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 №5**

**Языки программирования**

**Вариант 6**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Пушкин Максим Алексеевич  2 курс, группа ИТС-б-о-21-1,  11.03.02 «Инфокоммуникационные технологии и системы связи», направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Р.А., кандидат технических наук доцент кафедры инфокоммуникаций  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  |  | |  |

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2022 г.

**Тема:** Работа с файлами 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.

# Открытие файла

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

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

Доступ к файлам можно получить с помощью различных режимов, таких как чтение, запись или добавление. посмотрим на простой пример, чтобы открыть файл с именем «file.txt»(хранящийся в том же каталоге) в режиме чтения и распечатать его содержимое на консоли

#opens the file file.txt in read mode fileptr = open("file.txt","r")

if fileptr:

print("file is opened successfully")

file is opened successfully

**Метод** close()

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()

# opens the file file.txt in read mode fileptr = open("file.txt","r")

if fileptr:

print("file is opened successfully") #closes the opened file fileptr.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.

try:

fileptr = open("file.txt") # perform file operations finally:

fileptr.close()

**Оператор** with

Он полезен в случае манипулирования файлами. Используется в сценарии, когда пара операторов должна выполняться с блоком кода между ними.

with open(<file name>, <access mode>) as <file-pointer>: #statement suite

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.

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

**Порядок выполнения работы:**

1. Создадим общедоступный репозиторий на GitHub
2. Решим задачи с помощью языка программирования Python3. и отправим их на GitHub.

Вывод пример 1



Рисунок 1. Результат.

Вывод пример 2



Рисунок 2. Результат.

Вывод пример 3

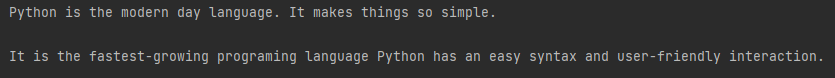


Рисунок 3. Результат.

Вывод пример 4

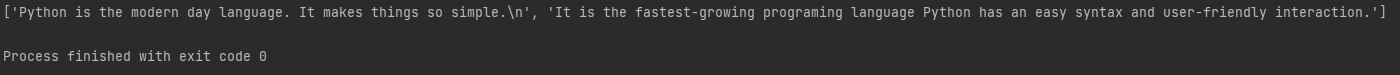


Рисунок 4. Результат.

Вывод пример 5

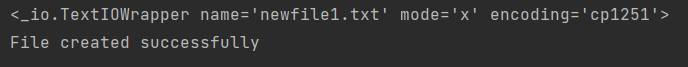


Рисунок 5. Результат.

Вывод пример 6

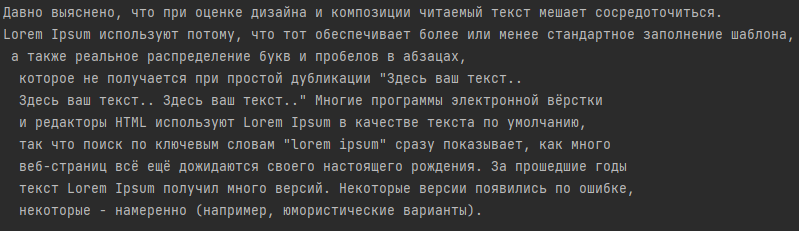
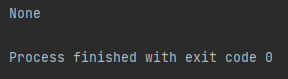


Рисунок 6. Результат.

Вывод пример 7

C:\Users\User\Documents\ShareX\Screenshots\2022-12\pycharm64_dY2Tvrgorf.png

Вывод пример 8



Вывод пример 9-10

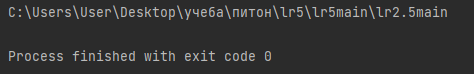


Рисунок 9. Результат

Вывод пример 11

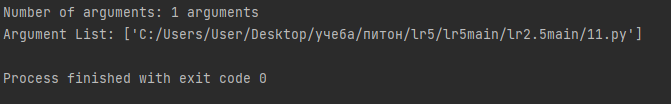


Рисунок 10. Результат

Вывод пример 12

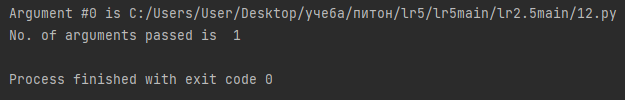


Рисунок 11. Результат

Индивидуальная 1

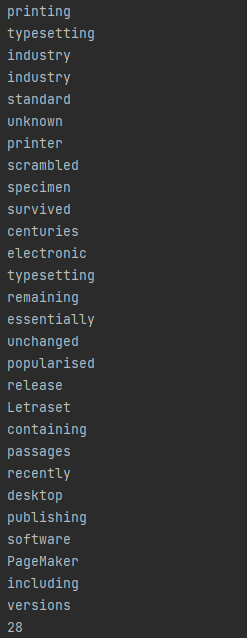
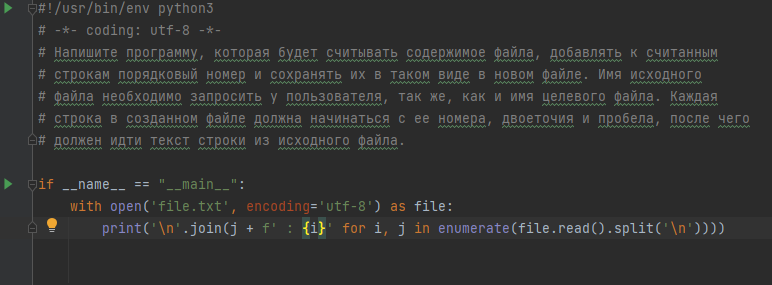


Рисунок 12. Результат

Индивидуальна 2



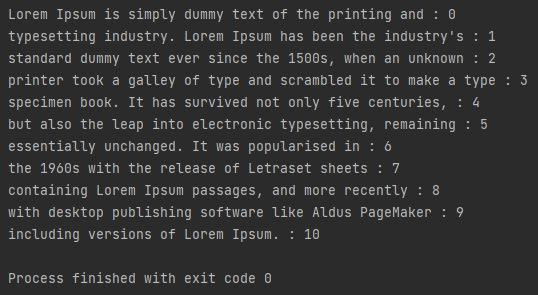


Рисунок 13. Результат

**Контрольные вопросы**

1. Что такое списки в языке Python?

Ответ: Список (*list*) – это структура данных для хранения объектов различных типов. Размер списка не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

2. Как осуществляется создание списка в Python?

Ответ: Для создания списка нудно заключить элементы в квадратные скобки:

my\_list = [1, 2, 3, 4, 5]

3. Как организовано хранение списков в оперативной памяти?

Ответ: При создании списка в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличии от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Ответ: Читать элементы списка можно с помощью следующего цикла: my\_list = ['один', 'два', 'три', 'четыре', 'пять']

for elem in my\_list: print(elem)

5. Какие существуют арифметические операции со списками?

Ответ: Для объединения списков можно использовать оператор сложения (+).

Список можно повторить с помощью оператора умножения (\*): 6. Как проверить есть ли элемент в списке?

Ответ: Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор in :

7. Как определить число вхождений заданного элемента в списке?

Ответ: Метод *count* можно использовать для определения числа сколько

раз данный элемент встречается в списке:

8. Как осуществляется добавление (вставка) элемента в список?

Ответ: Метод *insert* можно использовать, чтобы вставить элемент в список.

Метод *append* можно использовать для добавления элемента в список. 9. Как выполнить сортировку списка?

Ответ: Для сортировки списка нужно использовать метод *sort*.

Для сортировки списка в порядке убывания необходимо вызвать метод *sort* с аргументом *reverse=True*.

10. Как удалить один или несколько элементов из списка?

Ответ: Удалить элемент можно, написав его индекс в методе *pop*: Если не указывать индекс, то функция удалит последний элемент. Элемент можно удалить с помощью метода *remove*.

Оператор del можно использовать для тех же целей:

Можно удалить несколько элементов с помощью оператора среза:

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

Ответ: *List Comprehensions* чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения списков.

В языке Python есть две очень мощные функции для работы с коллекциями: map и filter. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как list, tuple, set, dict и т.п. Списковое включение позволяет обойтись без этих функций.

12. Как осуществляется доступ к элементам списков с помощью срезов? Ответ: Слайсы (срезы) являются очень мощной составляющей *Python*, которая позволяет быстро и лаконично решать задачи выборки элементов из

списка.

Слайс задается тройкой чисел, разделенных запятой: *start:stop:step*. *Start*

– позиция с которой нужно начать выборку, *stop* – конечная позиция, *step* – шаг. При этом необходимо помнить, что выборка не включает элемент определяемый *stop*.

13. Какие существуют функции агрегации для работы со списками? Ответ: Для работы со списками Python предоставляет следующие

функции:

len(L) - получить число элементов в списке L . min(L) - получить минимальный элемент списка L . max(L) - получить максимальный элемент списка L .

sum(L) - получить сумму элементов списка L , если список L содержит только числовые значения.

14. Как создать копию списка?

Ответ: Воспользоваться командой copy.copy(x)

15. Самостоятельно изучите функцию *sorted* языка Python. В чем ее отличие от метода *sort* списков?

Ответ: Функция sorted() в Python возвращает отсортированный список из элементов в итерируемом объекте. list.sort() на 13% быстрее, чем sorted().

Вывод: Входе работы мы изучили и приобрели по работе со списками при написании программ на языке Python