

C2184 Úvod do programování v Pythonu

Nepovinné úkoly

V úkolech v této sadě je potřeba (narozdíl od předchozích sad) vždy zadefinovat nějakou funkci. Není třeba načítat vstup pomocí funkce `input`, testy budou volat Vaši funkci tak, jak uvádí **vzorové volání**.

Pokud je uvedené **vzorové volání** a **vzorový výsledek**, chce se od Vás, aby návratovou hodnotou vzorového volání byl vzorový výsledek (`return`).

Pokud je uvedené **vzorové volání** a **vzorový výstup**, nemá funkce vracet žádnou návratovou hodnotou, ale má vypisovat vzorový výstup (`print`).

Cvičení 6.1: Objem kužele

Úkol:

Napište funkci `cone_volume`, která bere jako parametry poloměr podstavy kužele (`radius`) a jeho výšku (`height`) a vrací objem kužele. Objem kužele lze spočítat podle vzorce:

$$V = \frac{1}{3}\pi r^2 h$$

Vzorové volání 1:	Vzorové volání 2:
<code>cone_volume(10, 5)</code>	<code>cone_volume(1.5, 3.2)</code>
Vzorový výsledek 1:	Vzorový výsledek 2:
523.5987755982989	7.5398223686155035

```
[ ]: def cone_volume(radius: float, height: float) -> float:
    ...

# print(cone_volume(10, 5))
# print(cone_volume(1.5, 3.2))
```

Cvičení 6.2: Dělitelé

Úkol:

Napište funkci `divisors`, která bere jako parametr přirozené číslo `n` a vrací seznam všech dělitelů čísla `n` (v pořadí od nejmenšího).

Funkce navíc bere nepovinný parametr `proper`:

- Pokud je `proper` nastaveno na `False` (default), pak funkce vrací všechny dělitele, včetně samotného `n`.
- Pokud je `proper` nastaveno na `True`, pak funkce vrací pouze *vlastní dělitele*, tj. všechny dělitele kromě samotného `n`.

Poznámka: na hledání dělitelů existují efektivnější algoritmy, ale v této úloze úplně postačí naivní řešení, tj. zkusit všechna čísla od 1 až po `n`.

Vzorové volání 1: <code>divisors(12)</code>	Vzorové volání 2: <code>divisors(30, proper=True)</code>
Vzorový výsledek 1: <code>[1, 2, 3, 4, 6, 12]</code>	Vzorový výsledek 2: <code>[1, 2, 3, 5, 6, 10, 15]</code>

```
[ ]: def divisors(n: int, proper: bool = False) -> list[int]:  
    ...  
  
    # print(divisors(12))  
    # print(divisors(30, proper=True))
```

Cvičení 6.3: Společná písmena

Úkol:

Napište definici funkce `common_letters`, která přijímá seznam řetězců. Funkce vrátí množinu znaků, které se vyskytují v každém z těchto řetězců. Pokud bude funkce volána na prázdném seznamu, vrátí prázdnou množinu.

Nápověda: při řešení lze využít množinové operace (průnik `intersection (&)`, sjednocení `union (|)`...).

Vzorové volání 1: <code>common_letters(['mrkev', 'krkavec', 'krabice'])</code>	Vzorové volání 2: <code>common_letters([])</code>
Vzorový výsledek 1: <code>{'e', 'k', 'r'}</code>	Vzorový výsledek 2: <code>set()</code>

```
[ ]: def common_letters(words: list[str]) -> set[str]:  
    ...  
  
    # print(common_letters(['mrkev', 'krkavec', 'krabice']))
```

```
# print(common_letters([]))
```

Cvičení 6.4: Úhel

Mějme dva vektory v rovině: $\vec{u} = (u_x, u_y)$, $\vec{v} = (v_x, v_y)$.

Pak velikost úhlu α mezi těmito vektory umíme spočítat podle vzorce

$$\cos(\alpha) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| |\vec{v}|}$$

V tomto vzorci $\vec{u} \cdot \vec{v}$ značí skalární součin vektorů a $|\vec{u}|$ značí velikost vektoru.

$$\vec{u} \cdot \vec{v} = u_x v_x + u_y v_y$$

$$|\vec{u}| = \sqrt{\vec{u} \cdot \vec{u}}$$

Úkol:

Napište funkci `angle`, která spočítá velikost úhlu (v radiánech) mezi dvěma zadanými vektory. Vektory budou reprezentovány dvojicemi reálných čísel.

Řešení je vhodné rozdělit na jednodušší podúkoly, například definovat nejdřív funkce `dot` (skalární součin) a `size` (velikost) a ty pak vhodně volat ve funkci `angle`.

Vzorové volání 1:	Vzorové volání 2:
<code>angle((1, 1), (2, -2))</code>	<code>angle((1.5, 2.2), (5.2, 4.8))</code>
Vzorový výsledek 1:	Vzorový výsledek 2:
1.5707963267948966	0.22695795704220176

Pro kontrolu: velikost vektorů ve vzorovém volání 1 má vyjít zhruba 1.414 a 2.828, skalární součin 0.0.

```
[ ]: def angle(u: tuple[float, float], v: tuple[float, float]) -> float:
    ...

def dot(u: tuple[float, float], v: tuple[float, float]) -> float:
    ...

def size(u: tuple[float, float]) -> float:
    ...

# print(angle((1, 1), (2, -2)))
# print(angle((1.5, 2.2), (5.2, 4.8)))
```