Systemy kodowania

Jolanta Bachan

Problemy z czcionką

W pełni wykorzystaj możliwości swojego konta

Konto HP Smart ułatwia zarządzanie usługami i podłączonymi drukarkami. Podane przez Ciebie informacje osobiste i informacje zgromadzone z jakichkolwiek połączonych przez Ciebie drukarek będą używane do tworzenia konta i świadczenia podstawowych usług.

Dowiedz się więcej

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_{16}, 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 plików

- Utwórz plik tekstowy, wpisz tekst z polskimi znakami, zmień kodowanie pliku.

Unicode in Python

- Zainstaluj Pythona (w domu):

  https://www.python.org/downloads/
- Przetestuj Pythona:

```
>>> print ('Hello world!')
```

Tekst

```
>>> x = "Don't worry"
>>> print (x)
Don't worry
>>> x = 'Don\'t worry'
>>> print (x)
Don't worry
```

```
>>> x = """Don't
worry"""
>>> print (x)
Don't
worry
```

Konkatenacja:

```
>>> x = 'Hello'
>>> y = 'world'
>>> print (x + '   ' + y)
Hello world
```

Pętla for

```
>>> for x in 'hello':
        print (x)
```

```
h
e
l
l
o
```

```
>>> for x in range(5):
        print (x)
```

```
0
1
2
3
4
```

Pętla for + enumerate

```
>>> word = 'Python'
>>> for i, letter in enumerate(word):
        print (i, letter)


0 P
1 y
2 t
3 h
4 o
5 n
```

input()

```
>>> moj_tekst = input("Wpisz tekst: ")
Wpisz tekst: Python jest fajny.
>>> print (moj_tekst)
Python jest fajny.
>>> type(moj_tekst)
<class 'str'>
```

Unicode in Python

```
>>> import unicodedata
>>> unicodedata.unidata_version
'13.0.0'
>>> unicodedata.category(u'A')  # 'L'etter,
                                # 'u'ppercase
'Lu'
>>> x = chr(0x1383)
>>> print (x)
```

Unicode in Python

```
>>> print(chr(954))  # liczba dziesiętna
κ
>>> print(chr(0x3ba)) # liczba dziesiętna 954
                      # w systemie szesnastkowym
κ
>>> print(chr(0x233))
ȳ
>>> print(chr(233))
é
```

Unicode in Python

```
>>> a = chr(0x2160)
>>> print (a)
 Ⅰ
>>> unicodedata.name(a)
'ROMAN NUMERAL ONE'
>>> b = chr(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')
Ჲ
>>> hex(ord(unicodedata.lookup('GEORGIAN CAPITAL LETTER TAR')))
'0x10b2'
>>> print(chr(0x10b2))
Ჲ
# Asomtawruli - najstarsze pismo gruzińskie
https://pl.wikipedia.org/wiki/Pisma_gruzi%C5%84skie
```

Unicode in Python

```
>>> print(chr(0x10E2))
ტ
>>> print( unicodedata.name(chr(0x10E2)))
GEORGIAN LETTER TAR
```

Mchedruli - współczesne pismo gruzińskie

https://pl.wikipedia.org/wiki/Pisma_gruzi%C5%84skie

Unicode in Python

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

Unicode in Python

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

```
u = chr(233) + chr(0x0bf2) + chr(3972) + chr(6000) + chr(13231)
print (unicodedata.numeric(u[1]))


for i, c in enumerate(u):
    print (i, '%5x' % ord(c), unicodedata.category(c), c, unicodedata.name(c))
```

```
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

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 = chr(0x0142)
>>> print (x)
ł
>>> hex(ord(x))
'0x142'
>>> x.encode('UTF-8')
b'\xc5\x82'
>>> print (x)
ł
```

Unicode in Python

```
>>> z = chr(0x0420)
>>> print (z)
P
>>> z.encode('utf8')
'\xd0\xa0'
```

1101 0000 1010 0000

1 0000 10 0000 => 420 (hex)

Unicode in Python

```
>>> x = chr(0x0d82)
>>> print (x)
◌ං
>>> x.encode('utf8')
'\xe0\xb6\x82'
```

1110 0000 1011 0110 1000 0010

0000 11 0110 00 0010 => d82

Do przeczytania

- https://docs.python.org/3/howto/unicode.html