

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 №20
дисциплины «Основы программной инженерии»

Выполнил:

Джараян Арег Александрович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Лабораторная работа 2.17. 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.

Ход работы.

1. Создание нового репозитория с лицензией MIT.

Import a repository.

Required fields are marked with an asterisk (*).

Owner * Repository name *

aregdz / lab20

lab20 is available.

Great repository names are short and memorable. Need inspiration? How about curly-doodle ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

Create repository

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab20
$ git clone https://github.com/aregdz/lab20.git
Cloning into 'lab20'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab20
$
```

Рисунок 2 – клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/

```

Рисунок 3 – Файл .gitignore

```

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab20/lab20 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/lab20/lab20 (develop)

```

Рисунок 4 – организация ветки

4. Создал виртуальное окружение и установил нужные пакеты.

```

(venv) PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> pip install -r requirements.txt
Collecting black==24.2.0 (from -r requirements.txt (line 1))
  Using cached black-24.2.0-cp312-cp312-win_amd64.whl.metadata (74 kB)
Collecting cfgv==3.4.0 (from -r requirements.txt (line 2))
  Using cached cfgv-3.4.0-py2.py3-none-any.whl.metadata (8.5 kB)
Collecting click==8.1.7 (from -r requirements.txt (line 3))
  Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting colorama==0.4.6 (from -r requirements.txt (line 4))
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting distlib==0.3.8 (from -r requirements.txt (line 5))
  Using cached distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock==3.13.1 (from -r requirements.txt (line 6))
  Using cached filelock-3.13.1-py3-none-any.whl.metadata (2.8 kB)
Collecting flake8==7.0.0 (from -r requirements.txt (line 7))
  Using cached flake8-7.0.0-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting identify==2.5.35 (from -r requirements.txt (line 8))
  Using cached identify-2.5.35-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting isort==5.13.2 (from -r requirements.txt (line 9))
  Downloading isort-5.13.2-py3-none-any.whl.metadata (12 kB)
Collecting mccabe==0.7.0 (from -r requirements.txt (line 10))
  Using cached mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting mypy-extensions==1.0.0 (from -r requirements.txt (line 11))

```

Рисунок 5 - создание виртуального окружения

5.Проработал примеры из лабораторной работы.

```

PS D:\Рабочий стол\4 семестр\опи\lab20> cd lab20
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python workers.py add data.json --name="Джарян Апер" --post="it специалист" --year=2023
C:\Users\anegd\AppData\Local\Programs\Python\Python312\python.exe: can't open file 'D:\Рабочий стол\4 семестр\опи\lab20\lab20\workers.py': [Errno 2] No such file or directory
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python primer1.py add data.json --name="Джарян Апер" --post="it специалист" --year=2023
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20>

```

Рисунок 6 – добавление нового сотрудника

```

17  {
18      "name": "Mark Markdown",
19      "post": "Main engeneer",
20      "year": 2012
21  },
22  {
23      "name": "Сидоров Сидор",
24      "post": "Главны инженер",
25      "year": 2012
26  },
27  {
28      "name": "Сидоров Сидор",
29      "post": "Главны инженер",
30      "year": 2012
31  },
32  {
33      "name": "Джараян Арег",
34      "post": "it специалист",
35      "year": 2023
36  }
37  ]

```

Рисунок 7 – содержание файла data.json

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

```

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

import argparse
import json
import os.path
from datetime import date

def add_flight(flights, destination, departure_date, aircraft_type):
    flights.append(
        {
            "destination": destination,
            "departure_date": departure_date,
            "aircraft_type": aircraft_type
        }
    )

    return flights

```

```

def display_flights(flights):
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Destination",
                "Departure Date",
                "Aircraft Type"
            )
        )
        print(line)
        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    flight.get('destination', ''),
                    flight.get('departure_date', ''),
                    flight.get('aircraft_type', '')
                )
            )
            print(line)
    else:
        print("List of flights is empty.")

def select_flights(flights, date):
    result = []
    for flight in flights:
        if flight.get('aircraft_type') == date:
            result.append(flight)
    return result

def save_flights(file_name, flights):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(flights, fout, ensure_ascii=False, indent=4)

def load_flights(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(
        "filename",
        action="store",
        help="The data file name"
    )

    parser = argparse.ArgumentParser("flights")
    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 flight"
    )
    add.add_argument(
        "-d",
        "--destination",
        action="store",
        required=True,
        help="Destination of the flight"
    )
    add.add_argument(
        "-dd",
        "--departure_date",
        action="store",
        required=True,
        help="Departure date of the flight"
    )
    add.add_argument(
        "-at",
        "--aircraft_type",
        action="store",
        required=True,
        help="Aircraft type of the flight"
    )

    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all flights"
    )

    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select flights by departure date"
    )
    select.add_argument(
        "-D",
        "--date",
        action="store",
        required=True,
        help="Departure date to select flights"
    )

    args = parser.parse_args(command_line)

    is_dirty = False
    if os.path.exists(args.filename):
        flights = load_flights(args.filename)
    else:
        flights = []

    if args.command == "add":
        flights = add_flight(
            flights,
            args.destination,
            args.departure_date,
            args.aircraft_type
        )
        is_dirty = True

    elif args.command == "display":
        display_flights(flights)

    elif args.command == "select":
        selected = select_flights(flights, args.date)
        display_flights(selected)

    if is_dirty:
        save_flights(args.filename, flights)

```



```
if __name__ == "__main__":
    main()
```

Рисунок 8 – код для реализации задания

9. Самостоятельно изучите работу с пакетом `click` для построения интерфейса командной строки (CLI). Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета `click`.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import click
import json
import os.path

@click.group()
@click.version_option(version='0.1.0')
def flights():
    """
    Manage flight data.
    """
    pass

@flights.command()
@click.option("-f", "--filename", default="flights.json", help="The data file name")
@click.option("-d", "--destination", prompt="Destination", help="Destination of the flight")
@click.option("-dd", "--departure_date", prompt="Departure Date", help="Departure date of the flight")
@click.option("-at", "--aircraft_type", prompt="Aircraft Type", help="Aircraft type of the flight")
def add(filename, destination, departure_date, aircraft_type):
    """
    Add a new flight.
    """
    flights = load_flights(filename)
    flights.append({"destination": destination, "departure_date": departure_date,
"aircraft_type": aircraft_type})
    save_flights(filename, flights)

@flights.command()
@click.option("-f", "--filename", default="flights.json", help="The data file name")
def display(filename):
    """
    Display all flights.
    """
    flights = load_flights(filename)
    display_flights(flights)

@flights.command()
@click.option("-f", "--filename", default="flights.json", help="The data file name")
@click.option("-at", "--aircraft_type", prompt="Aircraft Type", help="Aircraft type to select flights")
def select(filename, aircraft_type):
    """
    Select flights by aircraft type.
    """
    flights = load_flights(filename)
    selected = select_flights(flights, aircraft_type)
    display_flights(selected)

def add_flight(flights, destination, departure_date, aircraft_type):
    """
    Add flight data.
    """
```

```

    flights.append(
        {
            "destination": destination,
            "departure_date": departure_date,
            "aircraft_type": aircraft_type
        }
    )
    return flights

def display_flights(flights):
    """
    Display list of flights.
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}+'.format('-'*4, '-'*30, '-'*20, '-'*8)
        print(line)
        print('| {:^4} | {:^30} | {:^20} | {:^8} |'.format("No", "Destination",
"Departure Date", "Aircraft Type"))
        print(line)
        for idx, flight in enumerate(flights, 1):
            print('| {:>4} | {:<30} | {:<20} | {:>8} |'.format(idx,
flight.get('destination', ''), flight.get('departure_date', ''),
flight.get('aircraft_type', '')))
            print(line)
        else:
            print("List of flights is empty.")

def select_flights(flights, aircraft_type):
    """
    Select flights by aircraft type.
    """
    result = []
    for flight in flights:
        if flight.get('aircraft_type') == aircraft_type:
            result.append(flight)
    return result

def save_flights(file_name, flights):
    """
    Save all flights to a JSON file.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(flights, fout, ensure_ascii=False, indent=4)

def load_flights(file_name):
    """
    Load all flights from a JSON file.
    """
    if os.path.exists(file_name):
        with open(file_name, "r", encoding="utf-8") as fin:
            return json.load(fin)
    else:
        return []

if __name__ == "__main__":
    flights()

```

Рисунок 9 – код для выполнения задания повышенной сложности

```

Aircraft Type: e33
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python zadanie2.py display
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|  1  | russia          | 12-03-2024      | e33  |
+-----+-----+-----+-----+
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python zadanie2.py select
|  2  | россия          | 12-03-2024      | a23  |
+-----+-----+-----+-----+
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python zadanie2.py select
Aircraft Type: e23
List of flights is empty.
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> python zadanie2.py select
Aircraft Type: e33
+-----+-----+-----+-----+
| No  | Destination      | Departure Date   | Aircraft Type |
+-----+-----+-----+-----+
|  1  | russia          | 12-03-2024      | e33  |
+-----+-----+-----+-----+
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20>

```

Рисунок 10 – выполнение задания

```

PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> flake8 zadanie1.py
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> flake8 zadanie2.py
primer.py:0:1: E902 FileNotFoundError: [Errno 2] No such file or directory: 'primer.py'

All done! 🎉👏 :
1 file reformatted.
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20> black zadanie1.py
reformatted zadanie1.py

All done! 🎉👏 :
1 file reformatted.
PS D:\Рабочий стол\4 семестр\опи\lab20\lab20>

```

Рисунок 11 – форматирование

Контрольные вопросы:

1.Отличие терминала и консоли:

Терминал (или терминальное окно) - это программа, которая предоставляет пользователю интерфейс для взаимодействия с операционной системой через текстовый интерфейс. Терминал обычно предоставляет доступ

к командной строке, где пользователь может вводить команды для выполнения различных задач.

Консоль - это текстовый интерфейс, предоставляемый операционной системой для взаимодействия с пользователем или другими программами. Он может быть встроенным в окружение рабочего стола или виртуальной машины. Консоль обычно работает поверх терминала и обеспечивает вывод текстовой информации и ввод команд пользователя.

2.Консольное приложение:

Консольное приложение - это приложение, которое работает в текстовом режиме и взаимодействует с пользователем через командную строку или терминал. Оно обычно используется для выполнения задач в операционной системе или автоматизации определенных процессов. Примеры консольных приложений включают текстовые редакторы, утилиты командной строки и скрипты.

3.Средства Python для построения приложений командной строки:

Модуль `sys`: для доступа к аргументам командной строки и другим системным параметрам.

Модуль `getopt`: для парсинга аргументов командной строки в стиле POSIX.

Модуль `argparse`: для создания гибких и мощных интерфейсов командной строки с поддержкой подпрограмм, позиционных и именованных аргументов, справочной информации и др.

4.Особенности построения CLI с использованием модуля sys:

Модуль `sys` предоставляет базовые инструменты для работы с аргументами командной строки, доступом к системным переменным и другими системными функциями.

С его помощью можно получить доступ к аргументам командной строки через список `sys.argv` и осуществить простой парсинг аргументов.

5.Особенности построения CLI с использованием модуля getopt:

Модуль `getopt` позволяет парсить аргументы командной строки в стиле POSIX с короткими и длинными опциями.

Он предоставляет более гибкие возможности по обработке аргументов, чем модуль `sys`, но требует более сложного кода для использования.

6. Особенности построения CLI с использованием модуля `argparse`:

Модуль `argparse` предоставляет мощные средства для создания гибких интерфейсов командной строки.

Он поддерживает позиционные и именованные аргументы, подпрограммы, группы аргументов, справочную информацию и многое другое.

`argparse` обеспечивает автоматическую проверку типов аргументов, генерацию справки и сообщений об ошибках.