Laboratorium programistyczne — Python: Napisy i szyfrowanie

Projekt „Matematyka dla Ciekawych Świata”,
Robert Paciorek & Łukasz Mazurek

2020-04-27

1 Napisy

Do tej pory używaliśmy zmiennych do przechowywania liczb i operowania na nich. Zmienne mogą również jako wartości przyjmować litery, słowa, a nawet całe zdania:

```
x = 'A'
a, b, c = 'Ala', "ma", " kota i psa"
d = """ ... a co ma ...
 "kotek"?"""
print(x, a[2])
print(c[1], c[-1], c[-3])
print(a + b)
print(3 * a)
print(a + " " + b + c + d)
```

```
A a
o a p
Alama
AlaAlaAla
Ala ma kota i psa ... a co ma ...
 "kotek"?
```

Zwróć uwagę na następujące rzeczy:

- Napisy muszą być otoczone pojedynczymi apostrofami lub podwójnym cudzysłowami (nie ma znaczenia, którą wersję wybierzemy), w przypadku napisów wieloliniowych używamy trzykrotnie apostrofu lub cudzysłowowa na początku i końcu napisu.
  Nie przypisane do żadnej zmiennej napisy wieloliniowe mogą być stosowane jako komentarze wieloliniowe.
- Przy użyciu liczby w nawiasie kwadratowym możemy poznać poszczególne litery napisu (***numeracja rozpoczyna się od 0***).
- Ujemny indeks oznacza odliczanie liter od końca napisu: ostatnia litera napisu `c` to `c[-1]`, przedostatnia to `c[-2]`, itd.
- Przy użyciu znaku dodawania możemy sklejać (*konkatenować*) napisy.
- Przy użyciu znaku gwiazdki możemy mnożyć napisy (czyli sklejać same ze sobą).

Innymi przydatnymi operacjami na napisach jest sprawdzanie długości napisu poleceniem `len()` oraz wycinanie podnapisu przy użyciu dwukropka:

```
tekst = 'Python'
dlugosc = len(tekst)
print(dlugosc, tekst[2:5], tekst[3:], tekst[:3])
```

```
6 tho hon Pyt
```

W powyższym przykładzie:

- komenda `tekst[2:5]` zwraca podnapis od znaku nr 2 (**włącznie**) do znaku nr 5 (**wyłącznie**),
- komenda `tekst[3:]` zwraca podnapis od znaku nr 3 (**włącznie**) do końca,
- komenda `tekst[:3]` zwraca podnapis od początku do znaku nr 3 (**wyłącznie**).

Podobnie jak w `range()` możemy podać trzeci argument określający przedział czyli krok. Pozwala to na wybieranie co n-tego znaku z napisu, zarówno zaczynając od początku jak i końca:

```
tekst = '123456789'
print(tekst[::2], tekst[1::2])
print(tekst[::-1], tekst[::-3])
print(tekst[::-1][::3], tekst[::3][::-1])
```

```
13579 2468
987654321 963
963 741
```

W powyższym przykładzie:

- komenda `tekst[::2]` zwraca co drugi znak,
- komenda `tekst[1::2]` zwraca co drugi znak od znaku nr 1,
- komenda `tekst[::-1]` zwraca napis od tyłu,
- komenda `tekst[::-3]` zwraca co 3 znak z napisu od tyłu (warto zauważyć że nie zawsze jest to równoważne wypisaniu napisu złożonego z co 3 znaku od tyłu).

1.1 Napis jako lista

Wszystkie listy, których do tej pory używaliśmy w pętli **for** były listami liczb. Okazuje się, że w Pythonie napisy mogą być traktowane jako lista, a dokładniej listą liter. Oznacza to, że po napisie można przejść przy użyciu pętli **for**, tak samo jak przechodziliśmy po liście liczb:

```
for l in 'Abc':
    print('litera', end = ' ')
    print(l)
```

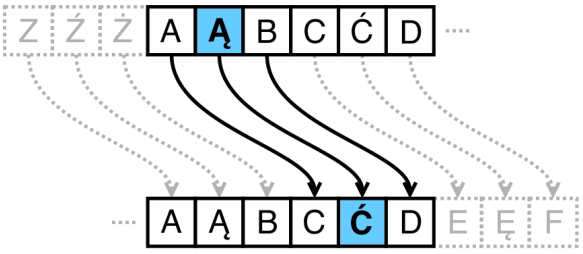
```
litera A
litera b
litera c
```

Zadanie 1.1.1

Napisz funkcję `wyiksuj(napis)`, która zwróci dany `napis`, zastępując każdą małą literę przez `x` i każdą wielką literę przez `X`, natomiast resztę znaków pozostawi bez zmian. Np. dla napisu `'Python 3.6.1 (default, Dec 2015, 13:05:11)'` funkcja powinna zwrócić napis: `Xxxxxx 3.6.1 (xxxxxxx, Xxx 2015, 13:05:11)`

2 Szyfr Cezara

Uwaga *We wszystkich zadaniach dotyczących szyfrowania (o ile nie powiedziano inaczej) zajmujemy się słowami składającymi się wyłącznie z wielkich liter 32-literowego polskiego alfabetu (AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ).*



Szyfr Cezara polega na zastąpieniu każdej litery tekstu jawnego literą występującą w alfabecie o 3 pozycje później — literę A szyfrujemy jako C, Ą szyfrujemy jako Ć, itd. Gdy „skończy nam się alfabet”, zaczynamy liczyć od początku — literę Z szyfrujemy jako A, Ź jako Ą, a Ż jako B. Czyli szyfrogramem dla tekstu jawnego CEZAR jest słowo EGACT, a szyfrogram DCÓCÓ po odszyfrowaniu daje tekst jawny BANAN.

Zadanie 2.0.1

Odszyfruj przy użyciu kartki i długopisu (nie wspomagając się komputerem) tekst jawny, który po zaszyfrowaniu szyfrem Cezara daje szyfrogram ŚŻAC.

2.1 Funkcja `translate`

Szyfr Cezara jest przykładem tzw. szyfru podstawieniowego: każda litera alfabetu jest szyfrowana na inną literę alfabetu zgodnie z pewną *regułą*. Reguła ta może być np. następującej postaci:

- $A \longrightarrow X$
- $B \longrightarrow Y$
- $C \longrightarrow Z$

Taka reguła oznacza, że każda litera A przechodzi na X, każda litera B przechodzi na Y, a każda litera C przechodzi na Z. Regułę taką możemy krócej zapisać w postaci:

$$ABC \longrightarrow XYZ$$

Zgodnie z powyższą regułą, słowo BABA po zaszyfrowaniu przejdzie na YXYX.

Bardzo przydatną funkcją do szyfrowania podstawieniowego jest funkcja `translate` w Pythonie. Aby z niej skorzystać, należy najpierw poleceniem `str.maketrans` stworzyć regułę, według której będziemy szyfrować:

```
> regula = str.maketrans('ABC', 'XYZ')
```

Stworzona powyżej reguła oznacza, że przy szyfrowaniu litera A przejdzie na X, B na Y, natomiast C przejdzie na Z. Mając taką regułę, możemy zaszyfrować dowolne słowo przy użyciu polecenia `translate`:

```
 > 'BABA'.translate(regula)
=> 'YXYX'
```

Zwróć uwagę na składnię polecenia `translate`: najpierw podaje się słowo do zaszyfrowania, następnie po kropce występuje słowo `translate`, a na końcu w nawiasie okrągłym podaje się regułę, według której chcemy zaszyfrować.

Zadanie 2.1.1

Korzystając z poleceń `str.maketrans` i `translate` zaszyfruj szyfrem Cezara słowo:

UZGŹHLFĆAFC

Pamiętaj, że używamy 32-literowego polskiego alfabetu: AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ, a szyfr Cezara zastępuję każdą literę alfabetu literą występującą o 3 pozycje później w alfabecie.

2.2 Szyfr Cezara z kluczem k

W oryginalnym szyfrze Cezara następowało przesunięcie o 3 pozycje w alfabecie. Możemy zajmować się również *szyfrem Cezara z kluczem k*, w którym szyfrogram otrzymuje się przesuwając każdą literę tekstu o k pozycji w alfabecie. Np. dla $k = 5$ szyfr przeprowadza A na D, Ą na E, itd.

Aby zmodyfikować nasz program do szyfrowania dowolnym kluczem, musimy mieć metodę do automatycznego generowania alfabetu „przesuniętego o k pozycji w lewo". Np. dla $k = 5$ chcielibyśmy wygenerować alfabet przesunięty o 5 pozycji w lewo:

DEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻAĄBCĆ

Jednak to zadanie rozwiązaliśmy już na poprzednich zajęciach! Takie przesunięcie nosi nazwę *rotacji* słowa i możemy je wygenerować sklejając z sobą:

- podsłowo powstałe przez usunięcie pierwszych pięciu liter

oraz

- podsłowo złożone z tych pięciu pierwszych liter:

```
> alfabet[5:] + alfabet[:5]
'DEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻAĄBCĆ'
```

Zadanie 2.2.1

Napisz funkcję `cezar(slowo, klucz)`, która wypisze `slowo` zaszyfrowane szyfrem Cezara o kluczu `klucz`. Np. słowo BANAN zaszyfrowane szyfrem Cezara o kluczu 2 brzmi ĆBOBO, więc wywołanie funkcji `cezar` w tym przypadku powinno wyglądać następująco:

```
 > cezar('BANAN', 2)
%* ĆBOBO*)
```

3 Zadania dodatkowe

Zadanie 3.0.1

Pewne słowo zostało zaszyfrowane szyfrem Cezara o kluczu 21 i uzyskano słowo MYŹCFMFUPEN. Jakie to było słowo? Spróbuj rozwiązać to zadanie nie tworząc nowej funkcji do odszyfrowywania, a jedynie używając funkcji `cezar`.

Zadanie 3.0.2

Korzystając z poleceń `str.maketrans` i `translate`, napisz funkcję `powieksz(slowo)`, która dla danego słowa `slowo` składającego się z małych lub wielkich liter polskiego alfabetu, wypisze je ze wszystkimi literami zamienionymi na wielkie. Np. dla słowa McGonagall funkcja powinna zwrócić słowo MCGONAGALL:

```
 > powieksz('McGonagall')
 MCGONAGALL
```

Zadanie 3.0.3

Rozważmy szyfr podstawieniowy według następującej reguły:

$$\text{AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ} \longrightarrow$$
$$\text{SĆTGLOAPNEJBFHŁZIRÓUĄKĘŃŹŻDYMWCŚ}$$

W tym szyfrze A przechodzi na S, Ą przechodzi na Ć, B na T, itd. Pewne polskie słowo s_0 zostało zaszyfrowane tym szyfrem i otrzymano szyfrogram s_1. Następnie słowo s_1 zostało ponownie zaszyfrowane tym samym szyfrem i otrzymano szyfrogram s_2. Następnie zaszyfrowano s_2, aby otrzymać s_3, itd. Łącznie procedurę szyfrowania powtórzono n razy i otrzymano słowo

$$\text{ÓŚZKŃTĆLHMALÓOYE}$$

Jakie było początkowe słowo s_0 i ile razy było zaszyfrowane?

Zadanie 3.0.4

Pewne polskie słowo zostało zaszyfrowane szyfrem Cezara z pewnym kluczem i w wyniku otrzymano szyfrogram ZWEŚŃHĆÓUBĘŚŃK. Przy użyciu pętli `for` i funkcji `cezar` spróbuj rozszyfrować szyfrogram wszystkimi 31 możliwymi kluczami i znajdź słowo, które zostało zaszyfrowane.

Zadanie 3.0.5

Napisz funkcję `szyfruj_liste(lista)`, która wypisze listę słów zaszyfrowaną w następujący sposób: każde słowo będzie zaszyfrowane szyfrem Cezara o kluczu równym długości tego słowa. Np. lista ['ALA', 'MA', 'KOTA'] powinna zostać zaszyfrowana do ['CNC', 'ŃB', 'NSZĆ'] (pierwsze słowo jest zaszyfrowane szyfrem Cezara o kluczu 3, drugie o kluczu 2, a trzecie o kluczu 4). Poszczególne słowa powinny zostać wypisane w oddzielnych wierszach. Przykładowe użycie funkcji powinno wyglądać następująco:

```
> szyfruj_liste(['ALA', 'MA', 'KOTA')
CNC
ŃB
NSZĆ
```

Zadanie 3.0.6

Napisz funkcję `szyfruj_napis(napis, klucz)`, która dla danego `napisu` składającego się z małych i wielkich liter polskiego alfabetu oraz ze spacji, cyfr i znaków specjalnych, wypisze ten napis zaszyfrowany w następujący sposób: każda mała lub wielka litera alfabetu zostanie zaszyfrowana szyfrem Cezara z kluczem `klucz`. Każdy inny znak (spacje, cyfry, znaki specjalne) zostanie wypisany w niezmienionej postaci. Przykładowe użycie funkcji powinno wyglądać następująco:

```
> szyfruj_napis('Polska - Włosi 8:8', 4)
Tsńwnć - Żoswł 8:8
```