

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»**

Выполнил:  
Гайчук Дарья Дмитриевна  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика  
и вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А.-доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

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

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

**Порядок выполнения работы:**

Задание 1. Изучила теоретический материал работы, создала общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавила файл .gitignore с необходимыми правилами.

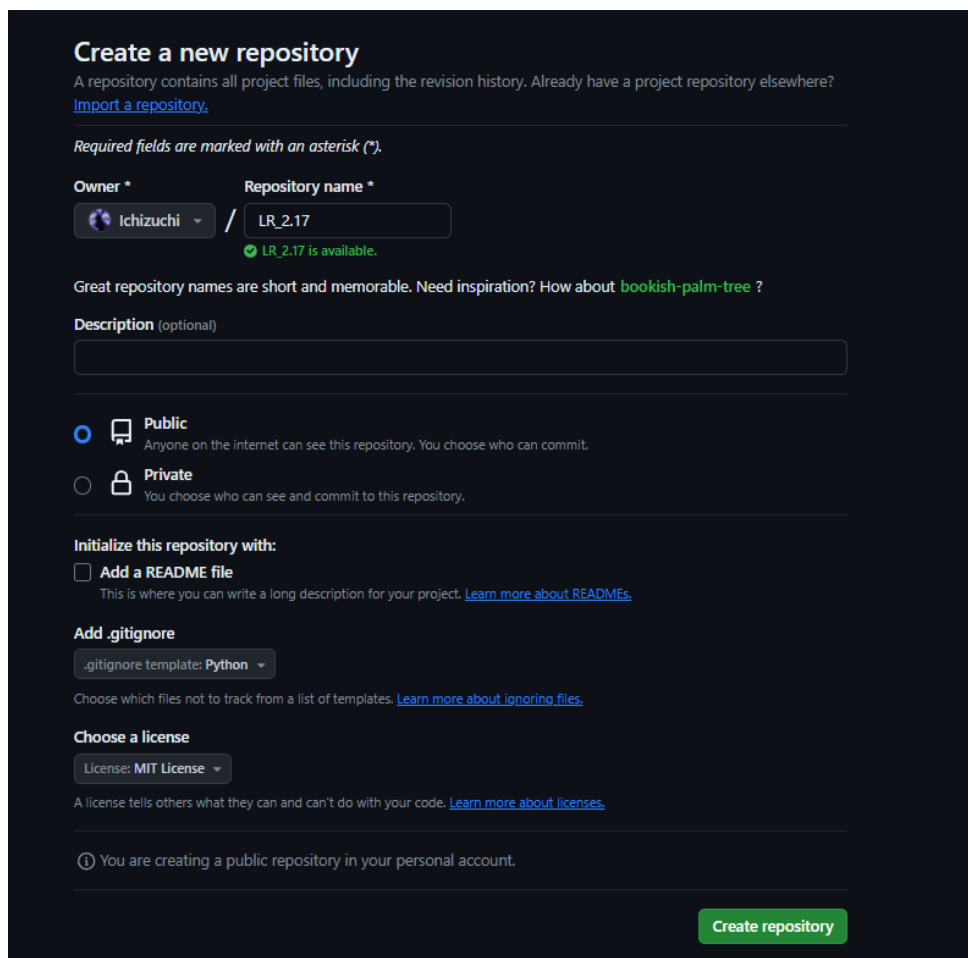
The image shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there's a section for 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'Ichizuchi' and the 'Repository name' field is 'LR\_2.17', with a green checkmark indicating it's available. There's a 'Description' field labeled '(optional)'. Under 'Visibility', the 'Public' option is selected, with a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is also visible. In the 'Initialize this repository with:' section, the 'Add a README file' checkbox is unchecked. The 'Add .gitignore' section shows the '.gitignore template: Python' selected. The 'Choose a license' section has 'License: MIT License' selected. At the bottom, there's a note: 'You are creating a public repository in your personal account.' and a green 'Create repository' button.

Рисунок 1. Создан новый репозиторий

Клонировала репозиторий на свой компьютер. В ходе данной лабораторной работы работала с моделью ветвления git-flow.

```
Switched to a new branch 'developer'
● @Ichizuchi →/workspaces/LR_2.17 (developer) $ git checkout developer
  Already on 'developer'
  Your branch is up to date with 'origin/developer'.
○ @Ichizuchi →/workspaces/LR_2.17 (developer) $
```

Рисунок 2. Клонирование и модель ветвления git-flow

Создала виртуальное окружение Anaconda

```
● @Ichizuchi →/workspaces/LR_2.17 (main) $ conda create -n myenv python=3.10
Channels:
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/conda/envs/myenv

added / updated specs:
  - python=3.10

The following packages will be downloaded:
```

package	build	
-----	-----	
_libgcc_mutex-0.1	main	3 KB
_openmp_mutex-5.1	1_gnu	21 KB
bzip2-1.0.8	h7b6447c_0	78 KB
ca-certificates-2023.12.12	h06a4308_0	126 KB

Рисунок 3. Создание виртуального окружения

Создала виртуальное окружение (ВО) Miniconda и активировала его, также установила необходимые пакеты isort, black, flake8.

```

● @Ichizuchi →/workspaces/LR_2.17 (main) $ conda install -c conda-forge black
Channels:
  - conda-forge
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/conda

added / updated specs:
  - black

The following packages will be downloaded:


```

package	build		
_libgcc_mutex-0.1	conda_forge	3 KB	conda-forge
_openmp_mutex-4.5	2_gnu	23 KB	conda-forge
black-24.2.0	py312h7900ff3_0	371 KB	conda-forge
ca-certificates-2024.2.2	hbcca054_0	152 KB	conda-forge
certifi-2024.2.2	pyhd8ed1ab_0	157 KB	conda-forge
click-8.1.7	unix_pyh707e725_0	82 KB	conda-forge
conda-24.1.2	py312h7900ff3_0	1.2 MB	conda-forge
libexpat-2.5.0	hcb278e6_1	76 KB	conda-forge
libgcc-ng-13.2.0	h807b86a_5	752 KB	conda-forge
libgomp-13.2.0	h807b86a_5	410 KB	conda-forge

Рисунок 4. Установка пакета black

```

● @Ichizuchi →/workspaces/LR_2.17 (main) $ conda install -c conda-forge flake8
Channels:
  - conda-forge
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/conda

added / updated specs:
  - flake8

The following packages will be downloaded:


```

package	build		
flake8-7.0.0	pyhd8ed1ab_0	108 KB	conda-forge
mccabe-0.7.0	pyhd8ed1ab_0	11 KB	conda-forge

Рисунок 5. Установка пакета flake8

```
Executing transaction: done
● @Ichizuchi →/workspaces/LR_2.17 (main) $ conda install -c conda-forge isort
Channels:
  - conda-forge
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/conda

added / updated specs:
  - isort

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
isort-5.13.2 | pyhd8ed1ab_0 | 72 KB | conda-forge
-----|-----|-----|-----
Total: | 72 KB
```

Рисунок 6. Установка пакета isort

Работа с примером №1.

Для примера 1 лабораторной работы 2.16 разработайте интерфейс командной

Добавление следующих подкоманд:

- add – добавление рабочего, имя которого задано в аргументе с параметром - name,
- должность – в аргументе с параметром --post , а год поступления – в аргументе с параметром -year .
- display – отображение списка всех работников.
- select – выбор и отображение требуемых работников, у которых заданный период передается через аргумент с параметром – period.

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'LR\_2.17 [CODESPACES: SYMMETRICAL EUREKA]' with files: 'flow', 'data.json', 'flights.json', 'ind\_1.py', 'primer\_1.py', '.gitignore', 'LICENSE', and 'README.md'. The 'primer\_1.py' file is selected and its content is displayed in the code editor. The code defines two functions: 'add\_worker' and 'display\_workers'. The 'add\_worker' function appends a worker to a list, and the 'display\_workers' function prints the list of workers in a formatted table. The terminal at the bottom shows the output of running the script with the '-h' flag, displaying the usage and options.

```
10 def add_worker(staff, name, post, year):
11     """
12     Добавить данные о работнике.
13     """
14     staff.append({
15         "name": name,
16         "post": post,
17         "year": year
18     })
19     return staff
20
21
22 def display_workers(staff):
23     """
24     Отобразить список работников.
25     """
26     # Проверить, что список работников не пуст.
27     if staff:
28         # Заголовок таблицы.
29         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
30             '-' * 4,
31             '-' * 30,
32             '-' * 20,
33             '-' * 8
34         )
35         print(line)
36         print('| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
37             "И.",
38             "Ф.И.О.",
39             "Должность",
40             "Год"
41         ))
42         print(line)
43         for worker in staff:
44             print('| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
45                 worker["name"],
46                 worker["post"],
47                 worker["year"],
48                 ""
49             ))
50         print(line)
```

```
if os.path.exists(args.filename):
    AttributeError: 'Namespace' object has no attribute 'filename'
@Ichizuchi →/workspaces/LR_2.17 (main) $ flow/primer_1.py -h
bash: flow/primer_1.py: Permission denied
@Ichizuchi →/workspaces/LR_2.17 (main) $ python flow/primer_1.py -h
usage: workers [-h] [--version] {add,display,select} ...

positional arguments:
  {add,display,select}
    add                Add a new worker
    display            Display all workers
    select            Select the workers

options:
  -h, --help            show this help message and exit
  --version            show program's version number and exit
@Ichizuchi →/workspaces/LR_2.17 (main) $
```

Рисунок 7. Работа с примером №1

Выполнение индивидуального задания. Для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

```
flow > ind_1.py > ...
4 import json
5 import os.path
6
7 data_file = "flights.json"
8
9 def input_flights():
10     flights = []
11     n = int(input("Введите количество рейсов: "))
12
13     for i in range(n):
14         flight = {}
15         flight["город назначения"] = input("Введите город назначения: ")
16         flight["номер рейса"] = input("Введите номер рейса: ")
17         flight["тип самолета"] = input("Введите тип самолета: ")
18         flights.append(flight)
19
20     flights.sort(key=lambda x: x["город назначения"])
21     return flights
22
23 def print_flights_with_plane_type(flights):
24     plane_type = input("Введите тип самолета: ")
25     found = False
26     for flight in flights:
27         if flight["тип самолета"] == plane_type:
28             print(f"Город назначения: {flight['город назначения']}, Номер рейса: {flight['номер рейса']}")
29             found = True
30     if not found:
31         print("Рейсы с указанным типом самолета не найдены")
32
33 def save_data_to_json(flights):
34     with open(data_file, 'w', encoding='utf-8') as file:
35         json.dump(flights, file, ensure_ascii=False)
36
37 def load_data_from_json():
38     if os.path.exists(data_file):
39         with open(data_file, 'r', encoding='utf-8') as file:
40             return json.load(file)
```

Рисунок 8. Выполнение индивидуального задания

```
primer_1.py U  COMMIT_EDITMSG  ind_1.py U  flights.json U x
flow > {} flights.json > ...
1 [{"город назначения": "Казань", "номер рейса": "334455", "тип самолета": "Аэробус"}, {"город назначения": "Москва", "номер рейса": "552233", "тип самолета": "Аэро
```

Рисунок 9. Файл json с данными о рейсах

Деактивировала виртуальное окружение.

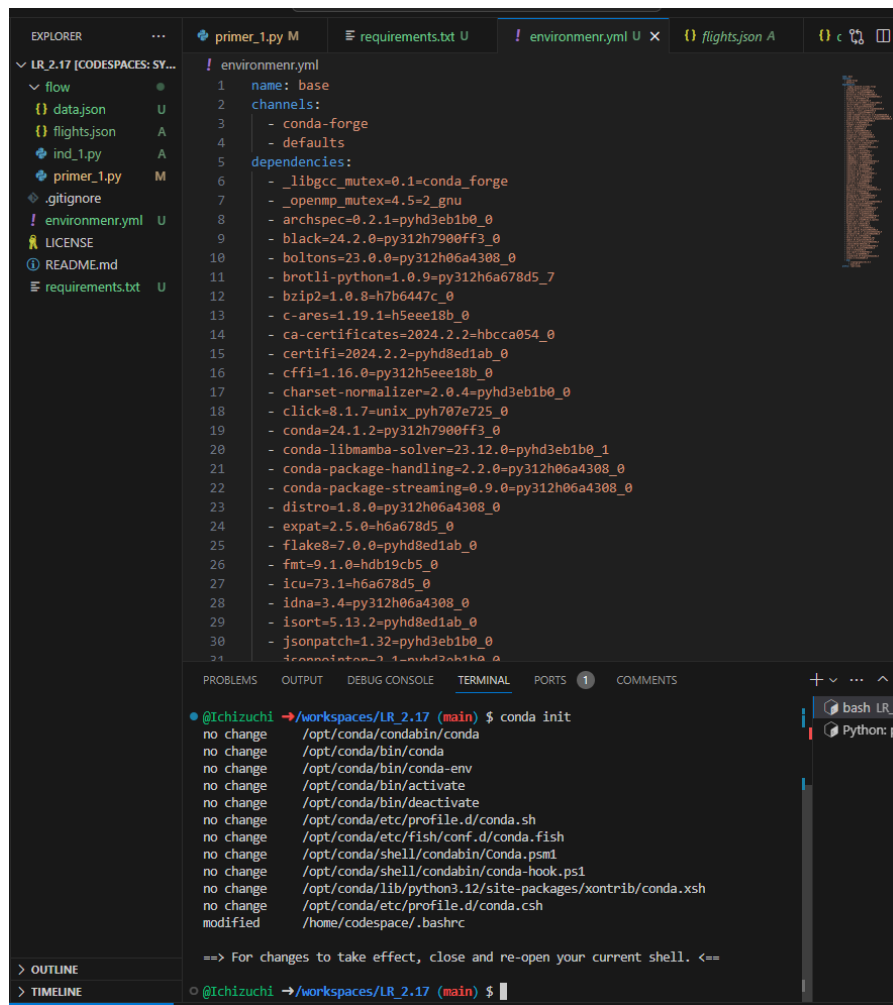


Рисунок 10. Деактивация ВО

Ответы на контрольные вопросы

1. В чем отличие терминала и консоли?

Терминал (Terminal): В общем смысле, терминал — это программа, предоставляющая текстовый интерфейс для взаимодействия с операционной системой. Это может быть командная строка в графической среде, также известная как терминал в Unix-подобных системах. В контексте Python, "терминал" может означать окно командной строки, в котором вы запускаете скрипты Python.

Консоль (Console): В Python термин "консоль" часто используется для обозначения интерактивной оболочки Python (REPL - Read-Eval-Print Loop), где вы можете вводить команды Python непосредственно и видеть результаты выполнения. В графической среде Windows, "консоль" может также относиться к окну командной строки (Command Prompt) или PowerShell.



## 2. Что такое консольное приложение?

Консольное приложение (или текстовое приложение) — это приложение, взаимодействие с пользователем которого осуществляется через текстовый интерфейс командной строки. В отличие от графических приложений, консольные приложения не используют графический пользовательский интерфейс (GUI) и обычно работают в текстовом режиме.

## 3. Какие существуют средства языка программирования Python для построения приложений командной строки?

Модуль `argparse`: Этот модуль предоставляет инструменты для анализа аргументов командной строки. Он позволяет определять, какие аргументы ожидаются при запуске программы и как они должны обрабатываться. Модуль `sys`: Модуль `sys` предоставляет доступ к некоторым переменным и функциям, связанным с интерпретатором Python.

Например, `sys.argv` содержит список аргументов командной строки, переданных скрипту.

Модуль `click`: это библиотека, которая облегчает создание красивых и удобных в использовании интерфейсов командной строки.

Модуль `subprocess`: Этот модуль позволяет запускать другие программы из Python и взаимодействовать с ними. Он может быть полезен для выполнения внешних команд из командной строки.

## 4. Какие особенности построения CLI с использованием модуля `sys`?

Построение интерфейса командной строки (CLI) с использованием модуля `sys` включает в себя работу с аргументами командной строки, переданными скрипту. Основные шаги включают в себя: Импорт модуля `sys`: Обработка аргументов: Аргументы командной строки доступны в списке `sys.argv`. Этот список содержит имя скрипта (индекс 0) и все переданные аргументы. Обработка флагов и значений: Модуль `sys` не предоставляет специальных инструментов для обработки флагов и значений. Запуск из командной строки: Ваш скрипт может быть запущен из командной строки с аргументами

4. Какие особенности построение CLI с использованием модуля getopt?

Модуль getopt - это более старый и менее удобный способ обработки аргументов командной строки по сравнению с более современным модулем argparse. Однако, если вам нужна простая и легкая в использовании альтернатива, getopt может быть полезным.

- 1) Импорт модуля getopt
- 2) Определение параметров командной строки
- 3) Обработка ошибок
- 4) Запуск из командной строки

6. Какие особенности построение CLI с использованием модуля argparse

Модуль argparse предоставляет более мощные и удобные средства для построения интерфейса командной строки (CLI) по сравнению с getopt или простым использованием sys.argv. Вот основные особенности построения CLI с использованием модуля argparse:

- 1) Импорт модуля argparse
- 2) Создание объекта парсера
- 3) Добавление аргументов
- 4) Парсинг аргументов
- 5) Использование аргументов
- 6) Запуск из командной строки
7. Поддержка справки и документации

Вывод: приобрела навыки построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x