

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.17

дисциплина «Программирование на языке Python»

Выполнил:

Тихоненко Борис Витальевич
2 курс, группа ИТС-б-з-22-1,
11.03.02

(подпись)

Проверил:

доцент, кандидат технических наук

Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г

Цель работы:

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

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.17>

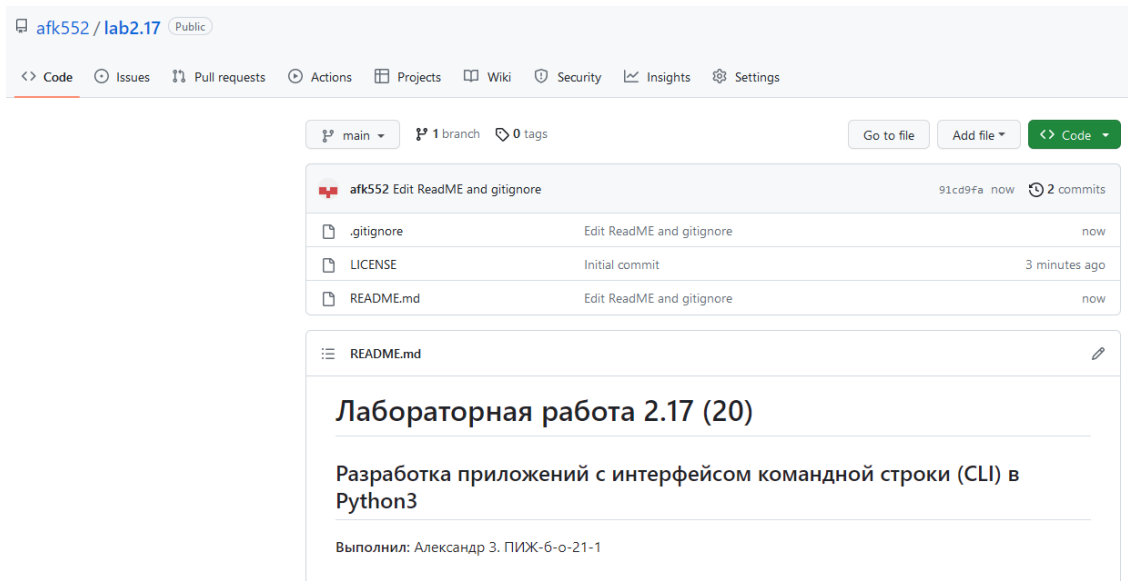


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

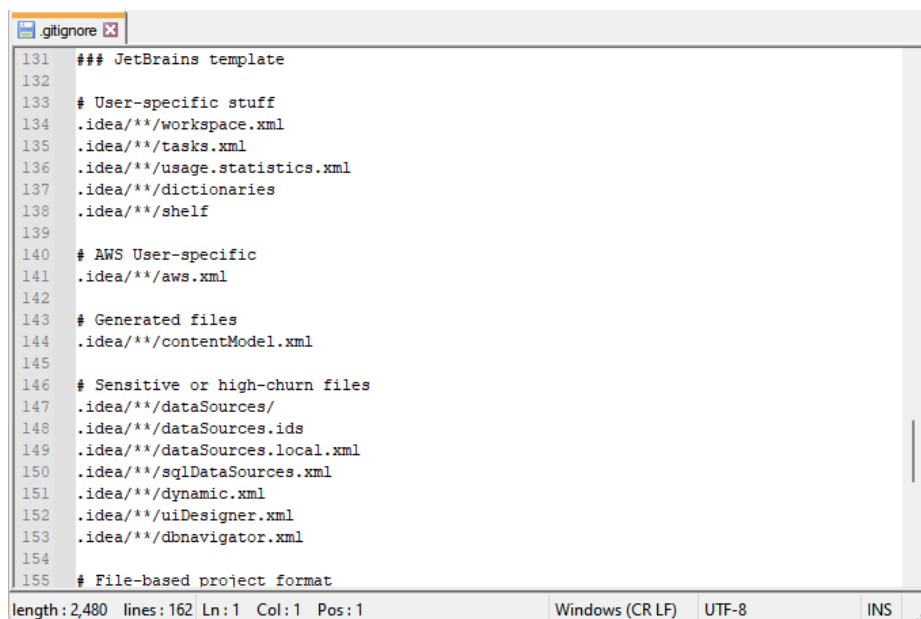
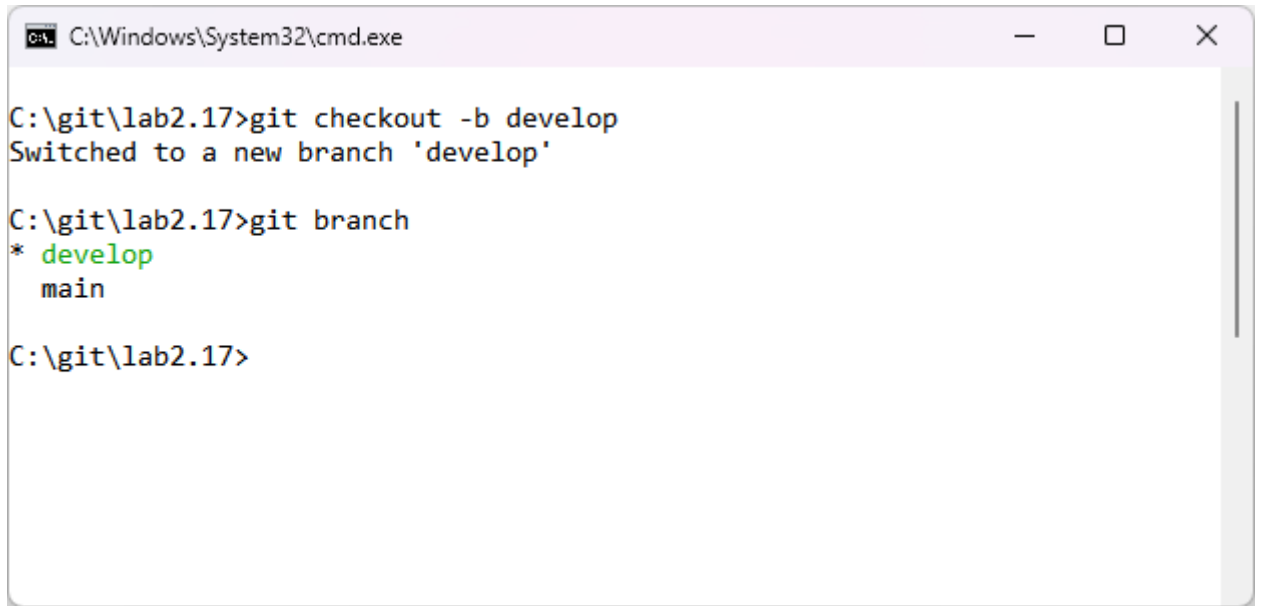


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.17>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.17>git branch
* develop
  main

C:\git\lab2.17>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

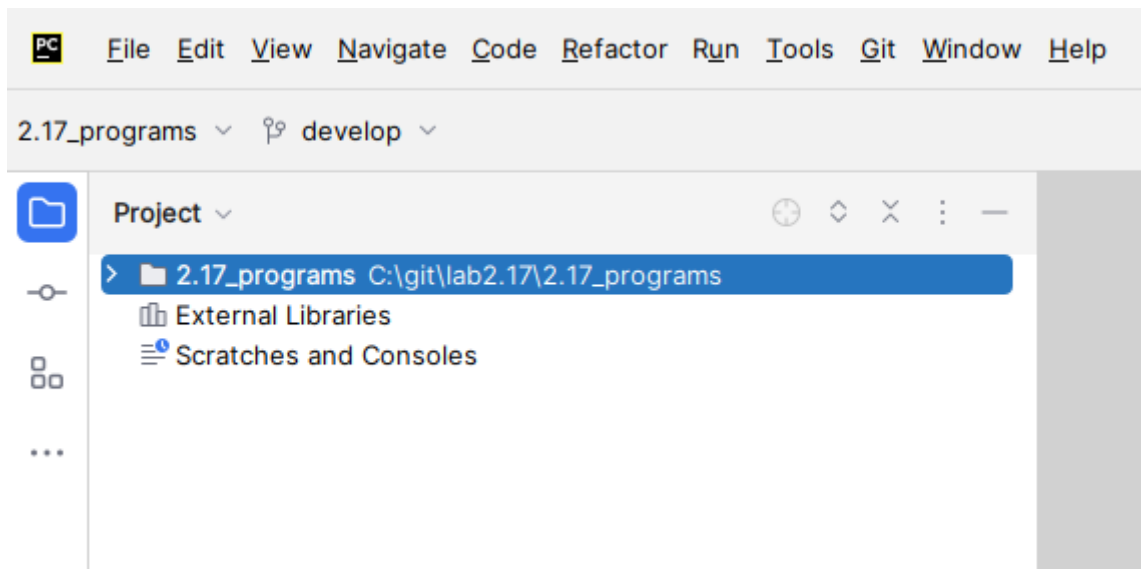


Рисунок 4 – Окно проекта в PyCharm

Индивидуальное задание.

Задание 1. Для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

```
138  def main(command_line=None):
139      """
140      Основная функция программы
141      """
142      # Создать родительский парсер для определения имени файла
143      file_parser = argparse.ArgumentParser(add_help=False)
144      file_parser.add_argument(
145          "filename", action="store", help="The data file name"
146      )
147
148      # Создать основной парсер командной строки
149      parser = argparse.ArgumentParser("people")
150      parser.add_argument(
151          "--version", action="version", version="%s alpha beta 0.0.1"
152      )
153
154      subparsers = parser.add_subparsers(dest="command")
155
156      # Создать субпарсер для добавления данных человека
157      add = subparsers.add_parser(
158          "add", parents=[file_parser], help="Добавить нового человека"
159      )
160      add.add_argument(
161          "-n",
162          "--name",
163          action="store",
164          required=True,
165          help="Имя и фамилия человека",
166          nargs="+",
167      )
168      add.add_argument("-p", "--pnumber", action="store", help="Номер телефона")
169      add.add_argument(
170          "-b",
171          "--birth",
172          action="store",
173          type=str,
174          required=True,
175          help="Person's birthday date",
176      )
177
```

Рисунок 5 – Код программы индивидуального задания 1 (1)

```

178     # Создать субпарсер для отображения всех людей
179     _ = subparsers.add_parser(
180         "display", parents=[file_parser], help="Отобразить всех людей"
181     )
182
183     # Создать субпарсер для выбора людей
184     select = subparsers.add_parser(
185         "select",
186         parents=[file_parser],
187         help="Выбрать людей по их месяцу рождения",
188     )
189
190     select.add_argument(
191         "-M",
192         "--month",
193         action="store",
194         type=str,
195         required=True,
196         help="Номер месяца рождения",
197     )
198
199     # Выполнить разбор аргументов командной строки
200     args = parser.parse_args(command_line)
201
202     # Загрузить всех людей из файла, если файл существует
203     is_dirty = False
204     if os.path.exists(args.filename):
205         people = load_workers(args.filename)
206     else:
207         people = []
208
209     # Добавить данные человека
210     if args.command == "add":
211         people = add_people(
212             people, " ".join(args.name), args.pnumber, args.birth
213         )
214         is_dirty = True
215
216     # Отобразить всех людей
217     elif args.command == "display":
218         display_people(people)

```

Рисунок 6 – Код программы индивидуального задания 1 (2)

```

220     # Выбрать требуемых людей
221     elif args.command == "select":
222         printed_month = args.month
223         corrected_month = correct_date(printed_month)
224         selected = select_people(people, corrected_month)
225         display_people(selected)
226
227     # Сохранить данные в файл, если список был изменен.
228     if is_dirty:
229         save_workers(args.filename, people)

```

Рисунок 7 – Код программы индивидуального задания 1 (3)

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1485]
(c) Microsoft Corporation. All rights reserved.

C:\git\lab2.17\2.17_programs>py ind1_arg.py --help
usage: people [-h] [--version] {add,display,select} ...

positional arguments:
  {add,display,select}
    add                Добавить нового человека
    display            Отобразить всех людей
    select             Выбрать людей по их месяцу рождения

options:
  -h, --help            show this help message and exit
  --version              show program's version number and exit

C:\git\lab2.17\2.17_programs> py ind1_arg.py add testo.json -n Test Testov -p +7947327472 -b 02.02.2002
[{'name': 'Test Testov', 'pnumber': '+7947327472', 'birth': datetime.datetime(2002, 2, 2, 0, 0)}]

C:\git\lab2.17\2.17_programs>py ind1_arg.py display testo.json
+-----+-----+-----+-----+
| №п/п | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Test Testov | +7947327472 | 02.02.2002 |
+-----+-----+-----+-----+

C:\git\lab2.17\2.17_programs>py ind1_arg.py select testo.json -M 2
+-----+-----+-----+-----+
| №п/п | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Test Testov | +7947327472 | 02.02.2002 |
+-----+-----+-----+-----+

C:\git\lab2.17\2.17_programs>

```

Рисунок 8 – Результат выполнения программы индивидуального задания 1 (1)

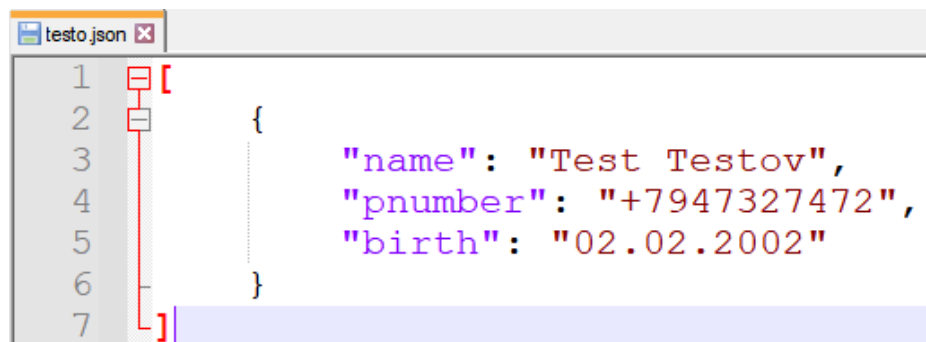


Рисунок 9 – Результат выполнения программы индивидуального задания 1 (2)

Задание повышенной сложности. 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`.



```
12  @click.group()
13  def cli():
14      pass
15
16
17  # afk552
18  @cli.command("display")
19  @click.argument("filename")
20  def display_data(filename):
21      """
22      Вывести людей из списка
23      """
24      if os.path.exists(filename):
25          people = load_workers(filename)
26      else:
27          people = []
28      display_people(people)
29
30  # afk552
31  @cli.command("select")
32  @click.argument("filename")
33  @click.option(
34      "--month",
35      help="Запросить людей, чьи дни рождения приходятся на месяц (число)",
36  )
37  def select_data(filename, month):
38      """
39      Выбрать людей по заданному месяцу рождения
40      """
41      if os.path.exists(filename):
42          people = load_workers(filename)
43      else:
44          people = []
45      select_people(f"select {month}", people)
46
```

Рисунок 11 – Код программы индивидуального задания 1 (1)

```

47 @cli.command("add")
48 @click.argument("filename")
49 @click.option("--name", help="Имя человека")
50 @click.option("--surname", help="Фамилия человека")
51 @click.option(
52     "--pnumber",
53     help="Номер телефона",
54 )
55 @click.option("--birth", help="Дата рождения человека (01.01.2077)")
56 def add_data(filename, name, surname, pnumber, birth):
57     """
58     Добавить людей
59     """
60     if os.path.exists(filename):
61         people = load_workers(filename)
62     else:
63         people = []
64         full_name = f"{name} {surname}"
65         birth = birth.split(".")
66         birth_dt = datetime(int(birth[2]), int(birth[1]), int(birth[0]))
67         people.append({"name": full_name, "pnumber": pnumber, "birth": birth_dt})
68     save_workers(filename, people)

```

Рисунок 12 – Код программы индивидуального задания 2 (2)

```

C:\Windows\System32\cmd.exe

C:\git\lab2.17\2.17_programs>py ind2_click.py --help
Usage: ind2_click.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  add      Добавить людей
  display  Вывести людей из списка
  select   Выбрать людей по заданному месяцу рождения

C:\git\lab2.17\2.17_programs>py ind2_click.py add --help
Usage: ind2_click.py add [OPTIONS] FILENAME

  Добавить людей

Options:
  --name TEXT      Имя человека
  --surname TEXT   Фамилия человека
  --pnumber TEXT   Номер телефона
  --birth TEXT     Дата рождения человека (01.01.2077)
  --help          Show this message and exit.

C:\git\lab2.17\2.17_programs>py ind2_click.py add testo.json --name Test --surname Testov --pnumber +79473274721 --birth 03.02.2002
[{'name': 'Test Testov', 'pnumber': '+79473274721', 'birth': datetime.datetime(2002, 2, 3, 0, 0)}]

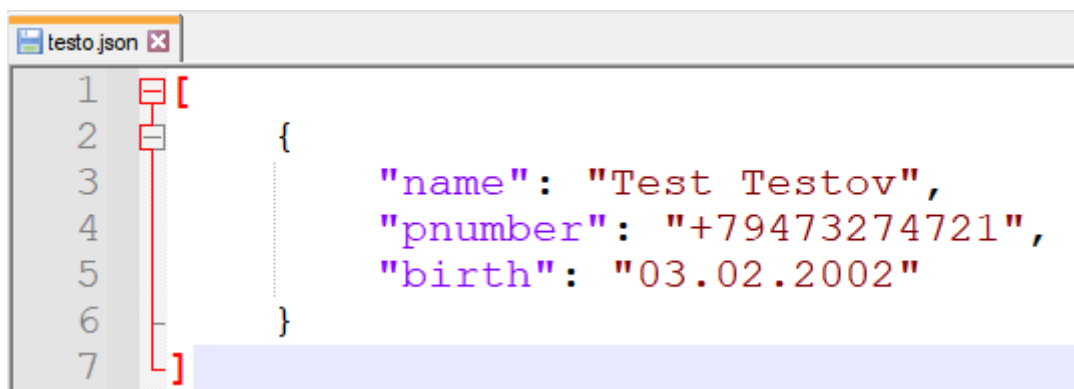
C:\git\lab2.17\2.17_programs>py ind2_click.py display testo.json
+-----+-----+-----+-----+
| №п/п | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Test Testov | +79473274721 | 03.02.2002 |
+-----+-----+-----+-----+

C:\git\lab2.17\2.17_programs>py ind2_click.py select testo.json --month 2
+-----+-----+-----+-----+
| №п/п | Фамилия Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Test Testov | +79473274721 | 03.02.2002 |
+-----+-----+-----+-----+

C:\git\lab2.17\2.17_programs>

```

Рисунок 13 – Результат выполнения программы индивидуального задания 2 (1)



```
1 [
2   {
3     "name": "Test Testov",
4     "pnumber": "+79473274721",
5     "birth": "03.02.2002"
6   }
7 ]
```

Рисунок 14 – Результат выполнения программы индивидуального задания 2
(2)

Вывод: В результате выполнения работы была изучена работа с модулями `argparse` и `click` создания интерфейса командной строки CLI для приложения.

Контрольные вопросы:

1. В чем отличие терминала и консоли?

Терминал (от лат. *terminus* — граница) — устройство или ПО, выступающее посредником между человеком и вычислительной системой. Обычно данный термин используется, когда точка доступа к системе вынесена в отдельное физическое устройство и предоставляет свой пользовательский интерфейс на основе внутреннего интерфейса (например, сетевых протоколов). Консоль — компьютер с клавиатурой и монитором.

2. Что такое консольное приложение?

Консольное приложение *console application* — вид ПО, разработанный с расчётом на работу внутри оболочки командной строки, т.е. опирающийся на текстовый ввод-вывод.

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

Python 3 поддерживает несколько различных способов обработки аргументов командной строки. Встроенный способ — использовать модуль `sys`.

С точки зрения имен и использования, он имеет прямое отношение к библиотеке C (libc). Вторым способом – это модуль `getopt`, который обрабатывает как короткие, так и длинные параметры, включая оценку значений параметров. Кроме того, существуют два других общих метода. Это модуль `argparse`, производный от модуля `optparse`, доступного до Python 2.7. Другой метод – использование модуля `docopt`, доступного на GitHub.

4. Какие особенности построения CLI с использованием модуля sys?

Это базовый модуль, который с самого начала поставлялся с Python. Он использует подход, очень похожий на библиотеку C, с использованием `argc` и `argv` для доступа к аргументам. Модуль `sys` реализует аргументы командной строки в простой структуре списка с именем `sys.argv`. Каждый элемент списка представляет собой единственный аргумент. Первый элемент в списке `sys.argv` [0] – это имя скрипта Python. Остальные элементы списка, от `sys.argv` [1] до `sys.argv` [n], являются аргументами командной строки с 2 по n. В качестве разделителя между аргументами используется пробел. Значения аргументов, содержащие пробел, должны быть заключены в кавычки, чтобы их правильно проанализировал `sys`. Эквивалент `argc` – это просто количество элементов в списке. Чтобы получить это значение, используйте оператор `len()`.

5. Какие особенности построения CLI с использованием модуля getopt?

Модуль `getopt` в Python – это анализатор, используемый для параметров командной строки, которые основаны на соглашении, организованном функцией UNIX `getopt()`. Он в основном используется для анализа последовательности аргументов, например `sys.argv`. Мы также можем истолковать этот модуль как помощника сценариям анализировать аргументы командной строки в `sys.argv`. Он работает как функция `getopt()` языка программирования C для анализа параметров командной строки.

6. Какие особенности построение CLI с использованием модуля argparse?

argparse - это модуль для обработки аргументов командной строки.

Примеры того, что позволяет делать модуль:

- создавать аргументы и опции, с которыми может вызываться скрипт
- указывать типы аргументов, значения по умолчанию
- указывать, какие действия соответствуют аргументам
- выполнять вызов функции при указании аргумента
- отображать сообщения с подсказками по использованию скрипта