Язык программирования

python

Лекция № 7. Работа с файлами (txt, docx, xlsx) Python

Евгений Сергеевич Чухланцев

4.Файлы.

Объекты-файлы – это основной интерфейс между программным кодом на языке Python и внешними файлами на компьютере. Файлы являются одним из базовых типов, но они представляют собой нечто необычное, поскольку для файлов отсутствует возможность создания объектов в виде литералов. Вместо этого, чтобы создать объект файла, необходимо вызвать встроенную функцию ***open***, передав ей имя внешнего файла и строку режима доступа к файлу.

```
>>> f = open('data.txt', 'w') # Создается новый файл для вывода
>>> f.write('Hello\n') # Запись строки байтов в файл
6
>>> f.write('world\n') # Возвращает количество записанных байтов
6
>>> f.close() # Закрывает файл и выталкивает выходные буферы на диск
>>> f = open('data.txt') # 'r' – это режим доступа к файлу по умолчанию
>>> text = f.read()  # Файл читается целиком в строку
>>> text
 Hello world
>>> text.split()  # Содержимое файла всегда является строкой
['Hello', 'world']
```

Добавление 'b' (binary) в строку режима означает работу с бинарными файлами.

4.1. Операции.

Операция	Интерпретация
output = open(r'C:\spam', 'w')	Открывает файл для записи ('a' - запись в конец)
input = open('data', 'r')	Открывает файл для чтения ('+' - чтение и запись)
input = open('data')	Открывает файл для чтения (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 (строки типа str)
open('f.bin', 'rb')	Файлы с двоичными данными (строки типа bytes)

4.2.Примеры.

- Если необходимо вывести содержимое файла, обеспечив правильную интерпретацию символов конца строки, его следует прочитать в строку целиком, с помощью метода *read*, и вывести:

```
>>> open('myfile.txt').read() # Прочитать файл целиком в строку
'hello text file\ngoodbye text file\n'
>>> print(open('myfile.txt').read()) # Более дружественная форма отображения
hello text file
goodbye text file
```

- Если необходимо просмотреть содержимое файла строку за строкой, лучшим выбором будет итератор файла:

```
for line in open('myfile'):
    print(line, end='')
...
hello text file
goodbye text file
```

4.2.Примеры.

Операция чтения двоичных данных из файла возвращает объект типа *bytes* – последовательность целых чисел, представляющих абсолютные значения байтов (которые могут соответствовать символам, а могут и не соответствовать), который во многих отношениях очень близко напоминает обычную строку:

```
>>> data = open('data.bin', 'rb').read() # Открыть двоичный файл для чтения
>>> data   # Строка  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()
'0b1110011'
```

python

4.2.Примеры.

Данные всегда записываются в файл в виде строк, а методы записи не выполняют автоматического форматирования строк:

```
>>> X, Y, Z = 43, 44, 45  # Объекты языка Python должны
>>> S = 'Spam'  # записываться в файл только в виде строк
>>> D = {'a': 1, 'b': 2}
>>> L = [1, 2, 3]
>>> F = open('datafile.txt', 'w')  # Создает файл для записи
>>> F.write(S + '\n')  # Строки завершаются 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 символом $
>>> F.close()
```

Функция автоматического вывода в интерактивной оболочке дает точное побайтовое представление содержимого, а инструкция *print* интерпретирует встроенные символы конца строки, чтобы обеспечить более удобочитаемое отображение:

```
>>> chars = open('datafile.txt').read() # Отображение строки
>>> chars       # в неформатированном виде
"Spam\n43,44,45\n[1, 2, 3]${'a': 1, 'b': 2}\n"
>>> print(chars)   # Удобочитаемое представление Spam
43,44,45
[1, 2, 3]${'a': 1, 'b': 2}
```

4.2.Примеры.

Обратные преобразования для получения из строк в текстовом файле объектов языка Python.

```
>>> F = open('datafile.txt')  # Открыть файл снова
>>> line = F.readline()  # Прочитать одну строку
>>> line
'Spam\n'
>>> line.rstrip()  # Удалить символ конца строки 'Spam' (можно было line[:-1])
>>> line = F.readline()  # Следующая строка из файла
>>> line   # Это - строка
'43,44,45\n'
>>> parts = line.split(',') # Разбить на подстроки по запятым
>>> parts
['43', '44', '45\n']
>>> int(parts[1])  # Преобразовать строку в целое число
44
>>> numbers = [int(P) for P in parts]  # Преобразовать весь список
>>> numbers
[43, 44, 45]
```

4.2.Примеры.

Чтобы преобразовать список и словарь можно воспользоваться встроенной функцией ***eval***, которая интерпретирует строку как программный код на языке Python (формально – строку, содержащую выражение на языке Python):

```
>>> line = F.readline()
>>> line
"[1, 2, 3]${'a': 1, 'b': 2}\n"
>>> parts = line.split('$')  # Разбить строку по символу $
>>> parts
['[1, 2, 3]', "{'a': 1, 'b': 2}\n"]
>>> eval(parts[0])
[1, 2, 3]
>>> objects = [eval(P) for P in parts]  # То же самое для всех строк в списке
 >>> objects
[[1, 2, 3], {'a': 1, 'b': 2}]
```

Функция ***eval*** представляет собой мощный инструмент. И иногда даже слишком мощный. Она без лишних вопросов выполнит любое выражение на языке Python, даже если в результате будут удалены все файлы в компьютере, если передать в выражение соответствующие права доступа! Если вам действительно необходимо извлекать объекты Python из файлов, но вы не можете доверять источнику этих файлов, идеальным решением будет использование модуля ***pickle***, входящего в состав стандартной библиотеки Python.

4.3.Извлечение объектов из файлов с помощью модуля *pickle*.

Модуль ***pickle*** выполняет сериализацию объектов, – преобразование объектов в строку байтов и обратно.

```
#Сохранить словарь в файле:
>>> D = {'a': 1, 'b': 2}
>>> F = open('datafile.pkl', 'wb')
>>> import pickle
>>> pickle.dump(D, F) # Модуль pickle запишет в файл любой объект
>>> F.close()

#Прочитать словарь обратно:
>>> F = open('datafile.pkl' 'rb')
>>> E = pickle.load(F)  # Загружает любые объекты из файла
>>> E
{'a': 1, 'b': 2}
```

Document Title

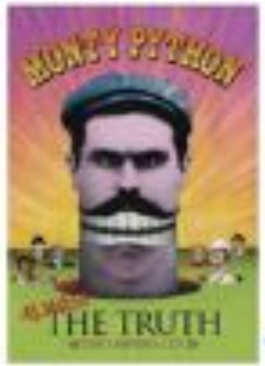
A plain paragraph having some **bold** and some *italic.*

Heading, level 1

Intense quote

- first item in unordered list

1. first item in ordered list



Qty	Id	Desc
1	101	Spam
2	42	Eggs
3	631	Spam, spam, eggs, and spam

```python
from docx import Document
from docx.shared import Inches

document = Document()

document.add_heading('Document Title', 0)

p = document.add_paragraph('A plain paragraph having some ')
p.add_run('bold').bold = True
p.add_run(' and some ')
p.add_run('italic.').italic = True

document.add_heading('Heading, level 1', level=1)
document.add_paragraph('Intense quote', style='IntenseQuote')

document.add_paragraph(
    'first item in unordered list', style='ListBullet'
)
document.add_paragraph(
    'first item in ordered list', style='ListNumber'
)

document.add_picture('monty-truth.png', width=Inches(1.25))

table = document.add_table(rows=1, cols=3)
hdr_cells = table.rows[0].cells
hdr_cells[0].text = 'Qty'
hdr_cells[1].text = 'Id'
hdr_cells[2].text = 'Desc'
for item in recordset:
    row_cells = table.add_row().cells
    row_cells[0].text = str(item.qty)
    row_cells[1].text = str(item.id)
    row_cells[2].text = item.desc

document.add_page_break()

document.save('demo.docx')
```

Как читать и редактировать Excel файлы при помощи openpyxl

Например, в книге с данными, которые вы пытаетесь получить на Python, есть следующие листы:

Sheet 1

	A	B	C
1	ID	AGE	SCORE
2	1	22	5
3	2	15	6
4	3	28	9
5			

Sheet 2

	A	B	C
1	X	Y	Z
2	1	2	3
3	4	5	6
4	7	8	9
5			

Sheet 3

	A	B	C	D
1	M	N	O	P
2	10	11	12	13
3	14	15	16	17
4	18	19	20	21
5				

```
# Import `load_workbook` module from `openpyxl`
from openpyxl import load_workbook
# Load in the workbook
wb = load_workbook('./test.xlsx')
# Get sheet names
print(wb.get_sheet_names())
```

Функция load_workbook () принимает имя файла в качестве аргумента и возвращает объект рабочей книги, который представляет файл. Это можно проверить запуском type (wb).

```
# Get a sheet by name
sheet = wb.get_sheet_by_name('Sheet3')
# Print the sheet title
sheet.title
# Get currently active sheet
anotherSheet = wb.active
# Check `anotherSheet`
anotherSheet
```

Фрагмент кода возвращает имена листов книги, загруженной в Python. Вы можете использовать эту информацию для получения отдельных листов книги. Также вы можете проверить, какой лист активен в настоящий момент с помощью wb.active.

```
# Retrieve the value of a certain cell
sheet['A1'].value
# Select element 'B2' of your sheet
c = sheet ['B2']
# Retrieve the row number of your element
c.row
# Retrieve the column letter of your element
c.column
# Retrieve the coordinates of the cell
c.coordinate
```

На первый взгляд, с этими объектами Worksheet мало что можно сделать. Однако, можно извлекать значения из определенных ячеек на листе книги, используя квадратные скобки [], к которым нужно передавать точную ячейку, из которой вы хотите получить значение.

Атрибут row вернет 2;
Добавление атрибута column к “C” даст вам «B»;
coordinate вернет «B2».

Вы также можете получить значения ячеек с помощью функции cell (). Передайте аргументы row и column, добавьте значения к этим аргументам, которые соответствуют значениям ячейки, которые вы хотите получить, и, конечно же, не забудьте добавить атрибут value

```
# Retrieve cell value
sheet.cell (row=1, column=2).value
# Print out values in column 2
for i in range(1, 4): print (i, sheet.cell (row=i, column=2).value)
```

Например, вы хотите сосредоточиться на области, находящейся между «A1» и «C3», где первый указывает левый верхний угол, а второй — правый нижний угол области, на которой вы хотите сфокусироваться. Эта область будет так называемой cellObj, которую вы видите в первой строке кода ниже.

```
# Print row per row
for cellObj in sheet['A1':'C3']:
    for cell in cellObj:
        print(cells.coordinate, cells.value)
    print('--- END ---')
```

```
('A1', u'M')
('B1', u'N')
('C1', u'O')
--- END ---
('A2', 10L)
('B2', 11L)
('C2', 12L)
--- END ---
('A3', 14L)
('B3', 15L)
('C3', 16L)
--- END ---
```

Чтение и форматирование Excel файлов xlrd

```
# Import `xlrd`
import xlrd
# Open a workbook
workbook = xlrd.open_workbook('example.xls')
# Loads only current sheets to memory
workbook = xlrd.open_workbook('example.xls', on_demand = True)
```

Если вы не хотите рассматривать всю книгу, можно использовать такие функции, как sheet_by_name () или sheet_by_index (), чтобы извлекать листы, которые необходимо использовать в анализе

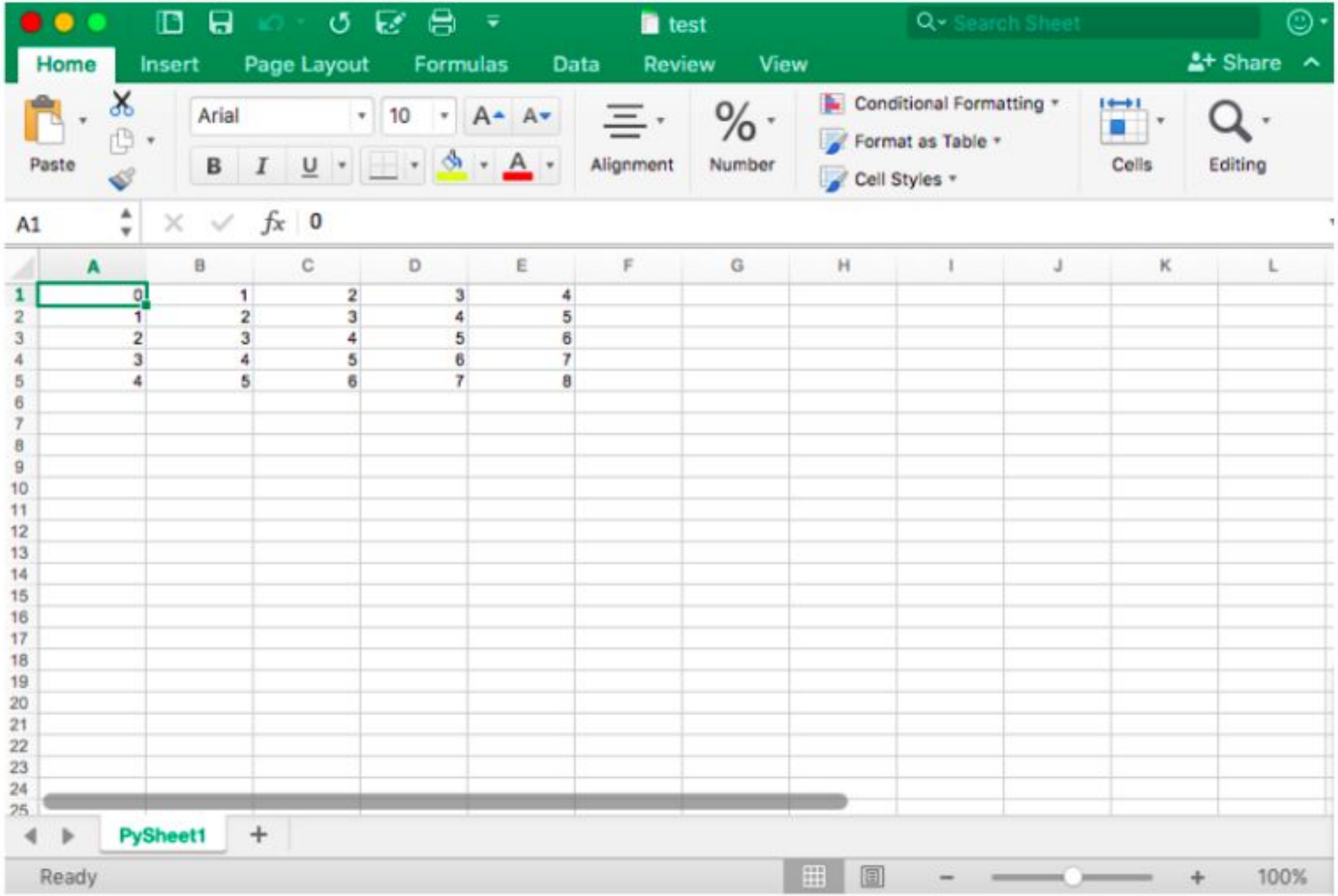
```
# Load a specific sheet by name
worksheet = workbook.sheet_by_name('Sheet1')
# Load a specific sheet by index
worksheet = workbook.sheet_by_index(0)
# Retrieve the value from cell at indices (0,0)
sheet.cell(0, 0).value
```

Когда вы вручную хотите записать в файл, это будет выглядеть так:

```
# Import `xlwt`
import xlwt
# Initialize a workbook
book = xlwt.Workbook(encoding="utf-8")
# Add a sheet to the workbook
sheet1 = book.add_sheet("Python Sheet 1")
# Write to the sheet of the workbook
sheet1.write(0, 0, "This is the First Cell of the First Sheet")
# Save the workbook
book.save("spreadsheet.xls")
```

Запись в файл (автоматический режим), это будет выглядеть так:

```
# Initialize a workbook
book = xlwt.Workbook()
# Add a sheet to the workbook
sheet1 = book.add_sheet("Sheet1")
# The data
cols = ["A", "B", "C", "D", "E"] txt = [0,1,2,3,4]
# Loop over the rows and columns and fill in the values
for num in range(5):
        row = sheet1.row(num)
        for index, col in enumerate(cols):
                value = txt[index] + num row.write(index, value)
# Save the result
book.save("test.xls")
```


test
Search Sheet
Home
Insert
Page Layout
Formulas
Data
Review
View
Share
Paste
Arial
10
Alignment
Number
Conditional Formatting
Format as Table
Cell Styles
Cells
Editing
A1
0
A B C D E F G H I J K L
1 0 1 2 3 4
2 1 2 3 4 5
3 2 3 4 5 6
4 3 4 5 6 7
5 4 5 6 7 8
PySheet1
Ready
100%

Спасибо за внимание !

Домашнее задание

Продолжаем читать книгу: Лутц М. “Изучаем Python" (4-е издание, в 2-х томах) (2011, PDF) !