

Python. Введение 2. Строки, кодировки, файлы.

Емельянов А. А.
login-const@mail.ru

Способ хранения строк

- Кодировка (часто называемая также *кодовой страницей*) – это набор числовых значений, которые ставятся в соответствие группе алфавитно-цифровых символов, знаков пунктуации и специальных символов.
- Символ – минимальная компонента текста.
- Стандарт, в котором перечислены все возможные символы – Unicode (unicode-table.com).

0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	CS	RS	US
0020		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z	{		}	~	¸
0080	xxx	xxx	BPH	NBH	IND	NEL	SBA	ESA	HTS	HTJ	VTB	PLD	PLU	RI	SS2	SS3
0090	DOBS	PU1	PU2	STB	CUH	MW	SPA	SPB	SOS	XXX	SOI	OSI	ST	OGC	PM	APC

- ASCII (American Standard Code for Interchange of Information).
 - В ASCII первые 128 символов всех кодовых страниц состоят из базовой таблицы символов. Первые 32 кода базовой таблицы, начиная с нулевого, размещают управляющие коды. Символы с номерами от 128 до 255 представляют собой таблицу расширения и варьируются в зависимости от набора скриптов, представленных кодировкой символов.
- Кодировки на основе Unicode.
 - utf16, utf32, ucs1, ucs2, ucs4.
 - utf8 (использует переменное количество байт (от 1 до 6)).

- ASCII (American Standard Code for Interchange of Information).
 - В ASCII первые 128 символов всех кодовых страниц состоят из базовой таблицы символов. Первые 32 кода базовой таблицы, начиная с нулевого, размещают управляющие коды. Символы с номерами от 128 до 255 представляют собой таблицу расширения и варьируются в зависимости от набора скриптов, представленных кодировкой символов.
- Кодировки на основе Unicode.
 - utf16, utf32, ucs1, ucs2, ucs4.
 - utf8 (использует переменное количество байт (от 1 до 6)).

```
import sys
import chardet

beer = "🍺 some"
[sys.getsizeof(beer[:index]) for index in range(len(beer) + 1)]
[49, 80, 84, 88, 92, 96, 100]
```

Кодировки в python

- Размер внутреннего представления символа меняется в зависимости от того, какой диапазон номеров символов в строке (ucs1, ucs2, ucs4).

```
import sys
import chardet
```

```
easy = "easy"
изич = "изич"
易易易易 = "易易易易"
```

```
[sys.getsizeof(easy[:index]) for index in range(len(easy) + 1)]
```

```
[49, 50, 51, 52, 53]
```

```
[sys.getsizeof(изич[:index]) for index in range(len(изич) + 1)]
```

```
[49, 76, 78, 80, 82]
```

```
[sys.getsizeof(易易易易[:index]) for index in range(len(易易易易) + 1)]
```

```
[49, 76, 78, 80, 82]
```

```
sys.getsizeof(изич + easy), sys.getsizeof(易易易易 + изич + easy)
```

```
(90, 98)
```

```
chardet.detect(easy.encode()), chardet.detect(изич.encode()), chardet.detect(易易易易.encode())
```

```
{'confidence': 1.0, 'encoding': 'ascii'},
{'confidence': 0.938125, 'encoding': 'utf-8'},
{'confidence': 0.73, 'encoding': 'windows-1252'}
```

- Одинарные или двойные кавычки.

```
: str1 = "think about 'it'"  
str2 = 'синк эбаут "ит"'
```

- Тройные кавычки для строк с переносами.

```
str3 = """想想吧 想想吧 想想吧  
想想吧  
"""
```

- escape-последовательности (не интерпретируются raw-strings)

```
In [49]: raw_str = r"\n \t \" \'"  
print(raw_str)  
  
\\n \\t \\\" \\'
```

Строки и Unicode

- В python встроены экранированные последовательности символов.

```
In [80]: import codecs
```

```
In [88]: u_str = "\u0045\u0041\u0053\u0059"  
print(u_str)  
codecs.encode(u_str)
```

EASY

```
Out[88]: b'EASY'
```

- Для кодирования и декодирования символов используют функции `chr` и `ord`.

```
In [94]: chr(65), ord('A')
```

```
Out[94]: ('A', 65)
```

Функции строк стандартной библиотеки str

- Поиск: `.find`, `.count`, `.index`, `.replace`
- Проверка префикса и постфикса строки: `.startswith`, `.endswith`
- Проверки на то, из каких символов состоит вся строка: `.isalnum`, `.isalpha`, `.isdigit`, `.islower`, `.isspace`, `.istitle`
- Преобразование регистра некоторых (или всех) символов строки: `.lower`, `.upper`, `.capitalize`, `.title`, `.swapcase`
- Часто применяемые функции для работы с файлами: `.strip`, `.split`, `.join`, `.partition`
- Функции для выравнивания чисел: `.ljust`, `.rjust`, `.center`
- Поиск подстроки в строке: `in`, `not in` (работает по алгоритму Boyer-Moore)

Форматирование строк

- Новый стиль:

```
In [185]: name = "Anton"
          second_name = "Emelianov"
          print("fn: {}, sn: {}".format(name, second_name))
```

fn: Anton, sn: Emelianov

```
In [186]: print("fn: {1}, sn: {0}".format(second_name, name))
```

fn: Anton, sn: Emelianov

```
In [188]: print("fn: {name}, sn: {second_name}".format(second_name=second_name, name=name))
```

fn: Anton, sn: Emelianov

```
In [194]: print("left{:>10}".format(10))
          print("{:<10}right".format(10))
          print("left{: ^10}right".format(10))
```

left 10
10 right
left 10 right

— Документация: <https://pyformat.info/>

Форматирование строк

- Старый стиль:

```
In [196]: print("old school %d year in town %s" % (1900, "Moscow"))
```

```
old school 1900 year in town Moscow
```

```
In [198]: print("old school in town %s" % ("Moscow"))
```

```
old school in town Moscow
```

- Документация: <https://pyformat.info/>

Модуль string

- Все маленькие символы латинского алфавита:
 - `.ascii_lowercase`
- Все большие символы латинского алфавита:
 - `.ascii_uppercase`
- Все цифры:
 - `.digits`
- Все буквы латинского алфавита (`ascii_lowercase + ascii_uppercase`):
 - `.ascii_letters`
- Список всех знаков:
 - `.punctuation`
- Список пробельных символов:
 - `.whitespace`

Тип bytes

- Тип bytes представляет собой неизменяемую последовательность байт.

```
byte_str = b"\xd0\xbb\xd0\xb5\xd0\xb3\xd0\xba\xd0\xbe"
```

- Поддерживает кодирование:

```
In [211]: "легко".encode("utf-8")  
Out[211]: b'\xd0\xbb\xd0\xb5\xd0\xb3\xd0\xba\xd0\xbe'
```

- И декодирование:

```
In [213]: byte_str.decode("utf-8")  
Out[213]: 'легко'
```

- Интерпретатор Python поддерживает множество кодировок (~100).

Полезное для кодировок

- decode принимает стратегию обработки ошибок «strict» «replace» или «ignore»
- По умолчанию используется дефолтная кодировка

```
In [81]: import sys  
sys.getdefaultencoding()
```

```
Out[81]: 'utf-8'
```

- Байты не соответствуют последовательности символов

```
In [82]: "b" in b"bytes"
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-82-572365c82a3b> in <module>()  
----> 1 "b" in b"bytes"
```

```
TypeError: a bytes-like object is required, not 'str'
```

- В каждой ОС файл есть шапка файла, отвечающая за машинную информацию о файле и непосредственно данные, сохраненные в нем.

```
In [251]: fin = open("buckets.py", "r")
```

```
In [252]: type(fin)
```

```
Out[252]: _io.TextIOWrapper
```

```
In [253]: fout = open("buckets_copy.py", "w")  
for line in fin.readlines():  
    fout.write(line)
```

```
In [254]: fin.close()  
fout.close()
```

Режимы открытия файлов

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newlines mode (deprecated)

Функции работы с файлами

- Чтение:
 - `.read()`, `.readline()`, `.readlines()`
- Запись:
 - `.write()`, `.writelines()`
- Заккрытие:
 - `.close()`
- Смещение указателя в файле:
 - `.seek`
- Указатель текущей позиции в файле:
 - `.tell()`
- Принудительная запись в файл кэшированных данных:
 - `.flush()`

Менеджеры контекста

1. Выполняется выражение в конструкции `with ... as`.
2. Загружается специальный метод `__exit__` для дальнейшего использования.
3. Выполняется метод `__enter__`. Если конструкция `with` включает в себя слово `as`, то возвращаемое методом `__enter__` значение записывается в переменную.
4. Выполняется `suite`.
5. Вызывается метод `__exit__`, причём неважно, выполнилось ли `suite` или произошло исключение. В этот метод передаются параметры исключения, если оно произошло, или во всех аргументах значение `None`, если исключения не было.

```
with open("buckets.py", "r", encoding="utf-8") as fin:
    for line in fin:
        print(line)
```

Потоки ввода и вывода

- Стандартный поток ввода:
 - `sys.stdin`
- Стандартный поток вывода:
 - `sys.stdout`
- Стандартный канал ошибок:
 - `sys.stderr`

```
In [297]: stdin = sys.stdin.read()
```

```
In [298]: sys.stdout.write("End of lecture! Yeah!")
```

```
End of lecture! Yeah!
```

```
In [299]: sys.stderr.write("Dead")
```

```
Dead
```

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

- *objects* – объекты для печати,
- *sep* – разделитель (строка или *None*),
- *end* – символ, напечатанный в конце вывода (строка или *None*),
- *file* – поток вывода (не работает файлами открытыми в бинарном режиме),
- *flush* – флаг «кэширование вывода».

Домашнее задание 2

- Целью этого задания является знакомство с библиотеками в python для работы со строками, байтами, кодировками.
- Deadline (получение полных баллов): 08.03.2018
- Адрес: login-const@mail.ru
- Задание состоит из двух частей:
 - Написать классификатор текста.
 - Реализовать base64 encode и decode
- Текст условия доступен по [ссылке](#).

СПАСИБО ЗА ВНИМАНИЕ