МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Установка пакетов в Python. Виртуальные окружения»

Отчет по лабораторной работе № 2.15

по дисциплине «Основы программной инженерии»

Выполнил студент группы

ПИЖ-б-о-21-1

Трушева В. О. .« » 2023г.

Подпись студента_____

Работа защищена «_____20__г.

Проверила Воронкин Р.А. _____

(подпись)

Ставрополь 2023

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.

2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

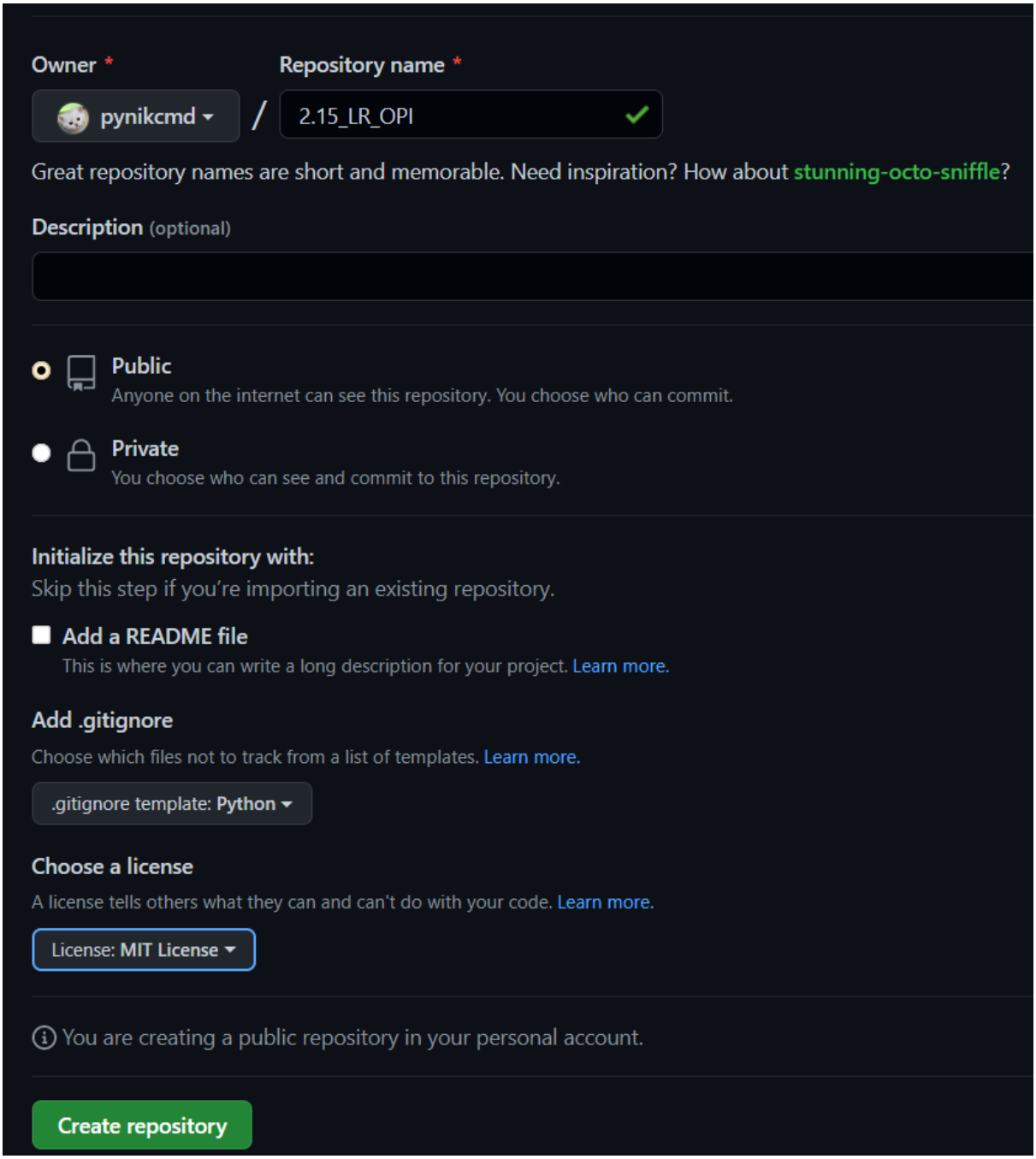


Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
D:\fgit>git clone https://github.com/pynikcmd/2.15_LR_OPI.git
Cloning into '2.15_LR_OPI'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 4.76 KiB | 2.38 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 – Клонирование репозитория

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

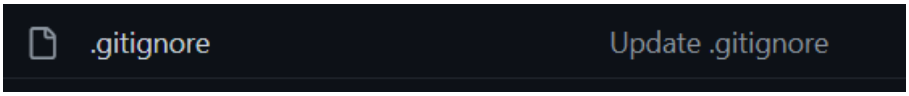


Рисунок 3 – Дополнен .gitignore

5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
D:\fgit\2.15_LR_OPI>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
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? [D:/fgit/2.15_LR_OPI/.git/hooks]
```

Рисунок 4 – Модель git-flow

6. Создайте проект PyCharm в папке репозитория.

7. Проработать примеры лабораторной работы.

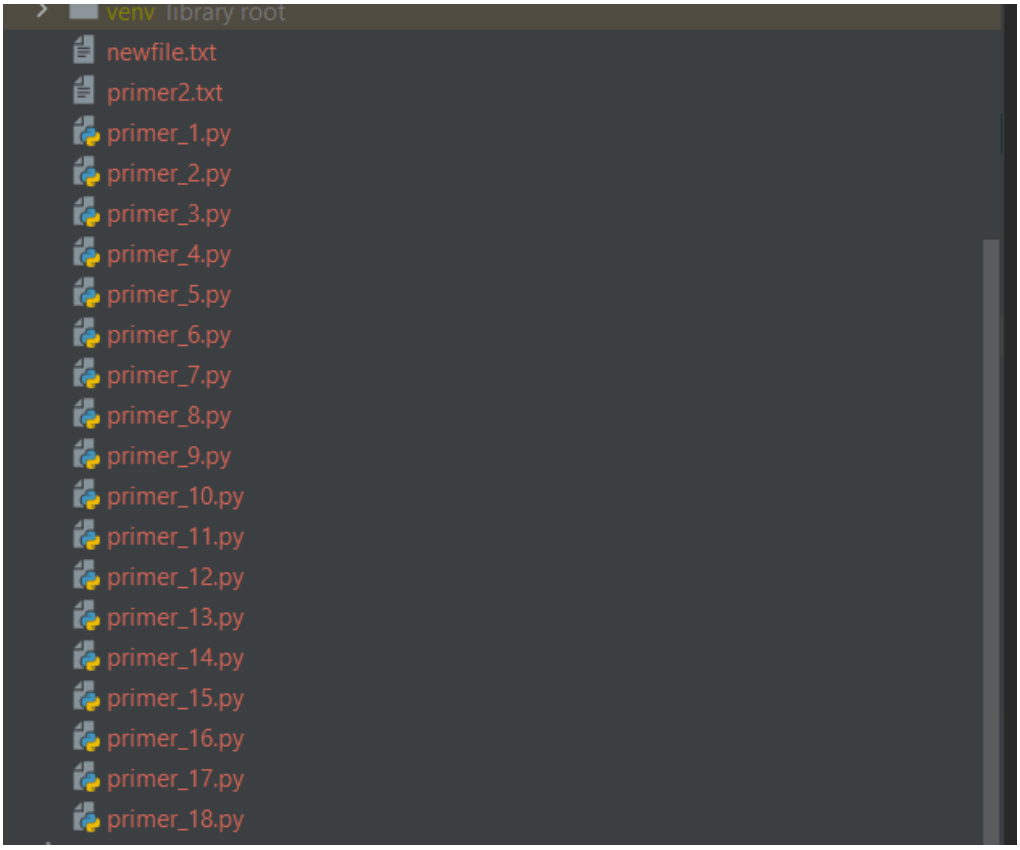


Рисунок 5 – Выполнены примеры лабораторной работы

8. Выполнить индивидуальные задания.

Вариант – 8. Условие.

Написать программу, которая считывает текст из файла и выводит на экран только цитаты, то есть предложения, заключенные в кавычки.

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

# 8. Написать программу, которая считывает текст из файла и выводит на экран только цитаты,
# то есть предложения, заключенные в кавычки.

if __name__ == '__main__':
    with open('1_ind.txt', 'r', encoding='utf-8') as f:
        text = f.read()
    quote = []
    for i in range(0, len(text)):
        if text[i] == '"':
            quote.append(i)
    i = 0
    while i < len(quote):
        print(text[quote[i]:quote[i+1]+1])
        i += 2

```

Рисунок 6 – Код программы 1 индивидуального задания

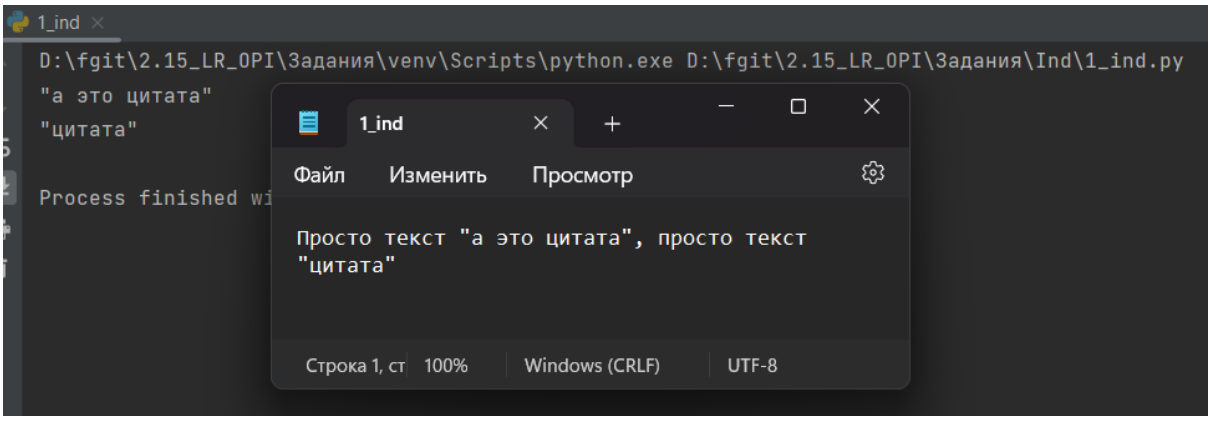


Рисунок 7 – Результат работы программы

Вариант – 12. Условие.

Автоматическая проверка орфографии не помешала бы многим из нас. В данном упражнении мы напишем простую программу, сверяющую слова из текстового файла со словарем. Неправильно написанными будем считать все слова, которых не нашлось в словаре. Имя файла, в котором требуется выполнить орфографическую проверку, пользователь должен передать при помощи аргумента командной строки. В случае отсутствия аргумента должна выдаваться соответствующая ошибка. Сообщение об ошибке также должно появляться, если не удается открыть указанный пользователем файл. Также Вам следует игнорировать регистр символов при выполнении проверки.

```python
import sys

if __name__ == "__main__":

    # Создаем словарь правильно написанных слов
    dictionary = ["привет", "как", "дела", "котик", "учеба", "зарплата", "пенсия", "улица"]

    try:
        # Проверка наличия аргументов
        if len(sys.argv) != 2:
            print("В качестве аргумента командной строки необходимо передать имя файла", file=sys.stderr)
            sys.exit(1)

        filename = (sys.argv[1])

        with open(filename, encoding='utf-8', mode='r') as file:
            content = file.read().split(" ")

        # Проверка корректности написания слов в тексте и вывод их
        for i, word in enumerate(content):
            if content[i].lower() not in dictionary:
                print(f"Неправильное написанное слово: {word}")

    except IOError:
        # При возникновении проблемы с чтением файла, отобразится ошибка
        print("Ошибка при доступе к файлу", file=sys.stderr)
```

Рисунок 8 – Код программы 2 индивидуального задания

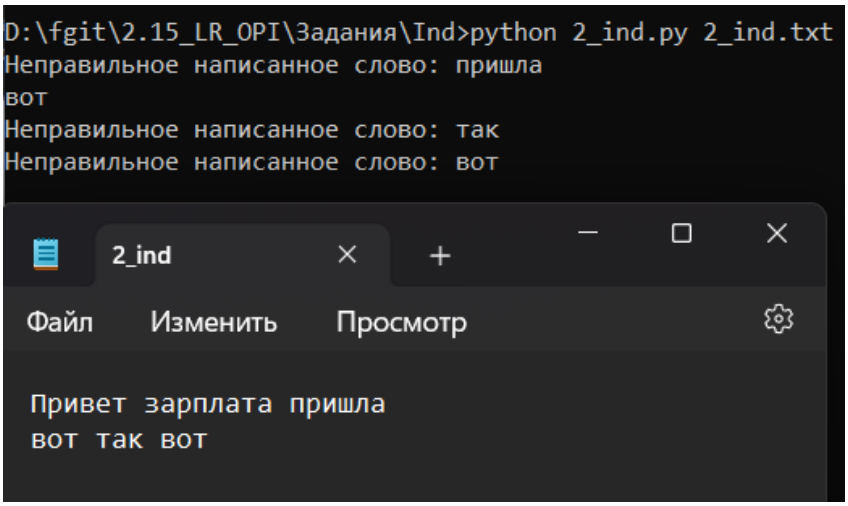
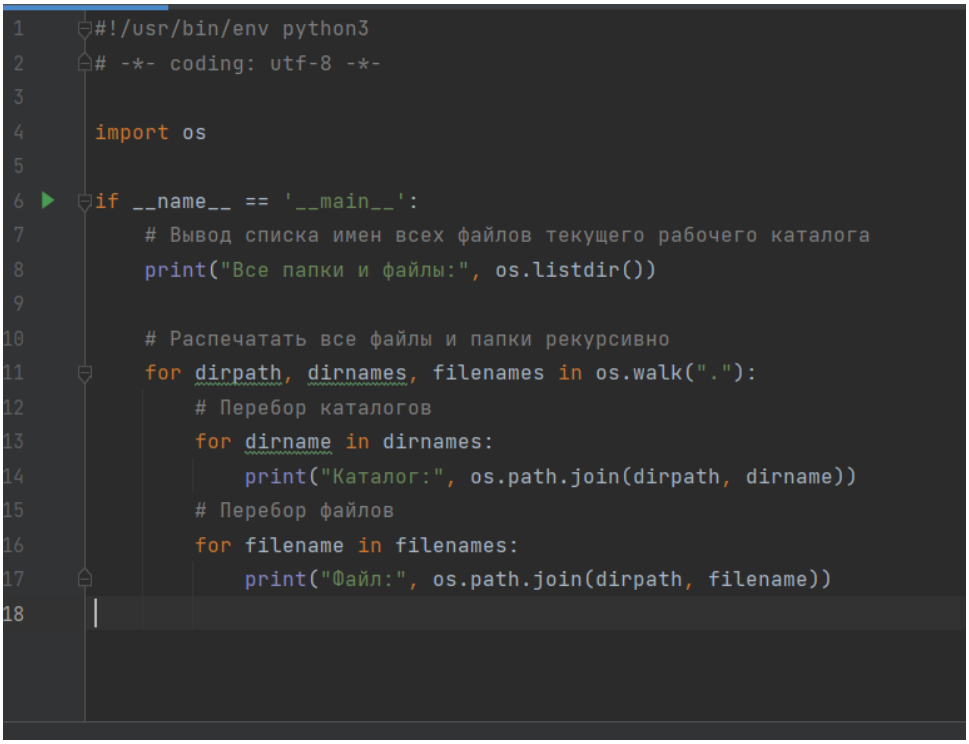


Рисунок 9 – Результат работы программы

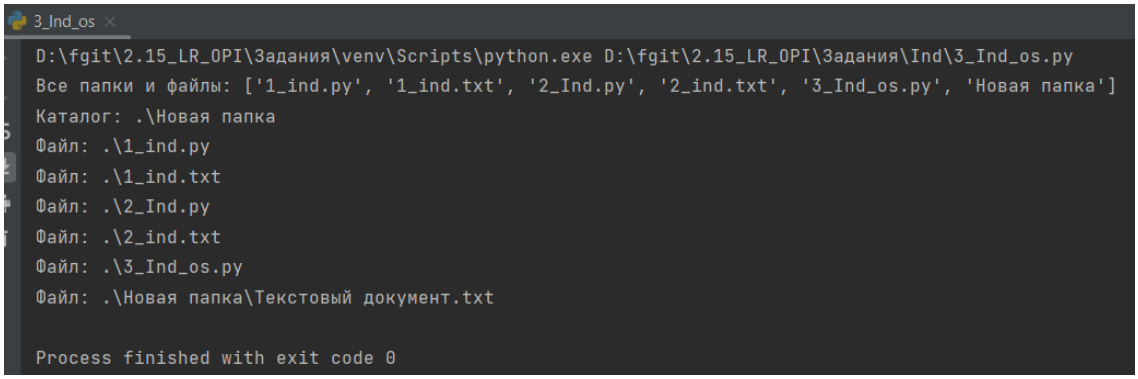
9. Зафиксируйте изменения в репозитории.

10. Самостоятельно подберите или придумайте задачу для работы с изученными функциями модуля os . Приведите решение этой задачи.



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

import os

if __name__ == '__main__':
    # Вывод списка имен всех файлов текущего рабочего каталога
    print("Все папки и файлы:", os.listdir())

    # Распечатать все файлы и папки рекурсивно
    for dirpath, dirnames, filenames in os.walk("."):
        # Перебор каталогов
        for dirname in dirnames:
            print("Каталог:", os.path.join(dirpath, dirname))
        # Перебор файлов
        for filename in filenames:
            print("Файл:", os.path.join(dirpath, filename))

```

Рисунок 10 – Код программы



```
3_Ind_os
D:\fgit\2.15_LR_OPI\Задания\venv\Scripts\python.exe D:\fgit\2.15_LR_OPI\Задания\Ind\3_Ind_os.py
Все папки и файлы: ['1_ind.py', '1_ind.txt', '2_Ind.py', '2_ind.txt', '3_Ind_os.py', 'Новая папка']
Каталог: .\Новая папка
Файл: .\1_ind.py
Файл: .\1_ind.txt
Файл: .\2_Ind.py
Файл: .\2_ind.txt
Файл: .\3_Ind_os.py
Файл: .\Новая папка\Текстовый документ.txt

Process finished with exit code 0
```

Рисунок 11 – Результат работы программы

11. Зафиксируйте изменения в репозитории.

12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

13. Выполните слияние ветки для разработки с веткой master/main.

14. Отправьте сделанные изменения на сервер GitHub.

15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Вопросы для защиты работы

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

file object = open(<file-name>, <access-mode>, <buffering>)

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

- `w` – только для записи. Он перезаписывает файл, если он существовал ранее, или создает новый, если файл с таким именем не существует. Указатель имеется в начале файла.

3. Как прочитать данные из файла в языке Python?

```
with open("file.txt",'r') as f:
    content = f.read();
    print(content)
```

Построчное чтение содержимого файла в цикле:

with open("file2.txt", "r") as fileptr:

for i in fileptr:

print(i)

Где i – одна строка файла.

Построчное чтение содержимого файла с помощью методов файлового объекта:

with open("file2.txt", "r") as fileptr:

content1 = fileptr.readline()

content2 = fileptr.readline()

print(content1)

print(content2)

Мы вызывали функцию readline() два раза, поэтому она считывает две строки из файла.

Чтение строк с помощью функции readlines():

```
with open("file2.txt", "r") as fileptr:
content = fileptr.readlines()
print(content)
```

readlines() считывает строки в файле до его конца (EOF)

4. Как записать данные в файл в языке Python?

Чтобы записать текст в файл, нам нужно открыть файл с помощью метода open с одним из следующих режимов доступа:

'w': он перезапишет файл, если какой-либо файл существует. Указатель файла находится в начале файла.

'a': добавит существующий файл. Указатель файла находится в конце файла. Он создает новый файл, если файл не существует.

Пример:

```
with open("file2.txt", "w") as fileptr:
fileptr.write(
"Python is the modern day language. It makes things so simple.\n"
"It is the fastest-growing programing language"
)
```

5. Как закрыть файл в языке Python?

После того, как все операции будут выполнены с файлом, мы должны закрыть его с помощью нашего скрипта Python, используя метод close().

Любая незаписанная информация уничтожается после вызова метода close()для файлового объекта.

```
fileobject.close()
```

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.

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

Данная конструкция является менеджером контекста. Помимо файлов может использоваться в работе с базами данных:

```
def get_all_songs():
with sqlite3.connect('db/songs.db') as connection:
cursor = connection.cursor()
cursor.execute("SELECT * FROM songs ORDER BY id desc")
all_songs = cursor.fetchall()
return all_songs
```

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

Есть возможность записать в файл большой объем данных, если он может быть представлен в виде списка строк.

```
with open("examp.le", "w") as f:
f.writelines(list_of_strings)
```

Существует еще один, менее известный, способ, но, возможно, самый удобный из представленных. И как бы не было странно, он заключается в использовании функции print(). Сначала это утверждение может показаться странным, потому что общеизвестно, что с помощью нее

происходит вывод вконсоль. И это правда. Но если передать в необязательный аргумент file объект типа io.TextIOWrapper, каким и является объект файла, с которым мы работаем, то поток вывода функции print() перенаправляется из консоли в файл.

with open("examp.le", "w") as f:

print(some_data, file=f)

С помощью file.seek() можно перемещать указатель в файле на определённое количество байтов.

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

os.name - имя операционной системы. Доступные варианты: 'posix', 'nt', 'mac', 'os2', 'ce', 'java'.

os.environ - словарь переменных окружения. Изменяемый (можно добавлять и удалять переменные окружения).

os.getlogin() - имя пользователя, вошедшего в терминал (Unix).

os.getpid() - текущий id процесса.

os.uname() - информация об ОС. возвращает объект с атрибутами: sysname - имя операционной системы, nodename - имя машины в сети (определяется реализацией), release - релиз, version - версия, machine - идентификатор машины.

os.access(path, mode, *, dir_fd=None, effective_ids=False, follow_symlinks=True) - проверка доступа к объекту у текущего

пользователя. Флаги: os.F_OK - объект существует, os.R_OK - доступен на чтение, os.W_OK - доступен на запись, os.X_OK - доступен на исполнение.

os.chdir(path) - смена текущей директории.

os.chmod(path, mode, *, dir_fd=None, follow_symlinks=True) - смена прав доступа к объекту (mode - восьмеричное число).

os.chown(path, uid, gid, *, dir_fd=None, follow_symlinks=True) - меняет id владельца и группы (Unix).

os.getcwd() - текущая рабочая директория.

os.link(src, dst, *, src_dir_fd=None, dst_dir_fd=None, follow_symlinks=True) - создаёт жёсткую ссылку.

os.listdir(path=".") - список файлов и директорий в папке.

os.mkdir(path, mode=0o777, *, dir_fd=None) - создаёт директорию. OSError, если директория существует.

os.makedirs(path, mode=0o777, exist_ok=False) - создаёт директорию, создавая при этом промежуточные директории.

os.remove(path, *, dir_fd=None) - удаляет путь к файлу.

os.rename(src, dst, *, src_dir_fd=None, dst_dir_fd=None) - переименовывает файл или директорию из src в dst.

os.renames(old, new) - переименовывает old в new, создавая промежуточные директории.

os.replace(src, dst, *, src_dir_fd=None, dst_dir_fd=None) - переименовывает из src в dst с принудительной заменой.

os.rmdir(path, *, dir_fd=None) - удаляет пустую директорию.

os.removedirs(path) - удаляет директорию, затем пытается удалить родительские директории, и удаляет их рекурсивно, пока они пусты.

os.symlink(source, link_name, target_is_directory=False, *, dir_fd=None) - создаёт символическую ссылку на объект.

os.sync() - записывает все данные на диск (Unix).

os.truncate(path, length) - обрезает файл до длины length.

os.utime(path, times=None, *, ns=None, dir_fd=None, follow_symlinks=True) - модификация времени последнего доступа и изменения файла. Либо times - кортеж (время доступа в секундах, время изменения в секундах), либо ns - кортеж (время доступа в наносекундах, время изменения в наносекундах).

os.walk(top, topdown=True, onerror=None, followlinks=False) - генерация имён файлов в дереве каталогов, сверху вниз (если topdown равен True), либо снизу вверх (если False). Для каждого каталога функция walk возвращает кортеж (путь к каталогу, список каталогов, список файлов).

os.system(command) - исполняет системную команду, возвращает код её завершения (в случае успеха 0).

os.urandom(n) - n случайных байт. Возможно использование этой функции в криптографических целях.

os.path - модуль, реализующий некоторые полезные функции на работы с путями.