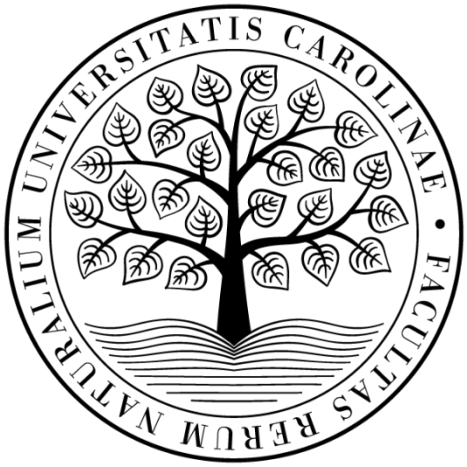
Univerzita Karlova

Pťírodovědecká fakulta



Doprovodná dokumentácia k programovým úlohám

Predmet Úvod do programování

Hana Struňáková

3. GEKA

1. Anotácia

Táto práca sa zaoberá riešením dvoch samostatných úloh v prostredí jazyka Python. Prvá úloha je zameraná na spracovanie textového vstupu, konkrétne na identifikáciu jednotlivých slov, určenie ich počtu a výpis slov v opačnom poradí. Druhá úloha pracuje s celočíselným vstupom a rieši obrátenie poradia číslic zadaného čísla bez použitia textovej reprezentácie, ako aj následný výpočet ciferného súčinu.

Obe úlohy sú riešené s využitím objektovo orientovaného prístupu, kladú dôraz na korektné ošetrenie okrajových prípadov, prehľadnú programovú štruktúru a dodržiavanie zadaných obmedzení.

2. Zadanie úloh

2.1 Úloha 1: Výpis zadaného textu po slovách v opačnom poradí

Program spracuje vstupný text obsahujúci písmená „A–Ž“, „a–ž“, číslice „0–9“ a vybrané špeciálne znaky „,.?!;“. Všetky ostatné znaky sú pri spracovaní ignorované. Program identifikuje jednotlivé slová oddelené medzerami, vypíše ich v opačnom poradí a určí ich počet. Sú ošetrené aj prípady, keď sú slová oddelené viacerými medzerami.

2.2 Úloha 2: Výpis čísla v opačnom poradí číslic

Program vypíše zadané celé číslo v opačnom poradí číslic a vypočíta jeho ciferný súčin. Výpočet nebude používať prevod čísla na reťazce. Prípadné nuly na začiatku čísla budú ignorované.

3. Úloha 1: Výpis zadaného textu po slovách v opačnom poradí

3.1 Rozbor problému

Cieľom prvej úlohy je spracovanie textového vstupu, ktorý môže obsahovať písmená latinskej abecedy vrátane diakritiky, číslice a vybrané špeciálne znaky.

Za slovo sa považuje neprázdny súbor povolených znakov oddelený jednou, alebo viacerými medzerami. Pri spracovaní je nutné ošetriť prípady s viacerými medzerami, aby nevznikali prázdne slová. Ďalej je potrebné ignorovať nepovolené znaky, ktoré nemajú ovplyvniť výsledok spracovania.

Výstupom úlohy je okrem výpisu slov v opačnom poradí aj počet slov. To zvýrazňuje dôležitosť správneho rozdelenia textu.

3.2 Zvolený algoritmus

Program je rozdelený do niekoľkých postupných krokov. Najprv je každý vstupný znak porovnaný so zoznamom povolených znakov. Tento zoznam je vytvorený s využitím štandardnej knižnice jazyka Python *string.* Ak znak patrí do zoznamu povolených znakov, je ponechaný, v opačnom prípade je pri spracovaní ignorovaný.

Povolené znaky sú postupne ukladané do zoznamu a následne spracovávané znak po znaku. Identifikácia jednotlivých slov je realizovaná manuálne, bez použitia vstavaných funkcií na delenie reťazcov. Znaky sú postupne pridávané do aktuálne spracovávaného slova, pričom výskyt medzery signalizuje koniec slova. Viacnásobné medzery sú týmto postupom ošetrené tak, že nevznikajú prázdne slová. Po identifikovaní všetkých slov je ich poradie obrátené pomocou cyklu, ktorý prechádza zoznam slov od posledného prvku po prvý..

Zvolený postup zabezpečuje správne spracovanie vstupu, prehľadnú implementáciu algoritmu a je v súlade s obmedzeniami zadania.

3.3 Diskusia výberu algoritmu

Pri návrhu riešenia bolo zvažované aj použitie regulárnych výrazov alebo spracovanie rozšírenej latinskej abecedy prostredníctvom knižnice unicodedata. Tieto prístupy je možné využiť pri riešení podobných úloh zameraných na spracovanie textu (Python Software Foundation, 2026d; Python Software Foundation, 2026f). Vzhľadom na rozsah a charakter zadania však boli v tomto prípade vyhodnotené ako zbytočne zložité.

Zvolený algoritmus pracuje s explicitne definovaným zoznamom povolených znakov, čo zvyšuje jeho čitateľnosť a zároveň umožňuje jednoduché rozšírenie alebo úpravu riešenia v budúcnosti.

3.4 Programová štruktúra riešenia

Úloha využíva objektovo orientovaný prístup k programovaniu. Hlavná logika spracovania textu je zapuzdrená v triede TextAnalyzer, ktorá uchováva vstupný text ako atribút objektu a obsahuje metódy určené na jeho spracovanie. Kontrola povolených znakov je realizovaná samostatnou metódou _char_ok, čo prispieva k prehľadnosti a znovupoužiteľnosti kódu. Obracanie poradia slov zabezpečuje metóda process, ktorá manuálne obracia slová zo zoznamu pomocou cyklov.

Vstup a výstup programu sú riešené mimo triedy, v hlavnej funkcii programu, čím je zabezpečené oddelenie aplikačnej logiky od používateľského rozhrania.

3.5 Reprezentácia vstupných a výstupných dát

Vstupom programu je jeden riadok textu zadaný používateľom prostredníctvom konzoly. Text môže obsahovať viacnásobné medzery a nepodporované znaky, ktoré sú počas spracovania ignorované. Výstupom programu je najskôr počet identifikovaných slov a následne výpis týchto slov v opačnom poradí v jednom riadku. Výstup je formulovaný tak, aby bol jednoznačný a ľahko čitateľný aj pre používateľa bez programátorských znalostí.

3.6 Možné vylepšenia

Ako možné vylepšenie riešenia je možné uviesť zlepšenie kontroly vstupných údajov, napríklad explicitné upozornenie používateľa na výskyt nepodporovaných znakov. Ďalším rozšírením by mohlo byť umožnenie spracovania textu zo súboru namiesto zadávania vstupu prostredníctvom konzoly.

4. Úloha 2: Výpis čísla v opačnom poradí číslic

4.1 Rozbor problému

Cieľom druhej úlohy je spracovanie celočíselného vstupu zadaného používateľom. Riešenie nesmie využívať prevod čísla ani jeho jednotlivých číslic na textovú reprezentáciu.

Pri spracovaní vstupu je nutné ošetriť prípady, keď sa v zadanom čísle nachádzajú nuly na konci, ktoré sa po obrátení poradia číslic objavia na začiatku výsledného čísla. Tieto nuly majú byť ignorované. Ďalším aspektom je správne ošetrenie nesprávneho vstupu, napríklad v prípade, že používateľ nezadá celé číslo.

4.2 Zvolený algoritmus

Riešenie úlohy je založené na opakovanom delení čísla desiatimi, pričom zvyšok po delení predstavuje aktuálnu poslednú číslicu spracovávaného čísla. Táto číslica je postupne pridávaná k výslednému otočenému číslu. Proces sa opakuje dovtedy, kým spracovávané číslo nenadobudne hodnotu nula. Týmto postupom sa automaticky zabezpečí ignorovanie prípadných núl na začiatku otočeného čísla.

Algoritmus pracuje s absolútnou hodnotou vstupného čísla, aby bolo možné jednotným spôsobom spracovať kladné aj záporné čísla. Znamienko pôvodného čísla je po dokončení obrátenia číslic aplikované dodatočne na výsledné otočené číslo.

Špeciálny prípad vstupnej hodnoty 0 je ošetrený samostatne, pretože bez tejto úpravy by sa výpočtové cykly nevykonali a ciferný súčin by zostal na inicializačnej hodnote 1 (Python Software Foundation, 2026c), čo by viedlo k nesprávnemu výsledku.

Následne sa z už otočeného čísla vypočíta ciferný súčin. Ak by sa ciferný súčin počítal v rovnakom kroku ako obrátenie čísla, bol by ovplyvnený prípadnými nulami na konci pôvodne zadaného čísla.

4.3 Diskusia výberu algoritmu

Pri riešení úlohy boli uvažované aj iné numerické postupy založené na celočíselných aritmetických operáciách. Jednou z možností je získavanie jednotlivých číslic pomocou delenia mocninami desiatky, pri ktorom sa číslo postupne rozkladá na jednotlivé číslice pomocou celočíselného delenia. Tento postup je funkčný, avšak z hľadiska implementácie zložitejší a menej prehľadný.

V praxi je na dosiahnutie rovnakého výsledku možno najjednoduchší prevod čísla na text, to však v tomto prípade nebolo brané do úvahy, lebo je to v priamom rozpore so zadaním.

4.4 Programová štruktúra riešenia

Riešenie úlohy je implementované objektovo orientovaným spôsobom. Logika spracovania čísla je zapuzdrená v samostatnej triede NumberReverser. Trieda uchováva vstupné číslo ako atribút objektu, pričom kontrola správnosti vstupu je vykonaná ešte pred vytvorením objektu v hlavnej funkcii programu. Tým je zabezpečené, že trieda pracuje výhradne s platným celočíselným vstupom. Obrátenie poradia číslic je implementované v hlavnej metóde triedy pomocou opakovaného delenia desiatimi a operácie modulo.

4.5 Reprezentácia vstupných a výstupných dát

Vstupom programu je celé číslo zadané používateľom prostredníctvom konzoly. V prípade, že vstup nie je vo správnom formáte, program vráti upozornenie o chybnom vstupe a ďalšie spracovanie neprebehne.

Výstupom programu je otočené číslo a hodnota jeho ciferného súčinu. Výstup je formulovaný textovou formou tak, aby bol jednoznačný a zrozumiteľný aj pre používateľa bez programátorských znalostí.

4.6 Možné vylepšenia

Program je možné vylepšiť rozsiahlejšou kontrolou vstupných údajov, napr. podrobnejším rozlíšením nesprávneho vstupu. Ďalším vylepšením by mohlo byť rozšírenie programu o spracovanie viacerých čísel zadaných postupne bez nutnosti opätovného spustenia programu.

5. Zoznam literatúry

PYTHON SOFTWARE FOUNDATION (2026a): Built-in Functions. https://docs.python.org/3/library/functions.html#reversed (cit. 6. 2. 2026)

PYTHON SOFTWARE FOUNDATION (2026b): Built-in Types. https://docs.python.org/3/library/stdtypes.html (cit. 7. 2. 2026)

PYTHON SOFTWARE FOUNDATION (2026c): math — Mathematical functions (math.prod). https://docs.python.org/3/library/math.html#math.prod (cit. 7. 2. 2026)

PYTHON SOFTWARE FOUNDATION (2026d): re — Regular expression operations. https://docs.python.org/3/library/re.html (cit. 7. 2. 2026)

PYTHON SOFTWARE FOUNDATION (2026e): string — Common string operations. https://docs.python.org/3/library/string.html (cit. 6. 2. 2026)

PYTHON SOFTWARE FOUNDATION (2026f): unicodedata — Unicode Database. https://docs.python.org/3/library/unicodedata.html (cit. 28. 1. 2026)