

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.18

Работа с переменными окружения в Python3.

Выполнил студент группы

ИВТ-б-о-21-1

Мальцев Н.А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

Цель работы: приобретение навыков построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Проработка примера.

Код программы:

```
#!/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):
    staff.append(
        {
            "name": name,
            "post": post,
            "year": year
        }
    )

    return staff

def display_workers(staff):
    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",
        help="The year of hiring"
    )
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Select the 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(command_line)
    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)

```

Результат выполнения программы:



```

+ FullyQualifiedTypeName : CommandNotFoundException

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Пример> python Example4.py add --name="Сидоров Сидор" --post="Главный инженер" --year=2012
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Пример>

```

Рисунок 1. Результат работы программы

Выполнение задания.

Код программы:

```

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

import json
import os.path
import argparse
import sys

```

```

def add_mans(list_man, name, number, date_):
    # Добавление данных
    list_man.append({"name": name, "number": number, "date": date_})
    return list_man

def list_d(list_man):
    """
    Отображение списка людей
    """
    # Проверить, что список не пуст
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-"
        " * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20} |".format(
                    idx, man.get("name", ""), man.get("number", 0),
                    man.get("date", 0)
                )
            )

            print(line)
    else:
        print("Список работников пуст: ")

def select_mans(mans_list, numb):
    """
    Отбор пользователей с заданным номером телеофна
    """
    count = 0
    if mans_list:
        for man in mans_list:
            if man.get("number") == numb:
                count += 1
                print("{:>4}: {}".format(count, man.get("name", "")))
                print("Номер телефона:", man.get("number", ""))
                print("Дата рождения:", man.get("date", ""))
    else:
        print("Список пользователей пуст.")
    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")

def save_workers(file_name_1, staff):
    """
    Сохранить всех пользователей в файл JSON.
    """
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)

```

```

def load_workers(file_name_2):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name_2, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создаем родительский парсер для определения имени файла
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument("filename", action="store", help="The date file name")
    # Основной парсер командной строки
    parser = argparse.ArgumentParser("workers")
    parser.add_argument("--version", action="version", version="% (prog)s 0.1.0")

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

    # Субпарсер для добавления пользователей
    add = subparser.add_parser("add", parents=[file_parser], help="Add a new worker")
    add.add_argument(
        "-n", "--name", action="store", required=True, help="The worker name"
    )
    add.add_argument(
        "-N", "--number", action="store", type=str, help="The workers phone number"
    )
    add.add_argument(
        "-y", "--year", action="store", type=int, required=True, help="Man's birthdate"
    )
    # Субпарсер для отображения всех пользователей
    _ = subparser.add_parser(
        "display", parents=[file_parser], help="Display information about users"
    )
    # Субпарсер для выбора пользователей
    select = subparser.add_parser("select", parents=[file_parser], help="Select users")
    select.add_argument(
        "-p", "--phone", action="store", type=str, help="The required period"
    )
    # Разбор аргументов командной строки
    args = parser.parse_args(command_line)
    data_file = args.filename
    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 пользователей из файла, если он существует
    is_dirty = False
    if os.path.exists(args.filename):
        mans = load_workers(args.filename)
    else:
        mans = []
    # Добавление пользователя
    if args.command == "add":
        mans = add_mans(mans, args.name, args.number, args.year)
        is_dirty = True
    # Отображение всех пользователей

```

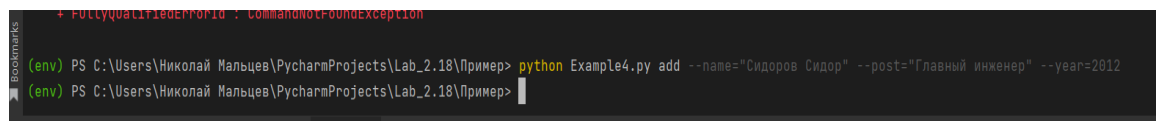
```

elif args.command == "display":
    list_d(mans)
# Отбор требуемых пользователей
elif args.command == "select":
    select_mans(mans, args.phone)
if is_dirty:
    save_workers(args.filename, mans)

if __name__ == "__main__":
    main()

```

Результат выполнения программы:



```

+ FullyQualifiedTypeName : CommandNotFoundException

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Пример> python Example4.py add --name="Сидоров Сидор" --post="Главный инженер" --year=2012
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Пример>

```

Рисунок 2. Результат выполнения

Задание повышенной сложности.

Код программы:

```

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

import json
import os.path
import argparse
import sys
from dotenv import load_dotenv

def add_mans(list_man, name, number, date_):
    # Добавление данных
    list_man.append({"name": name, "number": number, "date": date_})
    return list_man

def list_d(list_man):
    """
    Отображение списка людей
    """
    # Проверить, что список не пуст
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-"
" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20} |".format(
                    idx, man.get("name", ""), man.get("number", 0),

```

```

man.get("date", 0)
        )
    )

    print(line)
else:
    print("Список работников пуст: ")

def select_mans(mans_list, numb):
    """
    Отбор пользователей с заданным номером телефона
    """
    count = 0
    if mans_list:
        for man in mans_list:
            if man.get("number") == numb:
                count += 1
                print("{:>4}: {}".format(count, man.get("name", "")))
                print("Номер телефона:", man.get("number", ""))
                print("Дата рождения:", man.get("date", ""))
            else:
                print("Список пользователей пуст.")
    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")

def save_workers(file_name_1, staff):
    """
    Сохранить всех пользователей в файл JSON.
    """
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)

def load_workers(file_name_2):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name_2, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создаем родительский парсер для определения имени файла
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-f", "--filename", action="store", required=False, help="The data
file name"
    )
    # Основной парсер командной строки
    parser = argparse.ArgumentParser("workers")
    parser.add_argument("--version", action="version", version="% (prog)s
0.1.0")

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

    # Субпарсер для добавления пользователей
    add = subparser.add_parser("add", parents=[file_parser], help="Add a new
worker")
    add.add_argument(
        "-n", "--name", action="store", required=True, help="The worker name"
    )

```



```

    add.add_argument(
        "-N", "--number", action="store", type=str, help="The workers phone
number"
    )
    add.add_argument(
        "-y", "--year", action="store", type=int, required=True, help="Man's
birthdate"
    )
    # Субпарсер для отображения всех пользователей
    _ = subparser.add_parser(
        "display", parents=[file_parser], help="Display information about
users"
    )
    # Субпарсер для выбора пользователей
    select = subparser.add_parser("select", parents=[file_parser],
help="Select users")
    select.add_argument(
        "-p", "--phone", action="store", type=str, help="The required period"
    )
    # Разбор аргументов командной строки
    args = parser.parse_args(command_line)
    data_file = args.filename
    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("STUDENTS_DATA")
    if not data_file:
        print("The data file name is absent", file=sys.stderr)
        sys.exit(1)
    # Загрузить всех пользователей из файла, если он существует
    is_dirty = False
    if os.path.exists(data_file):
        mans = load_workers(data_file)
    else:
        mans = []
    # Добавление пользователя
    if args.command == "add":
        mans = add_mans(mans, args.name, args.number, args.year)
        is_dirty = True
    # Отображение всех пользователей
    elif args.command == "display":
        list_d(mans)
    # Отбор требуемых пользователей
    elif args.command == "select":
        select_mans(mans, args.phone)
    if is_dirty:
        save_workers(data_file, mans)

if __name__ == "__main__":
    main()

```

Результат выполнения программы:

```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Индивидуальное задание> python Ind_task_2.py add --name="Nik" --number=123 --year=2003
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Индивидуальное задание> python Ind_task_2.py display
+-----+-----+-----+-----+
| No |      Имя      | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Nik           | 123           | 2003          |
| 2 | Nik           | 123           | 2003          |
+-----+-----+-----+-----+
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.18\Индивидуальное задание>

```

Рисунок 3. Результат работы программы

Вывод: в ходе работы были приобретены навыки работы с переменными с помощью языка программирования Python версии 3.x.