МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет по лабораторной работе №2.17 по дисциплине:

основы программной инженерии

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Объединение обработки всех команд в одной программе, с помощью subparsers

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


import argparse


parser = argparse.ArgumentParser()
subparsers = parser.add_subparsers(help='List of commands')

# A list command
list_parser = subparsers.add_parser('list', help='List contents')
list_parser.add_argument('dirname', action='store', help='Directory to list')

# A create command
create_parser = subparsers.add_parser('create', help='Create a directory')
create_parser.add_argument('dirname', action='store', help='New directory to create')
create_parser.add_argument(
    '--read-only',
    default=False,
    action='store_true',
    help='Set permissions to prevent writing to the directory',
)
```

2. Простой разбор аргументов

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


import argparse


parser = argparse.ArgumentParser()
parser.add_argument("square", help="display a square of a given number")
args = parser.parse_args()
print(args.square ** 2)
```

```
Terminal:  Local  +
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py
usage: ex1.py [-h] square
ex1.py: error: the following arguments are required: square
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4
Traceback (most recent call last):
  File "D:\УЧЕБА\ОПИ\2.17\ex1.py", line 11, in <module>
    print(args.square ** 2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
PS D:\УЧЕБА\ОПИ\2.17>
```

3. Позиционные аргументы. Программа, высчитывающая квадрат аргумента.

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


import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    help="display a square of a given number",
    type=int
)
```

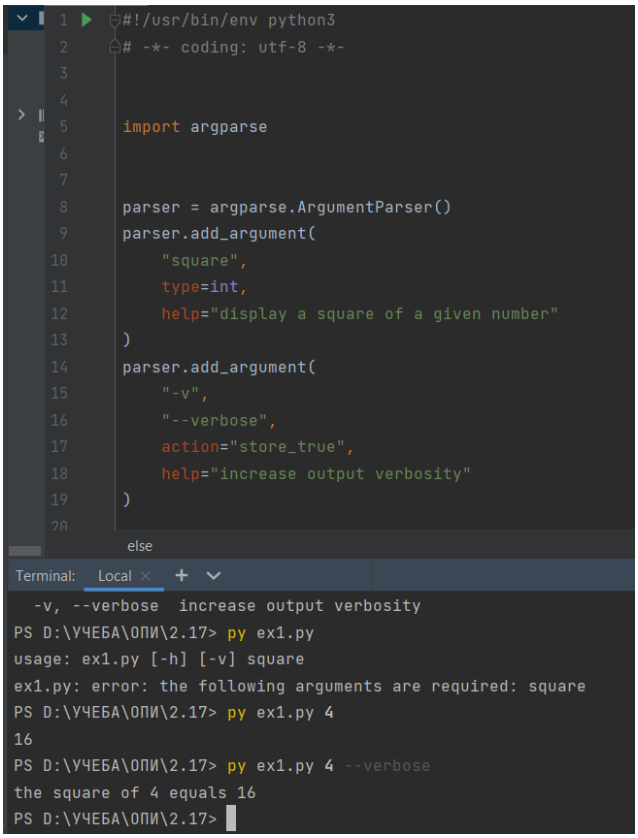
```
Terminal:  Local
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscor
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py
usage: ex1.py [-h] square
ex1.py: error: the following arguments are required: square
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4
Traceback (most recent call last):
  File "D:\УЧЕБА\ОПИ\2.17\ex1.py", line 11, in <module>
    print(args.square ** 2)
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4
16
```

4. Короткие имена –v и --verbosity

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


import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "-v",
    "--verbose",
    help="increase output verbosity",
    action="store_true"
)

args = parser.parse_args()
if args.verbose:
    print("verbosity turned on")
```

```
Terminal:  Local
16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py -v
verbosity turned on
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py --help
usage: ex1.py [-h] [-v]

options:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
```

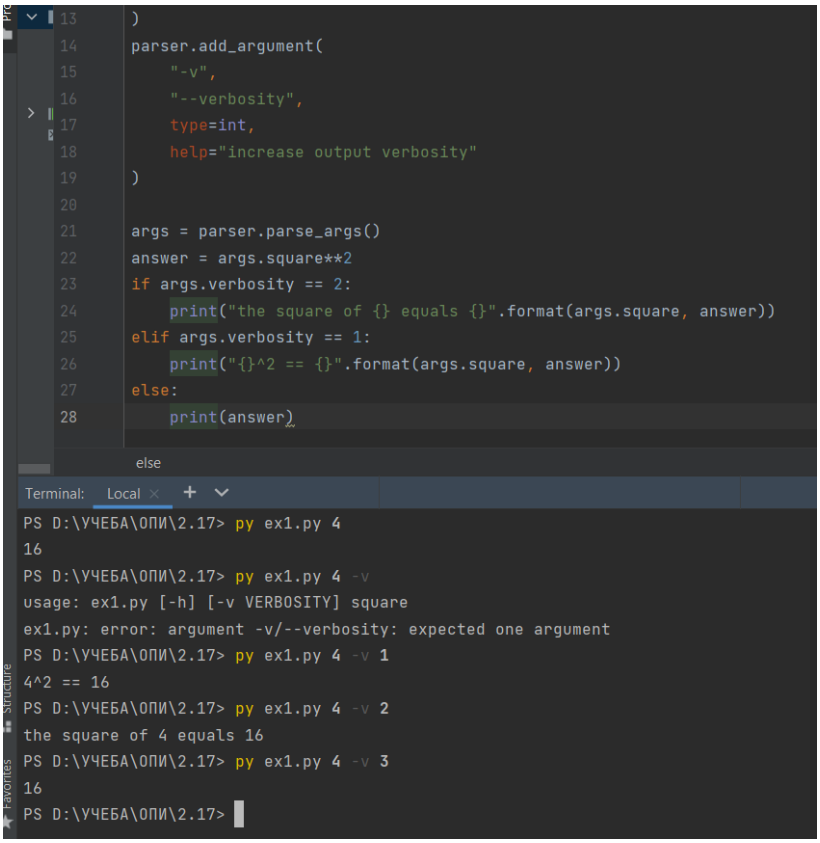
5. Применение одновременно позиционных и опциональных аргументов.



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


import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v",
    "--verbose",
    action="store_true",
    help="increase output verbosity"
)
```

```
else
Terminal:  Local
  -v, --verbose  increase output verbosity
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py
usage: ex1.py [-h] [-v] square
ex1.py: error: the following arguments are required: square
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4
16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 --verbose
the square of 4 equals 16
PS D:\УЧЕБА\ОПИ\2.17>
```

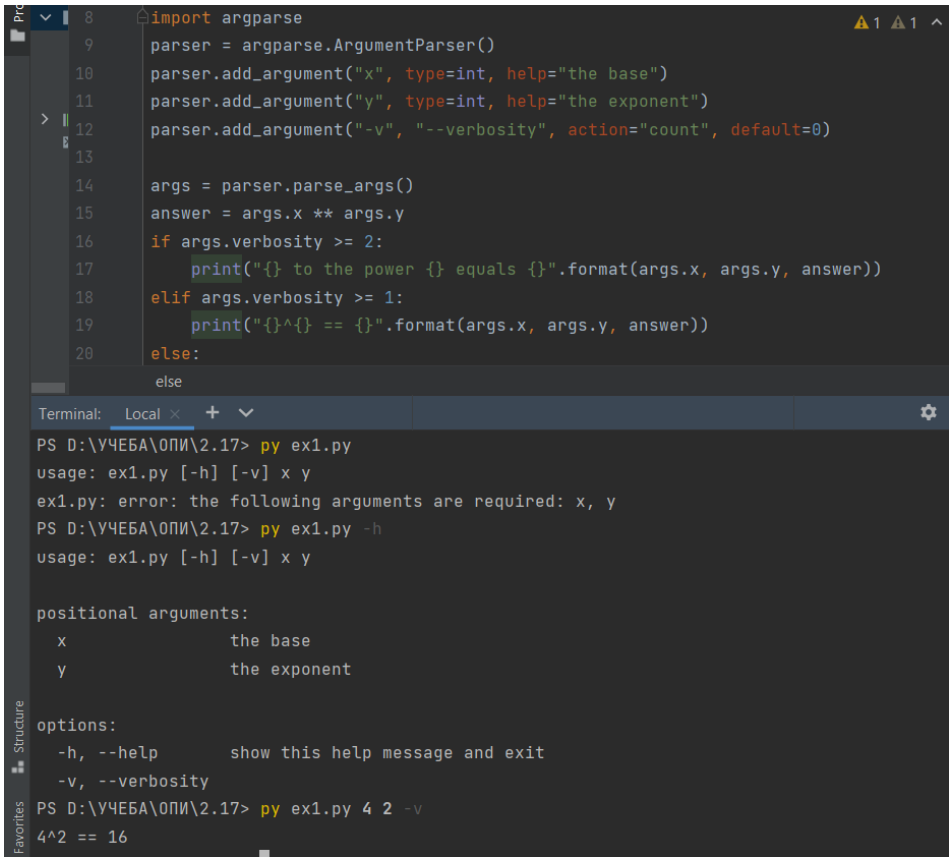
6. Опциональный аргумент с параметром и позиционный аргумент.



```
)
parser.add_argument(
    "-v",
    "--verbosity",
    type=int,
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square**2
if args.verbosity == 2:
    print("the square of {} equals {}".format(args.square, answer))
elif args.verbosity == 1:
    print("{}^2 == {}".format(args.square, answer))
else:
    print(answer)
```

```
else
Terminal:  Local
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4
16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 -v
usage: ex1.py [-h] [-v VERBOSITY] square
ex1.py: error: argument -v/--verbosity: expected one argument
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 -v 1
4^2 == 16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 -v 2
the square of 4 equals 16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 -v 3
16
PS D:\УЧЕБА\ОПИ\2.17>
```

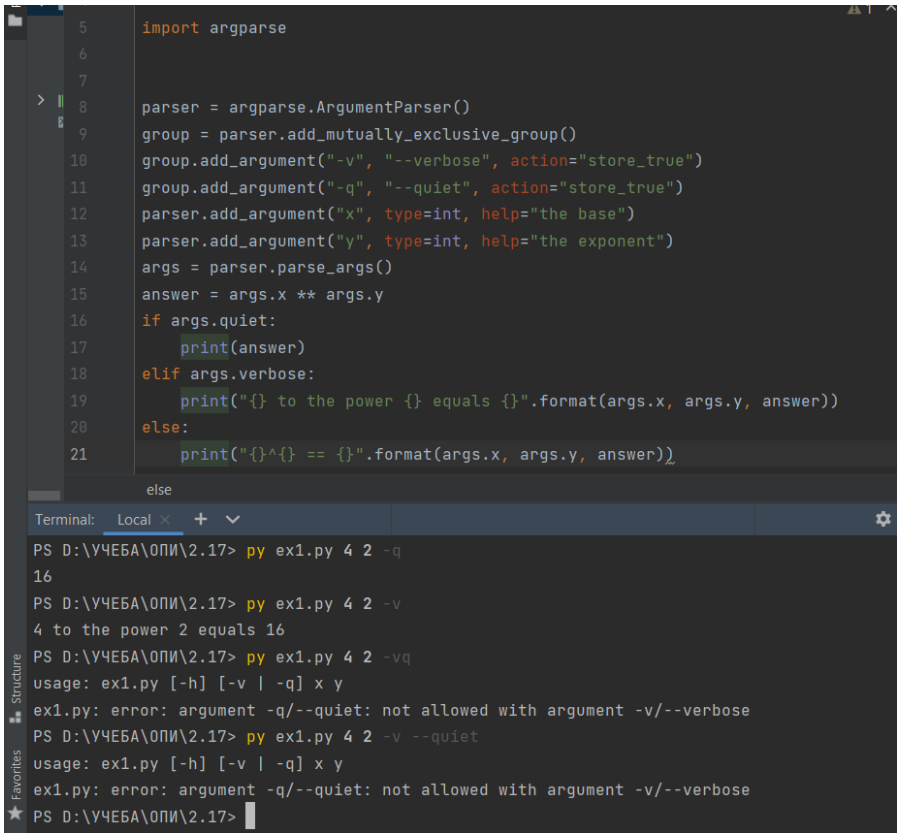
7. Выбор значения из заранее определенного списка.



```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("x", type=int, help="the base")
parser.add_argument("y", type=int, help="the exponent")
parser.add_argument("-v", "--verbosity", action="count", default=0)

args = parser.parse_args()
answer = args.x ** args.y
if args.verbosity >= 2:
    print("{} to the power {} equals {}".format(args.x, args.y, answer))
elif args.verbosity >= 1:
    print("{}^{} == {}".format(args.x, args.y, answer))
else:
```

```
else
Terminal: Local
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py
usage: ex1.py [-h] [-v] x y
ex1.py: error: the following arguments are required: x, y
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py -h
usage: ex1.py [-h] [-v] x y

positional arguments:
  x                the base
  y                the exponent

options:
  -h, --help       show this help message and exit
  -v, --verbosity
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -v
4^2 == 16
```

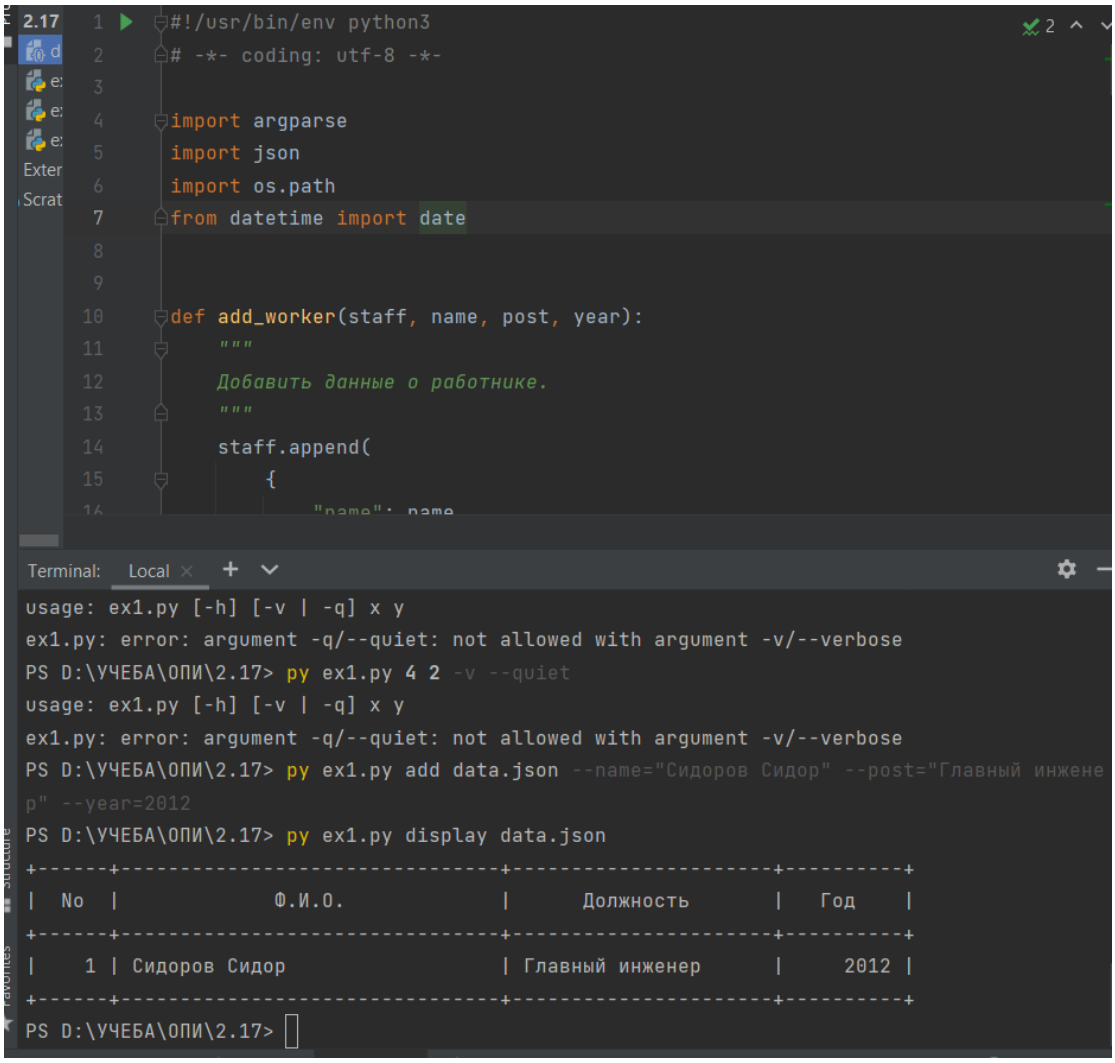
8. Опциональный аргумент с параметром.



```
import argparse


parser = argparse.ArgumentParser()
group = parser.add_mutually_exclusive_group()
group.add_argument("-v", "--verbose", action="store_true")
group.add_argument("-q", "--quiet", action="store_true")
parser.add_argument("x", type=int, help="the base")
parser.add_argument("y", type=int, help="the exponent")
args = parser.parse_args()
answer = args.x ** args.y
if args.quiet:
    print(answer)
elif args.verbose:
    print("{} to the power {} equals {}".format(args.x, args.y, answer))
else:
    print("{}^{} == {}".format(args.x, args.y, answer))
```

```
else
Terminal: Local
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -q
16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -v
4 to the power 2 equals 16
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -vq
usage: ex1.py [-h] [-v | -q] x y
ex1.py: error: argument -q/--quiet: not allowed with argument -v/--verbose
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -v --quiet
usage: ex1.py [-h] [-v | -q] x y
ex1.py: error: argument -q/--quiet: not allowed with argument -v/--verbose
PS D:\УЧЕБА\ОПИ\2.17>
```

9. Выполнение примера 1.



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

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


def add_worker(staff, name, post, year):
    """
    Добавить данные о работнике.
    """
    staff.append(
        {
            "name": name
```

```
Terminal:  Local
usage: ex1.py [-h] [-v | -q] x y
ex1.py: error: argument -q/--quiet: not allowed with argument -v/--verbose
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py 4 2 -v --quiet
usage: ex1.py [-h] [-v | -q] x y
ex1.py: error: argument -q/--quiet: not allowed with argument -v/--verbose
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py add data.json --name="Сидоров Сидор" --post="Главный инжене
p" --year=2012
PS D:\УЧЕБА\ОПИ\2.17> py ex1.py display data.json
+------+-------------------------------+----------------------+----------+
|  No  |             Ф.И.О.            |      Должность       |   Год    |
+------+-------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                 | Главный инженер      |     2012 |
+------+-------------------------------+----------------------+----------+
PS D:\УЧЕБА\ОПИ\2.17>
```

10.Выполнение индивидуального задания.

```
PS D:\УЧЕБА\ОПИ\2.17> python ex2.py add data1.json --surname="Sid" --name="S" --number="123445" --date_obj="10.10.2010"
PS D:\УЧЕБА\ОПИ\2.17> python ex2.py add data1.json --surname="Lazareva" --name="Darya" --number="898989" --date_obj="10.10.2010"
PS D:\УЧЕБА\ОПИ\2.17> python ex2.py add data1.json --surname="Petrovaa" --name="Anya" --number="121212" --date_obj="02.03.2004"
PS D:\УЧЕБА\ОПИ\2.17> python ex2.py add data1.json --surname="Ivanov" --name="Sergey" --number="454545" --date_obj="25.12.2000"
```

data1.json – Блокнот

Файл Правка Формат Вид Справка

```
[
    {
        "surname": "Lazareva",
        "name": "Darya",
        "number": "898989",
        "date_obj": "10.10.2010"
    },
    {
        "surname": "Petrovaa",
        "name": "Anya",
        "number": "121212",
        "date_obj": "02.03.2004"
    },
    {
        "surname": "Ivanov",
        "name": "Sergey",
        "number": "454545",
        "date_obj": "25.12.2000"
    }
]
```

```
PS D:\УЧЕБА\ОПИ\2.17> python ex2.py display data1.json
+------+----------------------------+----------------------+----------------+----------------------+
|  №   |          Фамилия           |         Имя          | Номер телефона |    Дата рождения     |
+------+----------------------------+----------------------+----------------+----------------------+
|  1   |          Lazareva          |        Darya         |     898989     |      10.10.2010      |
|  2   |          Petrovaa          |         Anya         |     121212     |      02.03.2004      |
|  3   |           Ivanov           |        Sergey        |     454545     |      25.12.2000      |
+------+----------------------------+----------------------+----------------+----------------------+
```

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.

Код решения задачи:

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

import click
import json
import os.path


def add_worker(staff, surname, name, number, date_obj):
    staff.append(
        {
            'surname': surname,
```

```
            'name': name,
            'number': number,
            'date_obj': date_obj,
        }
    )

    return staff


def display_workers(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15,
            '-' * 20
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} | {:^20} |'.format(
                "№",
                "Фамилия",
                "Имя",
                "Номер телефона",
                "Дата рождения"
            )
        )
        print(line)

        for idx, worker in enumerate(staff, 1):
            print(
                '| {:^4} | {:^30} | {:^20} | {:^15} | {:^20} |'.format(
                    idx,
                    worker.get('surname', ''),
                    worker.get('name', ''),
                    worker.get('number', ''),
                    str(worker.get('date_obj', '')),
                )
            )
        print(line)

    else:
        print("Список пуст.")


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


def load_workers(file_name):
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


@click.command()
@click.argument('command')
@click.argument('filename')
@click.option('--surname', help='The surname')
@click.option('--name', help='The name')
@click.option('--number', help='The number')
@click.option('--date.obj',  help='The date')
def main(command, filename, surname, name, number, date_obj):
```

```
    is_dirty = False
    if os.path.exists(filename):
        workers = load_workers(filename)
    else:
        workers = []

    if command == "add":
        workers = add_worker(
            surname,
            name,
            number,
            date_obj
        )
        is_dirty = True
    elif command == "display":
        display_workers(workers)
    if is_dirty:
        save_workers(filename, workers)


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

```
PS D:\УЧЕБА\ОПИ\2.17> python ind2.py add data.json --surname="Lazareva" --name="Nadya" --number="123456" --date_obj="10.12.2004"
```

Ответы на вопросы:

1. В чем отличие терминала и консоли?

Терминал – устройство или ПО, выступающее посредником между человеком и вычислительной системой.

Консоль – исторически реализация терминала с клавиатурой и текстовым дисплеем.

Терминал, это уже надстройка над консолью и под собой больше подразумевает удалённый доступ с мало мощной машины.

2. Что такое консольное приложение?

Консольное приложение – 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.

Второй способ – 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. Другой метод – использование модуля docopt, доступного на GitHub.

4. Какие особенности построение 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, являются аргументами командной строки. В качестве разделителя между аргументами используется пробел. Значения аргументов, содержащие пробел, должны быть заключены в кавычки, чтобы их правильно проанализировал sys.

5. Какие особенности построение CLI с использованием модуля getopt?

Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений.

6. Какие особенности построение CLI с использованием модуля argparse?

Рассмотрим, что интересного предлагает argparse:

- Анализ аргументов sys.argv;
- Конвертирование строковых аргументов в объекты Вашей программы и работа с ними;
- Форматирование и вывод информативных подсказок.