Библиотеки. Часть 4

План урока

1. Библиотеки для работы с текстами и текстовыми документами
2. Морфология
3. Работа с документами (текст, презентации, таблицы)

Аннотация

Мы уже говорили, что в Python имеется множество библиотек, реализующих абстракции для различных предметных областей. Поэтому мы занимаемся в основном верхним «этажом» логики. Мы уже посмотрели несколько библиотек для работы с массивами числовых данных, теперь поговорим про работу с документами (тексты, презентации, таблицы).

Библиотеки для работы с текстами и текстовыми документами

Поскольку в памяти компьютера хранятся двоичные числа, для записи нечисловой информации (текстов, изображений, видео, документов) прибегают к кодированию. Самый простой случай кодирования — сопоставление кодов текстовым символам. Один самых распространенных форматов такого кодирования — таблица ASCII.

ASCII

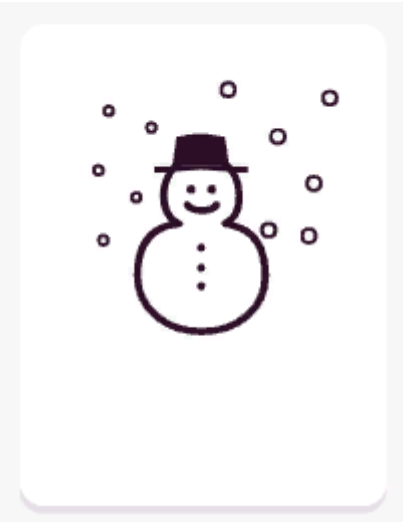
	32	5	53	J	74	_	95	t	116	Й	137	Ю	158	┤	179	╚	200	▌	221	Є	242
!	33	6	54	K	75	`	96	u	117	К	138	Я	159	╡	180	╔	201	▐	222	є	243
"	34	7	55	L	76	a	97	v	118	Л	139	а	160	╢	181	╩	202	▀	223	Ї	244
#	35	8	56	M	77	b	98	w	119	М	140	б	161	╖	182	╦	203	р	224	ї	245
$	36	9	57	N	78	c	99	x	120	Н	141	в	162	╕	183	╠	204	с	225	Ў	246
%	37	:	58	O	79	d	100	y	121	О	142	г	163	╣	184	═	205	т	226	ў	247
&	38	;	59	P	80	e	101	z	122	П	143	д	164	╤	185	╬	206	у	227	°	248
'	39	<	60	Q	81	f	102	{	123	Р	144	е	165	╥	186	╧	207	ф	228	·	249
(	40	=	61	R	82	g	103	|	124	С	145	ж	166	╙	187	╨	208	х	229	∙	250
)	41	>	62	S	83	h	104	}	125	Т	146	з	167	╘	188	╤	209	ц	230	√	251
*	42	?	63	T	84	i	105	~	126	У	147	и	168	╒	189	╥	210	ч	231	№	252
+	43	@	64	U	85	j	106	⌂	127	Ф	148	й	169	╓	190	╙	211	ш	232	¤	253
,	44	A	65	V	86	k	107	А	128	Х	149	к	170	╫	191	╘	212	щ	233	■	254
-	45	B	66	W	87	l	108	Б	129	Ц	150	л	171	└	192	╒	213	ъ	234		255
.	46	C	67	X	88	m	109	В	130	Ч	151	м	172	┴	193	╫	214	ы	235		
/	47	D	68	Y	89	n	110	Г	131	Ш	152	н	173	┬	194	╪	215	ь	236		
0	48	E	69	Z	90	o	111	Д	132	Щ	153	о	174	├	195	┘	216	э	237		
1	49	F	70	[	91	p	112	Е	133	Ъ	154	п	175	─	196	┌	217	ю	238		
2	50	G	71	\	92	q	113	Ж	134	Ы	155	▒	176	┼	197	▄	218	я	239		
3	51	H	72	]	93	r	114	З	135	Ь	156	▓	177	╞	198	▌	219	Ё	240		
4	52	I	73	^	94	s	115	И	136	Э	157	▒	178	╟	199	▄	220	ё	241		

Изначально это была 7-битная кодировка, что позволяло идентифицировать 128 различных символов. Этого хватало на буквы обоих регистров, знаки препинания и спецсимволы — например перевод строки или разрыв страницы. Позже её расширили до 1 байта, что позволяло хранить уже 256 различных значений: в таблицу помещались буквы второго алфавита (например, кириллица) и дополнительные графические элементы (псевдографика).

В некоторых относительно низкоуровневых языках типа C можно в любой момент перейти от представления строки в памяти к последовательности байтов, начинающейся по какому-либо адресу.

Сейчас однобайтные кодировки отошли на второй план, уступив место Юникоду.

Юникод — это таблица, которая содержит соответствия между числом и каким-либо знаком, причем количество знаков может быть любым. Это позволяет одновременно использовать любые символы любых алфавитов, а также дополнительные графические элементы. В Юникод все время добавляются новые элементы, а сам размер этой таблицы не ограничен и будет только расти, поэтому сейчас при хранении в памяти одного юникод-символа может потребоваться от 1 до 8 байт. Отсутствие ограничений привело к тому, что стали появляться символы на все случаи жизни. Например, есть несколько снеговиков.



Этого вы можете увидеть, если наберете:

```
print('\u2603')
```

☃

Важно понять, что все строки в Python хранятся именно как последовательность юникод-символов. Есть парные функции `chr` и `ord`, которые отвечают за соответствие символов и кодов. Мы с вами уже обращались к ним при решении задач курса.

```
ord('Б')
```

```
1041
```

```
chr(1041)
```

```
'Б'
```

Кроме того, в Юникоде каждый символ помимо кода имеет некоторые свойства: например, буква это или цифра. Это позволяет более гибко работать с текстами.

Морфология

Вы уже знаете стандартные строковые функции и пользовались ими. Давайте перейдем на уровень выше. Оказывается, строка и текст состоят из слов, и иногда нужно работать именно со словами, а не просто с последовательностями байтов.

Возьмём для примера склонение существительных с числительными. Например, на форуме надо написать, что в теме «21 комментар**ий**», но «24 комментар**ия**». То же самое нужно делать и для других слов: например, "новость", "пользователь". Иногда на сайтах обходят эту проблему и вставляют машинное «комментариев: 21».

Давайте посмотрим, какие средства работы со словами есть в Python, и познакомимся с библиотекой `pymorphy2` (морфология). Если эта библиотека отсутствует в вашем `Python`, то надо ее установить с помощью утилиты `pip`.

```
import pymorphy2
morph = pymorphy2.MorphAnalyzer()

morph.parse('Ваня')
```

```
[Parse(word='ваня', tag=OpencorporaTag('NOUN,anim,masc,Name sing,nomn'), n
ormal_form='ваня', score=1.0, methods_stack=((<DictionaryAnalyzer>, 'ван
я', 407, 0),))]
```

Для любого слова библиотека делает несколько предположений, что оно может означать, а также обозначает свою уверенность в этом предположении. В данном случае предположение одно с уверенностью `score=1.0`. Итак, мы имеем дело с существительным **NOUN**, именем собственным, одушевленным, мужского рода.

Конечно же предположений может быть несколько:

```
morph.parse('пила')
```

```
[Parse(word='пила', tag=OpencorporaTag('NOUN,inan,femn sing,nomn'), normal
_form='пила', score=0.428571, methods_stack=((<DictionaryAnalyzer>, 'пил
а', 55, 0),)),
 Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,ind
c'), normal_form='пить', score=0.285714, methods_stack=((<DictionaryAnalyz
er>, 'пила', 444, 8),)),
 Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,gent'), n
ormal_form='пил', score=0.142857, methods_stack=((<DictionaryAnalyzer>, 'п
ила', 1124, 1),)),
 Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,accs'), n
ormal_form='пил', score=0.142857, methods_stack=((<DictionaryAnalyzer>, 'п
ила', 1124, 3),))]
```

```
p = morph.parse('пила')[1]
p
```

```
Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,ind
c'), normal_form='пить', score=0.285714, methods_stack=((<DictionaryAnalyz
er>, 'пила', 444, 8),))
```

Обратите внимание на свойство **normal_form**. Например, для глаголов в нем будет храниться инфинитив. Таким образом, можно привести любую форму глагола к единому виду.

За параметры отвечают **теги**.

```
p.tag.POS
```

```
'NOUN'
```

```
p.tag.cyr_repr
```

```
'СУЩ,неод,жр ед,им'
```

Теги бывают (например, для глагола):

```
p.tag.POS           # часть речи
p.tag.animacy       # одушевленность
p.tag.aspect        # вид: совершенный или несовершенный
p.tag.case          # падеж
p.tag.gender        # род (мужской, женский, средний)
p.tag.involvement   # включенность говорящего в действие
p.tag.mood          # наклонение (повелительное, изъявительное)
p.tag.number        # число (единственное, множественное)
p.tag.person        # лицо (1, 2, 3)
p.tag.tense         # время (настоящее, прошедшее, будущее)
p.tag.transitivity  # переходность (переходный, непереходный)
p.tag.voice         # залог (действительный, страдательный)
```

Обозначения для **граммем** (грамматических единиц) можно посмотреть тут (http://pymorphy2.readthedocs.io/en/latest/user/grammemes.html).

Итак, мы можем разбирать большие тексты на части и узнавать информацию о словах, например, искать глаголы, подсчитывать имена и т.д.

Помимо разбора слов, библиотека может их изменять, например, сопоставлять с числами. Давайте посмотрим, как можно решить задачу с подсчетом и выводом количества комментариев на форуме:

```
comment = morph.parse('комментарий')[0]
comment.make_agree_with_number(1).word
```

```
'комментарий'
```

```
comment.make_agree_with_number(2).word
```

```
'комментария'
```

```
comment.make_agree_with_number(7).word
```

```
'комментариев'
```

Интересно, что библиотека пытается работать даже со словами, которых не знает, обращаясь к **оракулу**:

```
word = morph.parse('Мегатрон')[0]
word.make_agree_with_number(7).word
```

```
'мегатронов'
```

```
word.make_agree_with_number(2).word
```

```
'мегатрона'
```

Для склонения существительных можно использовать метод `inflect`:

```
word = morph.parse('случай')[0]
word.inflect({'gent'})
```

```
Parse(word='случая', tag=OpencorporaTag('NOUN,inan,masc sing,gent'), norma
l_form='случай', score=1.0, methods_stack=((<DictionaryAnalyzer>, 'случа
я', 175, 1),))
```

И во множественном числе:

```
word.inflect({'gent', 'plur'})
```

```
Parse(word='случаев', tag=OpencorporaTag('NOUN,inan,masc plur,gent'), norm
al_form='случай', score=1.0, methods_stack=((<DictionaryAnalyzer>, 'случае
в', 175, 7),))
```

Вернемся к параметру **score**.

```
morph.parse('пила')
```

```
[Parse(word='пила', tag=OpencorporaTag('NOUN,inan,femn sing,nomn'), normal
_form='пила', score=0.428571, methods_stack=((<DictionaryAnalyzer>, 'пил
а', 55, 0),)),
 Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,ind
c'), normal_form='пить', score=0.285714, methods_stack=((<DictionaryAnalyz
er>, 'пила', 444, 8),)),
 Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,gent'), n
ormal_form='пил', score=0.142857, methods_stack=((<DictionaryAnalyzer>, 'п
ила', 1124, 1),)),
 Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,accs'), n
ormal_form='пил', score=0.142857, methods_stack=((<DictionaryAnalyzer>, 'п
ила', 1124, 3),))]
```

Мы видим, что предложенные четыре варианта разбора имеют параметр **score**, который говорит нам о том, какой вариант предпочтительнее. Pymorphy2 использует статистические методы и ориентируется на данные проекта OpenCorpora (http://opencorpora.org/) для вычисления значения параметра **score**. Мы не будем останавливаться на данном моменте подробно. Интересующиеся могут прочитать о внутренней кухне на странице документации по Pymorphy2. Скажем только, что эти вычисления не всегда точны.

Работа с документами

Работая с текстами разной направленности в программах, нужно принять во внимание, что тексты иногда хранятся в более сложных форматах, чем .txt. Они могут содержать встроенное форматирование, быть разделенными на страницы, перемежаться с медиаконтентом (графиками, рисунками).

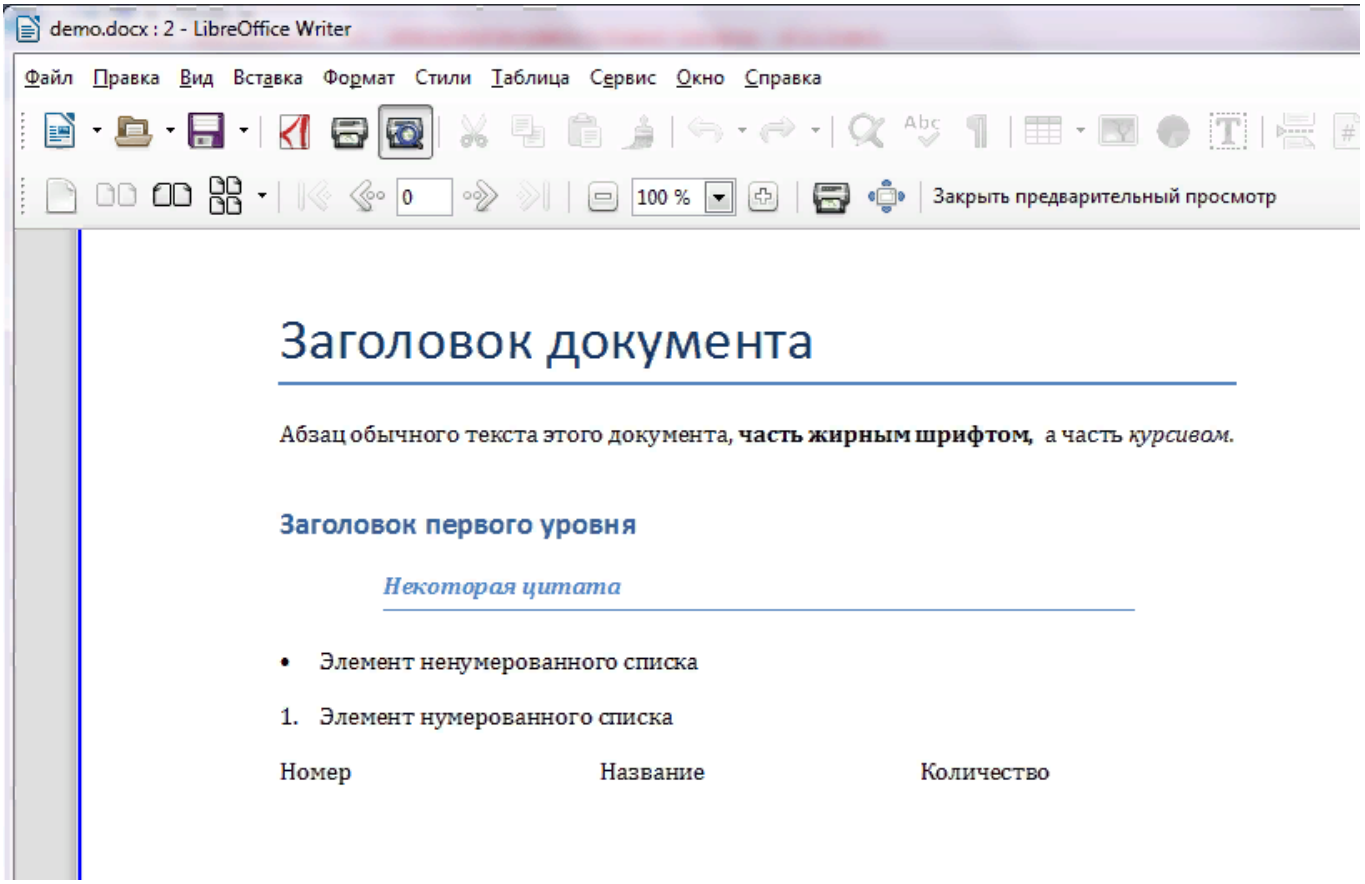
Python умеет работать со многими такими документами. Давайте посмотрим, что можно сделать, чтобы создавать документы в формате **Word**, **Excel** или **PowerPoint** прямо из Python.

Стоит отметить, что форматы docx, xlsx и pptx — открытые, что позволяет разработчикам довольно просто писать библиотеки для работы с ними.

Давайте воспользуемся модулем python-docx для создания docx-документа:

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

document = Document()

document.add_heading('Заголовок документа', 0)

p = document.add_paragraph('Абзац обычного текста этого документа, ')
p.add_run('часть жирным шрифтом, ').bold = True
p.add_run(' а часть ')
p.add_run('курсивом.').italic = True

document.add_heading('Заголовок первого уровня', level=1)
document.add_paragraph('Некоторая цитата', style='Intense Quote')

document.add_paragraph(
    'Элемент ненумерованного списка', style='List Bullet'
)
document.add_paragraph(
    'Элемент нумерованного списка', style='List Number'
)

table = document.add_table(rows=1, cols=3)
hdr_cells = table.rows[0].cells
hdr_cells[0].text = 'Номер'
hdr_cells[1].text = 'Название'
hdr_cells[2].text = 'Количество'

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

Установите при необходимости библиотеку `python-docx` и выполните приведенный код. Затем откройте созданный файл **test.docx**. Вы должны увидеть что-то такое:



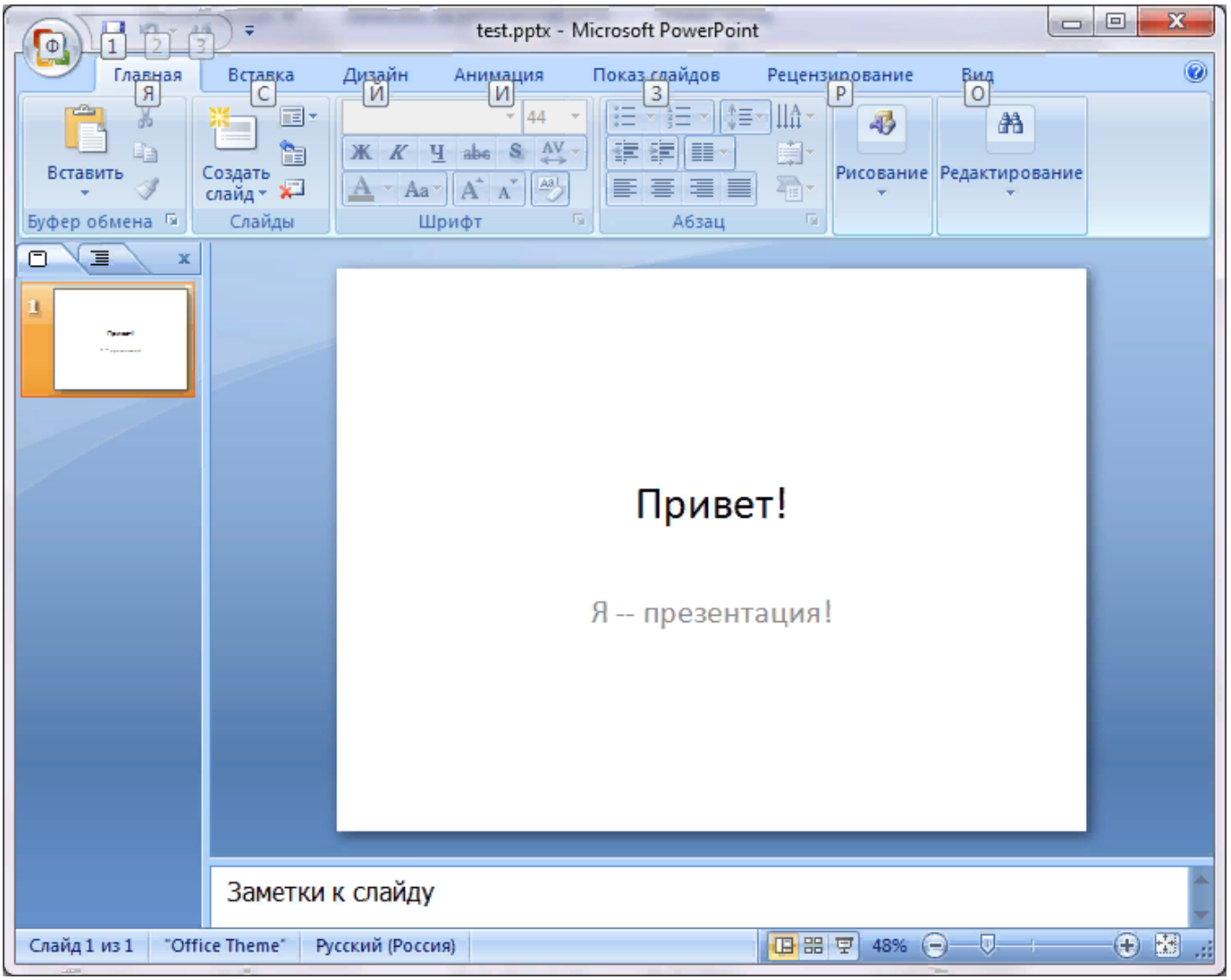
Как мы увидим дальше, работа с подобными модулями примерно одинакова. Все элементы управления и форматирования:

- списки,
- абзацы,
- таблицы,
- ячейки есть в наличии в том или ином виде, их можно изменять и комбинировать.

Давайте посмотрим, как же создавать презентации. Для начала установим библиотеку `python-pptx`. Затем напишем следующий код:

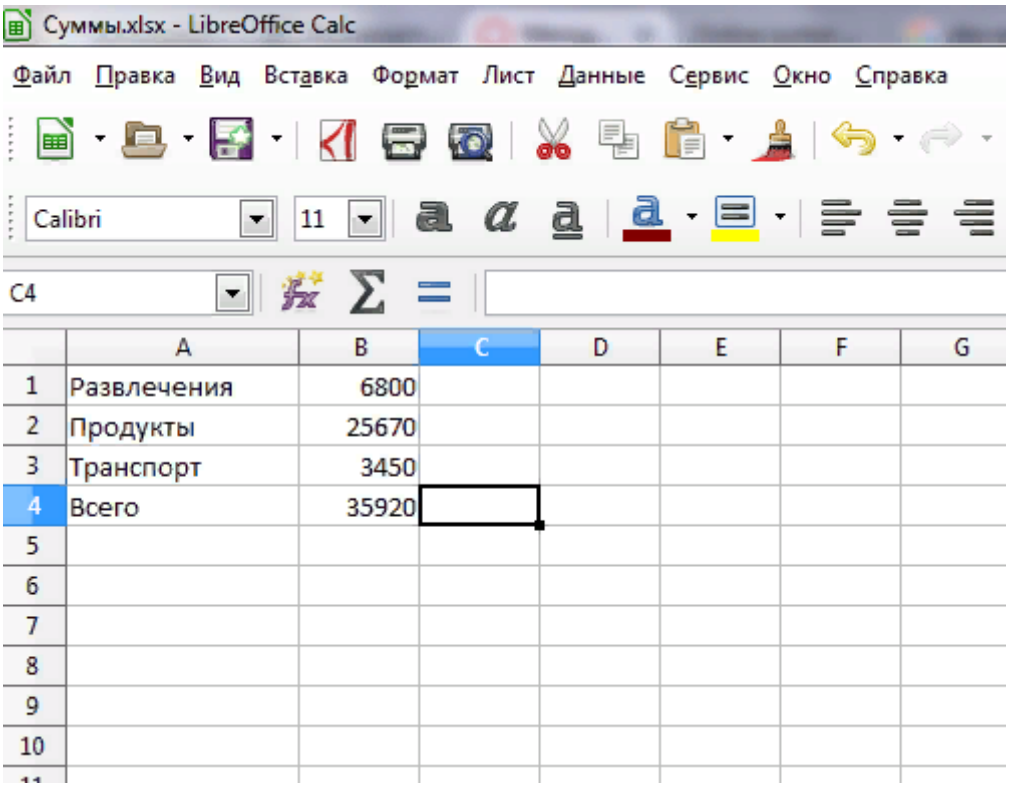
```
from pptx import Presentation

prs = Presentation()

title_slide_layout = prs.slide_layouts[0]
slide = prs.slides.add_slide(title_slide_layout)
title = slide.shapes.title
subtitle = slide.placeholders[1]

title.text = "Привет!"
subtitle.text = "Я -- презентация!"

prs.save('test.pptx')
```

И получится:



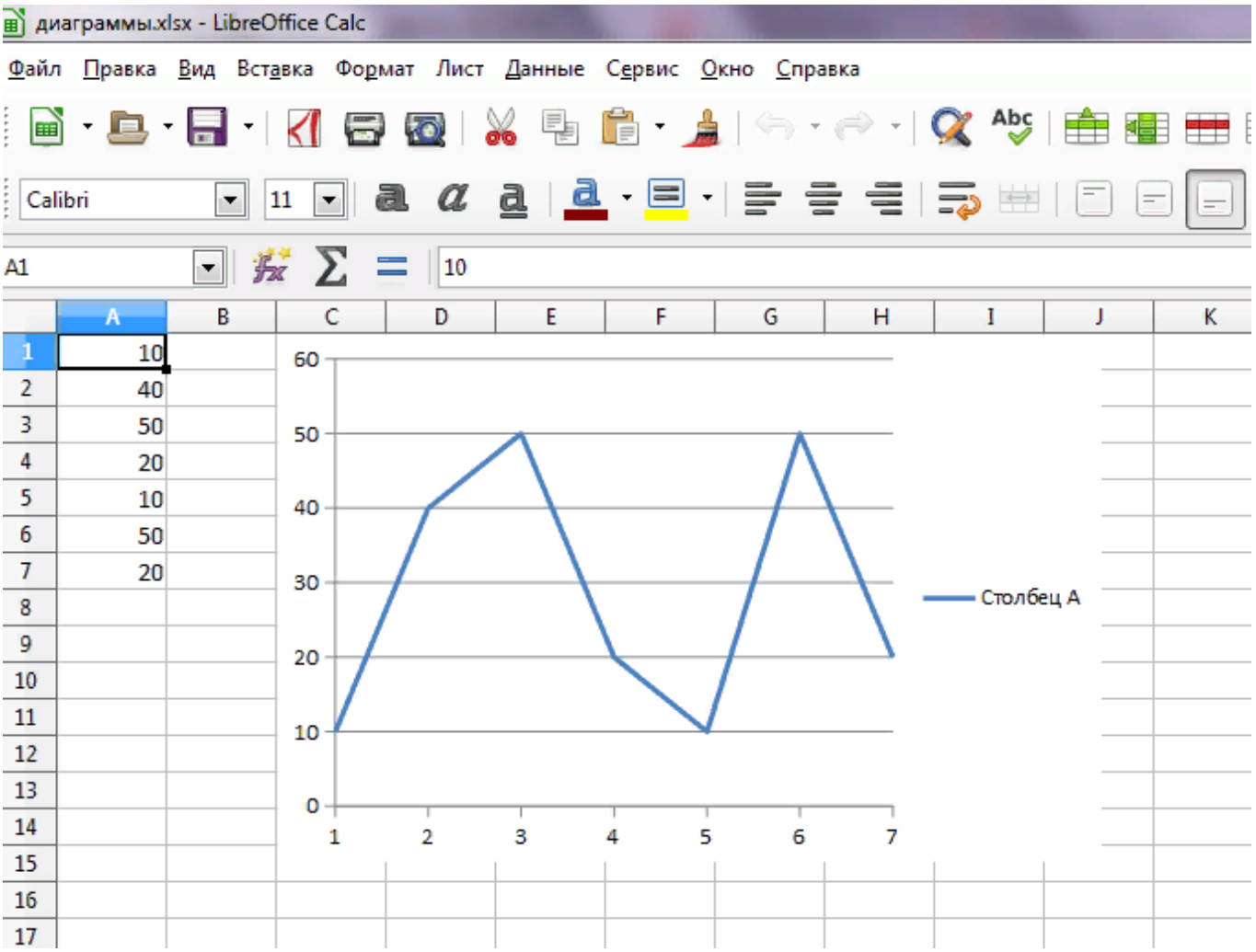
Для Excel-файлов все аналогично. Используется модуль `xlsxwriter`.

```
import xlsxwriter


workbook = xlsxwriter.Workbook('Суммы.xlsx')
worksheet = workbook.add_worksheet()

data = [('Развлечения', 6800), ('Продукты', 25670), ('Транспорт', 3450),]


for row, (item, price) in enumerate(data):
    worksheet.write(row, 0, item)
    worksheet.write(row, 1, price)

row += 1
worksheet.write(row, 0, 'Всего')
worksheet.write(row, 1, '=SUM(B1:B3)')

workbook.close()
```



Мы можем построить, например, диаграмму:

```
import xlsxwriter


workbook = xlsxwriter.Workbook('диаграммы.xlsx')
worksheet = workbook.add_worksheet()

# Данные
data = [10, 40, 50, 20, 10, 50, 20]
worksheet.write_column('A1', data)

# Тип диаграмы
chart = workbook.add_chart({'type': 'line'})

# Строим по нашим данным
chart.add_series({'values': '=Sheet1!A1:A7'})

worksheet.insert_chart('C1', chart)

workbook.close()
```



Эти библиотеки можно использовать для автоматизации документооборота. Например, можно формировать документы: макеты презентаций, налоговую отчетность, открытки с поздравлениями.

Итоги

Мы выяснили, что процесс создания различных документов и работа со словами в Python довольно просты. Сторонние библиотеки позволяют автоматизировать процесс создания документов и упростить работу с морфологией языков.

Мы не рассматривали средства и библиотеки для извлечения знаний и фактов из текстов на естественных языках. Например, по тексту новости (цитируется портал lenta.ru):

> Россиянка Дарья Виролайнен досрочно стала обладательницей Большого Хрустального глобуса, вручаемого победительнице общего зачета Кубка Международного союза биатлонистов (IBU). Об этом сообщается на сайте (http://www.biathlonworld.com/news/detail/alexander-loginov-wins-ibu-cup-pursuit) IBU.
>
> На всех этапах Виролайнен набрала 684 очка, ее соотечественница Анна Никулина, идущая второй, — 526 баллов. На последнем этапе Кубка IBU в Эстонии россиянка не выступит: она вызвана в основной состав сборной России на этап Кубка мира. Тем не менее, Никулина не сумеет догнать ее.
>
> Также Виролайнен стала обладательницей Малого Хрустального глобуса в зачете гонок преследования.
>
> На чемпионате мира по биатлону, прошедшем с 8 по 19 февраля в австрийском Хохфильцене, спортсменка была в составе сборной России, однако не провела ни одной гонки.

Можно узнать, что речь идет о конкретном человеке, странах и датах, что тут упоминается вид спорта и т. д. Со временем вы усвоите и эти возможности библиотек Python.