

# C2184 Úvod do programování v Pythonu (podzim 2020)

## Povinné domácí úkoly

Úkoly v této sadě řešte do připravených souborů `monthly_temperatures.py`, `minmax.py`, `sort.py`, `hydrolase_cofactors.py`, `birthdays.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`. V některých úlohách je už připravené i načítání argumentů z příkazové řádky pomocí modulu `argparse`, u dalších si ho musíte doplnit sami. 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 9.1: Teploty v Brně

Doplňte program `monthly_temperatures.py`, který bude volán z příkazové řádky se dvěma argumenty. První argument bude název vstupního CSV souboru s tabulkou denních teplot. Druhý argument bude název výstupního souboru, do kterého program zapíše průměrné měsíční teploty (zaokrouhlené na 1 des. místo), opět ve formátu CSV.

První 4 řádky vstupního souboru jsou hlavička, v dalších řádcích je vždy rok, měsíc a teploty v jednotlivých dnech (pozor, buňky na konci mohou být prázdné, když má měsíc méně než 31 dní).

Ve výstupním souboru bude každý řádek odpovídat jednomu roku a každý sloupec jednomu měsíci. Vzorový výsledek najdete v souboru `data/Brno_monthly-expected.csv`.

```
$ python monthly_temperatures.py data/Brno_daily.csv  
data/Brno_monthly.csv
```

## DÚ 9.2: Minimum maximum

Doplňte program `minmax.py`, který bude volán z příkazové řádky se dvěma argumenty – názvem vstupního a výstupního CSV souboru.

Program načte vstupní soubor ve formátu CSV, spočítá minimum a maximum z každého sloupce a výsledek uloží do výstupního CSV souboru.

Dále může uživatel zadat:

- že první řádek tabulky ve vstupním souboru se má brát jako hlavička a tato hlavička se má zkopírovat i do výstupního souboru (volba `--header` nebo `-H`)
- jaký je oddělovač ve vstupním a výstupním souboru (volba `--delimiter` nebo `-d`, default čárka)

Čísla načítejte a vypisujte jako float. Pokud se některé hodnoty v sloupci nepodaří převést na float, nechte buňky odpovídající tomuto sloupci ve výstupní tabulce prázdné (viz první sloupec v `table1.csv`).

Příklady spuštění z příkazové řádky:

```
$ python minmax.py data/table1.csv data/minmax1.csv  
$ python minmax.py data/table2.csv data/minmax2.csv --header  
--delimiter ';'`
```

(Všimněte si, že znak středník musí být v uvozovkách. To proto, že některé znaky (`;` `|` `<` `>` `?` `*` atd.) jsou speciální znaky příkazové řádky (bashu nebo powershellu). Aby se jejich speciální funkce zrušila, zadáme je do příkazové řádky v uvozovkách. Tyto uvozovky se automaticky odstraní, tj. zadáme-li napr. `' ; '`, tak Python dostane už jen řetězec o jednom znaku – středník.)

Vzorové výstupní soubory jsou `data/minmax1-expected.csv`, `data/minmax2-expected.csv`.

### DÚ 9.3: Sort

Doplňte program `sort.py`, který bude volán z příkazové řádky s jedním argumentem – názvem vstupního souboru. Z tohoto souboru načte všechny řádky a vypíše je na standardní výstup abecedně seřazené. Dále bude možné spustit program s volbou `--reverse (-r)`, pak bude výpis řádků v opačném pořadí. Když bude program spouštěn s nesprávnými argumenty, pak má vypsát `usage:...` (toto automaticky zaručí modul `argparse`).

```
$ python sort.py data/muses.txt -r
```

```
Urania  
Thalia  
Terpsichore  
Polyhymnia  
Melpomene  
Euterpe  
Erato  
Clio  
Calliope
```

### DÚ 9.4: Kofaktory

V souboru `data/cofactors.json` jsou různé sloučeniny a k nim informace o tom, jestli jsou kofaktorem nějakého enzymu. Konkrétně každá sloučenina má v "EC" seznam enzymů, pro které je kofaktorem. (Pro přehledné zobrazení souboru ve VSCode klikněte pravým tlačítkem na text a zvolte `Format document`.)

EC čísla enzymů vyjadřují zařazení enzymů do tříd, např. enzymy začínající trojkou se nazývají *hydrolázy*.

Doplňte program `hydrolase_cofactors.py`, který načte soubor `data/cofactors.json` a vypíše všechny sloučeniny, které jsou kofaktorem nějaké hydrolázy (tj. které mají v "EC" uvedené aspoň jedno číslo začínající trojkou).

```
$ python hydrolase_cofactors.py data/cofactors.json
```

```
Phosphopantetheine  
Nicotinamide-adenine dinucleotide  
Adenosylcobalamin  
Flavin adenine dinucleotide  
Tetrahydrofolic acid  
Coenzyme A  
Flavin Mononucleotide  
Menaquinone  
Heme  
Glutathione  
S-adenosylmethionine  
Thiamine diphosphate  
Pyridoxal 5'-phosphate
```

(Zdroj dat: <https://www.ebi.ac.uk/pdbe/api/doc/compounds.html>, část Cofactors)

## DÚ 9.5: Narozeninový problém

Alice a Bob pořádali večírek, na který přišlo 30 lidí. Dva lidi na večírku zjistili, že mají narozeniny ve stejný den. „To je ale náhoda,“ říká Bob.

Ve skutečnosti je však pravděpodobnost, že mezi 30 lidmi se najdou dva se stejným dnem narozenin, nečekaně velká (kolem 70 %). Tato skutečnost se označuje jako **narozeninový paradox**.

Naším úkolem je pro zadané číslo  $n$  spočítat pravděpodobnost, že mezi  $n$  lidmi dojde ke kolizi narozenin (tj. že se najdou aspoň dva, kteří mají narozeniny ve stejný den).

Tuto úlohu budeme řešit pomocí simulace: vygenerujeme  $n$  náhodných dní v roce a podíváme se, jestli došlo ke kolizi. Toto zopakujeme dostatečný počet krát (10 000) a spočítáme, v jakém procentu pokusů došlo ke kolizi.

V souboru `birthdays.py` doplňte funkci `collision_probability`, která vezme jako argument počet lidí  $n$  a vrátí nasimulovanou pravděpodobnost, že mezi  $n$  lidmi dojde ke kolizi narozenin.

### Poznámky:

- Pro zjednodušení uvažujme, že nikdo nemá narozeniny 29. února. Dále předpokládejme, že ze zbylých 365 dní je každý den stejná pravděpodobnost narození.
- Na generování náhodných dní použijte vhodnou funkci z modulu `random`. Tip: není třeba generovat konkrétní den a měsíc (např. 11.2.), stačí nám generovat pořadové číslo dne v roce, tj. číslo od 1 do 365.
- Použijte počet pokusů = 10 000.
- Když chceme zjistit, jestli se v seznamu nacházejí dva stejné prvky, jde to zapsat velmi jednoduše pomocí funkcí `len` a `set` (nemusíme procházet všechny kombinace prvků a zkoušet jestli jsou stejné).
- Samozřejmě pokaždé, když funkci spustíme, nám může vrátit trochu jiný výsledek. Proto je v doctestech vždy uvedený přípustný interval výsledků (např. `0.65 <= collision_probability(30) <= 0.75`). Svou funkci si můžete zavolat i z příkazového řádku:

```
$ python birthdays.py 30
```