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 №3
дисциплины «Анализ данных»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

_____
(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокомуникаций, доцент
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Разработка приложений с интерфейсом командной строки (CLI) в Python3

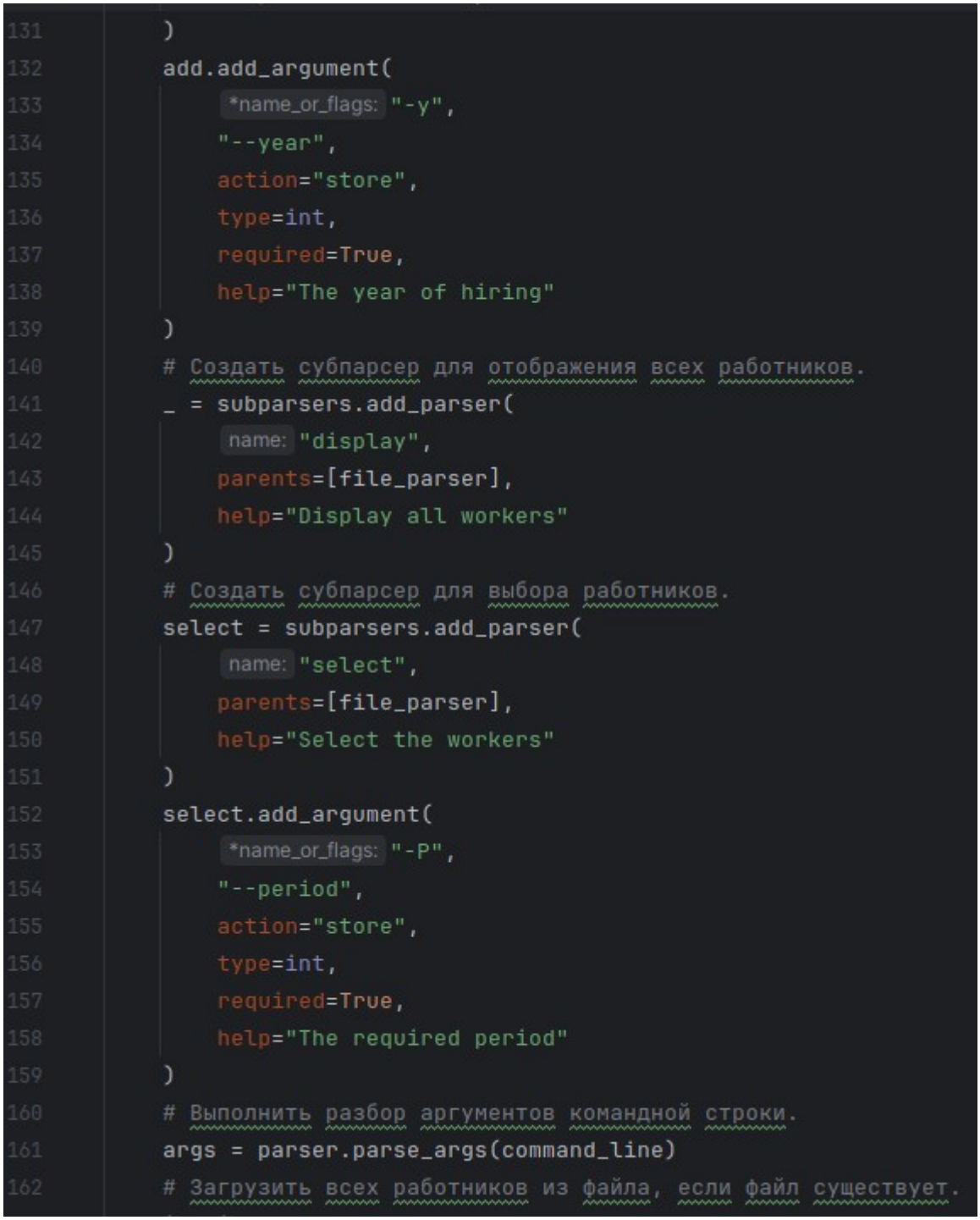
Цель: Приобретение построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Проработал пример:

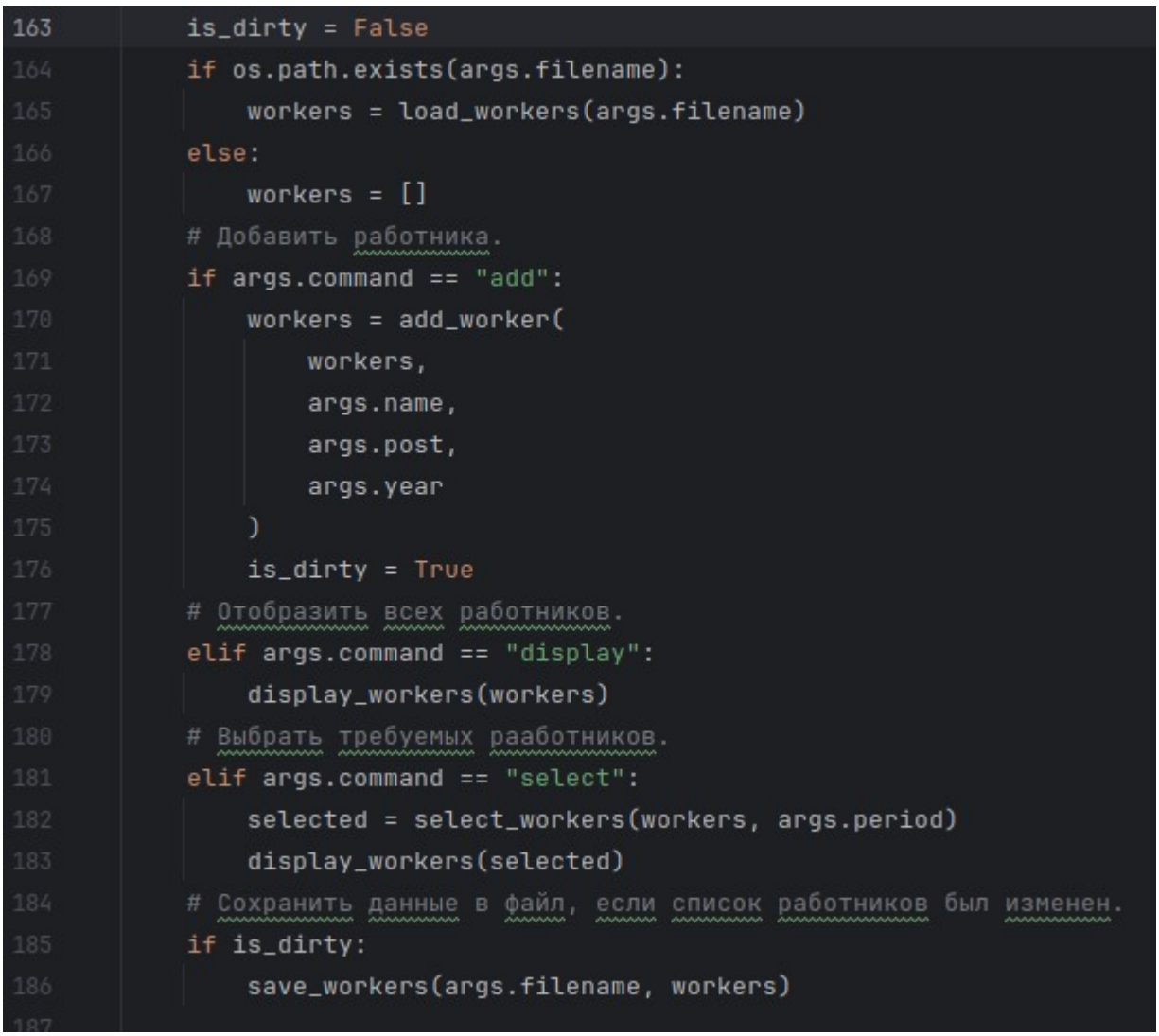
```python
def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        *name_or_flags: "filename",
        action="store",
        help="The data file name"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
        *name_or_flags: "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления работника.
    add = subparsers.add_parser(
        name: "add",
        parents=[file_parser],
        help="Add a new worker"
    )
    add.add_argument(
        *name_or_flags: "-n",
        "--name",
        action="store",
        required=True,
        help="The worker's name"
    )
    add.add_argument(
        *name_or_flags: "-p",
        "--post",
        action="store",
        help="The worker's post"
```

Рисунок 1. Измененный код примера (часть 1)



```
    )
    add.add_argument(
        *name_or_flags: "-y",
        "--year",
        action="store",
        type=int,
        required=True,
        help="The year of hiring"
    )
    # Создать субпарсер для отображения всех работников.
    _ = subparsers.add_parser(
        name: "display",
        parents=[file_parser],
        help="Display all workers"
    )
    # Создать субпарсер для выбора работников.
    select = subparsers.add_parser(
        name: "select",
        parents=[file_parser],
        help="Select the workers"
    )
    select.add_argument(
        *name_or_flags: "-P",
        "--period",
        action="store",
        type=int,
        required=True,
        help="The required period"
    )
    # Выполнить разбор аргументов командной строки.
    args = parser.parse_args(command_line)
    # Загрузить всех работников из файла, если файл существует.
```

Рисунок 2. Измененный код примера (часть 2)



```
    is_dirty = False
    if os.path.exists(args.filename):
        workers = load_workers(args.filename)
    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(args.filename, workers)
```

Рисунок 3. Измененный код примера (часть 3)

Выполнил индивидуальное задание: Нужно в программе из прошлой лабораторной работы реализовать интерфейс командной строки:

Рисунок 4. Измененный код индивидуального задания (часть 1)

```
add_parser.add_argument(
    *name_or_flags: "--second",
    action="store",
    required=True,
    help="Место прибытия"
)

list_parser = subparsers.add_parser(
    name: "list",
    parents=[file_parser],
    help="Показать данные из JSON-файла"
)

# Разбор аргументов командной строки.
args = parser.parse_args(command_line)

# Загрузить маршруты, если файл существует
fill = False
if os.path.exists(args.filename):
    routes = import_json(args.filename)
else:
    routes = []

# Добавить маршрут.
if args.command == "add":
    add_route(routes, args.first, args.second)
    fill = True
```

Рисунок 5. Измененный код индивидуального задания (часть 2)

```
if args.command == "list":
    import_json(args.filename)

if fill:
    export_to_json(args.filename, routes)

# Показать список маршрутов.
elif args.command == "list":
    list_of_routes(routes)
```

Рисунок 6. Измененный код индивидуального задания (часть 3)

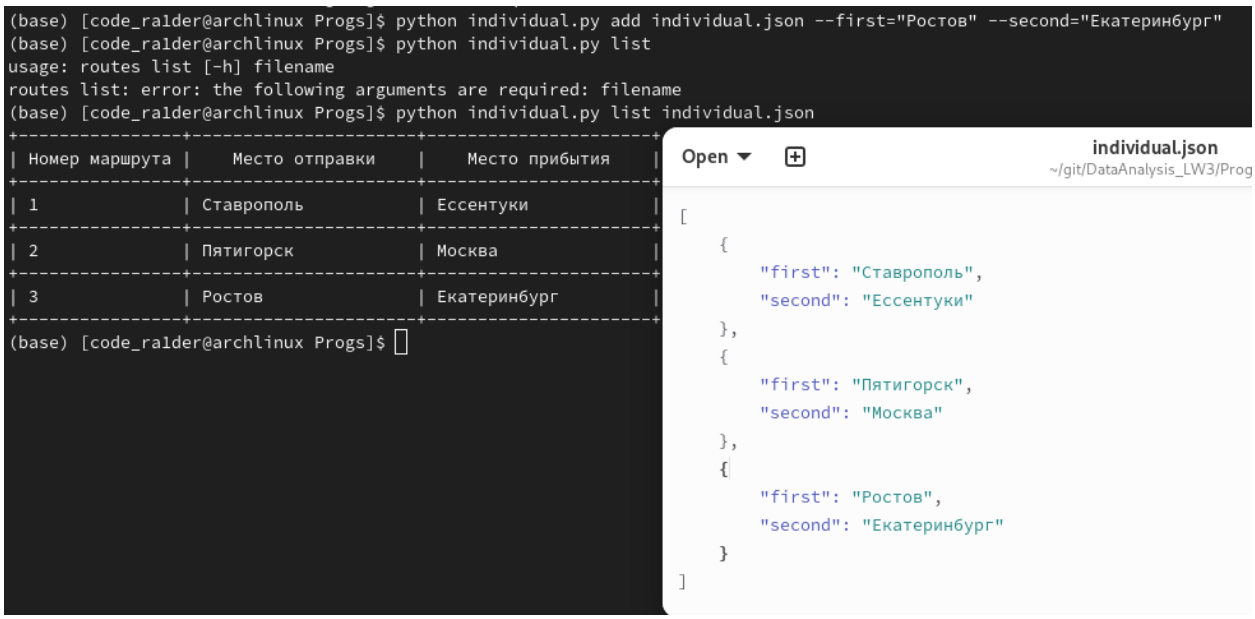


Рисунок 6. Проверка работоспособности программы

Ответы на вопросы:

1. Терминал — программа, которая обеспечивает взаимодействие пользователя с командной строкой. Он может быть как графическим, так и текстовым, и имеет внедренное распознавание команд от обычного текста

Консоль — окно, в котором отображается командная строка ОС. Является частью терминала, ведь именно благодаря консоли и обеспечивается взаимодействие пользователя и компьютером

2. Консольное приложение — программа, запускаемая и выполняемая в командной строке. Они обычно не имеют графического интерфейса, а значит выполняются обычно в текстовом режиме

3. Средства Python для построения приложений командной строки:

- argparse — модуль, предоставляющий простые инструменты для создания интерфейса командной строки

- getopt — низкоуровневый вариант argparse, обеспечивающий парсинг опций командной строки в стиле Unix

- sys.argv — стандартный список аргументов командной строки в Python

4. sys.argv включает в себя аргументы командной строки, которые запускаются вместе со скриптом. Парсинг аргументов в нем должен быть реализован вручную с использованием методов строк и списков

5. getopt поддерживает короткие и длинные опции с корректным разбором аргументов, однако работа с ним значительно сложнее, чем с argparse

6. argparse включает в себя расширенные и удобные инструменты для создания интерфейса командной строки. Он автоматически генерирует справку на основе определенных аргументов, поддерживает многие типы аргументов и их проверку и легко интегрируется с кодом программы в ясный синтаксис для определения аргументов

Вывод: В ходе данной лабораторной работы были приобретены навыки создания интерфейса командной строки с помощью языка программирования Python версии 3.x.