Cvičenie 10

Inštrukcie:

- Vyriešte všetky úlohy. Skúste čo najviac z nich spraviť už na cvičení. Ak nejakú úlohu nestihnete na cvičení a nebudete ju vedieť vyriešiť ani doma, opýtajte sa cvičiaceho na niektorom z ďalších cvičení.

Úloha č. 1

Definujte funkciu *najviac_nacitane()* bez parametrov, ktorá bude načítavať celé čísla z klávesnice, kým používateľ nezadá nulu. Funkcia vráti to číslo, ktoré bolo načítané najväčší počet krát. Ak existuje takýchto čísiel viacero, funkcia vráti to, ktoré dosiahlo tento maximálny počet načítaní ako prvé!

Vstup/výstup

Ak volanie *najviac_nacitane()* načíta čísla 3, 105, 2, 105, 3, 3, 3, 105, 2, 0, funkcia **vráti** číslo 3 (číslo 3 bolo načítané celkovo štyrikrát, číslo 105 trikrát, číslo 2 dvakrát a 0 jedenkrát.)
Ak volanie najviac_nacitane() načíta čísla 2, 2, **5**, **5**, 1, 2, **5**, 1, **5**, 2, 0 funkcia **vráti** číslo 5 (aj číslo 5, aj číslo 2 boli obe načítané najväčší počet krát – štyrikrát, avšak číslo 5 bolo načítané štyrikrát **skôr** ako číslo 2).

Úloha č. 2

Definujte funkciu *stred(t)*, ktorej parametrom je zoznam *t*. Funkcia **vráti** zoznam, ktorý vznikne zo zoznamu *t* odstránením prvého a posledného prvku. Ak je argumentom funkcie pri volaní prázdny zoznam, funkcia vypíše chybu a vráti hodnotu *None*. Funkcia **nesmie zmeniť** zoznam, ktorý tvorí jej argument pri volaní.

Vstup/výstup

Nech zoznam *z* = [5, [], ‘a‘, [1,2,3], -5.5]. Volanie *stred(z)* vráti zoznam [[], ‘a‘, [1,2,3]], pričom zoznam *z* zostane nezmenený aj po volaní funkcie.
Volanie *stred*([]) vypíše, že vstupom funkcie je prázdny zoznam a vráti *None*.

Úloha č. 3

Definujte funkciu *vyrez(t)*, ktorej parametrom je zoznam *t*. Funkcia **odstráni** zo zoznamu, ktorý tvorí jej argument pri volaní prvý a posledný prvok. Samotné volanie funkcie vráti hodnotu *None*. Ak je argumentom funkcie pri volaní prázdny zoznam, funkcia vypíše chybu a taktiež vráti hodnotu *None*. Funkcia teda **musí zmeniť** zoznam, ktorý tvorí jej argument pri volaní.

Vstup/výstup

Nech zoznam *z* = [5, [], ‘a‘, [1,2,3], -5.5]. Volanie *vyrez(z)* vráti *None* a zoznam *z* má po volaní funkcie hodnotu [[], ‘a‘, [1,2,3]].

Úvod k úlohám č. 4 a 5

V úlohách č. 4 a 5 budeme definovať funkcie, ktorých vstupy budú predstavovať matice. Naša matica bude reprezentovaná ako **dvojrozmerný zoznam**, teda zoznam zoznamov.

Uvažujme teda, že máme zoznam *t*, ktorý má *n* prvkov – každý prvok zoznamu *t* je zoznam obsahujúci *k* čísiel. Tieto vnorené zoznamy obsahujúce *k* prvkov budú reprezentovať **riadky** matice, teda zoznam *t* obsahujúci *n* prvkov – zoznamov *k* čísiel – reprezentuje maticu rozmerov *n* * *k*.

Napríklad matica $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ rozmerov 2*3 (2 riadky, 3 stĺpce) by teda bola reprezentovaná v jazyku Python pomocou zoznamu *t* = [[1,2,3],[4,5,6]] obsahujúceho *n*=2 zoznamy: [1,2,3] a [4,5,6], pričom každý z týchto zoznamov obsahuje *k*=3 čísla.

Úloha č. 4

Definujte funkciu *sucet_matic(A, B)*, ktorá má 2 parametre: matice *A* a *B*, ktoré sú implementované podľa pokynov vyššie. Funkcia **vráti súčet** matíc *A* + *B*. V prípade, že matice nie je možné sčítať (kvôli nekompatibilným rozmerom), funkcia vypíše „Zadane matice nie je mozne scitat“ a vráti hodnotu *None*.

Vstupy/výstupy:

Volanie *sucet_matic*([[1,2],[3,5]] , [[5,2],[-3,4]]) vráti zoznam [[6,4],[0,9]], čo predstavuje súčet:

$$\begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix} + \begin{pmatrix} 5 & 2 \\ -3 & 4 \end{pmatrix} = \begin{pmatrix} 6 & 4 \\ 0 & 9 \end{pmatrix}$$

Úloha č. 5

Definujte funkciu *sucin_matic(A, B)*, ktorá má 2 parametre: matice *A* a *B*, ktoré sú implementované podľa pokynov vyššie. Funkcia **vráti súčin** matíc *A* * *B*. V prípade, že matice nie je možné vynásobiť (kvôli nekompatibilným rozmerom), funkcia vypíše „Zadane matice nie je mozne vynasobit“ a vráti hodnotu *None*.

Vstupy/výstupy:

Volanie *sucin_matic*([[1,2],[3,5],[-2,-1]] , [[2],[-2]]) vráti zoznam [[-2],[-4],[-2]], čo predstavuje súčin:

$$\begin{pmatrix} 1 & 2 \\ 3 & 5 \\ -2 & -1 \end{pmatrix} * \begin{pmatrix} 2 \\ -2 \end{pmatrix} = \begin{pmatrix} -2 \\ -4 \\ -2 \end{pmatrix}$$

Volanie *sucin_matic*([[1,2],[3,5],[-2,-1]] , [[2],[-2],[1]]) vypíše „Zadane matice nie je mozne vynasobit“, pretože matice nemajú kompatibilné rozmery pre násobenie (prvá matica má 2 stĺpce a druhá matica 3 riadky).

Úvod k úlohe č. 6

V úlohe č. 6 budeme definovať funkciu, ktorá bude pracovať s otvorenými intervalmi. Otvorený interval (*a*, *b*) v matematike je množina všetkých reálnych čísiel *x*, kde platí $a < x < b$. My budeme takýto interval reprezentovať pomocou zoznamu [*a*, *b*]. Teda napríklad zoznam [-1,1] bude predstavovať otvorený interval (-1,1).

Úloha č. 6

Definujte funkciu *spolocny_prienik(intervaly),* ktorej vstupný parameter *intervaly* je zoznam intervalov podľa popisu vyššie, teda *intervaly* je zoznam dvojprvkových zoznamov tvaru [*a, b*], ktoré reprezentujú otvorené intervaly (*a*,*b*). Môžete predpokladať, že v každom intervale platí, že $a < b$. Funkcia vráti hodnotu *True*, ak je spoločný prienik všetkých intervalov zo zoznamu *intervaly* neprázdny. V opačnom prípade funkcia vráti hodnotu *False*.

Vstupy/výstupy:

Volanie *spolocny_prienik*([[-5,4],[0,7],[3,5]]) vráti *True*, pretože otvorené intervaly (-5,4), (0,7) a (3,5) majú spoločný prienik – interval (3,4).

Volanie *spolocny_prienik*([[0,4],[2,5],[4,7]]) vráti *False*, keďže intervaly (0,4), (2,5), (4,7) nemajú spoločný prienik.

Úloha č. 7
Definujte funkciu *ma_duplikat(t)*, ktorej parametrom je zoznam *t*. Funkcia vráti *True*, ak sa v zozname *t* nachádza nejaký prvok viac ako jedenkrát. V opačnom prípade vráti *False*. Funkcia nesmie zmeniť vstupný zoznam.

Vstupy/výstupy:
Nech zoznam *a* = [1, 2, [], 3, [], "a"]. Volanie *ma_duplikat*(*a*) vráti *True*, pretože sa v zozname *a* dvakrát nachádza ako prvok prázdny zoznam. Zoznam *a* zostane nezmenený aj po volaní funkcie *ma_duplikat(a).*

Úloha č. 8 (prevzatá z Think Python 2)
Naprogramujte v jazyku Python program, ktorý zistí, aká je pravdepodobnosť, že ak vezmeme 23 ľudí, tak majú v ten istý deň narodeniny.

Hint: Naprogramujte program, ktorý bude náhodne generovať dátumy narodení (deň a mesiac) – tým sa bude simulovať výber náhodných ľudí. Urobte program, ktorý vygeneruje 23 takýchto dátumov a zistí, či nastala kolízia – teda, že sa vygeneroval aspoň 2-krát ten istý dátum narodenia. Zopakujte takéto generovanie väčší počet krát – napríklad 1000-krát a zistite, koľkokrát nastala kolízia vo vygenerovaných dátumoch narodení. Tým získate hrubý experimentálny odhad pravdepodobnosti, že spomedzi 23 ľudí majú aspoň 2 v ten istý deň narodeniny.

Riešenie úlohy podľa autora knihy Think Python 2 nájdete tu:
https://github.com/AllenDowney/ThinkPython2/blob/master/code/birthday.py

Úloha č. 9 (prevzatá z Think Python 2)
Uvažujme súbor *words.txt*, ktorý sme používali v úlohách pre reťazce počas 8. týždňa. Definujte 2 funkcie: *vytvor_zoznam_slov_append*() a *vytvor_zoznam_slov_plus*(). Obe funkcie vytvoria zoznam slov zo slov zo súboru *words.txt*, pričom funkcia *vytvor_zoznam_slov_append*() vytvára zoznam pomocou metódy **append()** a funkcia *vytvor_zoznam_slov_plus*() pomocou operátora +, t.j. konštrukciou *t = t + [x]*. Ktorá funkcia je rýchlejšia?

Riešenie úlohy podľa autora knihy Think Python 2 nájdete tu:
https://github.com/AllenDowney/ThinkPython2/blob/master/code/wordlist.py

Úloha č. 10 (prevzatá z Think Python 2)
Ak chceme skontrolovať, či sa nejaký reťazec nachádza v zozname (napríklad nejaké slovo v zozname slov zo súboru *words.txt*), môžeme použiť Pythonovský operátor **in**. Avšak operátor **in** je pomerne pomalý, keďže prehľadáva zoznam postupne, prvok za prvkom.

Keďže súbor *words.txt* a aj zoznam, do ktorého načítame reťazce (pozri úloha č. 9), je utriedený **abecedne**, existuje aj rýchlejší spôsob vyhľadávania – tzv. **binárne vyhľadávanie**. Binárne vyhľadávanie je algoritmus vyhľadávania v utriedených dátach – najprv porovnáme hľadanú hodnotu s hodnotu, ktorá sa nachádza v utriedených dátach v **strede**. Ak je hľadaná hodnota rovná strednej hodnote, **našli** sme hľadanú hodnotu. Ak je hľadaná hodnota **menšia** ako stredná hodnota, následne prehľadávame **ľavú polovicu** utriedených dát. Ak je hľadaná hodnota **väčšia** ako stredná hodnota, následne prehľadávame **pravú polovicu** utriedených dát. Tento postup opakujeme, kým nenájdeme hľadanú hodnotu, alebo kým nezistíme, že sa v dátach nenachádza. Definujte funkciu *in_bisect(zoznam_slov, slovo)*, ktorá vezme utriedený zoznam reťazcov *zoznam_slov* a reťazec *slovo* a podľa uvedeného spôsobu vráti *True*, ak sa *slovo* nachádza v *zoznam_slov,* inak vráti False.

*Riešenie úlohy podľa autora knihy Think Python 2 (všimnite si, že autor používa **rekurziu**):*
https://github.com/AllenDowney/ThinkPython2/blob/master/code/inlist.py

Úloha č. 11
Definujte funkciu *najdi_reverzny_par()* ktorá v slovách zo súboru *words.txt* nájde všetky *reverzné slová*, teda slová, kde jedno slovo je zrkadlovým obrazom druhého slova (napríklad *bat* a *tab*).
V definícii funkcie môžete využiť funkciu *in_bisect()* z predošlej úlohy pre rýchle vyhľadávanie.

Riešenie úlohy podľa autora knihy Think Python 2:
https://github.com/AllenDowney/ThinkPython2/blob/master/code/reverse_pair.py

Úloha č. 12
Definujte funkciu *najdi_prepletene_slova()* ktorá v slovách zo súboru *words.txt* nájde také 2 slová, ktorých prepletením vznikne znovu slovo zo súboru *words.txt*. Napríklad ak vezmeme slová „shoe“ a „cold“ a prepletieme ich = vezmeme prvý znak z prvého slova, prvý znak z druhého slova, druhý znak z prvého slova, druhý znak z druhého slova, atď., t.j. dostaneme reťazec „schooled“, ktorý sa tiež nachádza vo *words.txt*.

Hint: Skúste sa zamyslieť, ako by bolo možné úlohu riešiť nie tak, že vezmete 2 rôzne slová, vytvoríte ich prepletenie a otestujete, či sa prepletenie nachádza v zozname slov, ale naopak, vychádzajte zo slova zo súboru a testujte, či sa 2 reťazce z jeho „rozpletenia“ nachádzajú v zozname slov.

Odporúčam si pozrieť riešenie úlohy podľa autora knihy Think Python 2:
https://thinkpython.com/code/interlock.py

Úloha č. 13
Definujte funkciu *rozloz_na_prvocisla(t),* ktorá má jeden parameter – zoznam celých čísiel *t*. Môžete predpokladať, že zoznam nie je prázdny a obsahuje len kladné celé čísla. Funkcia vráti zoznam obsahujúci prvočíselné faktory čísiel zo zoznamu *t*. Faktory sú pre jednotlivé čísla zo zoznamu *t* vo výslednom zozname oddelené nulami a zároveň sú usporiadané od najmenšieho po najväčší.

Vstup/výstup:
Volanie *rozloz_na_prvocisla*([14, 11, 18]) vráti zoznam [2, 7, 0, 11, 0, 2, 3, 3, 0]

Úloha č. 14
Definujte funkciu *usporiadaj_trojice(t),* ktorá má jeden parameter – zoznam celých čísiel *t*. Môžete predpokladať, že zoznam nie je prázdny a obsahuje celé čísla, ktorých počet je násobok troch. Funkcia **vráti** verziu zoznamu *t,* ktorá je usporiadaná **po trojiciach** vo vzostupnom poradí, pričom pôvodný zoznam *t* zostane bezo zmeny.

Vstup/výstup:
Volanie *usporiadaj_trojice*([5, 9, -10, 0, 2, 0, 4, -1, 10]) vráti zoznam [-10, 5, 9, 0, 0, 2, -1, 4, 10]

Úloha č. 14b
Upravte funkciu z úlohy 14, aby vrátila hodnotu *None* a **priamo upravila zoznam *t*.**

Vstup/výstup:
Nech je daný zoznam *t* = [5, 9, -10, 0, 2, 0, 4, -1, 10]. Volanie *usporiadaj_trojice(t)* v tejto upravenej verzii vráti hodnotu *None* a po volaní bude hodnota zoznamu *t* = [-10, 5, 9, 0, 0, 2, -1, 4, 10].