Ekstrakcja informacji

March 6, 2022

[306]:
```
NR_INDEKSU = 434784
```

1 UTF-8

1.1 Kodowanie znaku na bity

Pierwszy znak	Ostatni znak	Bajt 1	Bajt 2	Bajt 3	bajt 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

[307]:
```
c = ' '
```

[308]:
```
ord(c)
```

[308]: 10755

[309]:
```
chr(10755)
```

[309]: ' '

[310]:
```
10755 - 2* 16**3 -  10* 16**2 - 0 * 16**1 - 3* 16**0
```

[310]: 0

$10755_{10} = 2 * 16^3 + 10 * 16^2 + 0 * 16^1 + 3 * 16^0 = \text{U+2A03}$

[311]:
```
10755 - 1*2**13 - 0*2**12 - 1*2**11 - 0*2**10 - 1*2**9 -0*2**8␣
↪-0*2**7-0*2**6-0*2**5-0*2**4-0*2**3-0*2**2-0*2**1  -   1*2**1  - 1*2**0
```

[311]: 0

$10755 = 1 * 2 * *13 - 0 * 2 * *12 - 1 * 2 * *11 - 0 * 2 * *10 - 1 * 2 * *9 - 0 * 2 * *8 - 0 * 2 * *7 - 0 * 2 * *6 - 0 * 2 * *5 - 0 * 2 * *4 - 0 * 2 * *3 - 0 * 2 * *2 - 0 * 2 * *1 - 1 * 2 * *1 - 1 * 2 * *0 = 10101000000011_2$

[312]:
```
len('10101000000011')
```

[312]: 14

[313]:
```
'0010101000000011'
```

[313]: '0010101000000011'

1110xxxx 10xxxxxx 10xxxxxx

11100010 10101000 10000011

[314]:
```
'11100010 10101000 10000011'
```

[314]: '11100010 10101000 10000011'

[315]:
```
!echo ' ' > '01_materialy/znak.txt'
```

[316]:
```
cat '01_materialy/znak.txt'
```

[317]:
```
!xxd -b '01_materialy/znak.txt'
```

00000000: 11100010 10101000 10000011 00001010 …

[318]:
```
chr(2*2+4*2)
```

[318]: '\x0c'

1.1.1 ZADANIE SAMODZIELNE 1 (10 punktów)

Zakoduj poniższe znaki na bity wykonując niezbędne oblicznia

START ZADANIA

[319]:
```
!mkdir -p outputs
```

[320]:
```
c1 = chr(NR_INDEKSU % 100)
c2 = chr(NR_INDEKSU % 1000)
c3 = chr(NR_INDEKSU % 100000 - 123)
print(f'{c1}  {c2}  {c3}')
```

T

[321]:
```
ord(c1)
```

[321]: 84

[322]:
```
print(f'{ord(c1)} - 5 * 16**1 - 4 * 16**0 == 0')
print(f'{ord(c2)} - 3*16**2 - 1*16**1 - 0*16**0 == 0')
print(f'{ord(c3)} - 8*16**3 - 7*16**2 - 6*16**1 - 5*16**0 == 0')
```

```
84 - 5 * 16**1 - 4 * 16**0 == 0
784 - 3*16**2 - 1*16**1 - 0*16**0 == 0
34661 - 8*16**3 - 7*16**2 - 6*16**1 - 5*16**0 == 0
```

$84_{10} = 5 * 16^1 + 4 * 16^0 = \text{U+}54$

$786_{10} = 3 * 16^2 + 1 * 16^1 + 0 * 16^0 = \text{U+}310$

$34661_{10} = 8 * 16^3 + 7 * 16^2 + 6 * 16^1 + 5 * 16^0 = \text{U+}8765$

[323]:
```
print(f'{ord(c1)} - 1*2**6 - 0*2**5 - 1*2**4 -0*2**3 - 1*2**2 - 0*2**1 - 0*2**0␣
  ↪= 0')
print(f'{ord(c2)} - 1*2**9 - 1*2**8 - 1*2**4 = 0')
print(f'{ord(c3)} -1*2**15 - 1*2**10 - 1*2**9 - 1*2**8 - 1*2**6 - 1*2**5 -␣
  ↪1*2**2 - 1*2**0 = 0')
c1_b = str(1010100)
c2_b = str(1100010000)
c3_b = str(1000011101100101)
```

```
84 - 1*2**6 - 0*2**5 - 1*2**4 -0*2**3 - 1*2**2 - 0*2**1 - 0*2**0 = 0
784 - 1*2**9 - 1*2**8 - 1*2**4 = 0
34661 -1*2**15 - 1*2**10 - 1*2**9 - 1*2**8 - 1*2**6 - 1*2**5 - 1*2**2 - 1*2**0 =
0
```

$84_{10} = 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 1010100_2$

$786_{10} = 1 * 2^9 + 1 * 2^8 + 0 * 2^7 + 0 * 2^6 + 0 * 2^5 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 0 * 2^0 = 1100010000_2$

$34661_{10} = 1 * 2^1 5 + 0 * 2^1 4 + 0 * 2^1 3 + 0 * 2^1 2 + 0 * 2^1 1 + 1 * 2^1 0 + 1 * 2^9 + 1 * 2^8 + 0 * 2^7 + 1 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 1000011101100101_2$

[324]:
```
c1_c = '0' + c1_b
c2_c = '1100' + c2_b[0:4] + ' 10' + c2_b[4:]
c3_c = '1110' + c3_b[0:4] + ' 10' + c3_b[4:10] + ' 10' + c3_b[10:16]
print(f'c1 - {c1_c}')
print(f'c2 - {c2_c}')
print(f'c3 - {c3_c}')
```

```
c1 - 01010100
c2 - 11001100 10010000
c3 - 11101000 10011101 10100101
```

[325]:
```
for i in range(len([c1, c2, c3])):
    with open(f'outputs/char{i}.txt', 'w') as f:
        f.write([c1, c2, c3][i])
```

[326]:
```
!xxd -b 'outputs/char0.txt'
print('00000000: ' + c1_c)
```

```
00000000: 01010100                                                T
00000000: 01010100
```

[327]:
```
!xxd -b 'outputs/char1.txt'
print('00000000: ' + c2_c)
```

```
00000000: 11001100 10010000                                       ..
00000000: 11001100 10010000
```

[328]:
```
!xxd -b 'outputs/char2.txt'
print('00000000: ' + c3_c)
```

```
00000000: 11101000 10011101 10100101                              …
00000000: 11101000 10011101 10100101
```

KONIEC ZADANIA

1.1.2 Jakie są zakresy znaków?

[329]:
```
!echo 'zażółć gęślą jaźń' > '01_materialy/polski_tekst.txt'
```

Pierwszy znak	Ostatni znak	Bajt 1	Bajt 2	Bajt 3	bajt 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

[330]:
```
!xxd -b '01_materialy/polski_tekst.txt'
```

```
00000000: 01111010 01100001 11000101 10111100 11000011 10110011  za…
00000006: 11000101 10000010 11000100 10000111 00100000 01100111  … g
0000000c: 11000100 10011001 11000101 10011011 01101100 11000100  …l.
00000012: 10000101 00100000 01101010 01100001 11000101 10111010  . ja..
00000018: 11000101 10000100 00001010                             …
```

1.2 ZADANIE SAMODZIELNE 2 (10 punktów)

Zamień poniższy ciąg binarny na tekst UTF-8. Jeżeli tekst nie zaczyna się od prawidłowego bitu/bitów należy je pominąć.

[331]:
```
```

[332]:
```
tekst = tekst.split(' ')
```

[333]:
```
tekst = ' '.join(tekst[(NR_INDEKSU % 10)*5:(NR_INDEKSU % 10) + 108 ])
```

[334]:
```
tekst
```

[334]:
```
'01111010 01110101 01100011 01101000 01100001 00101100 00001010 01001010
01100001 01101011 00100000 01100010 01111001 11000101 10000010 00100000 01010011
01110100 01100101 01100110 01100101 01101011 00100000 01000010 01110101 01110010
01100011 01111010 01111001 01101101 01110101 01100011 01101000 01100001 11100010
10000000 10100110 00001010 11100010 10000000 10010100 00100000 01001010 01100001
00100000 01101110 01101001 01101011 01101111 01100111 01101111 00100000 01110011
01101001 11000100 10011001 00100000 01101110 01101001 01100101 00100000 01100010
01101111 01101010 11000100 10011001 00100001 00001010 01000011 01101000 01101111
11000100 10000111 01100010 01111001 00100000 01101110 01101001 01100101 01100100
11000101 10111010 01110111 01101001 01100101 01100100 11000101 10111010 11100010
10000000 10100110 00100000'
```

START ZADANIA

[335]:
```
string = bytes([int(c,2) for c in tekst.split(' ')]).decode('utf-8', 'ignore')
string = string.split('\n')
for line in string:
    print(line)
```

```
zucha,
Jak był Stefek Burczymucha…
- Ja nikogo się nie boję!
Choćby niedźwiedź…
```

KONIEC ZADANIA

1.3 ZADANIE SAMODZIELNE 3 (10 punktów)

Zamień poniższy ciąg binarny na tekst UTF-8. Jeżeli tekst nie zaczyna się od prawidłowego bitu/bitów należy je pominąć.

[336]:
```
```

[337]:
```
tekst = tekst.split(' ')
```

[338]:
```
tekst = ' '.join(tekst[(NR_INDEKSU % 10)*5:(NR_INDEKSU % 10) + 108 ])
```

[339]:
```
tekst
```

[339]:
```
'0x69 0x20 0x77 0x20 0x72 0x2e 0x20 0x31 0x36 0x37 0x30 0x2c 0x20 0x70 0x72 0x7a
0x65 0x64 0x20 0x75 0x70 0x61 0x64 0x6b 0x69 0x65 0x6d 0x20 0x4b 0x61 0x6d 0x69
0x65 0xc5 0x84 0x63 0x61 0x20 0x69 0x20 0x68 0x61 0x6e 0x69 0x65 0x62 0x6e 0x79
0x6d 0x69 0x20 0x75 0x6b 0xc5 0x82 0x61 0x64 0x61 0x6d 0x69 0x20 0x62 0x75 0x63
0x7a 0x61 0x63 0x6b 0x69 0x6d 0x69 0x2c 0x20 0x6b 0x74 0xc3 0xb3 0x72 0x65 0x20
0x6f 0x62 0x6f 0x77 0x69 0xc4 0x85 0x7a 0x79 0x77 0x61 0xc5'
```

START ZADANIA

[340]:
```
string = bytes([int(c,16) for c in tekst.split(' ')]).decode('utf-8', 'ignore')
print(string)
```

```
i w r. 1670, przed upadkiem Kamieńca i haniebnymi układami buczackimi, które
obowiązywa
```

KONIEC ZADANIA

1.4 ZADANIE SAMODZIELNE 4 (5 punktów)

Wykonaj następujące operacje w jednym bashowym/shellowym pipelinie:

- scal a.txt i b.txt łącząc je spacją (tak, że pierwszy wiersz a.txt i pierwszy wiersz b.txt są połaczone spacją w nowy wiersz, drugi wiersz a.txt i drugi wiersz b.txt są połaczone spacją w następny wiersz, itp.)
- wyfiltruj tylko linijki gdzie nie ma cyfr z Twojego numeru indeksu
- usuń wszystkie litery ‘a’,
- zamień wszystkie litery ‘c’ na literę ‘d’
- potrój każde wystąpienie litery e (małe i wielkie)
- przestaw kolejność wiersz od końca (ostatni wiersz jest pierwszym, przedostani drugim, itp.)
- wyfiltruj linijki od piątej do szóstej od końca (wg nowej kolejności)
- zapisz pliku c.txt

Następnie wyprintuj zawartość pliku c.txt do tego notebooka

Możesz użyć nastepujących programów: pipe, paste, sed, awk, tr, grep, head, tail, cut, echo, redirect. Nie używaj pythona, perla, ani innych podobnych języków.

START ZADANIA

[341]:
```
!paste '01_materialy/a.txt' '01_materialy/b.txt' -d ' ' | awk '!/[4378]/' | sed␣
↪'s/a//g' | sed 's/c/d/g'| sed 's/E/EEE/g'  | sed 's/e/eee/g' | sed '1!G;h;$!
↪d' | tail -n +5 | head -n -5 > outputs/c.txt
!cat < "./outputs/c.txt"
```

```
 Znk € w Unidodeee m ozndzeeenieee U+20AC.
Chwł tobieee, któryś mnieee zwydiężył!»
«EEEurydyko, porżk jeeest słodk. Kodownieee n podstwieee znku eeeuro €:
 Przykład
6. Porżk itd.
                          Aleee dąb mildzł potężnieeej.
w nieeebieee drżądym jeeeszdzeee… Probleeemu teeego możn byłoby uniknąć, przy
okzji skrdjąd nieeezndznieee wieeelkość dnydh, jeeeśli wykorzystno by zsdę
przeeesunięć typu:
Wteeedy przeeerwł, bowieeem udzuł, żeee mu głos uwięźnieee
Już nieee słoweeem, leee głoseeem w twrdą korę nieeeb 11100000 10000000
10101111itd.
 11000000 10101111
przeeediw tobieee, któreeej kształty dzrno się mrmurzą». 00101111
 t rozpdz w gnieeew urst,  teeen gnieeew jeeest burzą
Chdeeedieee? Błyskwidą dhłostm, seeerd gryzę gromeeem,
Trgnął strunę, bo nieee szeeepteeem śmieeerć mił głudhą przeeemód, 0x10000 do
0x1FFFFF     - bity 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
Jeeeno wod podhwydił to dzułeee weeezwnieee,
żeee nieee będąd jeeeszdzeee szeeepteeem, szeeeptł: «EEEurydyko…» Mpownieee
znków Unidodeee n diągi bjtów:
 w tydh włosdh smuteeek ngły wyląg się tk didho, Sposób kodowni
 żołędzieee meeelodyjnieee trądjąd się wzjeeem, Znki lfbeeetów nieeełdińskidh
zjmują po 2 bjty zmist jeeedneeego w kodownidh nrodowydh.
Wzniósł się w górę, ręką jeeeno dieżr strun odmieeerzł, Wdy
nd brzeeegmi, do jeee więżą, w lirę się wygięły. Jeeest domyślnym kodownieeem w
XML (równieeeż w jeeego plikdjdh: XHTML, SVG, XSL, CML, MthML).
żeee strumieeenieee, do pod zieeemią dieeemno się podzęły, Nieee m probleeemów z
littleee eeendin vs big eeendin.
żeee wzleeediły pond drzeeew kreeety uskrzydloneee, O każdym bjdieee widomo, dzy
jeeest podzątkieeem znku, dzy teeeż leeeży w jeeego środku, do nieee jeeest
dostępneee np. w kodowniu EEEUC.
I pieeeśń podzął. Nieee zwieeer bjtów 0xFF i 0xFEEE, więd łtwo możn go odróżnić
od teeekstu UTF-16.
strunę trądił, już obłoki przeeebudzoneee dzwonią. Ždeeen znk spoz ASCII nieee
zwieeer bjtu z ASCII.
 Zleeety
meee mildzeeenieee jeeest strumieeenieeem,  m pieeeśń liśdist!» Zleeety i wdy
W nieeebo śpieeeweeem rosnę, w zieeemię korzeeenimi wrstm, 5    Linki
zeeewnętrzneee
mnieee pokonsz - śmieeerć przeeemożeeesz, którą zwydiężyłeeem. 2        Sposób
kodowni
```

```
Zeee mną, boski Orfeeeuszu, zmieeerz muzydzną siłę, 1.2 Wdy
```

KONIEC ZADANIA

1.5 ZADANIE SAMODZIELNE 5 (5 punktów)

Napisz funkcję sortującą dla stringów, która będzie działać następująco:

- sortujemy słowa zgodnie z polskim alfabetem
- jeżeli są małe litery i wielkie to wielkie przed małymi
- jeżeli wyraz `x` jest początkiem wyrazu `y`, to wyraz `x` ma być pierwszy

Posortuj poniższy tekst

[342]:
```
przykladowa_lista = ['ą', 'a','b','B', 'cef', 'ce', 'A','Ą', 'ż', ]
```

tak nie chcemy:

[343]:
```
sorted(przykladowa_lista)
```

[343]:
```
['A', 'B', 'a', 'b', 'ce', 'cef', 'Ą', 'ą', 'ż']
```

tak chcemy:

[344]:
```
['A', 'Ą', 'a', 'ą' ,'B', 'b', 'ce', 'cef', 'ż']
```

[344]:
```
['A', 'Ą', 'a', 'ą', 'B', 'b', 'ce', 'cef', 'ż']
```

[345]:
```
tekst = !cat 01_materialy/magiczny-ogrod.txt
tekst = [x for x in ' '.join(tekst).split(' ') if x ][NR_INDEKSU % 1000 :␣
↪NR_INDEKSU % 1000 + 100]
```

START ZADANIA

Version 1 using locale, output doesn't match expected list

[346]:
```
!locale -a
# !sudo locale-gen pl_PL.UTF-8
```

```
C
C.UTF-8
POSIX
en_US.utf8
pl_PL
pl_PL.iso88592
pl_PL.utf8
polish
```

[347]:
```
import locale
locale.setlocale(locale.LC_COLLATE, "pl_PL.UTF-8")
```

```
print(sorted(przykladowa_lista, key = locale.strxfrm))
```

```
['a', 'A', 'ą', 'Ą', 'b', 'B', 'ce', 'cef', 'ż']
```

Version 2, ugly, but works as expected

[348]:
```
alphabet = "AĄaąBbCĆcćDdEĘeęFfGgHhIiJjKkLŁlłMmNŃnńOÓoóPpRrSŚsśTtUuWwYyZŹźŻżż !?.
↪,--_~`@#$%^&*()"
order = dict(zip(alphabet, range(len(alphabet))))
sorted(przykladowa_lista, key=lambda word: [order[c] for c in word])
```

[348]:
```
['A', 'Ą', 'a', 'ą', 'B', 'b', 'ce', 'cef', 'ż']
```

[349]:
```
s = sorted(tekst, key=lambda word: [order[c] for c in word])
for x in s:
    print(x, end=' | ')
```

```
Ayę | a | a | był | Chodź | chodź! | cholera | chorą, | Do | dnia | dnia, | do |
domu. | drugim. | formie | i | i | i | i | i | Jej | jak | jeden | jej | jeszcze
| lament | ludzie | ludzie | Mary | Mary. | małej | marli | mną, | muchy. |
Nastąpiły | Nic | najgroźniejszej | następnego | następnego | nie | nocy | Oto |
o | odwróciwszy | okropne | opanował | ów | Paniczny | padali | pan | po |
popłochu | poranka | potem | powstał | przerażeniu | przez | reszta | rzeczy |
się | się, | służby | służby, | strach | swej | śmierci. | tajemniczość | tego |
troje | trwodze | tym | uciekła. | ukryła | W | w | w | w | w | wbiegła |
wiedziałam! | właśnie | wnętrza | wskutek | wszystkich, | wszystkich, |
wybuchnęła | wytłumaczona | zabrano | zabudowaniach | zamęcie | zapomniana |
zawołała. | ze | zmarło | została | - | - | - | - |
```

KONIEC ZADANIA

1.6 WYKONANIE ZADAŃ

- skopiuj niniejszy notebook
- podmień wartość zmiennej NR_INDEKSU na własny numer indeksu
- zadania wykonaj w tym jupyterze- dodawaj własne komórki tylko miedzy komórkami START ZADANIA, a KONIEC ZADANIA
- Zadania wykonaj tak, żeby po kliknięciu w Kernel → Restart & Run All notebook wykonał się bez błędów
- następnie wygeneruj z notebooka PDF (File → Download As → PDF via Latex).
- notebook z kodem oraz PDF zamieść w zakładce zadań w MS TEAMS