

C2184 Úvod do programování v Pythonu (podzim 2020)

Povinné domácí úkoly

Úkoly v této sadě řešte do připravených souborů `header.py`, `find.py`, `recursive_find.py`, `sum_paper.py`, `collect_sequences.py`, `count_atoms.py`, které pak odevzdáte do odevzdávnice. Doctesty a typové anotace v těchto souborech považujte za součást zadání.

V každé úloze je naším úkolem napsat program, který budeme moct volat z příkazové řádky. Základní struktura je už připravena (blok `if __name__ == '__main__': main()`). Naším úkolem je doplnit samotnou funkci `main`. Samozřejmě lze definovat další pomocné funkce a ty pak volat v `mainu`.

V tomto zadání je vždy uvedený příklad volání programu – řádek začínající `$` popisuje, jak budeme program volat z příkazové řádky; další řádky jsou očekávaný výstup, který se má vypsát na terminál. Tento příklad je pouze ilustrační – více příkladů je vždy doctestech ke konkrétnímu programu.

Příklad doctestu:

```
$ python foo.py x 5
>>> run('python foo.py x 5') # doctest: +NORMALIZE_WHITESPACE
xxxxx
```

Program `foo.py` voláme z příkazové řádky s argumenty `x` a `5`. Očekávaný výstup je jeden řádek `xxxxx`. Řádek s `>>>` nemusíte brát v potaz (`run` je pomocná funkce, která zabezpečí při testování takové chování, jakoby byl program volán z příkazové řádky – proto neměňte funkci `run` ani blok `if __name__...`).

Všechny soubory ve složce `data` jsou v kódování UTF-8, stejné kódování použijte i při zápisu výstupních souborů.

Doctesty spustíte z příkazové řádky:

```
$ python -m doctest ***.py           # default mode
$ python -m doctest -v ***.py        # verbose mode
```

Při spouštění testů je důležité, abyste byli přímo ve složce, kde máte uložené programy, jinak nebudou sedět relativní cesty a testy neprojdou.

DÚ 8.1: Hlavička souboru

Doplňte program `header.py`, který bude volán z příkazové řádky s jedním argumentem – názvem souboru. Program vypíše prvních 10 řádků tohoto souboru (nebo méně, je-li kratší). Pokud soubor neexistuje nebo ho nelze načíst, vypíše se `{nazev_souboru} not found`

```
$ python header.py data/1tqn.pdb
```

```
HEADER      OXIDOREDUCTASE                      17-JUN-04    1TQN
TITLE       CRYSTAL STRUCTURE OF HUMAN MICROSOMAL P450 3A4
COMPND      MOL_ID: 1;
COMPND      2 MOLECULE: CYTOCHROME P450 3A4;
COMPND      3 CHAIN: A;
COMPND      4 EC: 1.14.14.1;
COMPND      5 ENGINEERED: YES
SOURCE      MOL_ID: 1;
SOURCE      2 ORGANISM_SCIENTIFIC: HOMO SAPIENS;
SOURCE      3 ORGANISM_COMMON: HUMAN;
```

```
$ python header.py meaning_of_life
```

```
meaning_of_life not found
```

DÚ 8.2: Hledá se Nemo

Doplňte program `find.py`, který bude volán z příkazové řádky se dvěma argumenty – hledaným slovem a názvem prohledávaného souboru. Program vypíše všechny řádky ze souboru, které obsahují hledané slovo.

U každého řádku bude uvedeno číslo řádku v původním souboru. Řádky číslovte od jedničky, číslo řádku vypisujte na šířku čtyř znaků se zarovnáním doprava, za číslem vypište tři mezery a pak samotný obsah řádku.

```
$ python find.py Nemo data/CSFD/Hleda_se_Nemo.txt
```

```
2   a odlehlém příbytku ze sasanek Marlin a jeho jediný syn Nemo. Marlin se s
4   snaží svého syna před nástrahami okolí ochránit. Nemo je však, stejně jako
23  postaviček. Scénáristou a režisérem filmu Hledá se Nemo je Andrew Stanton,
27  Hledá se Nemo a jeho úžasný podmořský svět je zcela novým uměleckým a
```

DÚ 8.3: Hledá se Dory

Doplňte program `recursive_find.py`, který bude volán z příkazové řádky se dvěma argumenty – hledaným slovem a názvem prohledávané složky. Program vypíše všechny soubory ve složce, které obsahují hledané slovo. Složku bude prohledávat rekurzivně (tj. včetně podsložek, podpodsložek atd.).

Doctesty jsou udělané tak, že nahradí ve vašem výpisu `\` za `/`, takže je jedno, jestli je spouštíte na Unixu nebo Windowsu.

Tip: Použijte metodu `.glob` nebo `.rglob`. Tyto metody nezaručují pořadí souborů, proto na jejich výsledek aplikujte funkci `sorted`.

```
$ python recursive_find.py Dory data/CSFD
```

```
data/CSFD/Hleda_se_Dory.txt
```

```
data/CSFD/Hleda_se_Nemo.txt
```

DÚ 8.4: Sběr papíru

Soubor `data/paper.txt` obsahuje informace o sběru papíru – kdy, kdo, a kolik kg donesl.

Doplňte program `sum_paper.py`, který bude volán z příkazové řádky s jedním argumentem – názvem souboru s informacemi o sběru. Program načte tento soubor a spočítá celkovou hmotnost sesbíraného papíru pro každou osobu. Výsledek vypíše na standardní výstup, osoby budou seřazeny podle abecedy.

```
$ python sum_paper.py data/paper.txt
```

```
Alice: 15.0
```

```
Bob: 28.0
```

```
Cyril: 19.5
```

```
Dana: 20.0
```

```
Emil: 7.5
```

```
Filip: 8.0
```

```
Gertruda: 24.0
```

```
Hanka: 27.5
```

```
Ivan: 20.0
```

```
John: 22.0
```

DÚ 8.5: Sběr sekvencí

Formát FASTA slouží k ukládání sekvencí nukleových kyselin (DNA, RNA) a proteinů. Tento formát je velmi jednoduchý – před každou sekvencí je řádek začínající znakem `>` s názvem sekvence, pak následuje samotná sekvence – viz soubor `data/collected_seqs-expected.fasta`.

Naším úkolem je doplnit program `collect_sequences.py`, který bude volán z příkazové řádky se dvěma argumenty – vstupní složkou `X` a výstupním souborem `Y`.

Program projde všechny soubory s příponou `.txt` ve složce `X`, načte z nich sekvence a uloží je do souboru `Y` ve formátu FASTA. Název každé sekvence bude název souboru, ze kterého byla načtena (bez přípony). Sekvence budou seřazeny abecedně podle svého názvu (nejvhodnější je seřadit soubory předtím, než je budeme procházet). Program nemá nic vypisovat na standardní výstup, sesbírané sekvence má uložit souboru `Y`.

```
$ python collect_sequences.py data/seqs data/collected_seqs.fasta
```

Pokud bude program fungovat správně, měl by se obsah výstupního souboru `data/collected_seqs.fasta` shodovat s `data/collected_seqs-expected.fasta`.

Poznámka: Druhý doctest volá pomocnou funkci `diff`. Ta srovnává Váš výstupní soubor se vzorovým výstupním souborem a vrací `True` pokud oba existují a jsou stejné.

Pokud nejsou stejné, vypíše rozdíly (- znamená chybějící řádek, + řádek navíc) a vrátí False. Funkce diff neupravujte.

DÚ 8.6: Počítáme atomy

Formát PDB slouží k ukládání struktur biomolekul. Obsahuje pro každý atom v molekule jeho typ (prvek), 3D souřadnice a další informace. PDB soubory obsahují řádky s různými doplňujícími informacemi, ale řádky týkající se atomů poznáme podle toho, že vždy začínají ATOM nebo HETATM. Označení prvku je vždy na 76.-77. znaku řádku (číslováno od nuly).

Naším úkolem je doplnit program `count_atoms.py`, který bude volán z příkazové řádky s jedním argumentem - názvem PDB souboru. Program vypíše počet atomů různých typů v tomto souboru. Typy atomů seřadte abecedně.

```
$ python count_atoms.py data/ibuprofen.pdb
```

```
C: 13  
H: 18  
O: 2
```