Cvičenie na iteráciu a zoznamy

Na zahriatie

Máte zoznam:

```
["banana", "apple", False,
 "apple", "banana", True,
 "banana", "pig", None,
 1, "banana", None, True]
```

Pomocou `for` cyklu spočítajte:

- Počet banánov
- Počet jabĺk
- Počet ovocia
- Počet datových typov typu `bool`

`middle`, `chop` a `is_sorted`

Nasledujúce sú viac-menej cvičenia 10.3, 10.4 a 10.5 z knihy Think Python.

Funkcie si otestujte!

- Naprogramujte funkciu `middle`, ktorá vezme zoznam a vráti nový zoznam, v ktorom chýba prvá a posledná hodnota, napr:

```
middle(['a', 'b', 'c', 'd'])  # -> ['b'. 'c']
```

- Urobte to isté, ale namiesto toho aby ste vrátili z funkcie nový zoznam, zoznam zmeňte, a vráťte `None`:

```
lst = [1, 2, 3]
chop(lst) # -> None
print(lst) # [2]
```

Návod:

Metóda `list.pop` môže brať jeden argumnent, ktorý špecifikuje index, z ktorého prvok vyhodíte:

```
a = [1, 2, 3, 4, 5]
a.pop() # -> 5
a.pop(1) # -> 2
```

- Naprogramujte funkciu `is_sorted`, ktorá vráti `True` pokiaľ je zoznam zoradený, od najmenšej do najväčšej hodnoty, inak vráti `False`.

```
is_sorted([2, 1]) # -> False
is_sorted([]) # -> True
is_sorted([0]) # -> True
is_sorted([1.21, 3.14, 5]) # -> True
```

Jednoduchá šifra

Rýchla referencia na slicing

Nasledujúca kombinácia slov nedáva zmysel:

```
to a steal TV ideas can from insult one your person
intelligence is but plagiarism nothing to rubs steal
it from in many like is a research computer
```

Ale ak budete čítať každé druhé slovo, tak dostanete dve zmyslu-plné vety. Naprogramujte program, ktorý vám slová usporiada v správnom poradí.

Návod:

Metóda `str.split()` rozdelí text zoznam slov.

Metóda `str.join()` zoznam spojí do reťazca.

Príklad použitia:

```
text = "a,b,c"
abc = text.split(',')
s_medzerou = " ".join(abc)
```

Poznámka: Ak vynecháte argument v metóde `str.split`, tak reťazec sa rozdelí podla *whitespace* – medzier, nových riadkov a tabulátorov. Toto je veľmi užitočné, hlavne pre prípady keď máte viac medzier a nechcete rozdelovať za každou jednou:

```
"   1   2   3   ".split() # -> ['1', '2', '3']
```

(Zdroj viet)

Deepcopy

Implementujte si vlastný `deepcopy`, t.j. rekurzívnu kópiu zoznamu, ktorá skopíruje aj zanorené zoznamy.

Pre tento účel budete chcieť použiť rekurziu a pre *base case* využiť funkciu `type`:

```
type([3, 1, 2]) is list # True
```

Tu sú nejaké testy, ktoré by vaša funkcia mala spĺňať:

```
def deepcopy(lst):
    # TODO

a = ['a', 'b', 'c']
b = [1, 2, 3, 4, a, a]
c = [b, 'hello']

c_dupl = deepcopy(c)

# All should be True:
print(c_dupl == c)
print(c_dupl is not c)
print(c_dupl[0] == b)
print(c_dupl[0] is not b)
print(c_dupl[0][-1] == a)
print(c_dupl[0][-1] is not a)
print(c_dupl[0][-2] == a)
print(c_dupl[0][-2] is not a)
```

Poznámka:

`deepcopy` z modulu `copy` funguje *trochu* inak, viz. toto.

1D celulárny automat

Jedno-dimenzionálny celulárny automat pozostáva z vektoru buniek a pravidiel ktoré určujú, ako sa bunky zmenia v ďalšom kroku. Bunky nadobúdajú dva možné stavy. 0-mŕtva bunka, 1-živá bunka.

Pravidlo 110

Každá bunka sa vyvíja podľa svôjho stavu a stavu dvoch susedných buniek. Napríklad bunka so stavom 1, ktorá je obklopená dvoma ďalšími bunkami so stavmi 1, zmení v ďalšom kroku svôj stav na 0. Prvá a posledná bunka budú mať vždy stav 1.

Zoznam pravidiel

1**1**1	1**1**0	1**0**1	1**0**0	0**1**1	0**1**0	0**0**1	0**0**0
↓	↓	↓	↓	↓	↓	↓	↓
0	**1**	**1**	**0**	**1**	**1**	**1**	**0**

Príklad výpočtu pre stav 11101

1. nastavíme prvú a poslednú bunku na 1

1	1	1	0	1
1	_	_	_	**1**

2. vypočítame nový stav pre bunku na pozícii 1 (číslujeme od 0). Použijeme pravidlo pre 111.

1	1	1	0	1
	↓			
1	**0**	_	_	1

3. vypočítame nový stav pre bunku na pozícii 2. Použijeme pravidlo pre 110.

1	1	1	0	1
		↓		
1	0	**1**	_	1

4. vypočítame nový stav pre bunku na pozícii 3. Použijeme pravidlo pre 101.

1	1	1	0	1
			↓	
1	0	1	**1**	1

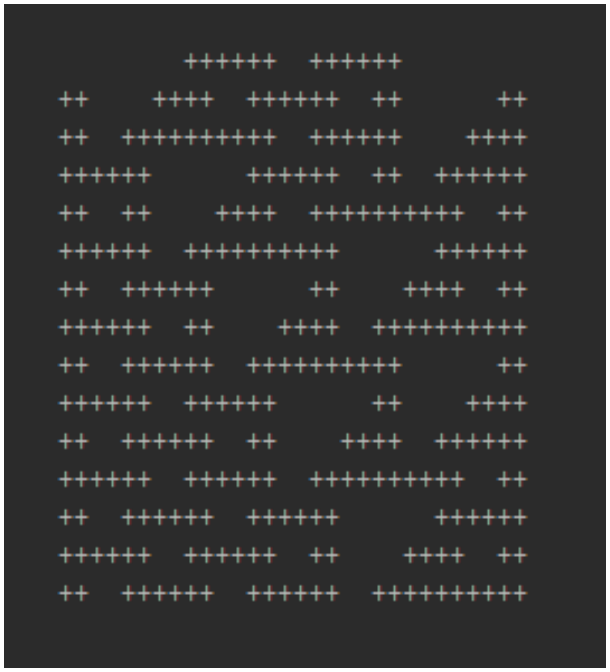
- Napíšte funkciu, ktorá vykreslí stav na príkazový riadok. Stav 0 vykreslite ako dve medzery a stav 1 vykreslite ako 2x plus `++`. Napríklad stav `[1, 0, 1]` sa má vykresliť na `++  ++`.

```
def print_state(cells):
    # TODO: implement
```

- Napíšte funkciu, ktorá bude brať počiatočný stav automatu a počet krokov a vráti list stavov automatu v krokoch 0 až n (počiatočný stav, stav automatu v prvom kroku, v druhom kroku, ...).

```
def cellular_automaton(cells, n):
    # TODO: implement automaton
```

- Vyskúšajte vykresliť prvých 15 krokov pre počiatočný stav `[0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0]`. Mal by sa vám vypísať nasledujúci výstup.



Obr. 1: Očakávaný výsledok

AoC

Advent of Code je veľmi pekná stránka, kde každý rok pred Vianocami nájdete kvalitne navrhnuté programovacie úlohy so stupňujúcou sa zložitosťou.

Vyriešte príklad 4 z roku 2019: Day 4.

Ak sa nechcete prihlasovať, môj input bol:

```
206938-679128
```