

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

Кафедра инфокоммуникаций

дисциплина Языки программирования

Отчет по лабораторной работе №2.18

Работа с переменными окружения в Python3

Выполнил: студент группы ИТС-б-о-21-1
Пушкин Максим Алексеевич

(подпись)

Проверил: кандидат технических наук, доцент кафедры
инфокоммуникаций,
Роман Александрович Воронкин

(подпись)

Ставрополь, 2022

1. **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>

Метод и порядок выполнения

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`

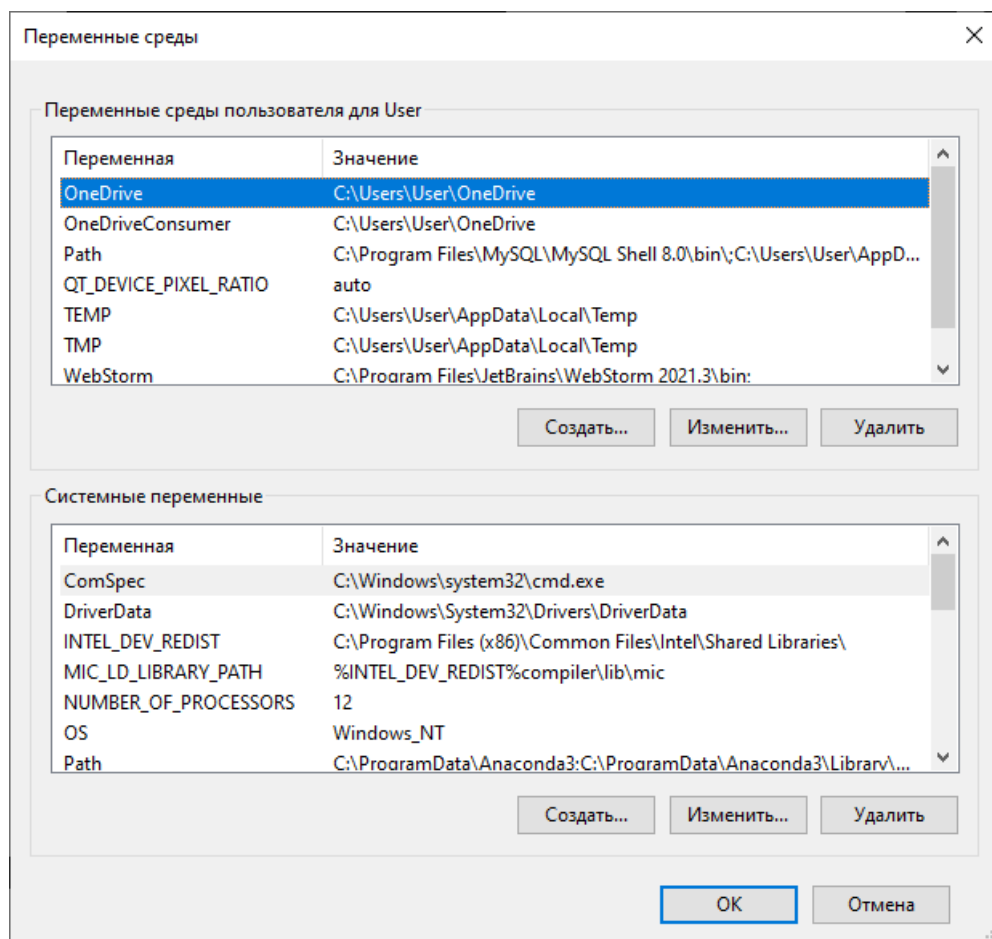
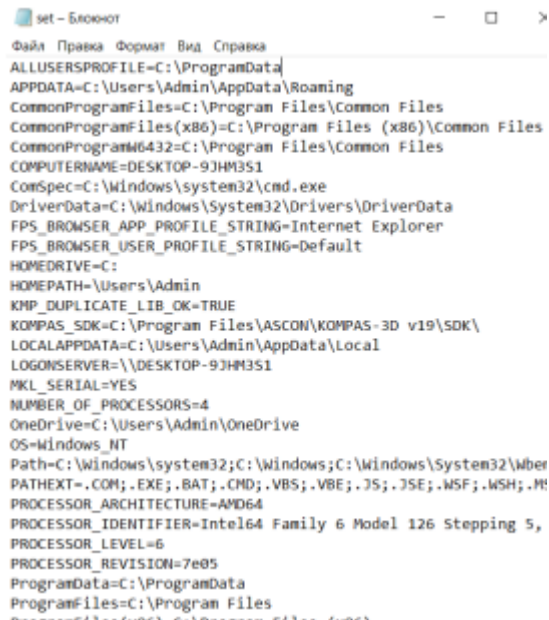


Рис 1.



```
set - Блокнот
Файл Правка Формат Вид Справка
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\Admin\AppData\Roaming
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=DESKTOP-9JHM351
ComSpec=C:\Windows\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\DriverData
FPS_BROWSER_APP_PROFILE_STRING=Internet Explorer
FPS_BROWSER_USER_PROFILE_STRING=Default
HOMEDRIVE=C:
HOMEPATH=\Users\Admin
KMP_DUPLICATE_LIB_OK=TRUE
KOMPAS_SDK=C:\Program Files\ASCON\KOMPAS-3D v19\SDK\
LOCALAPPDATA=C:\Users\Admin\AppData\Local
LOGONSERVER=\\DESKTOP-9JHM351
MKL_SERIAL=YES
NUMBER_OF_PROCESSORS=4
OneDrive=C:\Users\Admin\OneDrive
OS=Windows_NT
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbin
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.M
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 126 Stepping 5,
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=7e05
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
```

Рис 2.

Код задания 1

```
#!/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:
        # Заголовки 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 кириллицы 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 работников.
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()

```

Результат

```
PS E:\ЯП 2 курс\8\project> python prim.py add --name="Sidorov" --post="инженер" --year=2012
The data file name is absent
```

Рис.3

Пример 2

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

def add_shop(list_race, name, number, time):
    """
    Добавить данные магазина.
    """
    list_race.append(
        {
            "name": name,
            "number": number,
            "time": time
        }
    )
    return list_race

def display_shop(list_race):
    """
    Отобразить список.
    """
    if list_race:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 6,
            '-' * 20,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^6} | {:^20} | {:^30} | {:^20} |'.format(
                "No",
                "пункт назначения",
                "номер",
                "время"
            )
        )
        print(line)
        for idx, listrace in enumerate(list_race, 1):
            print(
                '| {:>6} | {:<20} | {:<30} | {:>20} |'.format(
                    idx,
                    listrace.get('name', ''),
                    listrace.get('number', ''),
                    listrace.get('time', '')
                )
            )
```

```

    )
    print(line)
else:
    print("Список рейсов пуст.")

def select_product(list_race, race_sear):
    """
    Выбрать.
    """
    search_race = []
    for race_sear_itme in list_race:
        if race_sear == race_sear_itme['name']:
            search_race.append(race_sear_itme)
    return search_race

def save_race(file_name, list_race):
    """
    Сохранить все в JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(list_race, fout, ensure_ascii=False, indent=4)

def load_list_race(file_name):
    """
    Загрузить все 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("races")
    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 race"
    )
    add.add_argument(
        "-nm",
        "--name",
        action="store",
        required=True,
        help="The race's name"
    )

```



```

)
add.add_argument(
    "-nb",
    "--number",
    action="store",
    help="The number"
)
add.add_argument(
    "-t",
    "--time",
    action="store",
    type=int,
    required=True,
    help="time"
)
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all races"
)
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the product"
)
select.add_argument(
    "-ss",
    "--name_sear",
    action="store",
    type=str,
    required=True,
    help="The name race"
)
args = parser.parse_args(command_line)

# Получить имя файла.
data_file = args.data
if not data_file:
    data_file = os.environ.get("RACES_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(args.filename):
    race = load_list_race(args.filename)
else:
    race = []
if args.command == "add":
    race = add_shop(
        race,
        args.name,
        args.number,
        args.time
    )
    is_dirty = True
elif args.command == "display":
    display_shop(race)

elif args.command == "select":
    selected = select_product(race, args.race_sear)

```

```

        display_shop(selected)
    if is_dirty:
        save_race(args.filename, race)

if __name__ == '__main__':
    main()

```

Результат

No	пункт назначения	номер	время
1	Ставрополь	15	13:25

Рис .4

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 PATHNEXT?

«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 каталогов, 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
PYTHONLEGACYWINDOWSSSTDIO 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.