Python2018

compscicenter.ru

aleksey.kladov@gmail.com

Лекция 4

Строки, Байты, IO

```
λ od -t x1 secret
0000000 48 65 6c 6c 6f 2c 20 57 6f 72 6c 64 21 0a
0000016
```

```
λ od -t x1 secret
0000000 48 65 6c 6c 6f 2c 20 57 6f 72 6c 64 21 0a
0000016

λ cat secret
Hello, World!
```

```
>>> bytes([0x48, 0x65, 0x6c, 0x6c, 0x6f]).decode()
'Hello'
>>> list('Hello'.encode())
[72, 101, 108, 108, 111]
```

~1967, ASCII

7 бит = 1 символ

~1987, latin1

1 байт = 1 символ

1991, Unicode 1.0

2 байта = 1 символ

1991, Unicode 1.0

16 bits per character are more than sufficient

1991, Unicode 1.0 / UCS-2

16 bits per character are more than sufficient

1996, Unicode 2.0 / UTF-16

- 1024 high surrogates
- 1024 low surrogates
- 1024 * 1024 дополнительных code points

1996, Unicode 2.0 / UTF-16

- Нельзя быстро получить n-ный символ (character)
- Нельзя быстро посчитать длину строки в символах

UTF-32

4 байта = символ

UTF-8

- совместима с ASCII
- компактна для частых символов
- не random access

Unicode это сложно

```
>>> a = 'e'
>>> b = chr(0x301)
>>> c = 'é'
>>> (a + b, c)
('é', 'é')
>>> a + b == c
False
>>> from unicodedata import normalize
>>> normalize('NFKC', a + b) == c
True
```

Unicode это сложно

```
// Swift
print("🇷🇺".count)

// 1, grapheme clusters
```

Unicode это сложно

```
# Python
print(len("🇷🇺"))

# 2, characters
```

Unicode это сложно

```kotlin
// Kotlin/Java/JavaScript
fun main(args: Array<String>) {
    println("🇷🇺".length)
    // 4, UTF-16 code points
    // две пары surrogate code points
}
```

Unicode это сложно

```rust
// Rust
fn main() {
    println!("{}", "🇷🇺".len())
    // 8, байты в UTF-8
}
```

Кодировки, итоги

- байты и текст -- разные вещи
- для интерпретации надо знать кодировку
- почти все современные данные -- UTF-8
- существенная часть программ -- UTF-16
- текст это сложно

Строки

```
>>> "hello"
'hello'
>>> '"hello"'
'"hello"'
>>> """'"hello"'"""
'\'"hello"\''
>>> ("hello"
...  "world")
'helloworld'
```

```
def foo():
    c_plus_plus = """
        #include <iostream>
        int main() {
            std::cout << "Hello, World!";
        }
    """
    print(c_plus_plus)


foo()
#
#        #include <iostream>
#        int main() {
#            std::cout << "Hello, World!";
#        }
```

```
from textwrap import dedent
def foo():
    c_plus_plus = dedent("""\
        #include <iostream>
        int main() {
            std::cout << "Hello, World!";
        }
    """)
    print(c_plus_plus)


foo()
##include <iostream>
#int main() {
#    std::cout << "Hello, World!";
#}
```

```
>>> '\x0a'
'\n'
>>> '\n'
'\n'
>>> '\x0a'
'\n'
>>> '\u000a'
'\n'
>>> '\N{HEAVY BLACK HEART}'
'❤'
```

```
>>> r"escape from raw string: \n"
'escape from raw string: \\n'
>>> "escape from raw string: \n"
'escape from raw string: \n'
```

```
>>> s = 'hello'
>>> type(s)
<class 'str'>
>>> list(s)
['h', 'e', 'l', 'l', 'o']
>>> type(list(s)[0])
<class 'str'>
```

```
>>> [ord(c) for c in "hello"]
[104, 101, 108, 108, 111] # байт/code point, Latin-1
>>> [ord(c) for c in "привет"] # 2 байта, UCS-2
[1087, 1088, 1080, 1074, 1077, 1090]
>>> [ord(c) for c in "🇷🇺"] # 4 байта, UCS-4
[127479, 127482]
```

Свойства строк

```
>>> "123".isdigit()
True
>>> "foo92".isalnum()
True
>>> "\t \n".isspace()
True
>>> "hello".isascii()
True # >= Python 3.7
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Регистр

```
"foo bar".capitalize() == "Foo bar"
"foo bar".title() == "Foo Bar"
"foo bar".upper() == "FOO BAR"
"foo bar".lower() == "foo bar"
"foo bar".title().swapcase() == ...
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Выравнивание

```
"foo bar".ljust(16, '~') == "foo bar~~~~~~~~~~"
"foo bar".rjust(16)      == "          foo bar"
"foo".center(16)         == "     foo bar     "
"foo".center(2)          == "foo"
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Разбиение

```
>>> "  hello world\n".split()     # *
['hello', 'world']
>>> "  hello world\n".split(" ")  # :(
['', '', 'hello', 'world\n']
>>> "hello\nworld\n".splitlines() # *
['hello', 'world']
>>> "hello\nworld\n".splitlines(keepends=True)
['hello\n', 'world\n']
>>> "hello\nworld\n".split("\n")  # :(
['hello', 'world', '']
>>> first, rest = "foo bar baz".split(maxsplit=1)
>>> rest
'bar baz'
>>> "foo bar baz".rsplit(maxsplit=1)
['foo bar', 'baz']
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Подстроки

```
>>> "foo" in "foobar"
True
>>> "spam" not in "foobar"
True
>>> "foobar".startswith("foo")
True
>>> "foobar".endswith(("boo", "bar"))
True
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Поиск подстрок

```
>>> "abracadabra".find("ra")
2
>>> "abracadabra".find("ra", 0, 3)
-1
>>> "abracadabra".index("ra", 0, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Модификация

```
>>> '1 + 2 + 3 + 4'.replace('+', "*")
'1 * 2 * 3 * 4'
>>> '1 + 2 + 3 + 4'.replace('+', "*", 2)
'1 * 2 * 3 + 4'
>>> "]>>foo bar<<[".lstrip("]>")
'foo bar<<['
>>> "]>>foo bar<<[".strip("[]<>")
'foo bar'
>>> "\t foo bar \r\n ".strip() # *
'foo bar'
```

https://docs.python.org/3/library/stdtypes.html#string-methods

Форматирование

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
...
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
...
```

Форматирование

```
>>> foo = 92
>>> f"foo = {foo}"
'foo = 92'
>>> "foo = {}".format(foo)
'foo = 92'
>>> "foo = %s" % foo
'foo = 92'
```

str / repr

```
>>> str(92)
'92'
>>> str("92")
'92'
>>> repr(92)
'92'
>>> repr("92")
"'92'"
```

Всегда определяйте repr

Опции форматирования

```
>>> s = "hello"
>>> f"{s}"
'hello'
>>> f"{s!r}"
"'hello'"

>>> a = 10
>>> b = 117
>>> f"{a:<4}: foo\n{b:<4}: bar"
10  : foo
117 : bar
>>> f"{a:+^12}"
'+++++10+++++'
```

https://docs.python.org/3/library/string.html#format-string-syntax

Байты

```
>>> hello = b'hello'
>>> type(hello)
<class 'bytes'>
>>> type(hello[0])
<class 'int'>
>>> h = bytearray(hello)
>>> h
bytearray(b'hello')
>>> h.pop()
111
>>> h
bytearray(b'hell')
>>> h.replace(b'l', b'm')
bytearray(b'hemm')
>>> h
bytearray(b'hell')
```

Байты

```
>>> 'hello'.encode(encoding='utf-8')
b'hello'
>>> b'hello'.decode(encoding='utf-8')
'hello'
>>> 'hello мир'.encode('ascii', 'strict')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: ...
>>> 'hello мир'.encode('ascii', 'ignore')
b'hello '
>>> 'hello мир'.encode('ascii', 'replace')
b'hello ???'
```

open

```
>>> help(open)
Help on built-in function open in module io:

open(file, mode='r', buffering=-1, encoding=None,
     errors=None, newline=None,
     closefd=True, opener=None)
    Open file and return a stream.
```

mode

- r -- чтение (по умолчанию)
- t -- декодировать байты в текст (по умолчанию)
- b -- не декодировать
- w -- запись (файл очищается)
- a -- запись в конец
- + -- чтение + запись
- x -- эксклюзивное создание

Текстовые файлы (t)

- Кодировка определяется системой
- -x UTF8 или PYTHONUTF8=1 (>= 3.7)
- universal newlines

file

- bytes или str
- или os.PathLike
- В Unix имя файлы -- произвольные байты
- В Windows -- UTF-16 строка

Как превратить имя файла-строку в байты?

file

- os.fsencode / os.fsdecode
- surrogateescape

```
import os


def list_files(dir):
    return [
        entry.path
        for entry in os.scandir(dir)
        if entry.is_file()
    ]

print(list_files("."))
print(list_files(b"."))
```

file

- Можно использовать файловый дескриптор
- closefd

Методы

```
>>> f = open('foo.txt', 'bw+')
>>> f.write(b'hello')
5 # !
>>> f.seek(0)
0
>>> f.read()
b'hello'
>>> f.seek(0)
0
>>> f.read(2)
b'he'
>>> f.readline()
b'llo'
>>> f.seek(0)
0
>>> list(f)
[b'hello']
```

Стандартные потоки

```
>>> import sys
>>> sys.stdin
<_io.TextIOWrapper mode='r' encoding='UTF-8'>
>>> sys.stderr
<_io.TextIOWrapper mode='w' encoding='UTF-8'>
>>> sys.stdout
<_io.TextIOWrapper mode='w' encoding='UTF-8'>
>>> sys.stdout.buffer
<_io.BufferedWriter name='<stdout>'>
>>> sys.stdout.buffer.write(bytes([92]))
\1
>>> ord('\\')
92
```

Файлы из воздуха

```
>>> import io
>>> handle = io.StringIO("foo\nbar")
>>> handle.readline()
'foo\n'
>>> handle.write("boo")
3
>>> handle.getvalue()
'foo\nboo'
>>> handle = io.BytesIO(b"foobar")
>>> handle.read(3)
b'foo'
```

Почитать в транспорте

PEP 383 -- Non-decodable Bytes in System Character Interfaces

PEP 393 -- Flexible String Representation