**Работа с данными: текстовые файлы, csv, excel, pickle**

Зачастую данные требующиеся для работы программы хранятся в файле. Для работы с файлами, имеющими расширение txt, csv, bin в языке Python существует ряд встроенных функций и модулей**.** Для работы с файлами, имеющими расширение xlsx/xls существует множество сторонних пакетов.

**Текстовые файлы**

Перед тем как начать работу с текстовым файлом, его требуется открыть с помощью встроенной функции open( file, mode='r', encoding=None), где :

* file – строка представляющая путь к файлу, который требуется открыть;
* mode – строка, обозначающая режим в котором следует открыть файл. Существуют следующий режимы доступа:

|  |  |
| --- | --- |
| Режим | Обозначение |
| 'r' | открытие на чтение (является значением по умолчанию). |
| 'w' | открытие на запись, содержимое файла удаляется, если файла не существует, создается новый. |
| 'x' | открытие на запись, если файла не существует, иначе исключение. |
| 'a' | открытие на дозапись, информация добавляется в конец файла. |
| '+' | открытие на чтение и запись |

* encoding - Имя кодировки для кодирования/декодирования файла. Если не указана, используется системная кодировка.

Если файл не может быть открыт, тогда возбуждается исключение OSError. После чтения/записи из файла, файл требуется закрыть с помощью функции close().

Пример:

import random as rnd  
import math  
  
# Пример 1   
# Для работы с файлом сначала вызываем встроенную open()  
# Здесь:  
# 'test.txt' - название файла, который лежит в папке проекта;  
# 'w' - режим доступа "Запись";  
# encoding='utf8' - кодировка файла 'utf8';  
# Изначально файла test.txt не существует,  
# после начала записи в файл, операционная система создаёт файл по заданному пути  
# Запишем в файл 10 случайных чисел  
f = open('test.txt', 'w', encoding='utf8')  
for \_ in range(1, 10):  
 # Генерируем случайное чило от 1 до 10 с помощью стандартного модуля random  
 numb = rnd.randint(1, 10)  
 # Для записи в файл у созданной переменной f воспользуемся функцией write()  
 f.write(str(numb) + '\n')  
# Не забываем закрыть файл, после его использования  
# Чтобы закрыть файл воспользуемся функцией close переменной f  
f.close()  
# Объявим список l1  
l1 = []  
# Обратите внимание, что для считывания из файла, вторым параметром отправляется строка 'r',  
# т.е. режима доступа "Чтение"  
r\_file = open('test.txt', 'r', encoding='utf8')  
# С помощью цикла for построчно считываем из файла test.txt  
for item in r\_file:  
 # С помощью встроенной функции int() приводим строку к типу int  
 l1.append(int(item))  
print(l1)  
# Не забываем закрыть файл  
r\_file.close()  
# Создаем список result  
result = []  
for item in l1:  
 # Добавим в список result округленный вниз квадратный корень числа из списка l1  
 result.append(math.floor(math.sqrt(item)))  
# Результат допишем в файл test.txt  
# Для этого, укажем режим доступа к файлу: "a"  
# т.е режим доступа Дозапись  
f\_result = open('test.txt', 'a', encoding='utf8')  
for item in result:  
 f\_result.write(str(item)+'\n')  
f\_result.close()

В языке Python существует оператор with … as, который упрощает работу с файлами. Оператор with … as автоматически закрывает файл после окончанию работы с ним. Общий синтаксис оператора with … as имеет вид:

with выражение as имя\_переменной:

набор инструкций

Для работы с файлами синтаксис оператора with примет вид:

with open() as file\_name:

чтение/запись в файл

Обратите внимание, что по окончанию работы с фалом не требуется его закрывать.

Пример :

# Создаем список texts  
texts = []  
# Пример 1  
# Открываем файл с помощью оператора with as  
# Этот код вызовет ошибку FileNotFoundError, т.к. такого файла не существует в папке нашего решения  
try:  
 # Этот код вызовет ошибку FileNotFoundError, т.к. такого файла не существует в папке нашего решения  
 with open('abc.txt', 'r', encoding='utf8') as f:  
 for line in f:  
 texts.append(line)  
# Отлавливаем не все ошибки, а только конкретную ошибку связанную с отсутствием файла в папке нашего решения  
except FileNotFoundError:  
 # Открываем файл abc.txt для записи случайных чисел в файл.  
 with open('abc.txt', 'w', encoding='utf8') as f:  
 for i in range(0, 5):  
 # Приводим i - типа int к строке с помощью функции str и в конце строки добавляем '\n'  
 f.write(str(i) + '\n')

**Работа с csv – файлами**

Файл csv – это текстовый файл, в котором каждая строка имеет несколько полей, разделенных запятыми, или другими разделителями. Для работы с файлами csv, т.е. чтение и запись, в стандартной библиотеке языка Python есть модуль под названием csv. В первой строке csv-файла описывается название полей. Этот модуль содержит большое число функций для работы с csv файлами, в этом пособии мы рассмотрим два основных метода:

* csv.reader(“file\_name”,delimiter=’,’) – метод предназначен для считывания из файла csv и имеет следующие параметры:
  + “file\_name” – путь до csv – файла;
  + delimiter – разделитель между колонками в csv – файле.
* csv.writer(“file\_name”,delimiter=’,’) – метод для записи в csv – файл. Параметры функции csv.writer() полностью аналогичны параметрам функции csv.reader().

Для примера использования модуля csv создадим пустой файл students.csv, содержащий следующую информацию:

number,name,surname,age,average\_rating  
1,Иван,Петров,19,3.39  
2,Тимофей,Васильев,20,4.33  
3,Петр,Иванов,17,4.0  
4,Виктор,Семенов,20,3.2

Примеры работы с файлами csv:

# Для работы с csv - файлами импортируем модуль csv из стандартной библиотеки языка Python  
import csv  
  
  
# Функция которая принимает переменную полученную путем вызова функции open()  
def csv\_read(file\_obj):  
 # Создаем переменную reader с помощью функции reader модуля csv  
 reader = csv.reader(file\_obj)  
 # считываем построчно и выводим на экран кнсоли  
 for row in reader:  
 print('\t'.join(row))  
  
  
# Функция которая принимает список списков и переменную полученную путем вызова функции open()  
def csv\_write(write\_data, file\_obj):  
 # создаём переменную writer с помощью функции csv.writer,   
 # которой передаем переменную файла и явно указываем разделитель  
 writer = csv.writer(file\_obj, delimiter=',')  
 for row in write\_data:  
 # Построчно записываем в файл  
 writer.writerow(row)  
  
  
# Т.к. csv-файл - это обычный текстовый файл, то для него применим оператор with as,  
# что не требует закрывать файл после его открытия  
# Открываем файл students.csv для чтения  
with open('students.csv', 'r', encoding='utf8') as f\_obj:  
 csv\_read(f\_obj)  
# Обратите внимание, что после оператора with as не требуется закрывать файл students.csv, он закроется автоматически   
# Создаем список data  
# Элементами data является список, полученный путем применения функции split() к строкам  
data = [  
 "5,Ярослав,Поляков,19,4.7".split(','),  
 "6,Владимир,Шляшко,21,3.7".split(',')  
]  
# Для записи в файл students.csv создадим переменную файла и вызовем функцию csv\_write  
with open('students.csv', 'a', encoding='utf8') as f\_obj:  
 csv\_write(data, f\_obj)

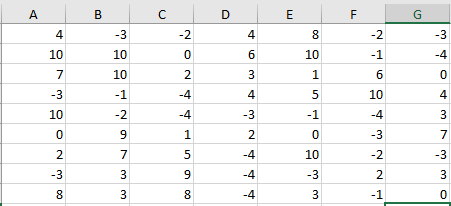
**Работа с файлами xls/xlsx**

Для работы с excel-файлами в стандартной библиотеке Python нет модулей, но существует ряд сторонних библиотек. Для записи и чтении данных из excel-файлов воспользуемся сторонней библиотекой openpyxl. Для установки openpyxl воспользуемся менеджером пакетов – pip. Установка openpyxl не отличается от установки пакта numpy, которая была описана в прошлых главах.

Рассмотрим пример создание xls файла, работы с ячейками, сохранения данных в файл и считывание данных из xls файла:

# Импортируем пакет openpyxl  
import openpyxl as opx  
import random as rnd  
  
# Перед началом работы с excel документом, требуется создать переменную с помощью метода Workbook  
wb = opx.Workbook()  
# Робота с excel - документами основана на работе с листами. По умолчанию документ создается с одним листом  
# Получаем рабочий лист с помощью свойства active, переменой wb  
ws = wb.active  
# К ячекам страницы можно обращаться как к ключам рабочей страницы  
c = ws['A4']  
# Запись значений в ячейку  
ws['A4'] = 10  
# Для записи большого числа данных в документ, воспользуемся методом cell переменной ws  
# Нумерация строк и колонок в excel - документе начинается с 1  
for row in range(1, 10):  
 for column in range(1, 8):  
 # Функция cell принимает 3 параметра:  
 # row - номер строки  
 # column - номер столбца  
 # value - значение, которое записывается в эту колонку  
 ws.cell(row=row, column=column, value=rnd.randint(-5, 10))  
# Для сохранения документа воспользуемся методом save(filename), где:  
# filename - путь по которому требуется сохранить файл, если такой файл существует, то он перезапишется  
wb.save('balances.xlsx')

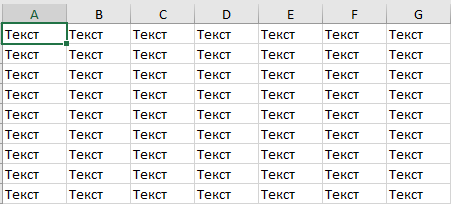
Созданный файл будет иметь следующий вид:



Теперь считаем из созданного файла, изменим некоторые значения и запишем обратно в файл:

# Импортируем пакет openpyxl  
import openpyxl as opx  
  
# Для открытия существующего excel - файла воспользуемся функцией load\_workbook(file\_name),  
# которая в качаестве параметра принимает путь до файла  
wb = opx.load\_workbook('balances.xlsx')  
# Получаем рабочий лист с помощью свойства active, переменой wb  
ws = wb.active  
# Для считывания из excel - документа воспользуемся свойством rows, переменной ws  
for row in ws.rows:  
 # row - это лист колонок, которые есть в этой строке  
 for cell in row:  
 # cell - представляет собой переменную, которая хранит значения одной ячейки,  
 # чтобы изменить её значение, обратимся к её свойству value  
 cell.value = "Текст"  
# Сохраним данные  
wb.save('balances.xlsx')

Итоговый файл примет следующий вид:



**Работа с модулем pickle**

Модуль pickle предоставляет функции для сериализации и десериализации объектов. В программировании под сериализацией понимают преобразование каких-либо данных в набор байтов, который потом обычно сохраняют в файл. Десериализация — это восстановление объектов из их байтовых представлений.

Сериализовать/десериализовать в языке Python можно следующие типы:

* None, True, False
* Строки
* Стандартные числовые типы данных
* Словари, списки, кортежи
* Функции
* Классы

Для сериализации и десериализации нам понадобятся следующие функции из модуля pickle:

* pickle.dump(obj,file) – записывает сериализованный объект в файл. Принимает следующие параметры:
  + obj – это объект, который мы сериализуем;
  + file – это путь до файла, в который мы записываем сериализованный объект.
* Pickle.load(file\_obj) – загружает объект из потока байт. Параметр file\_obj – это переменная созданная с помощью функции open().

Пример:

# Импортируем модуль pickle  
import pickle  
# Создаем словарь data  
data = {  
 'a': [1, 2.0, 3],  
 'b': ("просто строка", 3+2),  
 'c': {None, True, False}  
}  
  
# Сохраняем словарь data в бинарный файл data.bin  
# Для работы с файлом используем оператор with as  
# Режим в котором открывается файл data.bin wb, где:  
# w - Режим доступа "Запись"  
# b - означает, что запись будет производиться в бинарный файл  
with open('data.bin', 'wb') as f:  
 pickle.dump(data, f)  
  
# Считываем из файла data.bin  
# Для считывания из бинарного файла укажем режим доступа 'rb'  
with open('data.bin', 'rb') as f:  
 data\_new = pickle.load(f)  
# На консоль выведет: {'a': [1, 2.0, 3], 'b': ('просто строка', 5), 'c': {False, True, None}}  
print(data\_new)