19 października 2022

Kurs rozszerzony języka Python

Lista 2.

Każde zadanie jest warte 2 punkty. Na pracowni do oceny należy przedstawić trzy zadania.

Zadanie 1.

Podczas wyborów powszechnych do parlamentu czy władz samorządowych mandaty pomiędzy poszczególne partie czy komitety wyborcze dzieli się *metodą d'Hondta*. Zaprogramuj funkcję, która jako argument przyjmuje *wynik wyborów* i liczbę miejsc do obsadzenia, a zwraca listę wybranych osób. Przyjmujemy, że

- szczegóły *metody d'Hondta* są takie jak opisane w `https://pl.wikipedia.org/wiki/Metoda_D%E2%80%99Hondta`;
- próg wyborczy to 5% (nie uwzględniamy komitetów wyborczych mniejszości narodowych czy etnicznych, których nie obowiązuje próg wyborczy).

Wynik wyborów to struktura danych pamiętająca liczbę głosów oddanych na poszczególne osoby, wraz z komitetami wyborczymi do jakich należą te osoby.

Jako przykład poszukaj wyników wyborów (np. `https://pkw.gov.pl`) i sprawdź, czy podział mandatów uzyskany przez Ciebie zgadza się z podziałem podanym przez Państwową Komisję Wyborczą.

Zadanie 2.

Korzystając ze wzoru

$$\sum_{i=1}^{k}(2i-1) = k^2$$

zaprogramuj funkcję `pierwiastek(n)` obliczającą $\lfloor\sqrt{n}\rfloor$

Zadanie 3.

Zaprogramuj w Pythonie funkcję `sudan(n, x, y)`[1] obliczającą następującą funkcję rekurencyjną:

$$\begin{aligned}
&F_0(x,y) = x+y\\
&F_{n+1}(x,0) = x, x \geq 0\\
&F_{n+1}(x,y+1) = F_n(F_{n+1}(x,y), F_{n+1}(x,y)+y+1)
\end{aligned}$$

Ponieważ funkcja ta bardzo szybko rośnie, trzeba być ostrożnym i nie testować dla $n > 2$. Aby przyspieszyć działanie tej funkcji, proszę w implementacji zaprogramować przechowywanie już policzonych wyników; taka technika nazywa się *memoizacją* albo *spamiętywaniem*.

Sprawdź eksperymentalnie, dla jakich największych argumentów sensowne jest wywołanie tej funkcji w wersji bez spamiętywania, a dla jakich w wersji ze spamiętywaniem. Wyniki zamieść w komentarzu w pliku źródłowym.

Zadanie 4.

Zbyt skomplikowane zdania bywają utrapieniem dla czytającego tekst. Dlatego wykonamy uproszczenie w następujący sposób:

- najpierw usuwamy zbyt długie słowa;

[1] Jest to funkcja odkryta przez Gabiela Sudana

- a potem usuwamy losowo wyrazy jeśli zdanie ma ich zbyt wiele.

Zaprogramuj odpowiednią funkcję `uprosc_zdanie(tekst, dl_slowa, liczba_slow)`, gdzie *dl_slowa* to maksymalna dopuszczalna długość słowa, *liczba_slow* to największa liczba słów jaka może się znaleźć w zdaniu. Przykładowo

```
tekst = "Podział peryklinalny inicjałów wrzecionowatych \
kambium charakteryzuje się ścianą podziałową inicjowaną \
w płaszczyźnie maksymalnej."

uprosc_zdanie(tekst, 10, 5)
```

powinno zwrócić coś takiego

```
Podział kambium się ścianą inicjowaną.
```

Zbadaj działanie swojego programu dla jakiegoś popularnego dzieła literackiego dostępnego legalnie w sieci. W pliku źródłowym zamieść kod który pobiera taki tekst bądź zamieść w komentarzu link do takiego tekstu.

Zadanie 5.

Jedną z prostszych metod kompresji tekstu jest metoda polegająca na zastąpieniu ciągu identycznych znaków parą `(znak, liczność)`, np. zamiast *'aaaaaa'* można użyć `[(5, 'a')]`, a pojedynczą literę piszemy jak literę. Na przykład *'suuuuper'* skompresuje się do `[(1, 's'), (4, 'u'), (1, 'p'), (1, 'e'), (1, 'r')]`. Zaprogramuj dwie funkcje: `kompresja(tekst)` i `dekompresja(tekst_skompresowany)`, które zwracają odpowiednio tekst skompresowany i tekst zdekompresowany. Możesz przyjąć, że kompresujemy tylko teksty zawierające litery i znaki przestankowe.

Wypróbuj swój program na dłuższym tekście legalnie dostępnym w internecie. W kodzie źródłowym podaj link do tego tekstu lub zamieść kod pobierający ten tekst i wywołujący tę funkcję.

Marcin Młotkowski