переменной среды.

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

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

Для начала потребуется импортировать модуль os, чтобы считывать переменные. Для доступа к переменным среды в Python используется объект os.environ . С его помощью программист может получить и изменить значения всех переменных среды. Далее мы рассмотрим различные способы чтения, проверки и присвоения значения переменной среды.

Вывод: в ходе лабораторной работы приобретены навыки построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.