Python Prático

Strings x bytes

Sequências de caracteres

- String é uma sequência de caracteres
  - o problema é como se define 'caractere'
- Legado: caractere == byte
  - codificações: ASCII, ISO-8859-1, CP-1252...
- Atualidade: caractere == codepoint Unicode
  - codificações: UTF-8, UTF-16
  - codificação obsoleta: UCS-2

PythonPro

Codepoints Unicode x bytes

- Um codepoint Unicode é um número inteiro de `0` a `1114111` (`0x10ffff`)
- Cada codepoint representa um caractere Unicode
- Um encoding define como cada codepoint é representado em bytes
  - são necessários no mínimo 21 bits (3 bytes) para representar todos os codepoints possíveis

PythonPro

Unicode x UTF-8 encoding

encode

Unicode1	
caractere	codepoint
B	U+0042
Ã	U+00C3
ç	U+00E7
氣	U+6C23

UTF-8 encoding	
bytes	bits
42	01000010
c3 83	11000011 10000011
c3 a7	11000011 10100111
e6 b0 a3	11100110 10110000 10100011

decode

PythonPro

Tipos: Python 2 x Python 3

- Python 2
  - str: bytes (usado também para texto codificado)
    - literal: 'abc'
  - unicode: texto Unicode
    - literal: u'abc'
- Python 3
  - str: texto Unicode
    - literal: 'abc'
  - bytes: dados binários
    - literal: b'abc'

PythonPro

tipo str: Python 2 x Python 3

- Python 2: **`str`** é sequência de bytes
- Python 3: **`str`** é sequência de caracteres Unicode

sequência de...	Python 2		Python 3	
...bytes:	**str**	**'abc'**	bytes	b'abc'
...caracteres Unicode:	unicode	u'abc'	**str**	**'abc'**

PythonPro

Sintaxe: Python 2 x Python 3

- Python 2
  - código-fonte: ASCII por default
  - usar comentário # coding: utf-8
  - identificadores: letras e números ASCII
- Python 3
  - código-fonte: UTF-8 por default
  - identificadores: letras e números Unicode

Demo: de bytes para texto

```
>>> qi_bytes = '\xe6\xb0\xa3'
>>> qi_bytes
'\xe6\xb0\xa3'
>>> print(qi_bytes)
氣
>>> type(qi_bytes)
<type 'str'>
>>> len(qi_bytes)
3
>>> qi = qi_bytes.decode('utf8')
>>> qi
u'\u6c23'
>>> print qi
氣
>>> type(qi)
<type 'unicode'>
>>> len(qi)
1
>>> qi == '氣'
False
>>> qi == u'氣'
True
```

Python 2.7

```
>>> qi_bytes = b'\xe6\xb0\xa3'
>>> qi_bytes
b'\xe6\xb0\xa3'
>>> print(qi_bytes)
b'\xe6\xb0\xa3'
>>> type(qi_bytes)
<class 'bytes'>
>>> len(qi_bytes)
3
>>> qi = qi_bytes.decode('utf8')
>>> qi
'氣'
>>> print(qi)
氣
>>> type(qi)
<class 'str'>
>>> len(qi)
1
>>> qi == '氣'
True
>>> qi == u'氣'
True
```

Python 3.4

from __future__ import unicode_literals

```
>>> from __future__ import unicode_literals
>>> qi = '氣'
>>> type(qi)
<type 'unicode'>
>>> qi
u'\u6c23'
>>> print qi
氣
>>> qi_utf8 = qi.encode('utf8')
>>> qi_utf8
'\xe6\xb0\xa3'
>>> qi2 = '\xe6\xb0\xa3'
>>> qi2
u'\xe6\xb0\xa3'
>>> print qi2
æ°£
>>> qi_utf8 = b'\xe6\xb0\xa3'
>>> qi_utf8
'\xe6\xb0\xa3'
>>> print qi_utf8
氣
>>> type(qi_utf8)
<type 'str'>
```

Python 2.7

- Assume que o literal 'abc' é do tipo unicode