Zadanie Litera – LOGIA 17 (2016/17), etap 3

Treść zadania

Napisz jednoparametrową funkcję **litera**, której parametrem jest niepusta lista słów składających się z małych liter alfabetu łacińskiego. Wynikiem funkcji jest ta litera, która występuje najczęściej. Jeśli jest więcej niż jedna taka litera, to wynikiem funkcji jest uporządkowana rosnąco lista liter mających tę własność.

Przykłady:

wynikiem **litera(['ala', 'ma', 'kota'])** jest **'a'**

wynikiem **litera(['julka', 'lubi', 'psy'])** jest **['l', 'u']**

Omówienie rozwiązania

Jest to zadanie typowo narzędziowe, związane z przetwarzaniem napisów. Nie wymaga wiedzy algorytmicznej, poza zliczaniem wystąpień elementów i poszukiwaniem maksimum. Ponieważ mamy 26 liter alfabetu łacińskiego należy na początku utworzyć listę 26 wyzerowanych liczników. Następnie przeglądamy w pętli poszczególne wyrazy, a dla konkretnego wyrazu jego litery i powiększamy odpowiednie liczniki. Następnie należy znaleźć maksimum na liście liczników i policzyć, ile razy ono występuje. Jeśli raz, to wynikiem zadania jest litera odpowiadająca tej wartości (indeks listy zamieniony na literę jej odpowiadającą). Natomiast, jeśli ta sama wartość maksimum występuje więcej razy na liście, należy przejrzeć liczniki i utworzyć listę z literami odpowiadającymi wartościom maksimum.

Alternatywnym rozwiązaniem jest połączenie wyrazów w jeden napis i zliczanie liter występujących w napisie. Rozważania dotyczące znajdowania maksimum pozostają bez zmian.

Rozwiązanie w języku Python

```
1.  def litera(slowa):
2.      liczniki = [0 for i in range(26)]
3.      for slowo in slowa:
4.          for litera in slowo:
5.              liczniki[ord(litera)-97] += 1
6.      m = max(liczniki)
7.      if liczniki.count(m) == 1:
8.          return chr(97+liczniki.index(m))
9.      else:
10.         wynik = []
11.         for i in range(26):
12.             if liczniki[i] == m:
13.                 wynik.append(chr(97+i))
14.         return wynik
```

W wierszu 2 generowana jest lista 26 liczników z wyzerowaną wartością początkową. Pętla w wierszu 3 przegląda słowa listy słów, a wewnętrzna pętla w wierszu 4 litery pojedynczego słowa. W wierszu 5 powiększany jest o jeden licznik odpowiadający danej literze. Ponieważ lista jest indeksowana od 0, od kodu ASCII litery odejmowane jest 97 (kod ASCII małej litery a). W wierszu 6 wyznaczana jest wartość maksimum (wykorzystujemy funkcję max w Pythonie). Jeśli wartość maksimum występuje tylko raz na liście (funkcja count), to wynikiem jest litera o kodzie ASCII równym indeksowi maksimum

powiększonemu o 97. W przeciwnym przypadku budujemy listę liter. Na początku jest ona pusta, następnie w pętli dodawane są kolejne litery, których liczba wystąpień jest równa maksimum. Są one od razu uporządkowane alfabetycznie.

Testy

Należy przeprowadzić testy zarówno dla sytuacji, gdy wynikiem jest pojedyncza litera, jak lista liter. Warto wykonać test, gdy poprawnym wynikiem jest cały alfabet. Słowa do testów powinny być zarówno krótkie (w szczególności jednoliterowe) jak i długie. Także liczba słów w testach powinna być różna. Przykładowe testy:

```
litera(['abc', 'bac', 'cab'])
```

wynik: `['a', 'b', 'c']`

```
litera(['bacdefghijklmnopqrstuvwxyz', 'aeouy', 'zyxwvutsrqponmlkjihgfedcab', 'diefghij'])
```

wynik: `['e', 'i']`

```
litera(['cabdefghijklmnopqrstuvwxyz','zyxwvutsrqponmlkjihgfedbac','acegikmoqsuwybdfhjlnprtvxz'])
```

wynik: `['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']`

```
litera(['d', 'b', 'c', 'a', 'ddd', 'eeu', 'eu', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'fug', 'hiju', 'klmno', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'])
```

wynik: `'u'`