Практическая работа №4

Разработка программ, обращающихся к файлам и каталогам.

Цель работы: изучить основные операции при работе с файлами и каталогами в Python, научиться разрабатывать и отлаживать программы, взаимодействующие с файлами

Задание на лабораторную работу:

1. Изучить теоретические сведения.
2. Написать программу в соответствии с вариантом.

Теоретические сведения

Файлом называются именованные области постоянной памяти в компьютере, которыми управляет операционная система.

Модуль os.path содержит множество функций, предназначенных для манипулирования путями доступа к файлам. Он содержится в модуле os, и для его импортирования достаточно выполнить инструкцию `import os`.
Полная документация модуля os.path приведена на сайте Python по адресу http://docs.python.org/3/library/os.path.html.

Функция os.path.join() возвращает строку, в которой путь доступа к файлу указан с использованием корректной версии разделителя.

```
>>> import os
>>> os.path.join('usr', 'bin', 'spam')
'usr\\bin\\spam'
```

Для получения значения ***текущего рабочего каталога*** в виде строки используется функция os.getcwd(), а для его изменения — функция os.chdir().

```
>>> import os
>>> os.getcwd()
'C:\\Python34'
>>> os.chdir('C:\\Windows\\System32')
>>> os.getcwd()
'C:\\Windows\\System32'
```

!!!! При попытке перейти в несуществующий каталог Python выведет сообщение об ошибке.

Проверка существования пути

Вызов os.path.exists (path) возвращает значение True, если файл (или папка), на который ссылается аргумент, существует.
Вызов os.path.isfile (path) возвращает значение True, если заданный аргументом путь существует и является файлом, и значение False в противном случае.
Вызов os.path.isdir (path) возвращает значение True, если заданный аргументом путь существует и является папкой; иначе — False.

С помощью функции os.makedirs() можно ***создавать новые папки*** (каталоги).

```
>>> import os
>>> os.makedirs('C:\\delicious\\walnut\\waffles')
# функция создаст все необходимые промежуточные папки, гарантируя
существование полного пути.
```

Обработка абсолютных и относительных путей

Существуют два способа определения пути доступа к файлу:
абсолютный путь, который всегда начинается с имени корневой папки;

относительный путь, который задается относительно текущего рабочего каталога программы.

При задании путей можно использовать:

• Одиночная точка является сокращенным обозначением, имеющим смысл "данная папка".

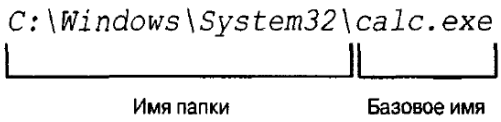
•• Двойная точка имеет смысл "родительская папка".

Вызов os.path.abspath(path) возвращает строку абсолютного пути аргумента. Это простой способ преобразования относительного пути в абсолютный.

Вызов os.path.dirname (path) возвращает строку, содержащую всю часть пути, которая предшествует последней косой черте в аргументе *path.*

Вызов os.path.basename (path) возвращает строку, содержащую всю ту часть пути, которая следует за последней косой чертой в аргументе *path.*

```
>>> path ='C:\\Windows\\System32\\calc.exe'
>>> os.path.basename(path)
'calc.exe'
>>> os.path.dirname(path)
'C:\\Windows\\System32'
```



Если нужны как имя папки, так и базовое имя, достаточно вызвать функцию os.path.split(), которая возвращает кортеж, включающий обе эти строки.

```
>>> os.path.split(path)
('C:\\Windows\\System32', 'calc.exe')
```

Определение размеров файлов и содержимого папок

Вызов функции os.path.getsize (path) возвращает выраженный в байтах размер файла, указанного в аргументе *path.*

```
>>> os.path.getsize('C:\\Windows\\System32\\calc.exe')
776192
```

Вызов os.listdir (path) возвращает список строк с именами всех файлов с путем доступа, указанным в аргументе path.

```
>>> os.listdir('C:\\Windows\\System32')
['0409', '12520437.cpx', '12520850.cpx', '5U877.ax', 'aaclient.dll',…]
```

Удаление файла, папки

Вызов os.unlink (путь) удаляет файл, расположенный по указанному пути.

Вызов os.rmdir (путь) удаляет папку, расположенную по указанному пути. Эта папка должна быть пуста.

Пример: удалить файлы с расширением .txt

```
import os
for filename in os.listdir():
    if filename.endswith('.txt'):
        os.unlink(filename)
```

Функции os.walk () – ***обход дерева каталогов***, передается единственное строковое значение — путь к папке. Функцию можно использовать в цикле for, она возвращает три значения на каждой итерации цикла:

1) строку, содержащую текущее имя папки;
2) список строк, представляющих имена папок, которые содержатся в текущей папке;
3) список строк, представляющих имена файлов, которые содержатся в текущей папке.

Поскольку функция os.walk () возвращает списки строк для переменных subfolder и filename, можно использовать эти списки в их циклах for.

Пример программы обхода дерева каталогов:

```
import os
for folderName, subfolders, filenames in os.walk('C:\\delicious'):
    print('Текущая папка - ' + folderName)
    for subfolder in subfolders:
```

```
        print('ПОДПАПКА ПАПКИ ' + folderName + ': ' + subfolder)
    for filename in filenames:
        print('ФАЙЛ В ПАПКЕ ' + folderName + ': ' + filename)
    print (` `)
```

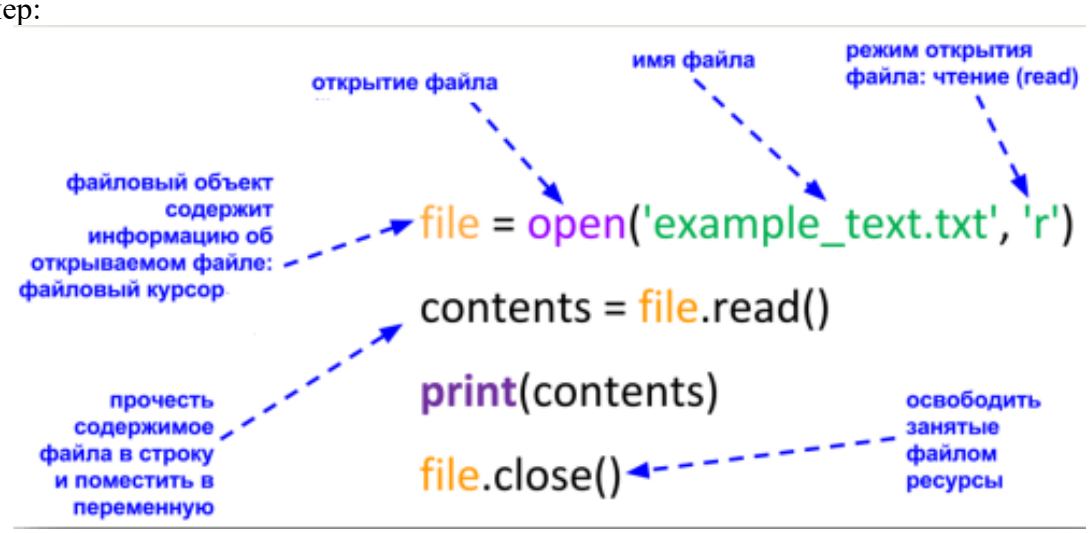
Чтение и запись файлов

В Python операции чтения/записи файлов выполняются в три этапа:
1) вызовите функцию open (), которая возвратит объект File;
2) вызовите метод read () или write () для объекта File;
3) закройте файл, вызвав метод close () для объекта File.

Режимы открытия файла:

'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
'b'	открытие в двоичном режиме
't'	открытие в текстовом режиме (является значением по умолчанию)
'+'	открытие на чтение и запись

Пример:



Операции над файлами:

Операция	Интерпретация
`output = open(r'C:\spam', 'w')`	Открывает файл для записи ('w' означает write – запись)
`input = open('data', 'r')`	Открывает файл для чтения ('r' означает read – чтение)
`input = open('data')`	То же самое, что и в предыдущей строке (режим 'r' используется по умолчанию)
`aString = input.read()`	Чтение файла целиком в единственную строку

`aString = input.read(N)`	Чтение следующих `N` символов (или байтов) в строку
`aString = input.readline()`	Чтение следующей текстовой строки (включая символ конца строки) в строку
`aList = input.readlines()`	Чтение файла целиком в список строк (включая символ конца строки)
`output.write(aString)`	Запись строки символов (или байтов) в файл
`output.writelines(aList)`	Запись всех строк из списка в файл
`output.close()`	Закрытие файла вручную (выполняется по окончании работы с файлом)
`output.flush()`	Выталкивает выходные буферы на диск, файл остается открытым
`anyFile.seek(N)`	Изменяет текущую позицию в файле для следующей операции, смещая ее на `N` байтов от начала файла.
`for line in open('data'):` *операции над line*	Итерации по файлу, построчное чтение
`open('f.txt',` `encoding='latin-1')`	Файлы с текстом Юникода в Python 3.0 (строки типа `str`)
`open('f.bin', 'rb')`	Файлы с двоичными данными в Python 3.0 (строки типа `bytes`)

!!!! Обратите внимание: данные, получаемые из файла, всегда попадают в сценарий в виде строки, поэтому необходимо будет выполнять преобразование данных в другие типы объектов языка Python, если эта форма представления не подходит. При выполнении операции записи данных в файл необходимо передавать методам уже сформированные строки (интерпретатор Python не выполняет автоматическое преобразование объектов в строки).

```
# Открывает файл в режиме записи (создает/очищает)
>>> myfile = open('myfile.txt', 'w')
>>> myfile.write('hello text file\n') # Записывает строку текста
16
>>> myfile.write('goodbye text file\n')
18
>>> myfile.close()    # Выталкивает выходные буферы на диск
# открыть файл для добавления информации в конец
>>> myfile = open('myfile.txt', 'a')
>>> myfile.write('new text\n')
9
>>> myfile.close()
#Открывает файл для чтения: 'r' - по умолчанию
>>> myfile = open('myfile.txt')
>>> content=myfile.read()        #Читает файл как одну строку
>>> myfile.close()
>>> print(content)
hello text file
goodbye text file
new text
```

Пример ***чтения файла*** plan.txt, находящегося в текущей директории, вывод его строкам с указанием длины строки:


```
with open('plan.txt', 'r') as file:
    for line in file:
        print(line)
        print(len(line.strip()))      #strip – удаляет символ \n
```

Пример: ***записать различные объекты в текстовый файл***.

```
>>>  X, Y,  Z = 43,  44,  45
>>>  S = 'Spam'
>>>  D = {'a':  1,   'b':  2}
>>>  L = [1,  2,  3]
>>>
>>>  F = open('datafile.txt', 'w')    # Создает файл для записи
>>>  F.write(S + '\n')                # Строку завершаем символом \n
>>>  F.write('%s,%s,%s\n' % (X,  Y,  Z))  # Преобразует числа в строки
>>>  F.write(str(L)+'$'+str(D)+'\n') # Преобразует и разделяет $
>>>  F.close()
```

Создав файл, его можно открыть файл и прочитать данные в строку целиком, с помощью метода read. Инструкция print интерпретирует встроенные символы конца строки, чтобы обеспечить более удобочитаемое отображение:

```
>>> chars = open('datafile.txt').read()
>>> print(chars)
Spam
43,44,45
[1,  2,  3]${'a':  1,   'b': 2}
```

Чтобы получить из строк объекты языка Python, нужно выполнить обратные преобразования:

```
>>> F = open('datafile.txt')     # Открыть файл снова
>>> line = F. readline()         # Прочитать одну строку
>>> line. rstrip()               # Удалить символ конца строки
'Spam'
>>> line = F.readline()          #  Следующая строка из файла
>>> parts = line.split(',')      # Разбить строку на подстроки по ,
>>> parts                        #получим список строк
['43', '44', '45\n']
#  Преобразовать весь список строк в список целых чисел
>>> numbers = [int(P) for P in parts]
>>> numbers
[43,   44,   45]
>>> line = F.readline()
>>> line
"[1,  2,  3]${'a':  1,   'b': 2}\n"
>>> parts = line.split('$')      #  Разбить на строки по символу $
>>> parts
[ ' [1,  2,   3]',   "{'a':  1,   'b':  2}\n"]
>>> eval(parts[0])               #  Преобразовать строку в объект
```

```
[1,   2,   3]
>>> objects = [eval(P) for P in parts] #  То же для всех строк в списке
>>> objects
[[1,  2,  3],   {'a':  1,   'b':  2}]
```
Встроенная функция eval интерпретирует строку как программный код на языке Python (формально - строку, содержащую выражение на языке Python).

Функция print() для записи файлов

Функция print() позволяет не только выводить информацию в терминал, но и записывать ее в файл. Для этого применяется ключевой аргумент file. Основное удобство (по сравнению с функцией write()) заключается в том, что перенос на новую строку осуществляется автоматически.

Пример:
```
with open('article.txt', 'w', encoding='utf-8') as text:
    print('Доброго времени', file=text)
    print('Пора прощаться', file=text)
```

Содержимое двоичных файлов представляется в виде строк типа bytes, и оно передается программе без каких-либо изменений.
```
# Файл открывается в двоичном режиме
>>> data = open('data.bin', 'rb').read()
>>> data                    # Строка байтов хранит двоичные данные
b' \x00\x00\x00\x07spam\x00\x08'
>>> data[4:8]               # Ведет себя как строка
b'spam'
>>> data[4:8][0]# Но в действительности хранит 8-битные целые числа
115
>>> bin(data[4:8][0])       # Функция bin() в Python 3.0
'0b110011'
```

Сохранение объектов Python с помощью модуля pickle

Модуль pickle позволяет сохранять в файлах практически любые объекты Python без необходимости с нашей стороны выполнять какие-либо преобразования.

Пример: сохранить словарь в файле с помощью модуля pickle:
```
>>> D = {'a': 1, 'b': 2}
>>> F = open('datafile.pkl', 'wb')    # открыть в двоичном режиме
>>> import pickle                     # подключаем модуль
>>> pickle.dump(D, F) # Модуль pickle запишет в файл любой объект
>>> F.close()

>>> F = open('datafile.pkl' 'rb')
>>> E = pickle.load(F) # Загружает любые объекты из файла
>>> E
{'a': 1, 'b': 2}
```
Модуль pickle выполняет то, что называется *сериализацией объектов,* – преобразование объектов в строку байтов и обратно.

Модуль shelve – инструмент, который использует модуль pickle для сохранения объектов Python в файлах с доступом по ключу.

Модуль shutil (от англ. "shell utilities" — утилиты командной оболочки)

Вызов shutil.copy (исходный_путь, путь_назначения) приведет к копированию одиночного файла. Если аргумент путь_на значения — это имя файла, то оно будет использовано в качестве нового имени скопированного файла.
```
>>> import shutil, os
```

```
>>> os.chdir('C:\\')
# копируем файл spam.txt, имя копии не меняем
>>> shutil.copy('C:\\spam.txt', 'C:\\delicious')
# копируем файл eggs.txt, имя копии меняем на eggs2.txt
>>> shutil.copy ('eggs.txt', 'C:\\delicious\\eggs2.txt')
```

Функция shutil.copytree (исходный_ путь, путь_назначения) копирует папку вместе со всеми папками и файлами, которые в ней содержатся.

Вызов функции shutil.move (исходный_путь, путь_назначения) перемещает файл или папку.

```
>>> shutil.move('C:\\bacon.txt',   'C:\\eggs')
```

Если в папке C:\eggs уже существует файл bacon.txt, то он будет заменен.

Параметр *путь_назна чения* также может задавать имя файла. Для перемещения и переименования исходного файла:

```
>>> shutil.move('C:\\bacon.txt', 'C:\\eggs\\new bacon.txt')
```

Если в каталоге *C:* не существует папка *eggs,* то функция move () переименует файл *bacon.txt* в файл *eggs.*

Вызов shutil.rmtree (путь) удаляет папку, расположенную по указанному пути, вместе со всеми содержащимися в ней другими папками и файлами.

Задания

Задание 1

- Вывести на экран текущий рабочий каталог.
- На диске D: создать папку «ваше ФИО_rabota4», а в ней папку Exercise1_variant Ваш.
- Сделать эту папку текущей.
- Вывести сообщение: «Создана папка «Exercise1_variant Ваш» в директории …»
- В текущей папке создать файл FIO.txt, в который записать вашу фамилию.
- В текущей папке создать файл age.txt, в который записать возраст, введенный пользователем.
- В папке «ваше ФИО_rabota4» создать папку с именем, которое задаст пользователь. Предусмотреть обработку ошибки, если такая папка уже существует.
- Скопировать файл FIO.txt в папку (с именем пользователя), переименовать его в my_FIO.dat
- Выполнить обход созданного дерева каталогов, вывести информацию на экран.

Задание 2

Найти суммарный размер всех файлов, находящихся в любой вашей папке, воспользовавшись функциями os.path.getsize() и os.listdir().

Задание 3

B1. Пользователь вводит строку. Проверить наличие введенного текста в файле. Если в файле содержится введенная строка, вывести сообщение «Файл содержит искомую строку».
B2. Написать программу, которая считает количество букв в текстовом файле.
B3. Написать программу, которая считает количество строк в текстовом файле.
B4. Напишите программу, которая заменяет все буквы F в текстовом файле на «99».
B5. Определите, есть ли в файле символ '%'. Выведите соответствующее сообщение.
B6. Пользователь вводит свою фамилию, имя, отчество, возраст. Записать все эти данные в файл, каждое с новой строки. Если файл уже существует, записать данные в конец.
B7. Пользователь вводит символ. Посчитать сколько раз этот символ встречается в файле.
B8. Дан файл, в тексте которого используется /. Создать другой файл с таким же содержимым, но заменить знак / четырьмя пробелами.
B9. Прочитать из файла определенные строки, номера которых ввел пользователь, и вывести их на экран.
B10. Файл содержит числа, каждое с новой строки. Посчитайте сумму первого и последнего числа.
B11. Файл содержит числа, каждое с новой строки. Посчитайте сумму всех чисел.
B12. Создайте файл 123.txt, содержащий 5 случайных чисел (от 1 до 20), записанных через пробел. Напишите программу, которая подсчитывает и выводит на экран общую сумму чисел, хранящихся в этом файле.

Задание 4

B1. Записать в файл, имя которого задал пользователь, 10 случайных чисел в диапазоне, указанном пользователем.
B2. Файл fio.dat содержит фамилии (каждая с новой строки). Отсортированное по алфавиту содержимое файла поместите в файл sort_fio.txt.
B3. Пользователь вводит имя создаваемого файла и число X. Создать X файлов, используя введенное имя_номер(от 1 до X).txt. Записать в каждый файл время его создания (год, месяц, день, часы, минуты, секунды).

B4. Напишите функцию read_last(lines, file), которая будет открывать определенный файл file и выводить на печать построчно последние строки в количестве lines (проверить, что задано положительное целое число).
B5. Написать программу, которая считает количество слов в текстовом файле.
B6. Дан текстовый файл, содержащий строку букв и цифр. Посчитать количество цифр.
B7. Файл 1.txt содержит названия городов (каждое с новой строки). Отсортированное по алфавиту содержимое файла поместите в файл sort.txt.
B8. В файле, содержащем фамилии студентов, изменить на прописные буквы фамилии тех студентов, которые начинаются с маленькой буквы.
B9. В файле содержится текстовая строка. Определить частоту повторяемости каждой буквы в тексте и записать букву и ее частоту в новый файл.
B10. В файле содержатся фамилии студентов и их оценки. Создать новый файл, в который записать тех студентов, которые имеют средний балл более «6».
B11. Дан текстовый файл. Создать новый файл, каждая строка которого получается из соответствующей строки исходного файла перестановкой слов в обратном порядке.
Например,

исходный файл:	новый файл:
мама папа я	я папа мама
яблоко компот груша изюм	изюм груша компот яблоко

B12. В данном текстовом файле удалить все слова, которые содержат хотя бы одну цифру.

Задание 5

Алгоритм выполнения задания может быть аналогичен приведенному в примере «модуль_zipfile.pdf».

B1-B6: Написать функцию, которая создаёт резервные копии папки в виде zip-архива.
Имя папки для архивирования (абсолютный путь) передается в функцию в качестве аргумента. Именем для zip-архива будет служить текущая дата и время.
Zip-архив создается в D:\архивы

```
Для задания имени архива можно использовать модуль time:
import time
target_dir = … # задать абсолютный путь к архиву
#формирование имени и  абсолютного пути к будущему архиву
target = target_dir + os.sep + time.strftime('%Y%m%d%H%M%S') + '.zip'
```

os.sep содержит разделитель пути для конкретной операционной системы ('/' в GNU/Linux и Unix3, '\\' в Windows, использование напрямую os.sep вместо этих символов делает программу переносимой).

Функция time.strftime() принимает в качестве аргумента формат вывода времени: символ %Y будет замещён годом и столетием, символ %m будет замещён месяцем и т.д.

B7-B13: Написать функцию, которая создаёт резервные копии папки в виде zip-архива.
Имя папки для архивирования (абсолютный путь) передается в функцию в качестве аргумента. Именем архива будет служить имя папки_версия.zip
Zip-архив создается в D:\копии

Контрольные вопросы

1. Какой функцией возвращаются объекты-файлы?
2. Какое значение используется по умолчанию в аргументе режима обработки файла в функции open?
3. Каким модулем можно воспользоваться для сохранения объектов Python в файл, чтобы избежать выполнения преобразований объектов в строки вручную?
4. Относительно чего задается относительный путь? С чего начинается абсолютный путь?
5. Что собой представляют папки **.** и **..**?
6. Что происходит при открытии существующего файла в режиме записи?
7. Чем различаются методы read () и readlines ()?