

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

**Кафедра инфокоммуникаций**

**Основы кросс платформенного программирования**

**Отчет по лабораторной работе №2.17**

**Разработка приложений с интерфейсом командной строки (CLI) в  
Python3**

Выполнил студент группы

ИТС-б-о-21-1 (2)

Гуляницкий Александр Евгеньевич

«    » \_\_\_\_\_ 20\_\_\_\_ г.

Подпись студента

\_\_\_\_\_  
Работа защищена

«    » \_\_\_\_\_ 20\_\_\_\_ г.

Проверил к.т.н., доцент

Кафедры инфокоммуникаций

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

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

Порядок выполнения работы:

- 1) Создадим общедоступный репозиторий на GitHub (<https://github.com/Alexander-its/laba-2.17>)
- 2) Решим задачи с помощью языка программирования Python3. И отправим их на GitHub.

Ход работы:

Пример1

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import argparse
4  import json
5  import os.path
6  import sys
7  from datetime import date
8
9
10 def add_worker(staff, name, post, year):
11     """
12     Добавить данные о работнике.
13     """
14     staff.append(
15         {
16             "name": name,
17             "post": post,
18             "year": year
19         }
20     )
21     return staff
22
```

Рис.1 примера 1

```

24  def display_workers(staff):
25      """
26      Отобразить список работников.
27      """
28      # Проверить, что список работников не пуст.
29      if staff:
30          # Заголовок таблицы.
31          line = '+-{}-+-{}-+-{}-+-{}-+'.format(
32              '-' * 4,
33              '-' * 30,
34              '-' * 20,
35              '-' * 8
36          )
37          print(line)
38          print(
39              '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
40                  "No",
41                  "Ф.И.О.",
42                  "Должность",
43                  "Год"
44              )
45          )
46          print(line)
47

```

Рис.2 примера 1

```

44      )
45      )
46      print(line)
47
48      # Вывести данные о всех сотрудниках.
49      for idx, worker in enumerate(staff, 1):
50          print(
51              '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
52                  idx,
53                  worker.get('name', ''),
54                  worker.get('post', ''),
55                  worker.get('year', 0)
56              )
57          )
58      print(line)
59      else:
60          print("Список работников пуст.")
61
62
63      def select_workers(staff, period):
64          """
65          Выбрать работников с заданным стажем.
66          """
67

```

Рис.3 примера 1

```

68     # Получить текущую дату.
69     today = date.today()
70
71     # Сформировать список работников.
72     result = []
73     for employee in staff:
74         if today.year - employee.get('year', today.year) >= period:
75             result.append(employee)
76
77     # Возвратить список выбранных работников.
78     return result
79
80
81 def save_workers(file_name, staff):
82     """
83     Сохранить всех работников в файл JSON.
84     """
85     # Открыть файл с заданным именем для записи.
86     with open(file_name, "w", encoding="utf-8") as fout:
87         # Выполнить сериализацию данных в формат JSON.
88         # Для поддержки кириллицы установим ensure_ascii=False
89         json.dump(staff, fout, ensure_ascii=False, indent=4)
90

```

Рис.4 примера 1

```

92  ✓ def load_workers(file_name):
93      """
94      Загрузить всех работников из файла JSON.
95      """
96      # Открыть файл с заданным именем для чтения.
97      ✓ with open(file_name, "r", encoding="utf-8") as fin:
98          | return json.load(fin)
99
100
101  ✓ def main(command_line=None):
102      # Создать родительский парсер для определения имени файла.
103      file_parser = argparse.ArgumentParser(add_help=False)
104      ✓ file_parser.add_argument(
105          | "filename",
106          | action="store",
107          | help="The data file name"
108          )
109
110      # Создать основной парсер командной строки.
111      parser = argparse.ArgumentParser("workers")
112      ✓ parser.add_argument(
113          | "--version",

```

Рис.5 примера 1

```

114         action="version",
115         version="% (prog)s 0.1.0"
116     )
117     subparsers = parser.add_subparsers(dest="command")
118
119     # Создать субпарсер для добавления работника.
120     add = subparsers.add_parser(
121         "add",
122         parents=[file_parser],
123         help="Add a new worker"
124     )
125     add.add_argument(
126         "-n",
127         "--name",
128         action="store",
129         required=True,
130         help="The worker's name"
131     )
132     add.add_argument(
133         "-p",
134         "--post",
135         action="store",
136         help="The worker's post"

```

Рис.6 примера 1

```

137     )
138     add.add_argument(
139         "-y",
140         "--year",
141         action="store",
142         type=int,
143         required=True,
144         help="The year of hiring"
145     )
146     # Создать субпарсер для отображения всех работников.
147     _ = subparsers.add_parser(
148         "display",
149         parents=[file_parser],
150         help="Display all workers"
151     )
152     # Создать субпарсер для выбора работников.
153     select = subparsers.add_parser(
154         "select",
155         parents=[file_parser],
156         help="Select the workers"
157     )
158     select.add_argument(
159         "-p",

```

Рис.7 примера 1

```

160         "--period",
161         action="store",
162         type=int,
163         required=True,
164         help="The required period"
165     )
166     # Выполнить разбор аргументов командной строки.
167     args = parser.parse_args(command_line)
168
169     # Загрузить всех работников из файла, если файл существует.
170     is_dirty = False
171     if os.path.exists(args.filename):
172         workers = load_workers(args.filename)
173     else:
174         workers = []
175
176     # Добавить работника.
177     if args.command == "add":
178         workers = add_worker(
179             workers,
180             args.name,
181             args.post,
182             args.year
183         )

```

Рис.8 примера 1

```

184         is_dirty = True
185
186     # Отобразить всех работников.
187     elif args.command == "display":
188         display_workers(workers)
189
190     # Выбрать требуемых работников.
191     elif args.command == "select":
192         selected = select_workers(workers, args.period)
193         display_workers(selected)
194
195     # Сохранить данные в файл, если список работников был изменен.
196     if is_dirty:
197         save_workers(args.filename, workers)
198
199
200 if __name__ == '__main__':
201     main()

```

Рис.9 примера 1

```

    return loads(fp.read(),
File "C:\Users\G0_PB\AppData\Local\Programs\Python\Python310\lib\json\__init__.py", line 346, in loads
    return _default_decoder.decode(s)
File "C:\Users\G0_PB\AppData\Local\Programs\Python\Python310\lib\json\decoder.py", line 337, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
File "C:\Users\G0_PB\AppData\Local\Programs\Python\Python310\lib\json\decoder.py", line 355, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

C:\Users\G0_PB>python primerlr7.py add data.json --name="Иванов Иван" --post="Стажер" --year=1965

C:\Users\G0_PB>python primerlr7.py add data.json --name="Иванов Сергей" --post="Студент" --year=2005

C:\Users\G0_PB>python primerlr7.py add data.json --name="Иван Грозный " --post="Старейшина" --year=1056

C:\Users\G0_PB>python primerlr7.py display data.json
+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Должность      |      Год      |
+-----+-----+-----+-----+
| 1 | Иванов Иван      | Стажер              | 1965          |
| 2 | Иванов Сергей    | Студент              | 2005          |
| 3 | Иван Грозный     | Старейшина           | 1056          |
+-----+-----+-----+-----+

C:\Users\G0_PB>python primerlr7.py select data.json --period=12
"python" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\G0_PB>python primerlr7.py select data.json --period=12
+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Должность      |      Год      |
+-----+-----+-----+-----+
| 1 | Иванов Иван      | Стажер              | 1965          |
+-----+-----+-----+-----+

C:\Users\G0_PB>

```

Рис 10.Результат работы

## Индивидуальная

```

+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Номер      |      Год      |
+-----+-----+-----+-----+
| 1 | Иванов Иван      | 2354252          | 1955          |
| 2 | Иванов Иван      | 2354252          | 1466          |
| 3 | Иванов Иван      | 2354252          | 2065          |
+-----+-----+-----+-----+

C:\Users\G0_PB>python individuallr7.py select data.json --period=12
+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Номер      |      Год      |
+-----+-----+-----+-----+
| 1 | Иванов Иван      | 2354252          | 1955          |
+-----+-----+-----+-----+

C:\Users\G0_PB>python individuallr7.py select data.json --period=66
+-----+-----+-----+-----+
| No |      Ф.И.О.      |      Номер      |      Год      |
+-----+-----+-----+-----+
| 1 | Иванов Иван      | 2354252          | 1955          |
+-----+-----+-----+-----+

C:\Users\G0_PB>

```

Рис.10 индивидуальная работа



### 1. В чем отличие терминала и консоли?

Консоль - это совокупность устройств, которые позволяют вам взаимодействовать с устройством. Раньше консолью мог быть обычный принтер (в принципе и сейчас может), тогда это будет текстовая консоль/терминал. Не надо путать с текстовым интерфейсом. Терминал - По сути это так же устройство для ввода и вывода информации но>. Консоль, это уже обёртка над терминалом.

### 2. Что такое консольное приложение?

Консольным приложением называется программа, которая не имеет графического интерфейса - окон, и которая работает в текстовом режиме в черно-белой консоли

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

Click — это Python-пакет для создания красивых интерфейсов командной строки с минимальным количеством кода и



возможностью компоновки. Это один из лучших Python- пакетов для создания CLI, и с ним очень удобно работать.

#### 4. Какие особенности построение CLI с использованием модуля sys ?

Модуль sys реализует аргументы командной строки в простой структуре списка с именем sys.argv На практике для правильной обработки входных данных требуется модуль sys . Для этого необходимо предварительно загрузить как модуль sys , так и модуль getopt . ... Генерация интерфейсов командной строки (CLI) с помощью Fire в Python. Модуль Shutil в Python.

#### 5. Какие особенности построение CLI с использованием модуля getopt ?

чтобы упростить написание кода, придерживающегося стандартных соглашений. Функция GNU getopt\_long(), является совместимой с getopt(), а также упрощает разбор длинных опций. getopt. Объявление: #include <unistd.h> int getopt(int argc, char \*argv[], const char \*optstring);

#### 6. Какие особенности построение CLI с использованием модуля argparse?

Использование модуля argparse в Python для создания интерфейса командной строки, обработки позиционных и необязательных аргументов, их комбинирование с подробными примерами Интерфейс командной строки в Python также известен как CLI,

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