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 №2.15**

**дисциплины «Анализ данных»**

**Вариант 13**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Выполнил:  Иващенко Олег Андреевич  2 курс, группа ИВТ-б-о-22-1,  09.03.02 «Информационные и вычислительные машины», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |  |
|  | | Руководитель практики:  Воронкин Роман Александрович, доцент кафедры инфокоммуникаций  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |  |
|  |  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

**Тема**: «Работа с фалами в языке Python»

**Цель**: 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.

Порядок выполнения работы

Пример 1.

Таблица 1 – Код программы example\_1.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  with open("file1.txt", "w") as fileptr:  fileptr.write(  "Python is the modern day language. It makes things so simple.\n"  "It is the fastest-growing programming language"  ) |

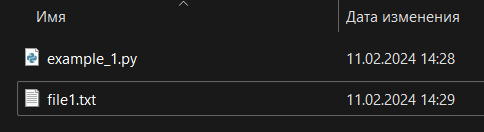


Рисунок 1.1 – Созданный программой example\_1.py файл file1.txt

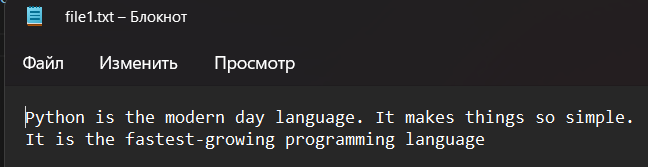


Рисунок 1.2 – Текст в текстовом файле file1.txt

Пример 2.

Таблица 2 – Код программы example\_2.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  with open("file1.txt", "a") as fileptr:  fileptr.write(" Python has an easy syntax and user-friendy interaction.") |

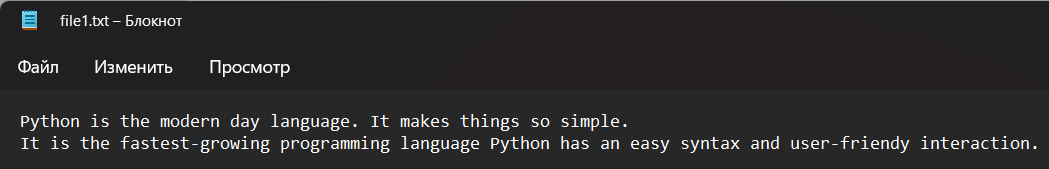


Рисунок 2 – Результат выполнения программы example\_2.py

Пример 3. Чтение строк с помощью метода readline()

Таблица 3 – Код программы example\_3.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  with open("file1.txt", "r") as fileptr:          content1 = fileptr.readline()          content2 = fileptr.readline()          print(content1)          print(content2) |

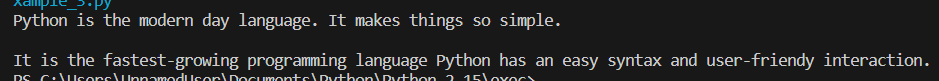


Рисунок 3 – Вывод программы example\_3.py

Пример 4. Чтение строк с помощью функции readlines().

Таблица 4 – Код программы example\_4.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":      with open("file1.txt", "r") as fileptr:          content = fileptr.readlines()          print(content) |



Рисунок 4 – Вывод программы example\_4.py

Пример 5.

Таблица 5 – Код программы example\_5.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":      with open("newfile.txt", "x") as fileptr:          print(fileptr)          if fileptr:              print("File created successfully") |



Рисунок 5.1 – Результат выполнения программы example\_5.py

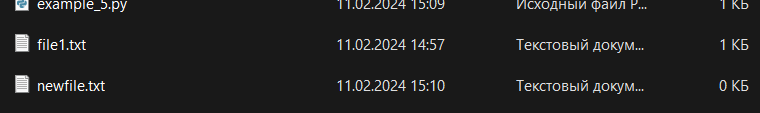


Рисунок 5.2 – Созданный новый файл newfile.txt

Пример 6.

Таблица 6 – Код программы example\_6.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  if \_\_name\_\_ == "\_\_main\_\_":      with open("text.txt", "w", encoding="utf-8") as fileptr:          print(              "UTF-8 is a variable-width characted encoding"              "used for electronic communication.",              file = fileptr          )          print(              "UTF-8 is capable of encoding all 1,112,064 valid character"              "code points.",              file = fileptr          )          print(              "In Unicode using one to four one-byte (8-bit) code units.",              file = fileptr          ) |

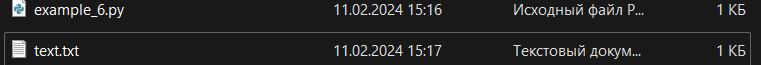


Рисунок 6.1 – Созданный новый текстовый файл text.txt

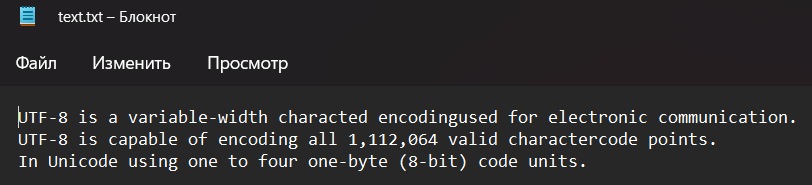


Рисунок 6.2 – Содержимое текстового файла text.txt

Пример 7. Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие запятые. Каждое предложение в файле записано на отдельной строке.

Таблица 7 – Код программы

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  if \_\_name\_\_ == "\_\_main\_\_":      with open("text.txt", "r", encoding="utf-8") as fileptr:          sentences = fileptr.readlines()      for sentence in sentences:          if "," in sentence:              print(sentence) |



Рисунок 7 – Вывод программы example\_7.py

Пример 8. Позиция указателя файла

Таблица 8 – Код программы example\_8.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*- if \_\_name\_\_ == "\_\_main\_\_":      with open("file1.txt", "r") as fileptr:          print("The filepointer is at byte: ", fileptr.tell())          fileptr.seek(10)          print("After reading, the filepointer is at: ", fileptr.tell()) |



Рисунок 8 – Вывод программы example\_8.py

Пример 9. Переименование файла.

Таблица 9 – Код программы example\_9.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_      os.rename("file1.txt", "renamed\_file.txt") |

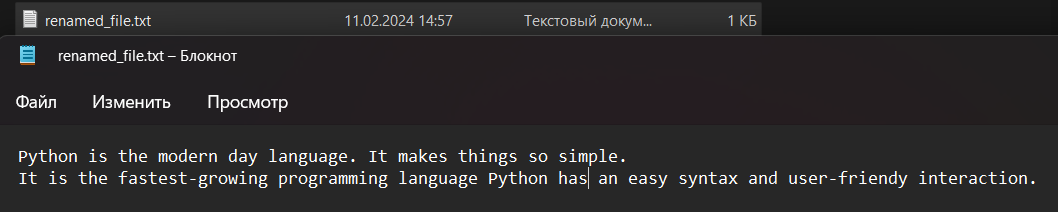


Рисунок 9 – Результат выполнения программы example\_9.py

Пример 10. Удаление файла.

Таблица 10 – Код программы example\_10.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      os.remove("renamed\_file.txt") |

Пример 11. Создание нового каталога.

Таблица 11 – Код программы example\_11.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      os.mkdir("new") |

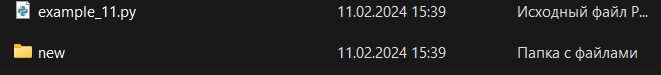


Рисунок 11 – Созданная программой example\_11.py новая директория

Пример 12. Получение текущего рабочего каталога.

Таблица 12 – Код программы example\_12.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      path = os.getcwd()      print(path) |



Рисунок 12 – Вывод программы example\_12.py

Пример 13. Изменение текущего рабочего каталога.

Таблица 13 – Код программы example\_13.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      os.chdir("C:\\Windows")      print(os.getcwd()) |



Рисунок 13 – Вывод программы example\_13.py

Пример 14. Удаление каталога.

Таблица 14 – Код программы example\_14.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      os.rmdir("new") |

Пример 15. Доступ к элементам командной строки.

Таблица 15 – Код программы example\_15.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import sys  if \_\_name\_\_ == "\_\_main\_\_":      print("Number of arguments: ", len(sys.argv), " arguments")      print("Argument list: ", str(sys.argv)) |



Рисунок 15.1 – Вывод программы example\_15.py при простом запуске

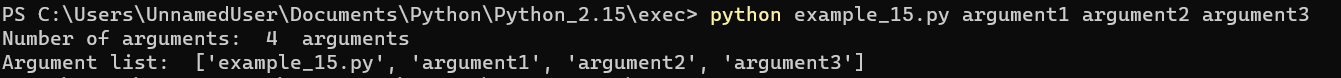


Рисунок 15.2 – Вывод программы example\_15.py при запуске через терминал с вводом аргументов

Пример 16.

Таблица 16 – Код программы example\_16.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import sys  if \_\_name\_\_ == "\_\_main\_\_":      for idx, arg in enumerate(sys.argv):          print(f"Argument #{idx} is {arg}")      print("No. of arguments passed is ", len(sys.argv)) |



Рисунок 16.1 – Вывод программы example\_16.py при обычном запуске

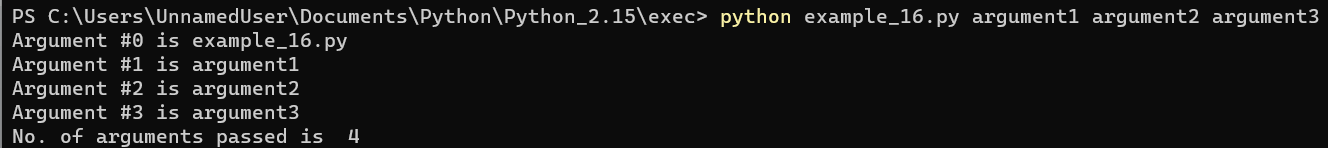


Рисунок 16.2 – Вывод программы example\_16.py при запуске через терминал с вводом аргументов

Пример 17. Написать программу для генерации пароля заданной длины. Длина пароля должна передаваться как аргумент командной строки сценария.

Таблица 17 – Код программы example\_17.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  import secrets  import string  import sys  if \_\_name\_\_ == "\_\_main\_\_":      if len(sys.argv) != 2:          print("The password length is not given!", file=sys.stderr)          sys.exit(1)      chars = string.ascii\_letters + string.punctuation + string.digits      length\_pwd = int(sys.argv[1])      result = []      for \_ in range(length\_pwd):          idx = secrets.SystemRandom().randrange(len(chars))          result.append(chars[idx])      print(f"Secret password: {''.join(result)}") |



Рисунок 17.1 – Вывод программы example\_17.py при обычном запуске



Рисунок 17.2 – Вывод программы example\_17.py при запуске через терминал с вводом аргумента

Задание 1. Написать программу, которая считывает текст из файла и выводит его на экран, заменив цифры от 0 до 9 на слова «ноль», «один», ..., «девять», начиная каждое предложение с новой строки.

Таблица 18 – Код программы individual\_1.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import os  if \_\_name\_\_ == "\_\_main\_\_":      words = {          '0': 'Ноль',          '1': 'Один',          '2': 'Два',          '3': 'Три',          '4': 'Четыре',          '5': 'Пять',          '6': 'Шесть',          '7': 'Семь',          '8': 'Восемь',          '9': 'Девять',          '10': 'Десять'      }      with open("individual\_1.txt", "r", encoding="utf-8") as file:          sentences = file.readlines()          for sentence in sentences:              for digit, word in words.items():                  sentence = sentence.replace(digit, word)              print(sentence) |

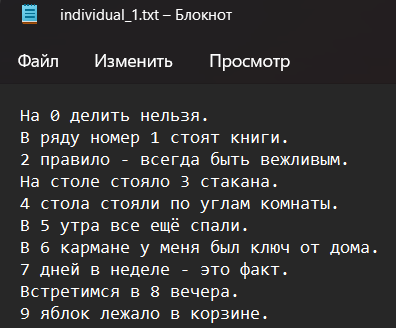


Рисунок 18.1 – Содержимое файла individual\_1.txt

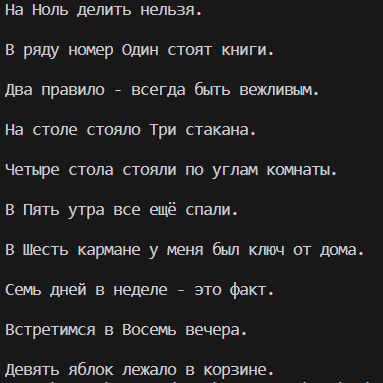


Рисунок 18.2 – Вывод программы individual\_1.py

Задание 2. Проверка орфографии – лишь составная часть расширенного текстового анализа на предмет наличия ошибок. Одной из самых распространённых ошибок в текстах является повторение слов. Например, автор может по ошибке дважды подряд написать одно слово. Некоторые текстовые процессоры умеют распознавать такой вид ошибок при выполнении текстового анализа. В данном упражнении вам предстоит написать программу для определения наличия дублей слов в тексте. При нахождении повтора на экран должен выводиться номер строки и дублирующееся слово. Удостоверьтесь, что программа корректно обрабатывает случаи, когда повторяющиеся слова находятся на разных строках. Имя файла для анализа должно быть передано программе в качестве единственного аргумента командной строки. При отсутствии аргумента или невозможности открыть указанный файл на экране должно появляться соответствующее сообщение об ошибке.

Таблица 19 – Код программы individual\_2.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import osimport sys  if \_\_name\_\_ == "\_\_main\_\_":      if len(sys.argv) != 2:          print("Имя файла не указано. Используйте python"                " individual\_2.py <имя файла>", file=sys.stderr)          sys.exit(1)      else:          with open("individual\_2.txt", "r", encoding="utf-8") as file:              strings = file.readlines()              prev\_word = None              words\_seen = {}              for idx, line in enumerate(strings):                  words = line.strip().split()                  for word in words:                      if (prev\_word == word):                          print(f"В строке {idx} слово '{word}' повторяется дважды")                      elif word in words\_seen:                          if idx != words\_seen[word]:                              print(f"В строке {idx} слово '{word}' повторяется"                                    " на строке {words\_seen[word]}")                      prev\_word = word |

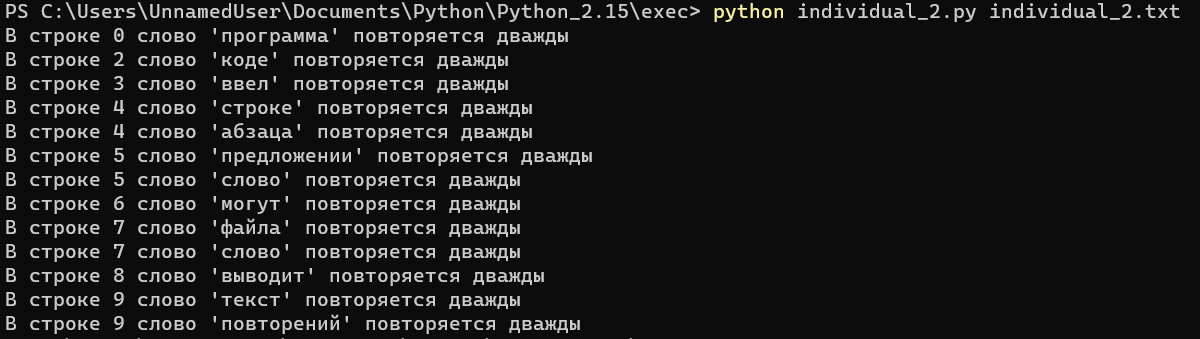


Рисунок 19.1 – Содержимое текстового файла individual\_2.txt

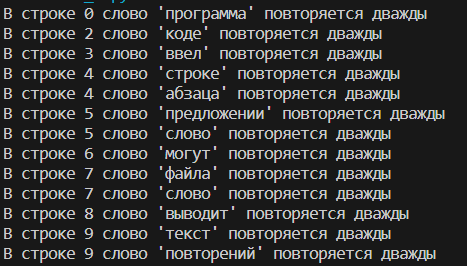


Рисунок 19.2 – Вывод программы individual\_2.py

Контрольные вопросы

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

Python предоставляет функцию open(), которая принимает два аргумента: имя файла и режим доступа, в котором осуществляется доступ к файлу. Функция возвращает файловый объект, который можно использовать для выполнения различных операций, таких как чтение, запись и т.д. Синтаксис:

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

Файл по умолчанию открывается в режиме чтения, но также открыть файл только для чтения можно следующий образом:

file\_object = open(<file-name>, "r")

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

Открыть файл только для записи можно следующий образом:

file\_object = open(<file-name>, "w")

1. Как прочитать данные из файла в языке Python?

Синтаксис для открытия файла и получения данных с помощью оператора with ... as:

with open("file\_name", "r") as f:

content = f.read();

print(content)

1. Как записать данные в файл в языке Python?

Для записи текста 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 создаёт новый файл, если файл не существует.

1. Как закрыть файл в языке Python?

Для закрытия файла в Python используется метод close(). Любая незаписанная информация уничтожается после вызова этого метода для файлового объекта.

Синтаксис использования метода close():

fileobject.close()

После закрытия файла мы не можем выполнять какие-либо операции с файлом. Файл необходимо правильно закрыть. Если при выполнении некоторых операций с файлом возникает какое-либо исключение, программа завершается, не закрывая файл. Для этого можно использовать следующий метод, чтобы решить такую проблему:

try:

fileptr = open(<file\_name>)

finally:

fileptr.close()

1. Изучите самостоятельно работу конструкции with ... as. Каково её назначение в языке Python? Где она может быть использована ещё, помимо работы с файлами?

Консутркция with ... as в языке Python используется для управления контекстом выполнения кода. Её основное назначение – гарантировать, что некоторый ресурс будет правильно управлен в рамках блока кода, и закрыт после завершения работы с ним, независимо от того, произошли ли ошибки в процессе выполнения кода или нет.

Помимо работы с файлами, конструкция with ... as может быть использована для управления другими типами ресурсов, такими как сетевые соединения, базы данных, блокировки, потоки и т.д.

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

* read(size=-1) – читает указанное количествро байтов из файла. Если значение параметра size не указано или отрицательно, то читается весь файл;
* readline(size=-1) – читает одну строку из файла;
* readlines(size=-1) – читает все строки файла и возвращает список строк;
* write(string) – записывает списоку в файл;
* writelines(lines) – записывает список строк в файл;
* seek(offset, whence=0) – перемещает указатель текущей позиции в файле на заданное смещение `offest` относительно начала файла. Аргумент `whence` определяет базу смещения (0 – начало файла, 1 – текущая позиция, 2 – конец файла).
* tell() – возвращает текущую позицию указателя в файле.

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

* chmod(path, mode) – изменяет права доступа к файлу или директории;
* chown(path, uid, gid) – изменяет владельца и группу файла или директории;
* stat(path) – возвращает информацию о файле или директории в виде объекта os.stat\_result;
* path.getsize(path) – возвращает размер файла в байтах;
* utime(path, times=None, \*, ns=None) – устанавливает временные метки доступа и модификации файла.

**Выводы**: В процессе выполнения лабораторной работы были приобретены навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x, было создано виртуальное окружение с помощью Anaconda, с помощью менеджера пакетов conda установлен ряд пакетов. Однако, с пакетом TenserFlow возникли проблемы при установке с помощью conda – решение заморожено. Но при помощи менеджера пакетов pip установка прошла успешно. Также были сформированы файлы environment.yml и requirements.txt. В первом файле находятся параметры окружения, нужные для восстановления окружения в любой момент. Во втором находятся пакеты и зависимости, которые были установлены на момент формирования файла.