Cvičenie č. 8

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 ďalšom cvičení.

Časť prvá (úlohy prevzaté zo sekcie 8.13 Exercises z knihy Think Python):

Úloha č. 1 – teoretický úvod do úlohy

Ako bolo spomínané na prednáške, na webstránke:
https://docs.python.org/3/library/stdtypes.html#string-methods
viete nájsť zoznam preddefinovaných metód pre reťazce v jazyku Python. Napríklad metóda **count**(), ktorej popis je na uvedenej stránke, hľadá počty neprekrývajúcich sa výskytov reťazca *sub* v reťazci, pre ktorý je metóda volaná. Navyše je možné špecifikovať, v akej časti reťazca sa podreťazec *sub* hľadá.

hlavička metódy tak, ako ju nájdete na uvedenej stránke:
str.count(*sub*[, *start*[, *end*]])

vysvetlenie hlavičky str.count(*sub*[, *start*[, *end*]]) :
1) str – znamená, že metóda sa týka reťazcov (str = string = reťazec). **Keď budete metódu používať, namiesto „str“ musíte uviesť alebo konkrétny reťazec, alebo premennú, kde sa reťazec, v ktorom hľadáte výskyty podreťazca *sub,* nachádza !**
2) count – identifikátor (názov) metódy, ktorým ju voláme
3) parametre: parametre uvedené v hranatých zátvorkách [] sú nepovinné (voliteľné parametre). To znamená, že v prípade metódy **count()** sú parametre nasledovné:
   3.1) *sub* – povinný parameter
   3.2) *start* – nepovinný parameter
   3.3) *end* – ak je uvedený parameter *start*, tak je to nepovinný parameter (ak nie je uvedený parameter *start*, tak parameter *end* nemožno použiť)

Metóda **count**() vráti počet výskytov podreťazca *sub* v reťazci, pre ktorý je metóda **count()** volaná. Ak sa navyše uvedie hodnota pre parameter *start*, tak táto hodnota predstavuje index, odkiaľ sa bude hľadať výskyt podreťazca *sub* – teda sa prehľadáva reťazec v zmysle slice-notácie [*start*:]. Ak je k parametru *start* navyše uvedená hodnota pre parameter *end*, tak táto hodnota parametra *end* predstavuje index, pokiaľ sa bude hľadať výskyt podreťazca *sub* – teda sa prehľadáva reťazec v zmysle slice-notácie [*start*:*end*].

Úloha č. 1 – zadanie

Definujte funkciu *pocet_vyskytov(retazec, podretazec)*, ktorá má 2 parametre – reťazce *retazec* a *podretazec.* Funkcia vráti počet, koľkokrát sa v reťazci *retazec* nachádza *podretazec* ako podreťazec s **neprekrývajúcim sa výskytom**.

*Hint:v definícii funkcie pocet_vyskytov(retazec, podretazec) použite metódu **count().***

Vstupy/výstupy:
Volanie *pocet_vyskytov*(“banana“, “a“) vráti číslo 3.
Volanie *pocet_vyskytov*(“banana“, “an“) vráti číslo 2.
Volanie *pocet_vyskytov*(“banana“, “ana“) vráti číslo 1.

Úloha č. 2

Definujte funkciu *je_palindrom(ret)*, ktorá má 1 vstupný parameter – reťazec *ret*. Funkcia vráti hodnotu *True* v prípade, že *ret* je palindróm – palindróm je reťazec, ktorý sa číta rovnako odpredu aj odzadu. V opačnom prípade vráti *False*.

Hint: na prednáške sme si ukazovali slice-notáciu a spomínali sme, že ret[::-1] vráti reťazec ret v obrátenom poradí.

Vstupy/výstupy:
Volanie *je_palindrom(“anna“)* vráti *True.*
Volanie *je_palindrom(“anka“)* vráti *False.*
Volanie *je_palindrom(“kobylamamalybok“)* vráti *True.*
Volanie *je_palindrom(““)* vráti *True.*(t.j. argumentom je prázdny reťazec)

Úloha č. 3

Definujte funkciu *je_palindrom(ret)*, ktorá má 1 vstupný parameter – reťazec *retazec*. Funkcia vráti hodnotu *True* v prípade, že *ret* je palindróm – palindróm je reťazec, ktorý sa číta rovnako odpredu aj odzadu. V opačnom prípade vráti *False*. **Definujte funkciu s využitím rekurzie!**

Hint: Slovo je palindrómom, ak sa jeho prvý a posledný znak rovnajú a zároveň zvyšok slova po odstránení prvého a posledného znaku je taktiež palindróm!

Vstupy/výstupy:
Volanie *je_palindrom(“anna“)* vráti *True.*
Volanie *je_palindrom(“anka“)* vráti *False.*
Volanie *je_palindrom(“kobylamamalybok“)* vráti *True.*
Volanie *je_palindrom(““)* vráti *True.*(t.j. argumentom je prázdny reťazec)

Úloha č. 4

Definujte funkciu *any_lowercase(s),* ktorá má 1 vstupný parameter – reťazec *s*. Funkcia vráti booleovskú hodnotu *True,* ak reťazec *s* obsahuje aspoň 1 malé písmeno anglickej abecedy. V opačnom prípade vráti *False*.

Vstupy/výstupy:
Volanie *any_lowercase(“anna“)* vráti *True.*
Volanie *any_lowercase(“ANNA“)* vráti *False.*
Volanie *any_lowercase(“ANNa“)* vráti *True.*
Volanie *any_lowercase(““)* vráti *False.*(t.j. argumentom je prázdny reťazec)

Úloha č. 5

```
def any_lowercase1(s):
    for c in s:
        if c.islower():
            return True
        else:
            return False

def any_lowercase2(s):
    for c in s:
        if 'c'.islower():
            return 'True'
        else:
            return 'False'

def any_lowercase3(s):
    for c in s:
        flag = c.islower()
    return flag
```

```
def any_lowercase4(s):
    flag = False
    for c in s:
        flag = flag or c.islower()
    return flag


def any_lowercase5(s):
    for c in s:
        if not c.islower():
            return False
    return True
```

Vľavo je uvedených 5 verzií funkcie *any_lowercase(s)*.

Skúste sa zamyslieť, bez ich spustenia, ktoré z nich sú korektnou implementáciou funkcie *any_lowercase(s)* z predošlej úlohy. V prípade, že neviete, tak si ich spustite s rôznymi vstupmi a snažte sa zistiť, ktoré fungujú korektne a pochopiť, prečo tie nekorektné nefungujú!

Úloha č. 6
Jednou z najznámejších historických klasických šifier je Cézarova šifra. Cézarova šifra spočíva v tom, že sa písmená správy, ktorú chceme zašifrovať, cyklicky posunú o nejaký definovaný posun.

Napríklad posun o 3 pozície posunie písmená A -> D, B -> E, C -> F, …, X -> A, Y -> B, Z -> C. Posun o 1 pozíciu posunie písmená A ->B, B -> C, C -> D, …, Y -> Z, Z -> A.

Teda napríklad, ak by sme správu "CHEER" posunuli o 7 pozícií, dostaneme správu "JOLLY" (každé písmeno správy "CHEER" posunieme o 7 pozícii).

Definujte funkciu *cezar(OT, posun)* ktorá má 2 parametre: reťazec *OT* a celé číslo *posun*. *OT* predstavuje reťazec, ktorý chceme zašifrovať pomocou cyklického posunu každého jeho písmena (OT sa v slovenčine označuje tzv. otvorený text, čo je nešifrovaná správa), *posun* predstavuje číslo udávajúce číselný posun písmen. Môžete predpokladať, že reťazec *OT* pozostáva z veľkých písmen anglickej abecedy. Funkcia **vráti** zašifrovanú verziu reťazca *OT* pomocou Cézarovej šifry s posunom o hodnote *posun.*

Upozornenie: Spomeňte si na to, čo sme hovorili na prednáške o zmene znakov v reťazci – v jazyku Python nie je možné priamo modifikovať znaky v reťazci – namiesto toho je nutné vytvoriť nový, modifikovaný reťazec!

*Hint:Využite funkcie **ord()** a **chr()** pre konverzie textových znakov na ich číselné reprezentácie!*

Vstupy/výstupy:
Volanie *cezar("CHEER",7)* vráti reťazec *"JOLLY"*
Volanie *cezar("AHOJ",1)* vráti reťazec *"BIPK"*
Volanie *cezar("AHOJ",26)* vráti reťazec *"AHOJ"*

Časť druhá (9. kapitola z knihy Think Python):

V tejto časti nájdete úlohy, ktoré budú pri práci s reťazcami využívať zoznam slov anglického jazyka, uložených v súbore **words.txt**. Tento súbor si môžete stiahnuť zo stránky s cvičeniami.

Celá táto časť je prevzatá z kapitoly č. 9 z knihy Think Python. **Odporúčam si túto kapitolu prečítať aj samostatne!**

Úloha č. 1 – úvod do práce so súbormi

Ukážeme si „rýchlokurz geniality", ako v jazyku Python otvoriť textový súbor a následne z neho čítať riadok-za-riadkom:

1) otvorenie súboru v programe vykonáme pomocou funkcie **open(),** ktorej argumentom bude cesta k súboru, ktorý chceme otvoriť (načítať). V našom prípade volanie **open("words.txt")** otvorí súbor **words.txt** (súbor sa **musí** nachádzať v tom istom priečinku ako program, v ktorom použijeme volanie **open("words.txt")**). Funkcia **open**() v prípade úspešného otvorenia súboru vráti reprezentáciu súboru, s ktorou môžeme v programe ďalej pracovať – pre tieto účely si návratovú hodnotu volania **open()** vložíme do nejakej premennej, napríklad:

fin = open("words.txt")

vytvorí premennú **fin** cez ktorú budeme následne pristupovať k súboru **words.txt**

2) postupné načítavanie obsahu súboru **words.txt** riadok-po-riadku vykonáme tak, že ak už máme načítanú premennú **fin**, cez ktorú budeme pristupovať k obsahu súboru, pomocou jednoduchého **for**-cyklu môžeme iteratívne čítať obsah daného súboru riadok-po-riadku ako:

for *riadok* **in** *fin*:
    #v cykle bude v premennej *riadok* v každej iterácii obsah ďalšieho riadku zo súboru
    #ku ktorému pristupujeme cez premennú *fin*

Ak by sme si dali vypísať premennú *riadok* v každom cykle na obrazovku:

```
fin = open("words.txt")
for riadok in fin:
    print(riadok)
```

```
aa

aah

aahed

aahing

aahs
```

tak vidíme, že po každom slove nasleduje prázdny riadok – to je kvôli tomu, že v súbore **words.txt** sú jednotlivé slová každé na novom riadku a Python pri ich načítaní považuje tento „skok“ na nový riadok za súčasť reťazca. Preto ak chceme, aby sme v každej iterácii mali načítané len slovo zo súboru **words.txt**, môžeme na obsah premennej *riadok* aplikovať metódu **strip()**, ktorá vráti verziu reťazca, ktorá obsahuje len tlačiteľné znaky (teda z reťazca odstráni „skok“ na nový riadok):

```
fin = open("words.txt")
for riadok in fin:
    slovo = riadok.strip()
    print(slovo)
```

```
aa
aah
aahed
aahing
aahs
```

Ak teda vezmeme program:

```
fin = open("words.txt")
for riadok in fin:
    slovo = riadok.strip()
```

Tak v cykle dostaneme v každej iterácii v premennej *slovo* 1 slovo zo súboru **words.txt.**

Úloha č. 1
Definujte funkciu *slova_dlhsie_nez(n)*, ktorá má 1 vstupný parameter – kladné celé číslo *n*. Funkcia načíta súbor **words.txt** a vypíše z neho **len tie slová**, ktorých dĺžka je aspoň *n*.

Vstupy/výstupy
Volanie *slova_dlhsie_nez(21)* vypíše slová zo súboru **words.txt**, ktorých dĺžka je aspoň 21, t.j. slová
```
counterdemonstrations
hyperaggressivenesses
microminiaturizations
```

Úloha č. 2
Definujte funkciu *neobsahuje_e(slovo)* s 1 parametrom – reťazcom *slovo*. Funkcia vráti booleovskú hodnotu *True,* ak reťazec *slovo* neobsahuje písmeno "e". Ak *slovo* obsahuje aspoň jeden výskyt znaku "e", vráti *False*.

Následne pomocou tejto funkcie vypíšte všetky slová zo súboru **words.txt**, ktoré neobsahujú znak "e" a spočítajte, aké je percento takýchto slov, teda koľko percent slov zo súboru **words.txt** neobsahuje znak "e".

Úloha č. 3
Definujte funkciu *neobsahuje(slovo,pismena)* s 2 parametrami – reťazcom *slovo* a reťazcom *pismena*. Funkcia vráti booleovskú hodnotu *True,* ak reťazec *slovo* neobsahuje ani jedno písmeno uvedené v reťazci *pismena*. Ak *slovo* obsahuje aspoň jeden výskyt niektorého zo znakov reťazca *pismena*, funkcia vráti *False.*

Vstupy/výstupy:
Volanie *neobsahuje("stlp","aeiouy")* vráti *True.*
Volanie *neobsahuje("stlpy","aeiouy")* vráti *False.*
Volanie *neobsahuje("abeceda","fg")* vráti *True.*
Volanie *neobsahuje("abeceda","efg")* vráti *False.*

Úloha č. 4
Definujte funkciu *pouziva_len(slovo,pismena)* s 2 parametrami – reťazcom *slovo* a reťazcom *pismena*. Funkcia vráti booleovskú hodnotu *True,* ak reťazec *slovo* používa **len** písmená uvedené v reťazci *pismena*. Ak *slovo* obsahuje aspoň jeden výskyt znaku, ktorý nie je uvedený v reťazci *pismena*, funkcia vráti *False.*

Vstupy/výstupy:
Volanie *pouziva_len("stlp","prstl")* vráti *True.*
Volanie *pouziva_len("stlpy","prstl")* vráti *False*
(pretože "stlpy" obsahuje "y" ktorý nie je v "prstl")
Volanie *pouziva_len("abeceda","abcdefgh")* vráti *True.*
Volanie *pouziva_len("abeciedka","abcdefgh")* vráti *False.*

Úloha č. 5
Definujte funkciu *pouziva_vsetky(slovo,pismena)* s 2 parametrami – reťazcom *slovo* a reťazcom *pismena*. Funkcia vráti booleovskú hodnotu *True,* ak reťazec *slovo* používa **každé** písmeno uvedené v reťazci *pismena* **aspoň jeden krát**. Ak *pismena* obsahuje aspoň jeden znak, ktorý nie je použitý v reťazci *slovo*, funkcia vráti *False.*

Vstupy/výstupy:
Volanie *pouziva_vsetky("stlp","prstl")* vráti *False.*
Volanie *pouziva_vsetky("stlpy","pstl")* vráti *True*
Volanie *pouziva_vsetky("abeceda","abcde")* vráti *True.*
Volanie *pouziva_vsetky("abeciedka","abcde")* vráti *True.*
Volanie *pouziva_vsetky("abeciedka","abcdef")* vráti *False.*

Úloha č. 6
Definujte funkciu *je_vzostupne(slovo)* s 1 parametrom – reťazcom *slovo.* Funkcia vráti booleovskú hodnotu *True,* ak sú písmená v reťazci *slovo* zároveň zoradené vo vzostupnom poradí. Inak vráti *False*. Uvažujte situáciu, že za sebou môžu ísť aj 2 rovnaké písmená.

Vstupy/výstupy:
Volanie *je_vzostupne("stlp")* vráti *False.*
Volanie *je_vzostupne("psy")* vráti *True.*
Volanie *je_vzostupne("abeceda")* vráti *False.*
Volanie *je_vzostupne("mor")* vráti *True.*

Pomocou funkcie *je_vzostupne(slovo)* zistite, koľko takýchto slov obsahuje súbor **words.txt.**

Časť tretia (úlohy prevzaté zo sekcie 9.7 Exercises z knihy Think Python):

Úloha č. 1

Definujte funkciu *tri_dvojite(ret)* s 1 parametrom – reťazcom *ret*, ktorá vráti booleovskú hodnotu *True* v prípade, že sa v reťazci *ret* nachádzajú neprekrývajúce sa tri za sebou idúce dvojice rovnakých písmen. V opačnom prípade funkcia vráti booleovskú hodnotu *False*.

Použite funkciu *tri_dvojite(ret)* na nájdenie všetkých slov v súbore **words.txt**, ktoré spĺňajú uvedenú vlastnosť.

Vstupy/výstupy:
Volanie *tri_dvojite("committee")* vráti *False*, pretože síce obsahuje 3 dvojice rovnakých písmen za sebou *mm*, *tt, ee*, avšak nie sú presne za sebou!
Volanie *tri_dvojite("commttee")* vráti *True*, pretože obsahuje 3 dvojice rovnakých písmen za sebou, *mm*, *tt, ee.*

Úloha č. 2

Napíšte program (v tejto úlohe nemusíte definovať žiadnu funkciu), ktorý nájde riešenie nasledovného hlavolamu:

Vodič sa vezie vo svojom aute. Všimol si, že jeho aktuálny odometer v aute (prístroj na meranie prejdených kilometrov) so 6 ciframi ukazuje taký aktuálny stav najazdených kilometrov, že platí:

1) Aktuálna hodnota odometra má tú vlastnosť, že posledné 4 číslice tvoria palindróm. (napríklad povedzme 2-0-**0-0-0-0**
2) O 1 kilometer neskôr, posledných 5 číslic bude tvoriť palindróm.
3) O ďalší 1 kilometer neskôr, prostredné 4 číslice budú tvoriť palindróm.
4) O ešte ďalší 1 kilometer neskôr, všetkých 6 číslic bude tvoriť palindróm.

Aká je aktuálna hodnota odometra?

Hint:
1) Ak chcete skonvertovať číslo na reťazec, funkcia ***str()*** *vráti hodnotu svojho argumentu ako reťazec, t.j. napríklad volanie* ***str(100000)*** *vráti reťazec "100000"*
*2) Metóda str.****zfill(****width****)*** *vráti verziu reťazca str, ktorá je doplnená nulami zľava tak, aby výsledná dĺžka reťazca bola width. Napríklad ak v premennej s = "37", potom s.zfill(4) vráti reťazec "0037"*

Úloha č. 3

Napíšte program (v tejto úlohe nemusíte definovať žiadnu funkciu), ktorý nájde riešenie nasledovného hlavolamu:

1) Dcéra si všimla, že aktuálny vek jej mamy je zrkadlovým obrazom jej aktuálneho veku (zrkadlový obraz znamená, že napríklad by jej mama mala 73 rokov a dcéra 37 rokov).
2) Dcéra si zároveň všimla, že tento jav nastal v ich životoch zatiaľ už 6-krát a že ak budú mať šťastie, tak nastane ešte jeden krát a ak budú mať veľké šťastie, tak nastane ešte ďalší jeden krát, teda dokopy by situácia nastala 8-krát.

Aký je vek dcéry?