

**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

Романов Платон Дмитриевич

«__» _____ 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 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()

```

Рисунок 1 – Код примера

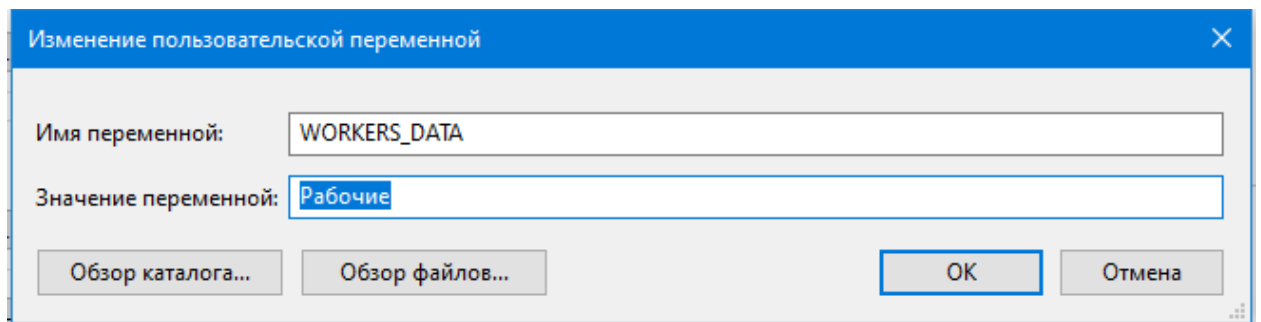


Рисунок 2 – Добавление переменной среды

```
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.

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

import sys
import json
import os.path
import argparse

def add_people(list_people, surname, name, post, datta):
    """
    Добавить данные магазина.
    """
    list_people.append(
        {
            "surname": surname,
            "name": name,
            "post": post,
            "datta": datta
        }
    )
    return list_people

def display_table(list_people):
    if list_people:
        line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 15,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)

        print('| {:^4} | {:^15} | {:^30} | {:^20} | {:^15} | '.format(
            "№",
            "Дата рождения",

```

```

39         "Фамилия",
40         "Имя",
41         "Знак Зодиака"
42     )
43 )
44
45 print(line)
46
47 for idx, spisok_new_new in enumerate(list_people, 1):
48     print(
49         '| {:>4} | {:<15} | {:<30} | {:<20} | {:<15} | '.format(
50             idx,
51             spisok_new_new.get('datta', ''),
52             spisok_new_new.get('surname', ''),
53             spisok_new_new.get('name', ''),
54             spisok_new_new.get('post', 0)
55         )
56     )
57     print(line)
58 else:
59     print("Список людей пуст.")
60
61
62 def select_zodiac(list_people, post_sear):
63     search_post = []
64     for post_sear_itme in list_people:
65         if post_sear == post_sear_itme['post']:
66             search_post.append(post_sear_itme)
67     return search_post
68
69
70 def save_people(file_name, list_people):
71     with open(file_name, 'w', encoding="utf-8") as fout:
72         json.dump(list_people, fout, ensure_ascii=False, indent=4)
73
74
75 def load_list_people(file_name):
76     with open(file_name, 'r', encoding="utf-8") as fin:
77         return json.load(fin)
78
79
80 def main(command_line=None):
81     file_parser = argparse.ArgumentParser(add_help=False)
82     file_parser.add_argument(
83         "-d",
84         "--data",
85         action="store",
86         required=False,
87         help="The datta file name"
88     )
89
90     parser = argparse.ArgumentParser("zodiac")
91     parser.add_argument(
92         "--version",

```

```
93         action="version",
94         version="%(prog)s 0.1.0"
95     )
96
97     subparsers = parser.add_subparsers(dest="command")
98
99     add = subparsers.add_parser(
100         "add",
101         parents=[file_parser],
102         help="Add a new people"
103     )
104
105     add.add_argument(
106         "-sn",
107         "--surname",
108         action="store",
109         required=True,
110         help="The peoples' surname"
111     )
112
113     add.add_argument(
114         "-n",
115         "--name",
116         action="store",
117         required=True,
118         help="The peoples' name"
119     )
120
121     add.add_argument(
122         "-p",
123         "--post",
124         action="store",
125         required=True,
126         help="The peoples' zodiac sign"
127     )
128
129     add.add_argument(
130         "-dt",
131         "--datta",
132         action="store",
133         type=int,
134         required=True,
135         help="Date of Birth"
136     )
137
138     _ = subparsers.add_parser(
139         "display",
140         parents=[file_parser],
141         help="Display all peoples"
142     )
143
144     select = subparsers.add_parser(
145         "select",
146         parents=[file_parser],
```



```

147         help="Select the zodiac"
148     )
149
150     select.add_argument(
151         "-PS",
152         "--post_sear",
153         action="store",
154         type=str,
155         required=True,
156         help="The find zodiac"
157     )
158
159     args = parser.parse_args(command_line)
160
161     data_file = args.data
162     if not data_file:
163         data_file = os.environ.get("PEOPLE")
164     if not data_file:
165         print("The data file name is absent", file=sys.stderr)
166         sys.exit(1)
167
168     is_dirty = False
169     if os.path.exists(data_file):
170         people = load_list_people(data_file)
171     else:
172         people = []
173     if args.command == "add":
174         people = add_people(
175             people,
176             args.surname,
177             args.name,
178             args.post,
179             args.datta
180         )
181     is_dirty = True
182     elif args.command == "display":
183         display_table(people)
184
185     elif args.command == "select":
186         selected = select_zodiac(people, args.post_sear)
187         display_table(selected)
188
189     if is_dirty:
190         save_people(data_file, people)
191
192
193     if __name__ == '__main__':
194         main()

```



Рисунок 4 – Код индивидуального задания

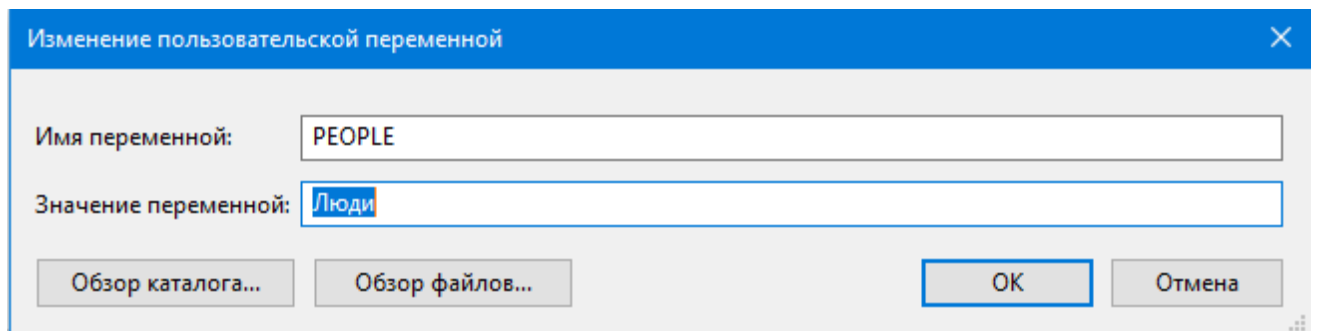


Рисунок 5 – Добавление переменной среды

```
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 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 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
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.