PYTHON – NAPISY – SZYFROWANIE – SORTOWANIE (24)

Praca z łańcuchami tekstowymi (napisami) jest kluczowa podczas obróbki danych.
Dostęp do pojedynczych znaków realizujemy za pomocą indeksowania []
Znaki w łańcuchu zaczynamy zliczać od zera
Bardzo pomocne podczas indeksowania są pętla FOR i polecenie RANGE
Rozszerzone indeksowanie ma postać ***napis[od: do: kolejność]***
Liczby dodatnie oznaczają zwykłą kolejność, liczby ujemne indeksowanie od końca

Indeksowanie (3)

- Do zmiennej OS przypisz swoje nazwisko i imię
- Wpisz i uruchom poniższe polecenia
  komentarzy nie musisz wpisywać

```
OS='Libront Wacław'
print(OS[0])     # pierwszy znak - indeks zero
print(OS[0:3])   # od pierwszego do trzeciego
print(OS[:3])    # trzy pierwsze znaki
print(OS[3:])    # od czwartego do końca
print(OS[3:6])   # od czwartego do szóstego
print(OS[-1])    # ostatni znak
print(OS[-3])    # trzeci od końca
print(OS[-3:])   # trzy ostatnie znaki
print(OS[-3:])   # trzy ostatnie znaki
print(OS[::2])   # co drugi znak
print(OS[::-1])   # napis wspak
print(OS[::-2])   # co drugi znak ale od końca
```

```
L
Lib
Lib
ront Wacław
ron
w
ł
ław
ław
LbotWca
wałcaW tnorbiL
wła nri
```

- Wklej zrzut ekranu z programem i wynikami

Range (3)

Pętla iteracyjna FOR korzysta z polecenia RANGE, które jest listą zawierającą liczby.
Rozszerzone polecenie ma postać ***range(od, do, kolejność)***

- Wpisz i uruchom poniższe polecenia
  komentarzy nie musisz wpisywać

```
for i in range(7):  # [0,1,2,3,4,5,6]
    print(OS[i])    # kolejne 7 znaków napisu OS

dl=len(OS)
for i in range(8,dl):   # [8,9,10,11,12,13]
    print(OS[i])        # od znaku o indeksie 8 do końca

for i in range(6,-1,-1):# [6,5,4,3,2,1,0]
    print(OS[i])        # od znaku o indeksie 6 do początku wspak
```

```
L   t
i W n
b a o
r c r
o ł b
n a i
t w L
```

- Wklej zrzut ekranu z programem i wynikami

Szyfr przestawny (3)

Ten typ szyfrowania polega na tym, że wszystkie znaki tekstu są zapisywane w innej kolejności.
Najprostszym przykładem takiego szyfrowania jest pisanie wspak.

Napisz funkcję, która zamienia miejscami kolejne dwa znaki w napisie.

- Wpisz i uruchom program

```
def PRZESTAWNY(napis):
    t=''
    d=len(napis)
    for i in range(0,d-1,2):
        z1=napis[i]
        z2=napis[i+1]
        t=t+z2+z1
    if d%2==1:
        t=t+napis[-1]
    return t

print(OS)
print(PRZESTAWNY(OS))
```

```
Libront Wacław
iLrbno taWłcwa
```

pętla wybiera co drugi znak od początku bez ostatniego, bo jeżeli napis ma nieparzystą liczbę znaków, to ostatnia pętla spowoduje błąd – nie ma znaku o indeksie [i+1]
dlatego na końcu sprawdzamy, czy napis ma nieparzystą liczbę znaków, za pomocą modulo 2 i doklejamy do wyniku końcowego ostatni znak napisu

- Uruchom program ze swoim nazwiskiem i imieniem
- Wklej zrzut ekranu z programem i wynikami

Szyfr Cezar

Szyfr przestawieniowy, w którym każdy znak w napisie jest zastępowany przez inny, oddalony o stałą liczbę pozycji (tzw. klucz) w zestawie znaków (tzw. alfabecie)
Nazwa szyfru pochodzi od Juliusza Cezara, rzymskiego wodza i polityka. Szyfrował on prywatną korespondencję do swoich przyjaciół, zapisaną po łacinie, używając szyfru przesuwającego z kluczem 3
W typowych zastosowaniach szyfr korzysta z tzw. tablicy kodów ASCII, my posłużymy się alfabetem

Przesuwanie znaków (3)

ALF="AĄBCĆDEĘFGHIJKLŁMNŃOÓPQRSŚTUVWXYZŻŹaąbcćdeęfghijklłmnńoópqrsśtuvwxyzżź "

UWAGA – zmienna ALF jest wpisana w postaci tekstu i będziesz ją mógł wkleić w swój program

- Wpisz funkcję i uruchom program

```
def PRZESUN(znak,klucz):
    ALF="AĄBCĆDEĘFGHIJKLŁMNŃOÓPQRSŚT
    d=len(ALF)
    poz=ALF.find(znak)
    poz=(poz+klucz) % d
    return ALF[poz]

print(OS[0],11,PRZESUN(OS[0],11))
[redacted]
```

```
L 11 Ś
w 11 Ć
```

Funkcja PRZESUN
za pomocą polecenia find znajduje pozycję znaku w napisie ALF
do pozycji dodaje klucz i wykonuje operację modulo
jeżeli nowa pozycja jest dłuższa niż ALF, to zliczanie od początku alfabetu
funkcja zwraca znak z nowej pozycji alfabetu ALF

Wiersz programu wypisuje pierwszą literę zmiennej OS, klucz i przesunięcie tej litery w alfabecie o klucz

- Napisz kolejną instrukcję, która wypisuje:
  - ostatnią literę zmiennej OS,
  - klucz równy Twojemu numerowi z dziennika
  - przesunięcie tej litery o klucz
- Wklej zrzut ekranu z programem i wynikami

Cezar (3)

- Wpisz funkcję i uruchom program

```
def CEZAR(napis,klucz):
    t=""
    for z in napis:
        t=t+PRZESUN(z,klucz)
    return t

print(OS)
print(11)
print(CEZAR(OS,11))
```

```
Libront Wacław
11
ŚqkźxvĄHdiltiĆ
```

Funkcja CEZAR

wybiera w pętli FOR kolejne znaki z napisu
„wyliczony" zostaje nowy znak za pomocą funkcji PRZESUN
nowe znaki sklejane są w łańcuch tekstowy

- Uruchom program ze swoim nazwiskiem i imieniem w zmiennej OS, klucz jest Twoim numer z dziennika
- Wklej zrzut ekranu z programem i wynikami

Sortowanie

Sortowanie, to zadanie polegające na uporządkowaniu zbioru danych względem jakiegoś klucza, na przykład alfabetycznie spisu książek w bibliotece lub według numerów uczniów w dzienniku.
Bez posortowanych danych nie moglibyśmy szybko i sprawnie korzystać na przykład z zasobów Internetu
Przyjmuje się, że prawie połowę czasu pracy komputera zajmuje porządkowanie danych, dlatego też bardzo ważna jest szybkość sortowania i zajętość pamięci, której potrzebuje algorytm sortujący.

Liczby losowe (3)

Do testowania sortowania potrzebne są nieuporządkowane dane. Zamiast wpisywać je ręcznie posłużymy się generatorem liczb losowych.

- Wpisz i uruchom program

```
def LOSUJ(ile, od, do):
    L=[]
    for i in range(ile):
        L=L+[randint(od, do)]
    return L

print(LOSUJ(20,0,9))
```

```
[9, 2, 2, 0, 0, 7, 5, 8, 5, 0, 1, 3, 9, 1, 2, 4, 0, 3, 1, 2]
[10, -3, -4, 5, -7, -9, 1, 6, -3, 1]
```

random *import biblioteki z poleceniem randint – liczba losowa*
L=[] *zmienna L jest pustą listą [nawiasy klamrowe]*
w pętli FOR doklejamy do listy L kolejne wylosowane liczby
po uruchomieniu tworzona jest 20-elementowa lista z losowymi liczbami pomiędzy 0..9

- Napisz kolejną instrukcję, która wypisuje 10 liczb z przedziału -10..10
- Wklej zrzut ekranu z programem i wynikami

Sortowanie bąbelkowe

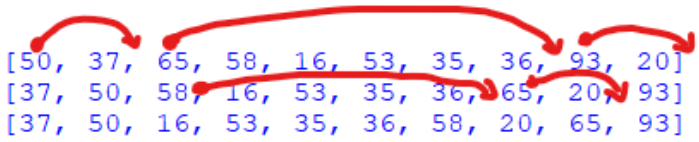
Porównujemy dwa kolejne elementy listy. Jeżeli nie są uporządkowane, to zamieniamy je miejscami.

Bąbelek (3)

- Wpisz i uruchom program

```
def BABEL(L):
    for i in range(len(L)-1):
        if L[i]>L[i+1]:
            L[i],L[i+1]=L[i+1],L[i]
    return L

lista=LOSUJ(10,0,99)
print(lista)
lista=BABEL(lista)
print(lista)
```



Funkcja BABEL przesuwa w prawo każdy większy element niż następny
Na koniec listy przesuwany jest element największy
pętla bez ostatniego elementu
jeżeli pierwszy jest większy od drugiego
to zamiana miejscami dwóch leżących obok siebie elementów listy,

- Napisz instrukcje, które jeszcze raz wykonają „bąbelka” i wypiszą wynik na ekranie
- Wklej zrzut ekranu z programem i wynikami

Sortowanie (3)

- Wpisz i uruchom program

```
def SORTUJ(L):
    for i in range(len(L)):
        L=BABEL(L)
    return L

lista=LOSUJ(10,0,99)
print(lista)
sort=SORTUJ(lista)
print(sort)
```

```
[65, 75, 58, 35, 53, 11, 0, 54, 80, 94]
[0, 11, 35, 53, 54, 58, 65, 75, 80, 94]
```

Funkcja SORTUJ powtarza wiele razy „bąbelka”
przesuwanie na koniec największego elementu

- Wklej zrzut ekranu z programem i wynikami