PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

UNICODE

ЗАЧЕМ ВООБЩЕ НУЖНА КОДИРОВКА?

КОМПЬЮТЕРЫ РАБОТАЮТ С БАЙТАМИ

Мы получаем байты при работе с:

- сетью
- файлами

КОМПЬЮТЕРЫ РАБОТАЮТ С БАЙТАМИ

Для записи символов в байты, нужна определенная договоренность как они будут выглядеть:

- A - 0x41
- F - 0x46

СТАНДАРТ ASCII

ASCII (American standard code for information interchange) - описывает соответствие между символом и его числовым кодом. Изначально описывал только 127 символов:

- коды от 32 до 127 описывали печатные символы
- коды до 32 описывали специальные управляющие символы

СТАНДАРТ ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ISO LATIN 1 (ISO 8859-1)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_																
1_																
2_		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8_																
9_																
A_		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

WINDOWS CP1252

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_		┌	┐	└	┘	│	─	•	◘			♂	□		♫	☼
1_	┼	◀	↕	‼	¶	┴	┬	┤	↑	├	→	←				
2_		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8_	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9_		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
A_		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

UNICODE 🦄

UNICODE

- 1,114,112 кодов 🤦‍♀️
- диапазон 0x0 - 0x10FFFF 🥕
- Стандарт Unicode версии 10.0 (Июнь 2017) определяет 136 690 символов 😇

UNICODE

- U+1F383 JACK-O-LANTERN 🎃
- U+2615 HOT BEVERAGE ☕
- U+1f600 GRINNING FACE 😀

SCHÖN

U+0073 U+0063 U+0068 U+00F6 U+006E

!РЕПУС ЫТ ,EDOCINU

КОДИРОВКИ

- UTF-8
- UTF-16
- UTF-32

UTF-8

- позволяет хранить символы Юникода
- использует переменное количество байт
- символы ASCII обозначаются такими же кодами

UTF-8

H	i			
48	69	01 f6 c0	01 f6 80	26 03

UNICODE В PYTHON 3

STR

STR

Строка в Python 3 - это последовательность кодов Unicode.

```
In [1]: s = 'привет'

In [2]: type(s)
Out[2]: str

In [3]: s.upper()
Out[3]: 'ПРИВЕТ'
```

STR

```
In [4]: hi = '\u043f\u0440\u0438\u0432\u0435\u0442'

In [5]: print(hi)
привет

In [6]: len(hi)
Out[6]: 6
```

ORD

Функция ord возвращает значение кода Unicode для символа:

```
In [7]: ord('п')
Out[7]: 1087

In [8]: hex(ord("a"))
Out[8]: '0x61'
```

CHR

Функция chr возвращает строку Unicode, которая символу, чем код был передан как аргумент:

```
In [9]: chr(1087)
Out[9]: 'п'

In [10]: chr(8364)
Out[10]: '€'

In [11]: chr(9731)
Out[11]: '☃'
```

BYTES

BYTES

```
In [12]: hi_bytes = b"Hello"

In [13]: type(hi_bytes)
Out[13]: bytes

In [14]: hi_bytes.upper()
Out[14]: b'HELLO'

In [15]: hi_bytes.find(b'l')
Out[15]: 2

In [16]: len(hi_bytes)
Out[16]: 5
```

BYTES

Можно работать с байтовыми строками, как с unicode строками:

```
In [17]: d = {b'hi':'Hello', b'by':'Goodbye'}

In [18]: d[b'hi']
Out[18]: 'Hello'

In [19]: d['hi']
---------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-38-259732fc8381> in <module>()
----> 1 d['hi']

KeyError: 'hi'
```

BYTES

```
In [20]: import subprocess

In [21]: result = subprocess.run('ls', stdout=subprocess.PIPE)

In [22]: output = result.stdout

In [23]: output
Out[23]: b'about.md\nacknowledgments.md\nbook\nbook.json\ncourse_presentations\ncourse_pre

In [24]: type(output)
Out[24]: bytes
```

NON ASCII

```
In [25]: test = b'привет'
  File "<ipython-input-39-e8b153ea3e66>", line 1
    test = b'привет'
           ^
SyntaxError: bytes can only contain ASCII literal characters.
```

ENCODE VS DECODE

UNICODE .ENCODE() → BYTES

BYTES .DECODE() → UNICODE

```
In [26]: hi_unicode = 'привет'

In [27]: hi_bytes = hi_unicode.encode('utf-8')

In [28]: hi_bytes
Out[28]: b'\xd0\xbf\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82'

In [29]: len(hi_bytes)
Out[29]: 12

In [30]: hi_bytes.decode('utf-8')
Out[30]: 'привет'
```

ENCODE VS DECODE

```
In [31]: import subprocess

In [32]: result = subprocess.run('ls', stdout=subprocess.PIPE)

In [33]: result.stdout
Out[33]: b'about.md\nacknowledgments.md\nbook\nbook.json\ncourse_presentations\ncourse_pre
```

ENCODE VS DECODE

```
In [34]: output_unicode = result.stdout.decode('utf-8')

In [35]: output_unicode
Out[35]: 'about.md\nacknowledgments.md\nbook\nbook.json\ncourse_presentations\ncourse_pres
```

ENCODE VS DECODE

```
In [36]: result = subprocess.run('ls', stdout=subprocess.PIPE, encoding='utf-8')

In [37]: result.stdout
Out[37]: 'about.md\nacknowledgments.md\nbook\nbook.json\ncourse_presentations\ncourse_pres
```

ENCODE VS DECODE

```
import telnetlib
import time

t = telnetlib.Telnet('192.168.100.1')

t.read_until(b"Username:")
t.write(b'cisco\n')

t.read_until(b"Password:")
t.write(b'cisco\n')
t.write(b'sh ip int br')

time.sleep(5)

output = t.read_very_eager().decode('utf-8')
print(output)
```

ENCODE VS DECODE

```
In [38]: de_hi_unicode = 'grüezi'

In [39]: bytes(de_hi_unicode, encoding='utf-8')
Out[39]: b'gr\xc3\xbcezi'

In [40]: bytes(de_hi_unicode, encoding='utf-16')
Out[40]: b'\xff\xfeg\x00r\x00\xfc\x00e\x00z\x00i\x00'

In [41]: bytes(de_hi_unicode, encoding='utf-32')
Out[41]: b'\xff\xfe\x00\x00g\x00\x00\x00r\x00\x00\x00\xfc\x00\x00\x00e\x00\x00\x00z\x00\x(
```

ОШИБКИ

ОШИБКИ

```
In [42]: hi_unicode = 'привет'

In [43]: hi_unicode.encode('ascii')
---------------------------------------------------------------
UnicodeEncodeError         Traceback (most recent call last)
<ipython-input-211-ec69c9fd2dae> in <module>()
----> 1 hi_unicode.encode('ascii')

UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-5: ordinal not in
```

ОШИБКИ

```
In [44]: de_hi_unicode = 'grüezi'

In [45]: de_hi_unicode.encode('ascii')
---------------------------------------------------------------
UnicodeEncodeError         Traceback (most recent call last)
<ipython-input-216-31c172a5bbb1> in <module>()
----> 1 de_hi_unicode.encode('ascii')

UnicodeEncodeError: 'ascii' codec can't encode character '\xfc' in position 2: ordinal not
```

ОШИБКИ

```
In [46]: hi_unicode = 'привет'

In [47]: hi_bytes = hi_unicode.encode('utf-8')

In [48]: hi_bytes.decode('ascii')
---------------------------------------------------------------
UnicodeDecodeError          Traceback (most recent call last)
<ipython-input-219-aa0ada5e44e9> in <module>()
----> 1 hi_bytes.decode('ascii')

UnicodeDecodeError: 'ascii' codec can't decode byte 0xd0 in position 0: ordinal not in rar
```

ОШИБКИ

```
In [49]: utf_16 = de_hi_unicode.encode('utf-16')

In [50]: de_hi_unicode = 'grüezi'

In [51]: utf_16 = de_hi_unicode.encode('utf-16')

In [52]: utf_16.decode('utf-8')
---------------------------------------------------------------
UnicodeDecodeError          Traceback (most recent call last)
<ipython-input-226-4b4c731e69e4> in <module>()
----> 1 utf_16.decode('utf-8')

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 0: invalid start byte
```

НАДО ЗНАТЬ КАКАЯ КОДИРОВКА ИСПОЛЬЗОВАЛАСЬ

```
In [53]: hi_unicode = 'привет'

In [54]: hi_bytes = hi_unicode.encode('utf-8')

In [55]: hi_bytes
Out[55]: b'\xd0\xbf\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82'

In [56]: hi_bytes.decode('utf-16')
Out[56]: '뿐胑룐닐뗐苑'
```

ОБРАБОТКА ОШИБОК

ОБРАБОТКА ОШИБОК

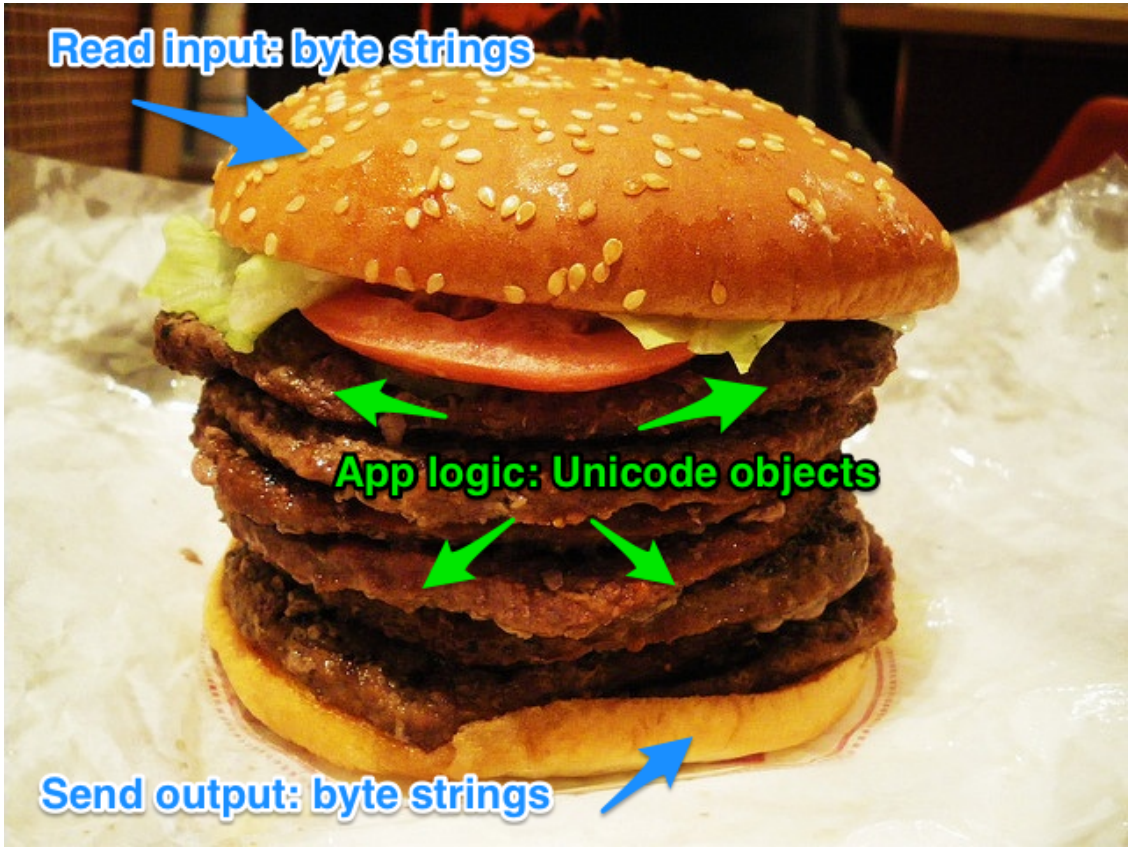
```
In [57]: de_hi_unicode = 'grüezi'

In [58]: de_hi_unicode.encode('ascii', 'replace')
Out[58]: b'gr?ezi'

In [59]: de_hi_unicode.encode('ascii', 'namereplace')
Out[59]: b'gr\\N{LATIN SMALL LETTER U WITH DIAERESIS}ezi'

In [60]: de_hi_unicode.encode('ascii', 'ignore')
Out[60]: b'grezi'
```

ОБРАБОТКА ОШИБОК

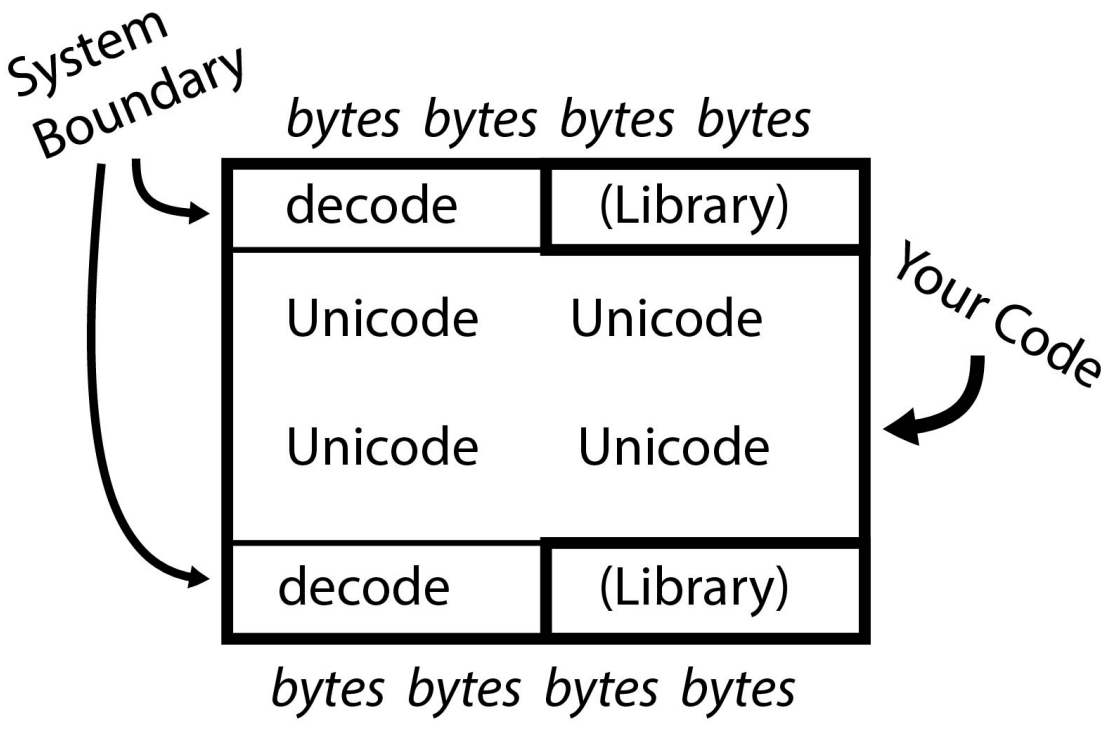
```
In [61]: de_hi_unicode = 'grüezi'

In [62]: de_utf8 = de_hi_unicode.encode('utf-8')

In [63]: de_utf8
Out[63]: b'gr\xc3\xbcezi'

In [64]: de_utf8.decode('ascii', 'ignore')
Out[64]: 'grezi'

In [65]: de_utf8.decode('ascii', 'replace')
Out[65]: 'gr??ezi'
```

UNICODE SANDWICH

UNICODE SANDWICH



UNICODE SANDWICH



РАБОТА С ФАЙЛАМИ

```
In [66]: de_hi_unicode = 'grüezi'

In [67]: f = open('test_unicode.txt', 'w')

In [68]: f.write(de_hi_unicode+'\n')
Out[68]: 7

In [69]: f.close()

In [70]: open('test_unicode.txt').read()
Out[70]: 'grüezi\n'
```

РАБОТА С ФАЙЛАМИ

По умолчанию:

```
In [71]: import locale

In [72]: locale.getpreferredencoding()
Out[72]: 'UTF-8'
```

РАБОТА С ФАЙЛАМИ

```
In [73]: de_hi_unicode = 'grüezi'

In [74]: f = open('test_unicode.txt', 'w', encoding='utf-8')

In [75]: f.write(de_hi_unicode+'\n')
Out[75]: 7

In [76]: f.close()

In [77]: open('test_unicode.txt',  encoding='utf-8').read()
Out[77]: 'grüezi\n'
```

РАБОТА С ФАЙЛАМИ

```
In [78]: file_content = open('test_unicode.txt', 'rb').read()

In [79]: file_content
Out[79]: b'gr\xc3\xbcezi\n'

In [80]: file_content.decode('utf-8')
Out[80]: 'grüezi\n'
```

1. Pragmatic Unicode
2. The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)
3. Unicode HOWTO
4. Standard Encodings