

UNIVERZITA KARLOVA
PŘÍRODOVĚDECKÁ FAKULTA



ÚVOD DO PROGRAMOVÁNÍ

Úloha 3:

Nalezení nejkratšího a nejdelšího slova v textu.

Anna Kožíšková

3. ročník, BGEKA

Mnichovice, 2022

Cílem úlohy je nalezení nejdelšího a nejkratšího slova v textu. Text, může být uložený v textovém souboru, nebo ho lze uložit do proměnné typu string, která bude součástí kódu programu.

Nejprve je třeba si stanovit, co je považováno za slovo. Pokud nahlédneme do slovníku českého jazyka, je slovo nositelem pojmového významu. Umožňuje tedy pojmenovat objekty, jevy, situace kolem nás. Jeho nedílnou součástí je grafická podoba, kdy je každé slovo složeno z několika znaků (grafémů). V neposlední řadě by mělo mít slovo svou společenskou ustálenost a fonetickou a fonologickou tvárnost (Hladká 2017).

Pro účely zpracování této úlohy je nejdůležitější grafická podoba slova. Společenská ustálenost a význam slova v úloze řešeny nebudou. Za slovo tedy bude považovaný takový řetězec znaků, respektive písmen, který bude oddělený mezerou, tečkou, čárkou, či jiným podobným oddělovačem. Číslice v textu nebudou považována za slova. V rámci hledání nejmenšího slova nebudou jako nejmenší slova vyhodnocována slova jednopísmenná (např. spojky a, i atd.). Tato podmínka je nastavena z toho důvodu, že se takováto jednopísmenná slova jednoduše hledají a jsou celkem v textu na první pohled dohledatelná.

Použitý algoritmus

Program je tvořen ze dvou kódů, a to ze souborů *code.py* a *functions.py*. Oba jsou součástí repozitáře na stránkách: https://github.com/koziskoa/the_shortest_and_longest_word.git.

Funkce

V souboru *functions.py* jsou definovány funkce, které jsou používány v hlavním kódu (*code.py*) programu. Je zde definována funkce na otevírání souboru, rozeznání oddělovačů a rozeznání číslic.

Otevírání souboru

Funkce na otevírání souboru (viz *Obrázek 1*) funguje jen pro otevírání souboru typu .txt. Je tedy za potřebí, aby se uživatel ujistil, že má svůj text, který chce procházet uložený v tomto typu souboru. Uživatel se musí také přesvědčit, že se jeho textový soubor nachází ve stejné složce, kde má uložené oba kódy programu. Pokud tak neučiní, programu mu nebude fungovat, respektive mu program nahlásí chybu, že soubor nebyl nalezen. Druhý typ chyby, kterou funkce vyhazuje, je v případě, že bude chtít uživatel otevřít soubor, ke kterému ovšem nemá přístup. Program tak logicky nemůže pokračovat dál a vyhlásí chybu.

Obrázek 1: Otevírání souboru

```
1  def open_load (name):
2      """opens .txt file
3      - parametr needs name of file e.g. "file.txt"
4      - returns content of file as str
5      - """
6      try:
7          with open(name, encoding="utf-8") as file:
8              reader = file.read()
9              return reader
10     except FileNotFoundError:
11         print(f"File {name} not found")
12         exit()
13     except PermissionError:
14         print(f"no permission to the file {name}")
15         exit()
```

Oddělovače

Aby bylo možné rozpoznat slovo, je nutné rozpoznat oddělovače, které slova oddělují. Nejčastěji je to mezera, čárka, nebo tečka. Funkce v programu však také umí rozpoznat různé typy závorek, lomítka, uvozovky atd. Funkce *separators* (viz *Obrázek 2*), která je použita v programu pro rozpoznání oddělovačů pracuje tak, že prochází předdefinovaný seznam znaků, které symbolizují oddělovače, a porovnává ji s proměnnou *val*. V momentě, kdy se bude *val* rovnat jednomu z oddělovačů, vrátí funkce hodnotu *True*. Pokud ani jeden ze znaků neodpovídá proměnné *val*, vrátí funkce hodnotu *False*.

Obrázek 2: Rozpoznávání oddělovačů

```
27 def separators (val):
28     """ Finds if value is separator
29     - e.g. of separators: . , ? ! - : / " ' "space" etc.
30     - if is separator: returns True"""
31     sep_list = ['-', '.', ',', '?', '!', '-', ':', '/', '"', "'", ' ', '*',
32                '(', ')', '[', ']', '{', '}', '<', '>', '°', '@']
33     for separator in sep_list:
34         if separator == val:
35             return(True)
36     return(False)
```

Číslice

Funkce *is_number*, kterou popisuje *Obrázek 3*, vrátí hodnotu *True*, pokud se podaří obsah proměnné *value* převést na typ *int*. Znamená to tedy, že obsah proměnné je číslice. V opačném případě vrátí funkce hodnotu *False*.

Obrázek 3: Převedení na číslo

```
17 def is_number (value):
18     """Tries convert value to number
19     - if number: returns True
20     - if not number: returns False"""
21     try:
22         value = int(value)
23         return(True)
24     except ValueError:
25         return(False)
```

Funkce jsou napsané samostatně pro lepší přehlednost a všechny jsou použité v hlavním kódu programu.

Hlavní kód

V hlavním kódu (*code.py*) se nachází stěžejní část algoritmu. Otevírá se zde textový soubor. Jsou zde definované proměnné, které jsou během pracování programu modifikovány (*max_length* = 0, *min_length* = 100000 a *word* = ""). Nejdůležitější částí programu, která je zodpovědná za vyhodnocení nejdelšího a nejkratšího slova v textu, je cyklus (viz *Obrázek 4*), který prochází text a vyhodnocuje délku slova. Pracuje na principu, že prochází každý znak v textu. Nejprve je potřeba poskládat z písmen slovo, aby mohl algoritmus vyhodnocovat, jak je slovo dlouhé. Pokud cyklus nevyhodnotí, že je znak, oddělovač, nebo číslice, tak se aktuální znak přidá do prázdného řetězce *word*. V tu chvíli bude řetězec

nově obsahovat znak a cyklus začíná od znova s novým znakem. V momentě, kdy se ve znaku objeví oddělovač, bude splněna podmínka na řádce 17 a začne se vyhodnocovat délka slova.

Nejdřív se spočítá počet znaků ve slově. Potom se zkoumají podmínky, jestli je aktuální délka slova větší než *max_length*. Jestli je tato podmínka splněna znamená to, že aktuální porovnávané slovo je nejdelší. Podmínka je tedy splněna a aktuální délka slova (*length*) se přeuloží do proměnné *max_length* a do proměnné *max_word* se uloží aktuální nejdelší slovo. Podobně funguje i podmínka pro nejkratší slovo. Bude-li aktuální délka slova menší než *min_length* a zároveň větší než 1 (slovo nebude jednopísmenné), pak je splněna podmínka pro nové nejkratší slovo. Po těchto dvou podmínkách následuje přepsání proměnné *word* na prázdný řetězec a cyklus může začít znovu s novým znakem. Cyklus běží tak dlouho, dokud se nedostane na konec textu. Podmínka *is_number* řeší číslice v textu. Je napsaná tak, že pokud je znak číslicí, cyklus začne znovu s novým znakem a číslice si nevšímá. Vyskytne-li se mi v textu řetězec písmen, který obsahuje na začátku, uprostřed, nebo na konci číslici, cyklus si jí nevšímá a dál načítá znaky do proměnné *word*, dokud nenarazí na oddělovač. Například řetězec „*Popo4catepetl21*“ se vyhodnotí jako „*Popocatepetl*“.

Obrázek 4: Hlavní cyklus

```
15 | #iterating through string
16 | for element in text2:
17 |     if separators(element):
18 |         length = len(word)
19 |         if length > max_length:
20 |             max_length = length # the highest number of characters in a word
21 |             max_word = word
22 |         if length < min_length and length > 1:
23 |             min_length = length
24 |             min_word = word
25 |         word = ""
26 |         continue
27 |     if is_number(element):
28 |         continue
29 |     word += element
```

Poslední část programu řeší případ posledního slova, protože v momentě, kdy dojde cyklus na konec textu a posledním znakem nebyl oddělovač, tak se poslední slovo nevyhodnotí. Je tedy nutné podmínky, které byly v cyklu zařadit ještě jednou za cyklus pro vyhodnocení posledního slova.

Pokud program proběhne správně na konci se vytiskne nejdelší a nejkratší slovo s počtem znaků. Slova se vytisknou vždy s malými písmeny

Příklad výstupu:

The longest word in the string: popocatepetl (length: 12).

The shortest word in the string: tree (length: 4).

Alternativy programu

Součástí repozitáře na stránkách jsou ještě další dva kódy *code_2.py* a *code_3.py*. Oba rovněž přebírají funkce ze souboru *functions.py* stejně jako *code.py*. Rozdíl je v tom, jak cyklus vyhodnocuje slovo, pokud se v řetězci vyskytne číslice.

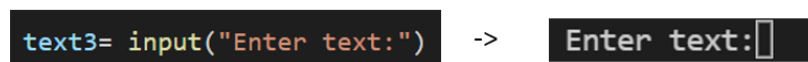
V souboru `code_2.py` je slovo, ve kterém se vyskytne číslice uprostřed řetězce, vyhodnoceno jako 2 samostatná slova. Bude-li se tedy vyhodnocovat řetězec „*Popo4catepetl21*“, vyhodnotí se jako „*Popo*“ a „*catepetl*“. Samotných číslic si stejně jako v prvním kódu nevšímá.

Ačkoliv byl v rámci úlohy stanoven cíl, že program nebude vyhodnocovat číslice za slova, tak soubor `code_3.py` tuto možnost nabízí. V průběhu chodu programu by pak algoritmus vyhodnocoval za slova i číslovky zapsané číslicemi, nebo řetězce písmen, které obsahují číslice. Příklad řetězce „*Popo4catepetl21*“, by byl vyhodnocen jako „*Popo4catepetl21*“.

Nedostatky programu

Ani jeden z algoritmů neumí vyhodnotit více stejně dlouhých nejkratších nebo nejdelších slov. Argumentem by však mohlo být, že cíl úlohy bylo nalezení nejdelšího a nejkratšího slova, ale není specifikováno, že se nejdelší a nejkratší slovo má na konci programu vypsat. Program neumí přijímat jiné typy souborů kromě jednoduchého textového souboru (.txt). Uživatel je tedy nucen si daný text uložit do textového souboru. Může však zvolit možnost, kdy zkopíruje text a uloží ji do proměnné přímo v těle programu. Daleko jednodušší by však bylo vytvořit interaktivní program, kdy se program uživatele zeptá na text, ze kterého chce nejdelší a nejkratší slovo vyhodnocovat (metoda *input* – Obrázek 5).

Obrázek 5: Zadání textu na vstupu



```
text3= input("Enter text:")
```

 -> `Enter text:`

Zdroje

HLADKÁ, Z. (2017): SLOVO. In: Karlík, P., Nekula, M., Pleskalová, J. a kol.: CzechEncy – Nový encyklopedický slovník češtiny. Masarykova univerzita, Brno. (cit. 9. 2. 2022)