**IT-Колледж “Сириус”**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ДОКЛАД**

по дисциплине “Программирование на языке Python”

на тему “Файловый ввод и вывод, структурированные текстовые файлы”

Выполнил:  
 Студент группы 7-1 Группа  
Веретеннникова Вероника Андреевна

Принял:

Преподаватель  
Тенигин Альберт Андреевич

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

IT-Колледж “Сириус”  
 2021

1

Файловый ввод и вывод, структурированные текстовые файлы

**Содержание**

* 3 Начало работы с файлами:
* 3 Встроенные средства Python
* 4 Типы файлов и операции над ними
* 5 Чтение из файла
* Файлы в Python:
* 6 Файловый вывод
* 7-8 Файловый ввод
* Правила работы с файлами в Python:
* 9-10 Структурированные текстовые файлы
* 11 Выводы

3

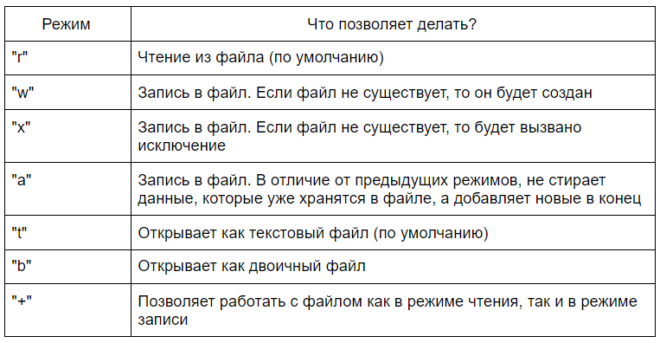
Файловый ввод и вывод, структурированные текстовые файлы

**Начало работы с файлами**

Файлы применяются для временного хранения информации, как для собственной работы (настроек и пр. ), так и полученной во время ее исполнения (результаты вычислений, таблицы, диаграммы). Большинство программ сейчас в некотором варианте используют файлы, в том числе сохраняя итоги работы между сеансами запуска.

## **Встроенные средства Python**

Основа для работы с файлами — built-in функция open() open(file, mode="rt") Эта функция имеет два аргумента. Аргумент file принимает строку, в которой хранится дорога к файлу. Второй аргумент, mode, позволяет указать режим, в котором нужно работать с файлом. По умолчанию данный аргумент принимает значение «rt», с которым, и с некоторыми другими, можно познакомиться в таблице ниже



3

Файловый ввод и вывод, структурированные текстовые файлы

**Типы файлов и операции над ними**

Файл — это набор данных, сохраненный в виде определенной последовательности битов на компьютере. Информация хранится в куче данных (структура данных) и имеет название «Filename»

В Python существует два типа файлов:

1. -Текстовые
2. -Бинарные

### **Текстовые файлы**

Это файлы с привычным человеку содержимым. В них хранятся последовательности символов, к примеру, на русском языке, которые понимает человек. Блокнот, Word и другие редакторы умеют читать и обрабатывать этот тип файлов.

Текст может храниться в нескольких форматах: (.txt) — обычный текст и (.rtf) — «формат обогащенного текста».

### Бинарные файлы

В бинарных файлах данные показываются в закодированной форме (с использованием только нулей и единиц ). Почти всегда это просто последовательности битов.

Они хранятся только в формате .bin.

Операцию с файлом можно разбить на три самых масштабных этапа:

1/3. Открытие/закрытие файла

2.Выполнение определенных операции

4

Файловый ввод и вывод, структурированные текстовые файлы

## **Чтение из файла**

При открытии файлов в режимах, позволяющих чтение, можно использовать некоторое количество подходов.

Для начала прочитаем файл целиком и все данные, находящиеся в нем, запищем в одну строку.

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

text = f.read()

Также при работе это функции с целочисленным аргументом, можно прочитать определенное количество символов.

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

part = f.read(16)

При этом способе будут получены только 16 первых символов текста. Важно, что при применении этой функции некоторое количество раз подряд будут считываться части этого текста — виртуальный курсор сдвигается на часть текста, которую считали.

Другой способ — считывание файла построчно.

Метод

readline()

read()

T.e сдвигает курсор на целую строку. Применения этот метод несколько раз - считывание нескольких строк.

5

Файловый ввод и вывод, структурированные текстовые файлы

**Файловый ввод и вывод**

**Вывод**

Один из распространенных способов вывода данных в Python– это печать их в консоли. Этот способ является основным для того, чтобы быстро просмотреть результат работы. Для вывода данных в консоль используется функция

print

Рассмотрим способы использования этой функции.

>>> print("Hello")

Hello

>>> print("Hello, " + "world!")

Hello, world!

>>> print("Age: " + str(**23**))

Age: 23

По умолчанию, для разделения элементов в функции print используется пробел.

>>> print("A", "B", "C")

A B C

Для замены разделителя необходимо использовать параметр sep функции print.

>>>print("A", "B", "C", sep="#")

A#B#C

6

Файловый ввод и вывод, структурированные текстовые файлы

**Файловый ввод и вывод**

**Ввод**

Для считывания вводимых с клавиатуры данных используется функция

input()

>>> input() test

'test'

Для сохранения данных в переменной используется следующий синтаксис.

>>> a = input()

hello

>>> print(a)

hello

Если считывается с клавиатуры целое число, то строку, получаемую с помощью функции

input()

можно передать сразу в функцию

int()

>>> val = int(input())

123

>>> print(val)

123

7

Файловый ввод и вывод, структурированные текстовые файлы

**Файловый ввод и вывод**

Вводить можно различные типы данных, операции. Мы можем использовать питон в качестве калькулятора, выполняя действия в коде. Пример:

print(5 + 10) #сложение

print(3 \* 7, (17 - 2) \* 8) #умножение, вычитание

print(2 \*\* 16) # две звёздочки означают возведение в степень

print(37 / 3) # один / — это деление с ответом-дробью

print(37 // 3) # два // считают частное от деления нацело

print(37 % 3) # процент считает остаток от деления нацело

Полученный результат мы можем преобразовывать в список, строку, переводить в другие системы счисления, переводить на другие языки и прочее

Преобразование строки в список осуществляется с помощью метода split(), по умолчанию, в качестве разделителя, используется пробел.

>>> l = input().split()

1 2 3 4 5 6 7

>>> print(l)

['1', '2', '3', '4', '5', '6', '7']

8

Файловый ввод и вывод, структурированные текстовые файлы

**Структурированные текстовые файлы**

with

Оператор Python был впервые представлен пять лет назад в версии Python 2.5. Это удобно, когда у вас есть две связанные операции, которые вы хотите выполнить как пара, с блоком кода между ними. Классический пример - открытие файла, изменение файла и его закрытие:

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

f.write ( ' Привет! ' )

Вышеупомянутый

with

оператор автоматически закроет файл после вложенного блока кода.

Преимущество использования

with

оператора состоит в том, что он гарантирует закрытие файла независимо от того, как закрывается вложенный блок. Если исключение возникает до конца блока, он закроет файл до того, как исключение будет перехвачено внешним обработчиком исключений.

Если бы вложенный блок содержал

return

оператор или оператор

continue / break / with

оператор автоматически закрыл бы файл и в этих случаях.

Context

объект используется для рисования шести прямоугольников, каждый с разным поворотом

9

Файловый ввод и вывод, структурированные текстовые файлы

**Структурированные текстовые файлы**

Каждый вызов

[rotate](http://cairographics.org/documentation/pycairo/2/reference/context.html" \l "cairo.Context.rotate)

фактически комбинируется с текущим преобразованием, поэтому мы используем пару вызовов

save

restore

для сохранения состояния рисования на каждой итерации цикла. Это предотвращает совмещение вращений друг с другом.

Структурированные текстовые файлы очень удобны в использовании благодаря большому набору функций, которые можно выполнять с ними. Одной из них является:

Объединение нескольких структурированных файлов :

Объединение файлов

myfile1.bin+myfile2.bin => myfile3.bin

1. Открыть файлы для чтения

f1 = open('myfile1.bin', 'rb')

f2 = open('myfile2.bin', 'rb')

2. Считать файлы в списки L1, L2

L1 = f1.readlines()

L2 = f2.readlines()

3. Объединить списки

L3 = L1 + L2

4. Закрыть файлы myfile1.bin, myfile2.bin

f1.close()

f2.close()

10

Файловый ввод и вывод, структурированные текстовые файлы

**Выводы**

В результате проведенной работы мы умеем вводить и выводить структурированные файлы в Python, открывать и закрывать файлы, использовать различные операторы вызова и пр. Для считывания вводимых с клавиатуры данных используется функция

input()

Самый распространенный способ вывода данных в консоль с помощью

print()

Структурированные текстовые файлы были представлены совсем недавно и полюбились пользователям. Это удобно, когда у вас есть две связанные операции, которые вы хотите выполнить как пара, с блоком кода между ними. Преимущество использования

with

оператора состоит в том, что он гарантирует закрытие файла независимо от того, как закрывается вложенный блок. Если исключение возникает до конца блока, он закроет файл до того, как исключение будет перехвачено внешним обработчиком исключений.

Структурированные текстовые файлы очень удобны и универсальны, они позволяют выполнять множество операций над собой, такие как приведение к одной системе, языку, объединяться между собой и прочее.

Их можно считывать построчно, вводить и выводить данные из них, сортировать, складывать между собой.

11

Файловый ввод и вывод, структурированные текстовые файлы

**Список литературы**

# The Python "with" Statement by Example:

# [Preshing on Programming](https://preshing.com/), sep20, 2011

# Файлы в python, ввод-вывод:

PythonRy, okt30, 2017

[Файлы и сериализация данных](https://www.yuripetrov.ru/edu/python/ch_08.html):

PythonCode, dec02, 2021

# Работа с текстовыми файлами Python 3:

Интернет технологии, Вадим Дворников, feb11, 2016

12