

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 «Основы программной инженерии»

Выполнил:
Мелтонян Одиссей
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:
Кандидат технических наук, доцент
кафедры инфокоммуникаций
Воронкин Р. А

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Тема: Работа с файлами в языке Python

Цель работы: приобретение навыков по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучение основных методов модуля os для работы с файловой системой, получение аргументов командной строки

Ход работы:

1. Изучен теоретический материал работы.
2. Создан общедоступный репозиторий на GitHub, в котором была использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *

odik8 / BSE18

✓ BSE18 is available.

Great repository names are short and memorable. Need inspiration? How about **redesigned-octo-garbanzo** ?

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.](#)

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

Create repository

Рисунок 1 – Создание репозитория

3. Выполнено клонирование созданного репозитория.

```
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР3
$ git clone https://github.com/odik8/BSE18.git
Cloning into 'BSE18'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР3
$ |
```

Рисунок 2 – Клонирование репозитория

4. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организован репозиторий в соответствии с моделью ветвления git-flow.

```
varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР3
$ git flow init
Initialized empty Git repository in C:/Users/varfe/Рабочий стол/4
 семестр/ОПИ/ЛР3/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/varfe/Рабочий стол/4 семес
тр/ОПИ/ЛР3/.git/hooks]

varfe@DESKTOP-E108KEN MINGW64 ~/Рабочий стол/4 семестр/ОПИ/ЛР3 (d
evelop)
$ |
```

Рисунок 3 – Инициализация git-flow

6. Создан проект PyCharm в папке репозитория.

7. Выполнены индивидуальные задания.

Вариант 14

Задание 1. Написать программу, которая считывает текст из файла, находит самое длинное слово и определяет, сколько раз оно встретилось в тексте.

```
4
5 > if __name__ == "__main__":
6     with open("file.txt") as f:
7         words = f.read().split()
8         max_word = str(max(words, key=len))
9         count_of_max_word = words.count(max_word)
10        print(f'Max word: {max_word}, Count of max word: {count_of_max_word}')
11
```

Рисунок 4 – Код решения

```
individual_1.py  file.txt  modified_text.txt
1 Python is the modern day language. It makes things so simple.
2 It is the fastest growing programming language Python has easy syntax and user-friendly interaction.
```

Рисунок 5 – Содержимое тестового файла

```
C:\Users\varfe\AppData\Local\Microsoft\WindowsA
Max word: user-friendly, Count of max word: 1

Process finished with exit code 0
```

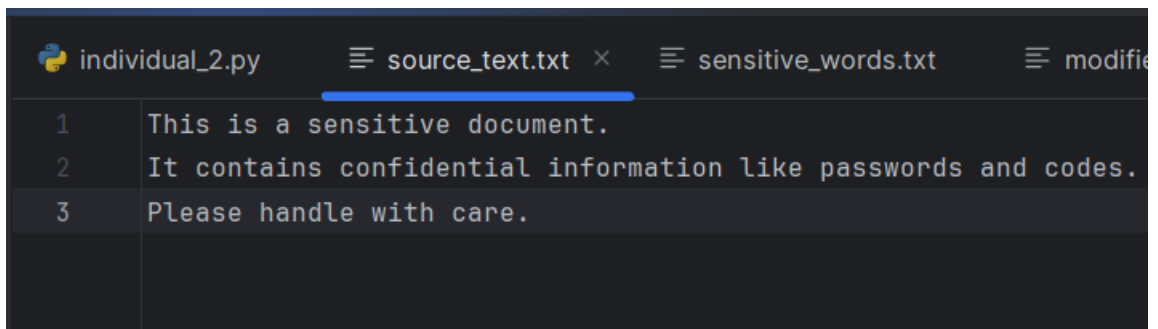
Рисунок 6 – Результат выполнения кода

Задание 2. Перед публикацией текста или документа обычно принято удалять или изменять в них служебную информацию. В данном упражнении вам необходимо написать программу, которая будет заменять все служебные слова в тексте на символы звездочек (по количеству символов в словах). Вы должны осуществлять регистрозависимый поиск служебных слов в тексте, даже если эти слова входят в состав других слов. Список служебных слов должен храниться в отдельном файле. Сохраните отредактированную версию исходного файла в новом файле. Имена исходного файла, файла со служебными словами и нового файла должны быть введены пользователем. В качестве дополнительного задания расширьте свою программу таким образом, чтобы она выполняла замену служебных слов вне зависимости от того, какой регистр символов используется в тексте. Например, если в списке служебных

слов будет присутствовать слово exam, то все следующие варианты слов должны быть заменены звездочками: exam, Exam, ExaM и EXAM.

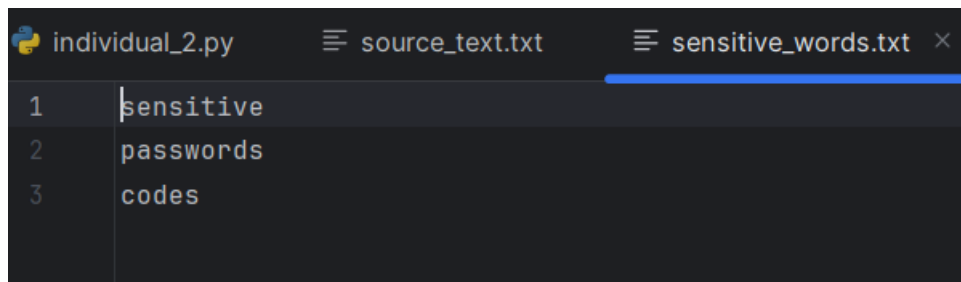
```
1 usage  👤 odyssey
16 def replace_sensitive_words(text: str, sensitive_words: list):
17     modified_text = text
18     for word in sensitive_words:
19         modified_text = modified_text.replace(word.lower(), '*' * len(word))
20     return modified_text
21
22
23 usage  👤 odyssey
24 def main():
25     input_file_name = input("Enter filename: ")
26     sensitive_words_file = input("Enter file name with sensitive_words: ")
27     output_file_name = input("Enter the name for a new file: ")
28
29     with open(input_file_name, 'r') as file:
30         text = file.read()
31
32     with open(sensitive_words_file) as file:
33         sensitive_words_file = file.read().splitlines()
34
35     modified_text = replace_sensitive_words(text, sensitive_words_file)
36
37     with open(output_file_name, 'w') as file:
38         file.write(modified_text)
39
40 ▶ if __name__ == "__main__":
41     💡 main()
42
```

Рисунок 7 – Код решения



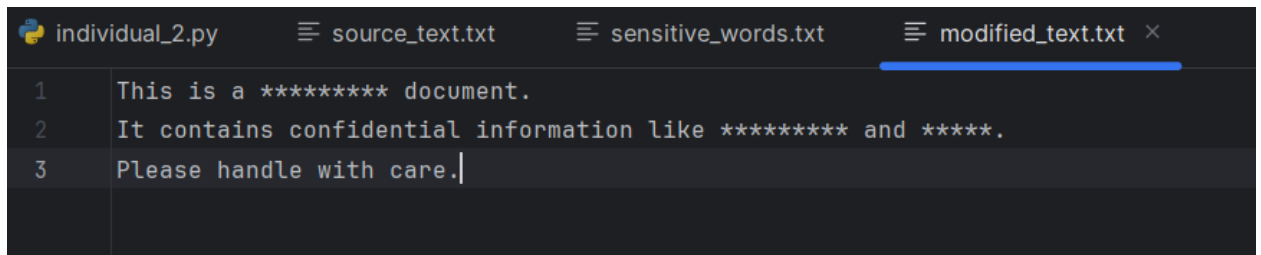
```
individual_2.py  source_text.txt  sensitive_words.txt  modified_...
1 This is a sensitive document.
2 It contains confidential information like passwords and codes.
3 Please handle with care.
```

Рисунок 8 – Содержимое исходного файла

A screenshot of a code editor with three tabs: 'individual_2.py', 'source_text.txt', and 'sensitive_words.txt'. The 'sensitive_words.txt' tab is active and shows a list of words: 'sensitive', 'passwords', and 'codes' on lines 1, 2, and 3 respectively.

```
1 sensitive
2 passwords
3 codes
```

Рисунок 9 – Содержимое файла с служебными словами

A screenshot of a code editor with four tabs: 'individual_2.py', 'source_text.txt', 'sensitive_words.txt', and 'modified_text.txt'. The 'modified_text.txt' tab is active and shows the result of code execution: 'This is a ***** document.', 'It contains confidential information like ***** and *****.', and 'Please handle with care.' on lines 1, 2, and 3 respectively.

```
1 This is a ***** document.
2 It contains confidential information like ***** and *****.
3 Please handle with care.
```

Рисунок 10 – Результат выполнения кода

8. Зафиксированы сделанные изменения в репозитории.
9. Добавлен отчет по лабораторной работе в формате PDF в папку doc репозитория.
10. Выполнено слияние ветки для разработки с веткой master/main.
11. Отправлены сделанные изменения на сервер GitHub.
12. Отправлен адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы:

1. Как открыть файл в языке Python только для чтения?

Для открытия файла только для чтения в Python используется функция **open()** с параметром **'r'**. Например:

```
file = open('file.txt', 'r')
```

Или:

```
with open('file.txt', 'r') as file:
```

...

2. Как открыть файл в языке Python только для записи?

```
file = open('file.txt', 'w')
```

Или:

```
with open('file.txt', 'w') as file:
```

3. Как прочитать данные из файла в языке Python?

Для чтения данных из файла в Python используются методы объекта файла, такие как **read()** – читает данные из файла в виде одной большой строки или в виде заданного количества символов, **readline()** – читает одну строку из файла. При каждом вызове он считывает следующую строку. Если файл достигает конца, возвращается пустая строка, **readlines()** – читает все строки из файла и возвращает их в виде списка строк. Каждый элемент списка соответствует строке в файле. Например:

4. Как записать данные в файл в языке Python?

```
file.write("Hello, world!")
```

5. Как закрыть файл в языке Python?

```
file.close()
```

6. Изучите самостоятельно работу конструкции `with ... as`. Каково ее назначение в языке Python? Где она может быть использована еще, помимо работы с файлами?

Преимущество использования оператора `with` заключается в том, что он обеспечивает гарантию закрытия файла независимо от того, как закрывается вложенный блок. Всегда рекомендуется использовать оператор `with` для файлов. Если во вложенном блоке кода возникает прерывание, возврат или исключение, тогда он автоматически закрывает файл, и нам не нужно писать функцию `close()`. Это не позволяет файлу исказиться.

7. Изучите самостоятельно документацию Python по работе с файлами. Какие помимо рассмотренных существуют методы записи/чтения информации из файла?

Метод `writelines()` (запись списка строк в файл):

```
lines = ["Строка 1\n", "Строка 2\n", "Строка 3\n"]
```

```
with open('file.txt', 'w') as f:
```

```
    f.writelines(lines)
```

Метод seek() (установка указателя чтения/записи в файле):

with open('file.txt', 'w') as f:

 f.write("Hello, World!")

 f.seek(0) # Установить указатель в начало файла

 f.write("Привет, мир!")

8. Какие существуют, помимо рассмотренных, функции модуля os для работы с файловой системой?

1. **os.listdir(path='.')** - возвращает список файлов и каталогов в указанном каталоге. По умолчанию возвращает содержимое текущего каталога.
2. **os.makedirs(path)** - создает все каталоги в пути, если они не существуют.
3. **os.removedirs(path)** - удаляет каталог и все его подкаталоги, если они существуют.
4. **os.path.exists(path)** - проверяет существование файла или каталога по указанному пути.
5. **os.path.isfile(path)** - возвращает True, если по указанному пути существует файл.
6. **os.path.isdir(path)** - возвращает True, если по указанному пути существует каталог.