Systemy kodowania

Jolanta Bachan

Bajt

- Bajt – najmniejsza adresowalna jednostka informacji pamięci komputerowej, składająca się z bitów.
- Zwykle przyjmuje się, że jeden bajt to 8 bitów, choć to nie wynika z powyższej definicji. W związku z tym w praktyce jeden bajt może zawierać dowolną liczbę bitów. Aby uniknąć niejednoznaczności, jednostka składająca się z ośmiu bitów zwana jest również *oktetem*.

Wikipedia

ASCII

- ASCII [aski] (ang. American Standard Code for Information Interchange) – 7-bitowy kod przyporządkowujący liczby z zakresu 0-127: literom (alfabetu angielskiego), cyfrom, znakom przestankowym i innym symbolom oraz poleceniom sterującym. Na przykład litera "a" jest kodowana liczbą 97, a znak spacji jest kodowany liczbą 32.
- Ponieważ kod ASCII jest 7-bitowy, a większość komputerów operuje na 8-bitowych bajtach, dodatkowy bit można wykorzystać na powiększenie zbioru kodowanych znaków do 256 symboli. *Wikipedia*

ASCII

- 0-127
- 0-7F
- 128 znaków
- 7-bitowy kod
- 1111111 = 127

1 bajt

- 0-255
- 0-FF
- 256 znaków
- 8-bitowy kod
- 11111111 = 255

Latin-1

- 128-255

UTF-16

- 2 bajty
- początkowo 2^16 = 65536, teraz 1.112.064 znaków
- 0-10FFFF
- UTF-16 (ang. 16-bit Unicode Transformation Format) – w informatyce jeden ze sposobów kodowania znaków standardu Unicode. Sposób ten wymaga użycia szesnastobitowych słów (ang. word), przy czym dla znaków na pozycjach poniżej 65536 (przedział 0000-FFFF) używane jest jedno słowo, którego wartość odpowiada pozycji znaku w standardzie. Dla znaków z wyższych pozycji używa się dwóch słów. *Wikipedia*

Unicode Planes

- The Unicode code space is divided into seventeen planes of 2^16 (65,536) code points each, though some code points have not yet been assigned character values, some are reserved for private use, and some are permanently reserved as non-characters. The code points in each plane have the hexadecimal values xx0000 to xxFFFF, where xx is a hex value from 00 to 10, signifying which plane the values belong to.
- https://en.wikipedia.org/wiki/Plane_(Unicode)

Wikipedia

UTF-8

- UTF-8 – system kodowania Unicode, wykorzystujący od 8 do 32 bitów do zakodowania pojedynczego znaku, w pełni kompatybilny z ASCII. Jest najczęściej wykorzystywany do przechowywania napisów w plikach i komunikacji sieciowej.

Wikipedia

UTF-8 - STANDARD

- 1 bajt
  - 0-7F
  - 0-127
- 2 bajty
  - 0080-07FF
  - 128-2047
- 3 bajty
  - 0800-FFFF
  - 2048-65535
- inne kodowane jako 4 bajty..

Kodowanie na stronach www

- Utwórz plik HTML, wpisz tekst z polskimi znakami, zmień kodowanie pliku oraz kodowania przeglądarki internetowej
- Kodowanie IPA: http://www.phon.ucl.ac.uk/home/wells/ipa-unicode.htm
- Ćwiczenie z θɪŋ
  - Inserting IPA symbols in web document

Unicode in Python

- Zainstaluj pakiet PyICU
  http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyicu
- import unicodedata

Unicode in Python

```
>>> unicodedata.unidata_version
'5.2.0'
>>> unicodedata.category(u'A')  # 'L'etter,
                                # 'u'ppercase
'Lu'
>>> x = unichr(0x1383)
>>> print x
ᎃᎮ
```

Unicode in Python

- ftp://ftp.unicode.org/Public/UNIDATA/UnicodeData.txt

```
u = unichr(233) + unichr(0x0bf2) + unichr(3972) + unichr(6000) + unichr(13231)

for i, c in enumerate(u):
        print i, '%40x' % ord(c), unicodedata.category(c),
        print unicodedata.name(c)

print unicodedata.numeric(u[1])
```

```
0  e9 Ll LATIN SMALL LETTER E WITH ACUTE
1  bf2 No TAMIL NUMBER ONE THOUSAND
2  f84 Mn TIBETAN MARK HALANTA
3  1770 Lo TAGBANWA LETTER SA
4  33af So SQUARE RAD OVER S SQUARED
```

- http://www.fileformat.info/info/unicode/category/index.htm

Unicode in Python

```
>>> a = unichr(0x2160)
>>> print a
 |
>>> unicodedata.name(unichr(0x2160))
'ROMAN NUMERAL ONE'
>>> b = unichr(0x0049)
>>> print b
I
>>> unicodedata.name(b)
'LATIN CAPITAL LETTER I'
```

Unicode in Python

```
>>> unicodedata.name(u'}')
'RIGHT CURLY BRACKET'
>>> unicodedata.name(u'A')
'LATIN CAPITAL LETTER A'
```

Unicode in Python

```
>>> unicodedata.lookup('GEORGIAN CAPITAL LETTER TAR')
u'\u10b2'
>>> x = unichr(0x10b2)
>>> print x
Ჲ
```

Unicode in Python

```
>>> x = unichr(0x006e) +  unichr(0x0334)
>>> print x
n̴
```

Unicode in Python

- set path=%path%;C:\python27

```
# -*- coding: UTF-8  -*-
print('Witaj świecie! €')
u = unichr(0x015b)
print u
x = 'Witaj '+ unichr(0x015b) + 'wiecie!' + unichr(0x00e1)
print x
```

Unicode in Python

- set path=%path%;C:\python27

```
# -*- coding: UTF-8  -*-
tekst = raw_input('Podaj tekst: ')
print tekst
out = open('output.txt', 'w')
out.write(tekst)
out.close()
# Test in Python Shell and on command line
```

Jak działa UTF-8

- *Co słychać? Co sAychaA? Co sĹychaA?*
- ł to U+0142

```
>>> unicodedata.name(unichr(0x0142))
'LATIN SMALL LETTER L WITH STROKE'
```

- UTF-8 ma 2 bajty C5 82
- Skąd wiem, że C5 82 to U+0142?
- C = 1100, 5 = 0101, 8 = 1000, 2 = 0010
- ciąg bitów 11000101 10000010
- „110” na początku mówi, że są 2 bajty
- wyrzuć to i 10 na początku drugiego bajta
- Pozostaje 00101000010... a to jest 0142 w hex!!

Unicode in Python

```
>>> x = unichr(0x0142)
>>> print x
ł
>>> x
u'\u0142'
>>> x.encode('UTF-8')
'\xc5\x82'
>>> print x
ł
```

Do przeczytania

- http://docs.python.org/2/howto/unicode.html