Модуль 8. Файлы, работа с файловой системой (2 ак. ч.)

[Работа с модулем os 2](#_Toc143117924)

[Получение системных сведений 2](#_Toc143117925)

[Создание и удаление папки 3](#_Toc143117926)

[Переименование файла 4](#_Toc143117927)

[Удаление файла 4](#_Toc143117928)

[Существование файла 4](#_Toc143117929)

[Вывод текущей директории 5](#_Toc143117930)

[Получить список файлов в директории/каталоге 5](#_Toc143117931)

[Изменение пути к рабочей папке 5](#_Toc143117932)

[Создание вложенных папок 7](#_Toc143117933)

[Работа с файлами и директориями 10](#_Toc143117934)

[Файлы и работа с файловой системой в Python 11](#_Toc143117935)

[Типы файлов 12](#_Toc143117936)

[Методы работы с файлами 12](#_Toc143117937)

[Практические задания 15](#_Toc143117938)

Для работы с файловой системой в Python используют модули os, os.path и shututil, а для операций с файлами – встроенные функции open(), close(), read(), readline(), write()

Как системный администратор, вы сталкиваетесь с многочисленными проблемами. Управление пользователями, дисковым пространством, процессами, устройствами и бэкапами могут стать причиной потери волос, юмора или вменяемости у многих администраторов.

Shell скрипты могут выручить, но они зачастую имеют множество ограничений. В этом случае полнофункциональный скриптовый язык, такой как Python, способен превратить утомительную задачу в легкую и, смею сказать, веселую.

По существу, модуль – это ресурс, который вы подключаете к программе, чтобы затем использовать его. Этот процесс можно сравнить с тем, что вы достанете из ящика лист бумаги и положите его на свой стол, подготовив тем самым к дальнейшему использованию. Модули подключаются при помощи команды import, которая присутствует в начале каждого примера.

И так, сегодня на конкретном примере я постараюсь рассказать о том, как можно работать с различным оборудованием при помощи Python, как этим облегчить свою работу и жизнь. Но Python не панацея. Всё зависит от ваших текущих знаний или какие знания и опыт вы хотели бы получить. Есть приверженцы Bash/Powershell.

Кто-то говорит, что это хороший вариант.

На мой взгляд — проще учить один язык, который на 90% одинаково будет себя вести на разных платформах, чем учить 2 языка.

Да и ни Powershell своими командлетами, ни Bash своим синтаксисом мне совершенно не нравятся.

Тем более, что у Python есть уже огромное количество разнообразных модулей и вы можете даже сделать для своих скриптом кросплатформенный GUI, если уж совсем скучно станет.

В нашей компании есть инженер, который в рамках своих задач по сопровождению приложений и серверов JEE написал не просто скрипт, а целую систему на Bash, которая была положительно отмечена руководством, используется не только в рамках нашей компании и нашими инженерами, но уже и разработчиками самой системы. Так что выбор инструмента, по сути, не так уж и важен.

Главное что бы вы были в состоянии реализовать на нём то, что задумываете.

# Работа с модулем os

## Получение системных сведений

Обработка файлов в Python с помощью модуля os включает создание, переименование, перемещение, удаление файлов и папок, а также получение списка всех файлов и каталогов и многое другое.

В индустрии программного обеспечения большинство программ тем или иным образом обрабатывают файлы: создают их, переименовывают, перемещают и так далее.

Любой программист должен обладать таким навыком. С этим руководством вы научитесь использовать модуль os в Python для проведения операций над файлами и каталогами вне зависимости от используемой операционной системы.

Важно знать, что модуль os используется не только для работы с файлами. Он включает массу методов и инструментов для других операций: обработки переменных среды, управления системными процессами, а также аргументы командной строки и даже расширенные атрибуты файлов, которые есть только в Linux.

Модуль встроенный, поэтому для работы с ним не нужно ничего устанавливать.

os.name

Модуль os обладает как вызываемыми функциями, так и обычными значениями. В случае с os.name, речь идет только о значениях. При доступе к os.name, вы получите информацию о том, с какой платформой вы работаете.

Вам откроются следующие значения: ‘posix’, ‘nt’, ‘os2’, ‘ce’, ‘java’, ‘riscos’. Давайте посмотрим на то, что выйдет, если запустить его на Windows 7:

import os

print(os.name)

Это говорит нам о том, что наш экземпляр Пайтон работает в окне Windows. Как мы об этом узнали? Майкрософт начали называть свою операционную систему NT много лет назад. Например, Windows 7 также носит имя Windows NT 6.1.

Ваши данные будут отличаться от моих, так как конфигурация компьютера у всех немного отличается, однако вы определенно увидите большое количество совпадений.

Как вы могли заметить, это вернуло словарь. Это значит, что вы можете получить доступ к значениям среды, пользуясь обычными словарными методами. Например:

print(environ["TMP"])

print(getenv("TMP"))

Полезность использования **os.getenv**() вместо словаря **os.environ** заключается в том, что если вы находитесь в положении, когда вам нужно получить доступ к переменной среды, которая не существует, **функция getenv попросту ничего не сделает**.

Если вы попытаетесь сделать то же самое, пользуясь **os.environ**, вы получите уведомление об ошибке. Давайте попробуем на примере:

print(getenv("TMP2"))

Ряд возможностей по работе с каталогами и файлами предоставляет встроенный модуль os. Хотя он содержит много функций, рассмотрим только основные из них:

*# mkdir(): создает новую папку  
# rmdir(): удаляет папку  
# rename(): переименовывает файл  
# remove(): удаляет файл*

### Создание и удаление папки

Для создания папки применяется функция mkdir(), в которую передается путь к создаваемой папке:

# путь относительно текущего скрипта

os.mkdir("hello")

# абсолютный путь

os.mkdir("c://somedir")

os.mkdir("c://somedir/hello")

Для удаления папки используется функция rmdir(), в которую передается путь к удаляемой папке:

# путь относительно текущего скрипта

os.rmdir("hello")

# абсолютный путь

os.rmdir("c://somedir/hello")

### Переименование файла

Для переименования вызывается функция rename(source, target), первый параметр которой - путь к исходному файлу, а второй - новое имя файла. В качестве путей могут использоваться как абсолютные, так и относительные. Например, пусть в папке C://SomeDir/ располагается файл somefile.txt. Переименуем его в файл "hello.txt":

os.rename("C://SomeDir/somefile.txt", "C://SomeDir/hello.txt")

### Удаление файла

Для удаления вызывается функция remove(), в которую передается путь к файлу:

os.remove("C://SomeDir/hello.txt")

### Существование файла

Если мы попытаемся открыть файл, который не существует, то Python выбросит исключение FileNotFoundError. Для отлова исключения мы можем использовать конструкцию try...except.

Однако можно уже до открытия файла проверить, существует ли он или нет с помощью метода os.path.exists(path). В этот метод передается путь, который необходимо проверить:

filename = input("Введите путь к файлу: ")

if os.path.exists(filename):

print("Указанный файл существует")

else:

print("Файл не существует")

### Вывод текущей директории

Для получения текущего рабочего каталога используется os.getcwd():

Если вам нужно знать, какой путь вы в данный момент используете, для этой нужно вызвать os.getcwd(). Попробуем воспользоваться обоими модулями:

import os

# Возвращает путь к текущей рабочей папке

pth=os.getcwd()

print(pth)

os.getcwd() возвращает строку в Юникоде, представляющую текущий рабочий каталог. Вот пример вывода:

Текущая директория: C:\python3\bin

os.getcwdb()

# b'C:\\Users\\Boss\\PycharmProjects\\pythonProject1'

### Получить список файлов в директории/каталоге

for fname in listdir('.'):

# Если у текущего имени файла расширение .py, то печатаем его

if ".py" in fname:

print(fname)

path = '.'

rez = sorted(os.listdir(path))

for n, item in enumerate(rez):

print(n+1, item)

### Изменение пути к рабочей папке

Функция chdir() модуля os изменяет текущий рабочий каталог. Аргумент path может принимать объекты, представляющие путь файловой системы, такие как pathlib.PurePath.

Эта функция может поддерживать указание дескриптора файла dir\_fd. Дескриптор должен ссылаться на открытый каталог, а не на открытый файл.

Устанавливает путь к текущей рабочей папке, в данном случае это диск

С:/

os.chdir(r'C:/')  
  
rez = sorted(os.listdir(path))  
for n, item in enumerate(rez):  
 print(n+1, item)

Если ты работаешь в Windows, то в пути к файлу или папке перед открывающей кавычкой указывай букву r (что означает raw) или вместо одной косой черты в пути ставь две.

Функция mkdir() модуля os создает каталог с именем path с режимом доступа к нему mode. Аргумент path может принимать объекты, представляющие путь файловой системы, такие как pathlib.PurePath.

Для создания папки/каталога в любой операционной системе нужна следующая команда:

# создать пустой каталог (папку)

os.mkdir("folder")

После ее выполнения в текущем рабочем каталоге тут же появится новая папка с названием «folder».

Если запустить ее еще раз, будет вызвана ошибка FileExistsError, потому что такая папка уже есть. Для решения проблемы нужно запускать команду только в том случае, если каталога с таким же именем нет. Этого можно добиться следующим образом:

# повторный запуск mkdir с тем же именем вызывает FileExistsError,

# вместо этого запустите:

if not os.path.isdir("folder"):

os.mkdir("folder")

Функция os.path.isdir() вернет True, если переданное имя ссылается на существующий каталог.

Изменение директории

Менять директории довольно просто. Проделаем это с только что созданным:

# изменение текущего каталога на 'folder'

os.chdir("folder")

Еще раз выведем рабочий каталог:

# вывод текущей папки

print("Текущая директория изменилась на folder:", os.getcwd())

Вывод:

Текущая директория изменилась на folder: C:\python3\bin\folder

### Создание вложенных папок

Предположим, вы хотите создать не только одну папку, но и несколько вложенных:

# вернуться в предыдущую директорию

os.chdir("..")

Функция makedirs() модуля os рекурсивно создает все промежуточные каталоги, если они не существуют. Функция работает подобно os.mkdir(), но создает все каталоги промежуточного уровня, необходимые для хранения конечного каталога.

# сделать несколько вложенных папок

os.makedirs("nested1/nested2/nested3")

Это создаст три папки рекурсивно, как показано на следующем изображении:

Функция rmdir() модуля os удаляет путь к каталогу path. Если директория path не существует или не является пустым каталогом, соответственно возникает исключение FileNotFoundError или OSError.

Функция removedirs() модуля os удаляет каталоги рекурсивно.

Работает подобно [функции os.rmdir()][os.rmdir] за исключением того, что, если конечный каталог успешно удален,os.removedirs()` пытается последовательно удалить каждый родительский каталог, указанный в пути, до появления ошибки.

Появления ошибки обычно означает, что родительский каталог не пуст.

Функция stat() модуля os получает статистическую информацию файла или дескриптора файла.

Выполняет эквивалент системного вызова stat() по заданному пути. Путь path может быть указан в виде строки или bytes, переданных прямо или косвенно через интерфейс os.PathLike или как дескриптор открытого файла. Функция stat() вернет объект os.stat\_result.

Функция os.lstat() эквивалентна вызову функции os.stat() с установленным аргументом follow\_symlinks=False, например os.stat(path, dir\_fd=dir\_fd, follow\_symlinks=False).

Переименование файлов

С помощью модуля os достаточно просто переименовать файл. Поменяем название созданного в прошлом шаге.

# переименовать text.txt на renamed-text.txt

os.rename("text.txt", "renamed-text.txt")

Функция os.rename() принимает 2 аргумента: имя файла или папки, которые нужно переименовать и новое имя.

Перемещение файлов

Функцию os.replace() можно использовать для перемещения файлов или каталогов:

# заменить (переместить) этот файл в другой каталог

os.replace("renamed-text.txt", "folder/renamed-text.txt")

Стоит обратить внимание, что это перезапишет путь, поэтому если в папке folder уже есть файл с таким же именем (renamed-text.txt), он будет перезаписан.

Список файлов и директорий

# распечатать все файлы и папки в текущем каталоге

print("Все папки и файлы:", os.listdir())

Функция os.listdir() возвращает список, который содержит имена файлов в папке. Если в качестве аргумента не указывать ничего, вернется список файлов и папок текущего рабочего каталога:

Все папки и файлы: ['folder', 'handling-files', 'nested1', 'text.txt']

Задание: Как вывести все файлы с расширением .py

А что если нужно узнать состав и этих папок тоже? Для этого нужно использовать функцию os.walk():

# распечатать все файлы и папки рекурсивно

for dirpath, dirnames, filenames in os.walk("."):

# перебрать каталоги

for dirname in dirnames:

print("Каталог:", os.path.join(dirpath, dirname))

# перебрать файлы

for filename in filenames:

print("Файл:", os.path.join(dirpath, filename))

os.walk() – это генератор дерева каталогов. Он будет перебирать все переданные составляющие. Здесь в качестве аргумента передано значение «.», которое обозначает верхушку дерева:

Каталог: .\folder

Каталог: .\handling-files

Каталог: .\nested1

Файл: .\text.txt

Файл: .\handling-files\listing\_files.py

Файл: .\handling-files\README.md

Каталог: .\nested1\nested2

Каталог: .\nested1\nested2\nested3

Метод os.path.join() был использован для объединения текущего пути с именем файла/папки.

Удаление файлов

Удалим созданный файл:

# удалить этот файл

os.remove("folder/renamed-text.txt")

os.remove() удалит файл с указанным именем (не каталог).

Удаление директорий

С помощью функции os.rmdir() можно удалить указанную папку:

# удалить папку

os.rmdir("folder")

Для удаления каталогов рекурсивно необходимо использовать os.removedirs():

# удалить вложенные папки

os.removedirs("nested1/nested2/nested3")

Это удалит только пустые каталоги.

Получение информации о файлах

Для получения информации о файле в ОС используется функция os.stat(), которая выполняет системный вызов stat() по выбранному пути:

open("text.txt", "w").write("Это текстовый файл")

# вывести некоторые данные о файле

print(os.stat("text.txt"))

Вывод:

os.stat\_result(st\_mode=33206, st\_ino=14355223812608232, st\_dev=1558443184, st\_nlink=1, st\_uid=0, st\_gid=0, st\_size=19, st\_atime=1575967618, st\_mtime=1575967618, st\_ctime=1575966941)

Это вернет кортеж с отдельными метриками. В их числе есть следующие:

st\_size – размер файла в байтах

st\_atime – время последнего доступа в секундах (временная метка)

st\_mtime – время последнего изменения

st\_ctime – в Windows это время создания файла, а в Linux – последнего изменения метаданных

Для получения конкретного атрибута нужно писать следующим образом:

# например, получить размер файла

print("Размер файла:", os.stat("text.txt").st\_size)

Вывод:

Размер файла: 19

Выводы

Работать с файлами и каталогами в Python очень просто. Не имеет значения даже используемая операционная система, хотя отдельные уникальные для системы функции можно использовать: например, os.chown() или os.chmod() в Linux.

### Работа с файлами и директориями

Системному администратору часто приходится работать с файлами и директориями в операционной системе Windows. Для упрощения этой задачи можно использовать Python.

Пример скрипта для вывода содержимого директории:

directory = '.'

for filename in os.listdir(directory):

print(filename)

Этот пример скрипта выводит список файлов и директорий в указанной директории.

Пример скрипта для удаления файлов соответствующих определенному шаблону имени:

directory = '.'  
  
for filename in os.listdir(directory):  
 if filename.startswith('t'):  
 remove(directory + filename)

Этот пример скрипта удаляет все файлы, имена которых начинаются со слова «example» в указанной директории.

Пример скрипта для создания новой директории:

directory = 'C:/Users/UserName/Documents/NewDirectory'

if not os.path.exists(directory):

os.makedirs(directory)

Этот пример скрипта создает новую поддиректорию «NewDirectory» в указанной директории, если ее еще не существует.

# Файлы и работа с файловой системой в Python

### Существование файла

Если мы попытаемся открыть файл, который не существует, то Python выбросит исключение FileNotFoundError. Для отлова исключения мы можем использовать конструкцию try...except. Однако можно уже до открытия файла проверить, существует ли он или нет с помощью метода **os.path.exists(path)**. В этот метод передается путь, который необходимо проверить:

filename = input("Введите путь к файлу: ")  
if os.path.exists(filename):  
 print("Указанный файл существует")   
else:  
 print("Файл не существует")

Открыть файл для проведения каких-либо манипуляций можно двумя способами:

С помощью функции open() – в этом случае после завершения работы нужно будет закрыть файл с помощью close():

import time  
  
file = open('file.txt', 'a', encoding='utf-8')  
  
x = 0  
while True:  
 print(x, file=file, end=',')  
 time.sleep(0.1)  
 x += 1

или с помощью функции write

f = open('task.txt', 'a', encoding='utf-8')

f.write('\n2) Написать модуль авторизации')

f.close()

С использованием менеджера контекста with, который автоматически и самостоятельно закроет файл, когда надобность в нем отпадет:

with open('task.txt', 'a', encoding='utf-8') as f:

f.write('\n2) Написать модуль авторизации')

### Типы файлов

В Python все файлы делятся на:

Текстовые 't'

Бинарные (двоичные) 'b'

Если никакой определенный тип файла не указан, по умолчанию Python считает, что пользователь работает с текстовыми файлами. Поэтому для работы с изображениями, мультимедийными файлами и pdf, например, следует указывать, что тип файла относится к 'b'.

Режимы доступа и записи

'r' Открывает файл для чтения. Возвращает ошибку, если указанный файл не существует.

'w' Открывает файл для записи, причем перезаписывает содержимое, если оно есть. Создает файл, если он не существует.

'a' Открывает файл для записи и добавляет новую информацию, не перезаписывая существующую. Создает файл, если он не существует.

'w+' Открывает файл для чтения и записи, перезаписывает содержимое.

'r+' Открывает файл для чтения и дополнения, не перезаписывает содержимое.

'x' Создает новый пустой файл. Возвращает ошибку, если файл с таким именем уже существует.

Примечание: для операций с двоичными файлами к указанным параметрам нужно добавлять b, то есть вместо 'r' должно быть 'rb', а вместо 'w+' – 'wb+'.

### Методы работы с файлами

Для чтения данных используют read(). Метод read() по умолчанию возвращает все содержимое файла:

with open('books.txt', 'r', encoding='utf-8') as f:

info = f.read()

print(info)

Вывод:

1. "Террор", Дэн Симмонс

2. "Она же Грейс", Маргарет Этвуд

3. "Облачный атлас", Дэвид Митчелл

4. "Искупление", Иэн Макьюэн

5. "Госпожа Бовари", Гюстав Флобер

При необходимости объем выводимой информации можно ограничить определенным количеством символов:

with open('movies.txt', 'r', encoding='utf-8') as f:

info = f.read(15)

print(info)

Метод readline() позволяет считывать информацию из текстовых файлов построчно:

with open('books.txt', 'r', encoding='utf-8') as f:

info = f.readline()

print(info)

Для получения всех строк файла используют метод readlines(), который возвращает содержимое в виде списка – вместе со всеми спецсимволами:

with open('books.txt', 'r', encoding='utf-8') as f:

info = f.readlines()

print(info)

['1. "Террор", Дэн Симмонс\n', '2. "Она же Грейс", Маргарет Этвуд\n', '3. "Облачный атлас", Дэвид Митчелл\n', '4. "Искупление", Иэн Макьюэн\n', '5. "Госпожа Бовари", Гюстав Флобер']

Чтобы избавиться от лишних пробелов, символа новой строки (и любых других спецсимволов), используют методы rstrip(), lstrip() или strip():

with open('books.txt', 'r', encoding='utf-8-sig') as f:

info = [line.strip() for line in f.readlines()]

print(info)

['1. "Террор", Дэн Симмонс', '2. "Она же Грейс", Маргарет Этвуд', '3. "Облачный атлас", Дэвид Митчелл', '4. "Искупление", Иэн Макьюэн', '5. "Госпожа Бовари", Гюстав Флобер']

Для записи информации в файл используют метод write():

with open('books.txt', 'a', encoding='utf-8') as f:

f.write('\n6. "Война и мир", Лев Толстой\n')

Или writelines():

with open('books.txt', 'a', encoding='utf-8') as f:

f.writelines(['7. "Преступление и наказание", Федор Достоевский\n',

'8. "Мизери", Стивен Кинг\n',

'9. "Джейн Эйр", Шарлотта Бронте\n'])

Кроме того, записывать данные в файл можно с помощью print():

with open('crm\_log.txt', 'a', encoding='utf-8') as file:

print('\nСергей Леонов, 11:18, 12:23', file=file)

Чтобы узнать текущую позицию курсора в файле, используют метод tell():

with open('books.txt', 'r', encoding='utf-8-sig') as f:

f.readline()

f.read(5)

print(f.tell())

Вывод:

51

Для считывания данных из определенной позиции используют seek():

with open('movies.txt', 'r', encoding='utf-8') as f:

f.seek(53)

print(f.read())

# Практические задания

Задание 1

Имеется файл books.txt, содержащий следующую информацию:

1. "Террор", Дэн Симмонс

2. "Она же Грейс", Маргарет Этвуд

3. "Облачный атлас", Дэвид Митчелл

4. "Искупление", Иэн Макьюэн

5. "Госпожа Бовари", Гюстав Флобер

6. "Война и мир", Лев Толстой

7. "Преступление и наказание", Федор Достоевский

8. "Мизери", Стивен Кинг

9. "Джейн Эйр", Шарлотта Бронте

Напишите программу, которая выводит первые 3 строки файла.

Решение:

with open('books.txt', 'r', encoding='utf-8') as file:

for i in range(3):

line = file.readline().strip()

print(line)

Задание 2

Напишите программу, которая находит самое длинное слово в списке книг из первого задания.

Ожидаемый результат:

Преступление

Решение:

with open('books.txt', 'r', encoding='utf-8') as file:

words = file.read().replace('"', '').split()

result = [w for w in words if len(w) == len(max(words, key=len))]

print(\*result)

3. В файле содержится последовательность целых чисел. Определите и запишите в ответе сначала количество пар элементов последовательности, в которых хотя бы одно число делится на 3, затем максимальную из сумм элементов таких пар.

В данной задаче под парой подразумевается два идущих подряд элемента последовательности. Например, для последовательности из пяти элементов: 6; 2; 9; –3; 6   — ответ: 4 11.

Ссылка на текстовый файл

<https://drive.google.com/file/d/1vPbdw-ooX6BR4MMbXWARtLrL75die0RQ/view?usp=sharing>

(Ответ: 2802 1990)

4. Текстовый файл состоит из символов X, Y и Z. Определите длину самой длинной последовательности, состоящей из символов X. Хотя бы один символ X находится в последовательности.

Для выполнения этого задания следует написать программу. Ниже приведён файл, который необходимо обработать с помощью данного алгоритма.

Содержимое файла

XXXYYXXYZXXYXZYXXXYYXXXXXXXXYZXYXXXYXYZYYZXYXXXXXXXXXYXZXYYXYYYXXXYXYZYYXYXYZXYXXYYXZXXXXYXZXYXYXXXYXXZXXYYZYXXXYYYYZXYYXXZXXXXYXXXYXXXZZXXXZYYZXXZXXXYXXYXXYYXYYXZYXYXZXZYXXYXXXYXXXZYYYYXXXXXZXZXXYXZXZXXZZZYYZXXXXYYZXZYXYYXXZYYXXYXZXXZXXYYXXZXXYYXZXXYXZYXXXXYZXZYXXXYXYYXYXYZYXXXXYXYXXXYXXXXXYYXYXYYXXYZXZYXYXYYZXXXZYXYYYYZYYYZYXXXYYYYYYXZXYYZXZYYXYZZXZXXZYYYYXXXYYXXXXYXZXXYXYXXYYZZZXXYXXXYXYXYXXYZXXYZXXXYXYYXZXYYXXXYXXYXZXZYYXXZXXXXYXYYXXXXZXXYYZZXXXZXXZYXXYXXZYZYXXYYZXYXYYXYYXY

Задание 4

Имеется файл fruit.txt со следующим содержимым:

Апельсин маракуйя папайя айва Яблоко

апельсин яблоко ананас банан персик Слива

Банан груша слива виноград авокадо Цитрон

Слива Груша яблоко мандарин цитрон

лимон Лайм апельсин ананас персик айва

Хурма киви хурма манго авокадо лайм

Нектарин Инжир гранат Папайя Гранат

Напишите программу, которая подсчитывает, сколько раз название каждого фрукта встречается в тексте.

Вывод:

Названия этих фруктов встречаются в тексте:

"апельсин" - 3 раз(а)

"маракуйя" - 1 раз(а)

"папайя" - 2 раз(а)

"айва" - 2 раз(а)

"яблоко" - 3 раз(а)

"ананас" - 2 раз(а)

"банан" - 2 раз(а)

"персик" - 2 раз(а)

"слива" - 3 раз(а)

"груша" - 2 раз(а)

"виноград" - 1 раз(а)

"авокадо" - 2 раз(а)

"цитрон" - 2 раз(а)

"мандарин" - 1 раз(а)

"лимон" - 1 раз(а)

"лайм" - 2 раз(а)

"хурма" - 2 раз(а)

"киви" - 1 раз(а)

"манго" - 1 раз(а)

"нектарин" - 1 раз(а)

"инжир" - 1 раз(а)

"гранат" - 2 раз(а)

Решение:

with open('fruit.txt', 'r', encoding='utf-8') as file:

result = {}

for line in file:

words = line.strip().lower().split()

for w in words:

result[w] = result.get(w, 0) + 1

print(f'Названия этих фруктов встречаются в тексте:')

for k, v in result.items():

print(f'"{k}" - {v} раз(а)'

Задание 4

Имеются два файла, first.txt и second.txt. В первом файле перечислены имена, во втором – должности:

Содержимое файла first.txt

Сергей Ковалев

Ирина Егорова

Никита Степанов

Марина Арефьева

Кирилл Евстегнеев

Елена Моисеева

Файл second.txt

мидл разработчик

девопс

тимлид

сеньор разработчик

продакт-менеджер

дизайнер

Напишите программу, которая открывает оба файла и выводит сведения о сотрудниках в следующем формате:

Сотрудник Сергей Ковалев, должность - мидл разработчик

Сотрудник Ирина Егорова, должность - девопс

Сотрудник Никита Степанов, должность - тимлид

Сотрудник Марина Арефьева, должность - сеньор разработчик

Сотрудник Кирилл Евстегнеев, должность - продакт-менеджер

Сотрудник Елена Моисеева, должность – дизайнер

Решение:

with open('first.txt', 'r', encoding='utf-8') as file1, \

open('second.txt', 'r', encoding='utf-8') as file2:

for line\_x, line\_y in zip(file1, file2):

print(f'Сотрудник {line\_x.strip()}, должность - {line\_y.strip()}')

Задание 6

Напишите программу, которая:

генерирует коды букв русского алфавита от а до я;

вставляет в полученный список код ё;

записывает полный алфавит (строчный и прописной варианты каждой буквы) в столбик в файл alphabet.txt.

Содержимое файла после выполнения программы:

Аа

Бб

Вв

Гг

Дд

Ее

Ёё

Жж

Зз

Ии

Йй

Кк

Лл

Мм

Нн

Оо

Пп

Рр

Сс

Тт

Уу

Фф

Хх

Цц

Чч

Шш

Щщ

Ъъ

Ыы

Ьь

Ээ

Юю

Яя

Решение:

alpha = [i for i in range(ord('а'), ord('я') + 1)]

alpha.insert(6, 1105)

with open('alphabet.txt', 'w', encoding='utf-8') as file:

for i in alpha:

file.write(chr(i).upper() + chr(i) + '\n')

Задание 7

Сведения о доходах и расходах компании хранятся в двух файлах, income.txt и outcome.txt. Напишите программу для подсчета прибыли компании.

Пример файла income.txt:

RUB100000

RUB200000

RUB200000

RUB500000

RUB600000

RUB100000

RUB700000

Пример файла outcome.txt:

-RUB1000

-RUB2000

-RUB50000

-RUB200000

-RUB10000

-RUB5000

-RUB6000

-RUB10000

-RUB19000

-RUB7000

Ожидаемый результат:

Прибыль за прошлый месяц: 2090000 RUB

Решение:

income, outcome = 0, 0

with open('income.txt', 'r', encoding='utf-8') as file1, \

open('outcome.txt', 'r', encoding='utf-8') as file2:

for line in file1:

num = line.strip()[3:]

income += int(num)

for line in file2:

num = line.strip()[4:]

outcome += int(num)

print(f'Прибыль за прошлый месяц: {income - outcome} RUB')

Задание 8

Напишите программу, которая считывает сведения об успеваемости по основным предметам из файла grades.txt, и определяет, кто из студентов имеет средний балл >= 4.5.

Пример файла grades.txt:

Арефьева И. 5 5 4 4

Богданов С. 5 5 3 4

Гаврилов Е. 4 4 3 3

Егорова А. 3 3 3 4

Семенова Е. 4 3 3 4

Хворостов И. 5 4 5 5

Ожидаемый результат:

Арефьева И., средний балл: 4.50

Хворостов И., средний балл: 4.75

Решение:

result = {}

with open('grades.txt', 'r', encoding='utf-8') as file1:

for line in file1:

l = line.strip().split()

grades = [int(i) for i in l[-4:]]

aver\_grade = sum(grades) / len(grades)

if aver\_grade >= 4.5:

result[l[0] + ' ' + l[1]] = aver\_grade

for student, aver\_grade in result.items():

print(f'{student}, средний балл: {aver\_grade:.2f}')

Задание 9

Напишите программу, которая получает от пользователя название файла с информацией на русском языке, и выполняет транслитерацию текста в соответствии с приведенным ниже словарем. Результат нужно записать в файл result.txt.

Словарь:

translit = {'а': 'a', 'к': 'k', 'х': 'h', 'б': 'b', 'л': 'l', 'ц': 'c', 'в': 'v', 'м': 'm', 'ч': 'ch', 'г': 'g', 'н': 'n',

'ш': 'sh', 'д': 'd', 'о': 'o', 'щ': 'shh', 'е': 'e', 'п': 'p', 'ъ': '\*', 'ё': 'jo', 'р': 'r', 'ы': 'y', 'ж': 'zh',

'с': 's', 'ь': "'", 'з': 'z', 'т': 't', 'э': 'je', 'и': 'i', 'у': 'u', 'ю': 'ju', 'й': 'j', 'ф': 'f', 'я': 'ya',

'А': 'A', 'К': 'K', 'Х': 'H', 'Б': 'B', 'Л': 'L', 'Ц': 'C', 'В': 'V', 'М': 'M', 'Ч': 'Ch', 'Г': 'G', 'Н': 'N',

'Ш': 'Sh', 'Д': 'D', 'О': 'O', 'Щ': 'Shh', 'Е': 'E', 'П': 'P', 'Ъ': '\*', 'Ё': 'Jo', 'Р': 'R', 'Ы': 'Y', 'Ж': 'Zh',

'С': 'S', 'Ь': "'", 'З': 'Z', 'Т': 'T', 'Э': 'Je', 'И': 'I', 'У': 'U', 'Ю': 'Ju', 'Й': 'J', 'Ф': 'F', 'Я': 'Ya'}

Пример русского текста:

Python – это язык программирования, который широко используется в интернет-приложениях, разработке программного обеспечения, науке о данных и машинном обучении (ML). Разработчики используют Python, потому что он эффективен, прост в изучении и работает на разных платформах.

Ожидаемое содержимое result.txt:

Python – jeto yazyk programmirovaniya, kotoryj shiroko ispol'zuetsya v internet-prilozheniyah, razrabotke programmnogo obespecheniya, nauke o dannyh i mashinnom obuchenii (ML). Razrabotchiki ispol'zujut Python, potomu chto on jeffektiven, prost v izuchenii i rabotaet na raznyh platformah.

Решение:

translit = {'а': 'a', 'к': 'k', 'х': 'h', 'б': 'b', 'л': 'l', 'ц': 'c', 'в': 'v', 'м': 'm', 'ч': 'ch', 'г': 'g', 'н': 'n',

'ш': 'sh', 'д': 'd', 'о': 'o', 'щ': 'shh', 'е': 'e', 'п': 'p', 'ъ': '\*', 'ё': 'jo', 'р': 'r', 'ы': 'y', 'ж': 'zh',

'с': 's', 'ь': "'", 'з': 'z', 'т': 't', 'э': 'je', 'и': 'i', 'у': 'u', 'ю': 'ju', 'й': 'j', 'ф': 'f', 'я': 'ya',

'А': 'A', 'К': 'K', 'Х': 'H', 'Б': 'B', 'Л': 'L', 'Ц': 'C', 'В': 'V', 'М': 'M', 'Ч': 'Ch', 'Г': 'G', 'Н': 'N',

'Ш': 'Sh', 'Д': 'D', 'О': 'O', 'Щ': 'Shh', 'Е': 'E', 'П': 'P', 'Ъ': '\*', 'Ё': 'Jo', 'Р': 'R', 'Ы': 'Y', 'Ж': 'Zh',

'С': 'S', 'Ь': "'", 'З': 'Z', 'Т': 'T', 'Э': 'Je', 'И': 'I', 'У': 'U', 'Ю': 'Ju', 'Й': 'J', 'Ф': 'F', 'Я': 'Ya'}

filename = input()

with open(filename, 'r', encoding='utf-8') as source, open('result.txt', 'w', encoding='utf-8') as tr\_result:

for l in source.read():

trans = translit.get(l.lower(), l)

tr\_result.write(trans if l.islower() else trans.capitalize())

Задание 10

Имеется лог-файл crm\_log.txt с информацией о времени входа пользователей в CRM-систему и о времени выхода. Напишите программу, которая определяет пользователей, работавших в системе более 4 часов, и записывает их имена и фамилии в файл best\_employees.txt.

Пример лога crm\_log.txt:

Егор Тимофеев, 09:10, 16:50

Марина Абрамова, 12:00, 15:59

Никита Круглов, 09:10, 12:45

Анна Семенова, 08:10, 12:30

Юлия Сафонова, 10:10, 10:50

Михаил Колесников, 11:10, 12:10

Ожидаемый результат в файле best\_employees.txt:

Егор Тимофеев

Анна Семенова

Решение:

with open('crm\_log.txt', encoding='utf-8') as file\_input, \

open('best\_employees.txt', 'w', encoding='utf-8') as file\_output:

for line in file\_input:

start, end = [int(h) \* 60 + int(m) for t in line.split(', ')[1:] for h, m in [t.split(':')]]

if end - start > 240:

file\_output.write(line.split(', ')[0] + '\n')

Подведем итоги

Задание 2

Напишите программу, которая получает от пользователя имя файла и возвращает следующие данные о его содержимом:

количество строк;

количество слов;

число символов без пробелов и точек.

Пример ввода:

series.txt

Содержимое файла:

1. Последнее королевство 2015

2. Рим 2005

3. Версаль 2015

4. Тюдоры 2007

5. Террор 2018

6. Человек в высоком замке 2015

7. Белая королева 2013

8. Братья по оружию 2001

9. Медичи 2016

10. Спартак 2010

Вывод:

Количество строк в файле series.txt: 10

Количество слов: 17

Число символов: 153

Решение:

filename = input()

lines, words, symbols = 0, 0, 0

with open(filename, 'r', encoding='utf-8') as file:

for i in file:

lines += 1

words += len([w for w in i.split() if w.isalpha()])

symbols += len([s for s in i if s.isalnum()])

print(f'Количество строк в файле {filename}: {lines}\n'

f'Количество слов: {words}\n'

f'Число символов: {symbols}\n'

)

7. Текстовый файл состоит из символов A, C, D, F и O. Определите максимальное количество идущих подряд пар символов вида согласная + гласная.

Ссылка на текстовый файл

<https://drive.google.com/file/d/1xyj12easUVi33Wr6-zaXLw7vZrnakZXG/view?usp=sharing>

(ответ 95)

Python предоставляет множество простых и удобных методов для работы с файловой системой и файлами. С помощью этих методов можно легко:

Получать любую информацию о файлах и директориях.

Создавать, изменять, перемещать и удалять директории и файлы.

Проводить любые операции с содержимым текстовых и двоичных файлов.

В следующей статье будем изучать регулярные выражения.