Python akademie - lekce 3 - 31.10.2024

03_02 🎾 Sety (množiny)

Zajímavé odkazy z této lekce:

- Oficiální dokumentace ke setům
- Seznam všech metod souvisejících se sety
- Odkaz na materiál k datovém typu frozenset

---

syntaxe:

```
muj_set = {"zena", "ruze", "pisen", "kost"}
```

- v Pythonu **standartní datový typ**,
- tvořený **unikatními hodnotami**,
- nepracuje s jednotlivými údaji, ale s daty jako celkem,
- hodnoty mohou být jak stringy, tak číselné hodnoty,
- nemá pořadí (podobné slovníkům),
- klíčové operace setů:
  - sjednocení `|`,
  - průnik `&`,
  - rozdíl `-`,
  - symetrický rozdíl `^`.

Nový set

```
muj_novy_set = set()
```

```
type(muj_novy_set)
```

```
set
```

```
muj_druhy_set = {}
```

```
type(muj_druhy_set)
```

```
dict
```

```
muj_druhy_set = set{}
```

```
  File "C:\Users\Radim Jedlicka\AppData\Local\Temp/ipykernel_23400/3541079430.py", line 1
    muj_druhy_set = set{}
                        ^
SyntaxError: invalid syntax
```

```
muj_novy_set = {"zena", "ruze", "pisen", "kost"}
```

```
type(muj_novy_set)
```

```
set
```

Metody setu

Podobně jako předchozí datové struktury (`list`, `tuple`, `dict`), mají i sety specifické metody. Níže je opět vypsaná tabulka s těmi častějšími (ne všemi):

Metoda	Použití
`add`	přidá hodnotu do setu
`discard`	odstraní hodnotu ze setu
`pop`	odstraní hodnotu ze setu
`remove`	odstraní hodnotu ze setu
`clear`	odstraní všechny hodnoty ze setu
`copy`	vytvoří *shallow copy* setu

Metoda	Použití
`difference`	vytvoří set obsahující rozdílné hodnoty ze dvou setů
`difference_update`	odstraní všechny hodnoty obsažené ve druhém setu z prvního setu
`intersection`	vytvoří set obsahující identické hodnoty ze dvou setů
`intersection_update`	aktualizuje stávající set s hodnotami z průniku s jiným setem
`union`	vrátí nový set jako spojení dvou původních setů
`symmetric_difference`	vrátí speciální objekt se všemi páry (v tuple)
`symmetric_difference_update`	aktualizuje stávající set s hodnotami z sym. rozdílu s dalším setem
`isdisjoint`	vrací `True` pokud dva sety nemají průnik. Jinak `False`
`issubset`	vrací `True` pokud všechny prvky 1. setu jsou obsaženy ve 2. setu. Jinak `False`

Procvičování metod

```
muj_set_A = {"zena", "ruze", "pisen", "kost"}
```

```
print(muj_set_A)
```

```
{'zena', 'pisen', 'ruze', 'kost'}
```

```
id(muj_set_A)
```

```
2479113769408
```

```
muj_set_B = muj_set_A.copy()
id(muj_set_B)
```

```
2479113768736
```

```
print(muj_set_A)
```

```
{'zena', 'pisen', 'ruze', 'kost'}
```

```
print(muj_set_B)
```

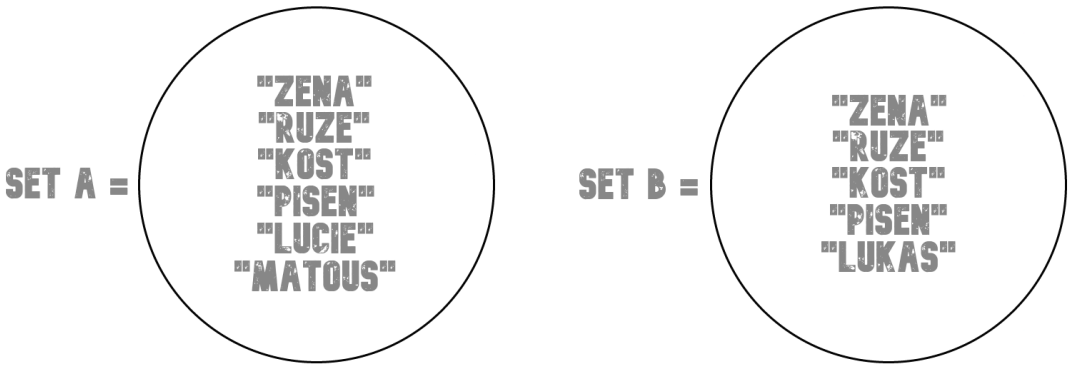
```
{'zena', 'pisen', 'ruze', 'kost'}
```

```
muj_set_A.add("Matous")
muj_set_A.add("Lucie")
muj_set_B.add("Lukas")
```

```
muj_set_A.add("Matous")
```

```
muj_set_A
```

```
{'Lucie', 'Matous', 'kost', 'pisen', 'ruze', 'zena'}
```



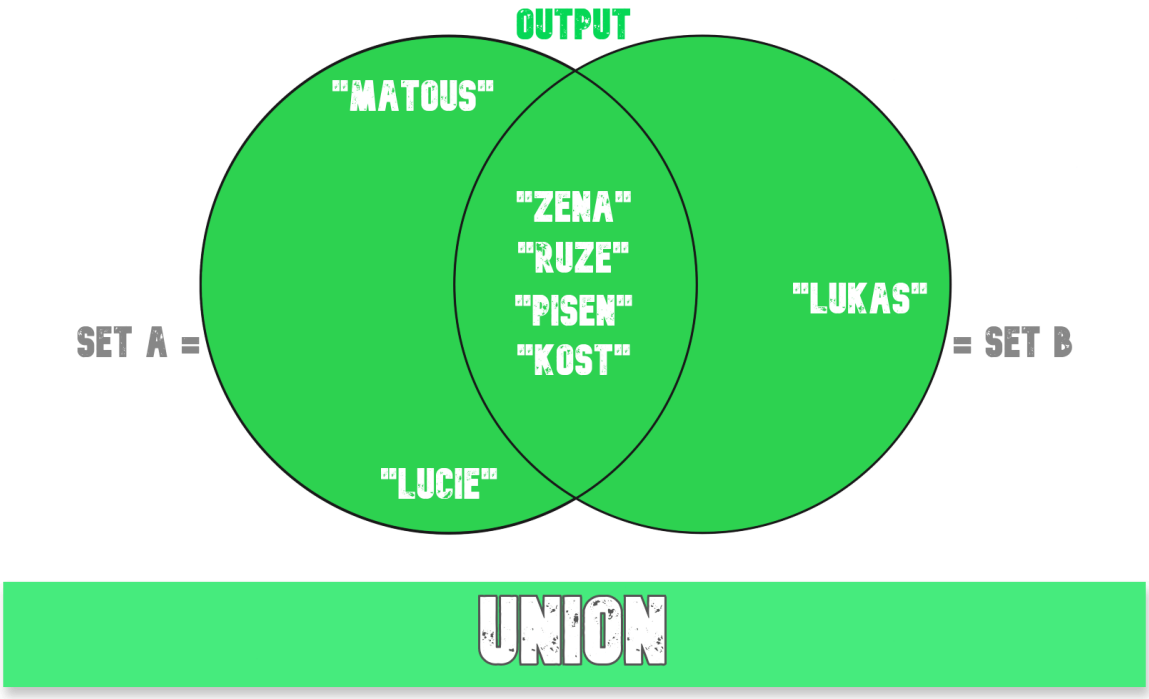
OPERACE SE SETY

```
print(muj_set_B)
```

```
{'kost', 'pisen', 'zena', 'ruze', 'Lukas'}
```

```
print(muj_set_A)
```

```
{'kost', 'Matous', 'Lucie', 'zena', 'ruze', 'pisen'}
```




```
print(muj_set_B.union(muj_set_A))
```

```
{'kost', 'Matous', 'Lucie', 'pisen', 'zena', 'ruze', 'Lukas'}
```

```
print(muj_set_A.union(muj_set_B))
```

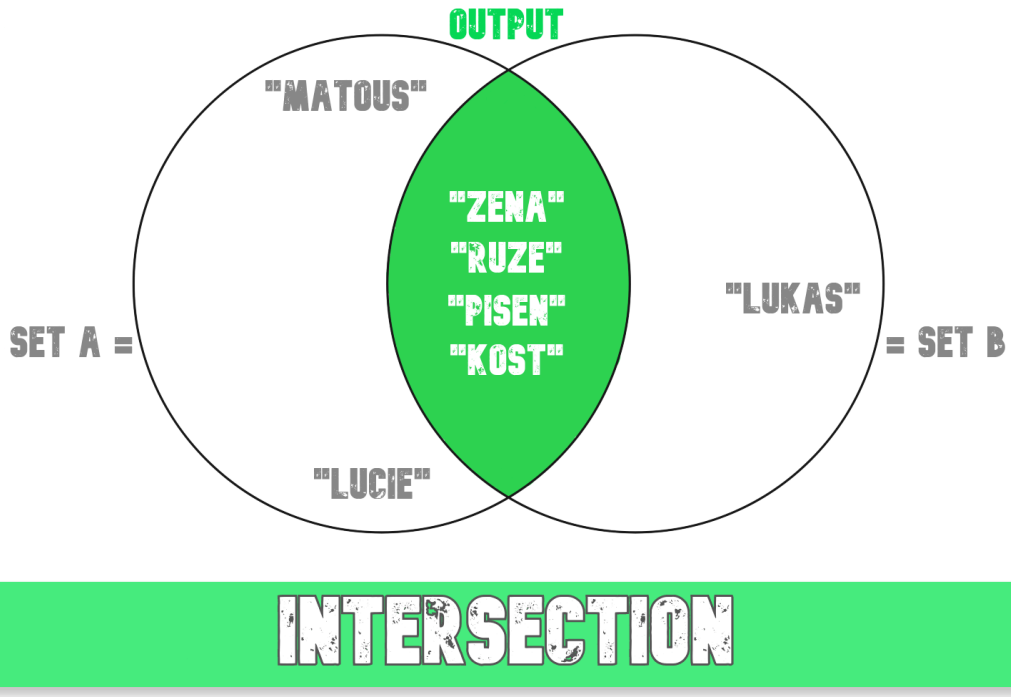
```
{'kost', 'Matous', 'Lukas', 'Lucie', 'zena', 'ruze', 'pisen'}
```

```
muj_set_B.union(muj_set_A) == muj_set_A.union(muj_set_B)
```

```
True
```

```
muj_set_A | muj_set_B
```

```
{'Lucie', 'Lukas', 'Matous', 'kost', 'pisen', 'ruze', 'zena'}
```




```
print(muj_set_A.intersection(muj_set_B))
```

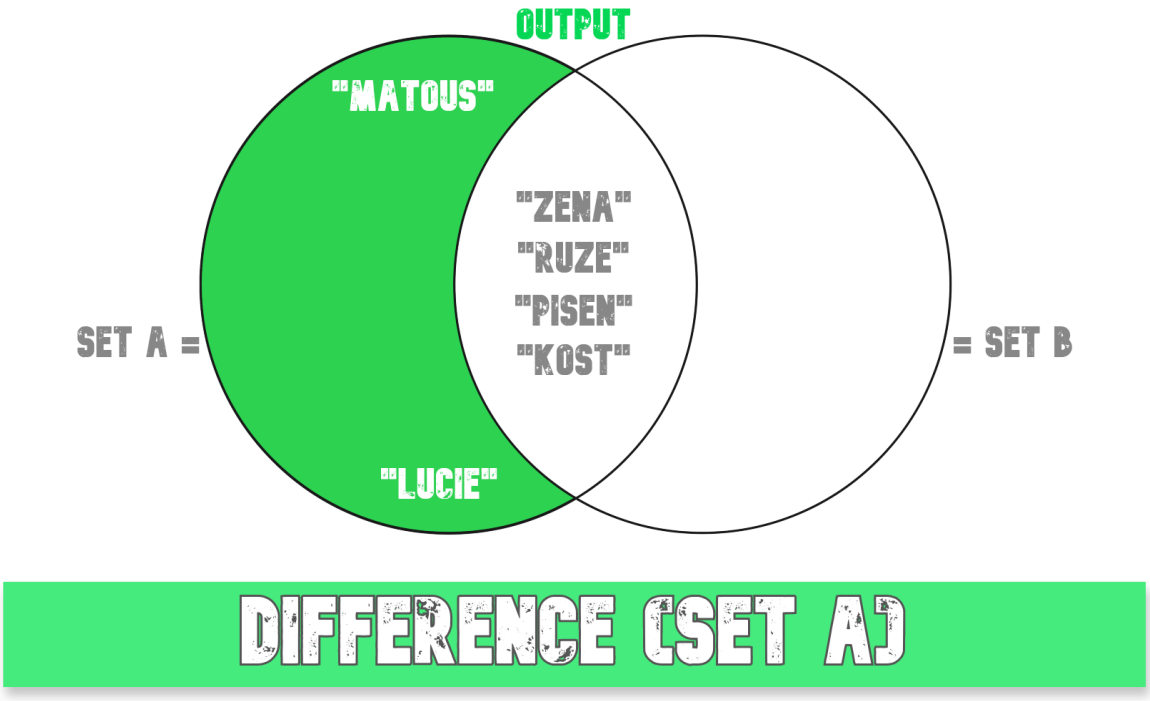
```
{'kost', 'ruze', 'zena', 'pisen'}
```

```
print(muj_set_B.intersection(muj_set_A))
```

```
{'kost', 'ruze', 'zena', 'pisen'}
```

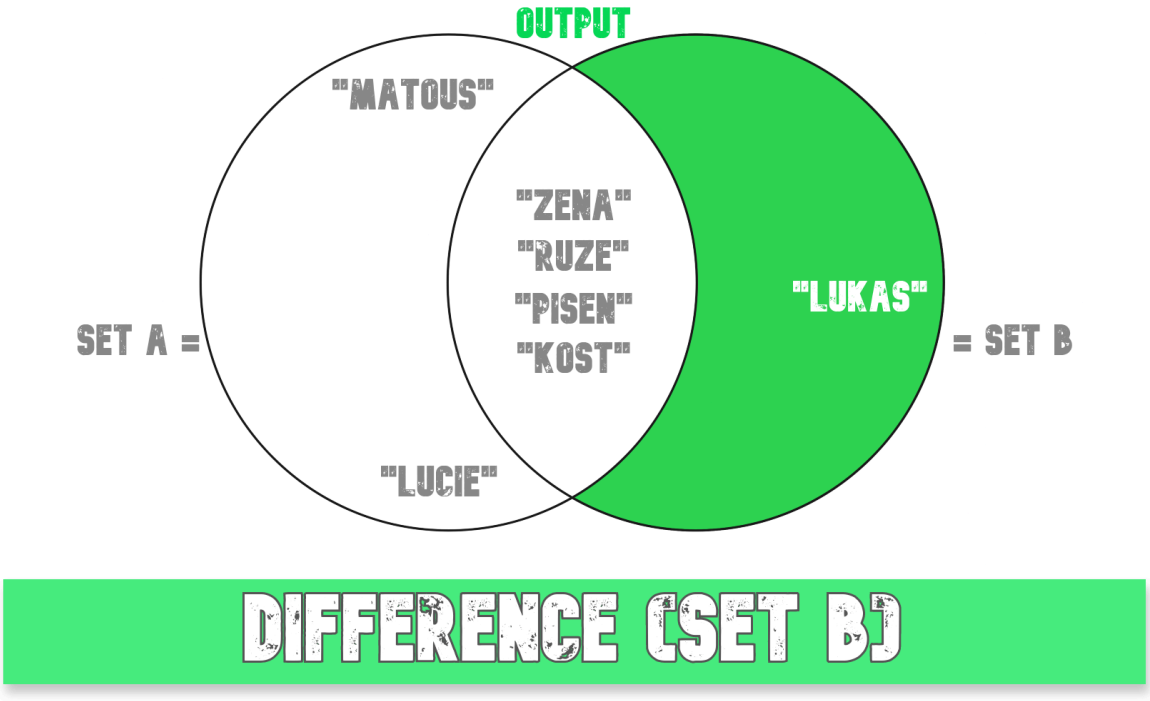
```
muj_set_A & muj_set_B
```

```
{'kost', 'pisen', 'ruze', 'zena'}
```



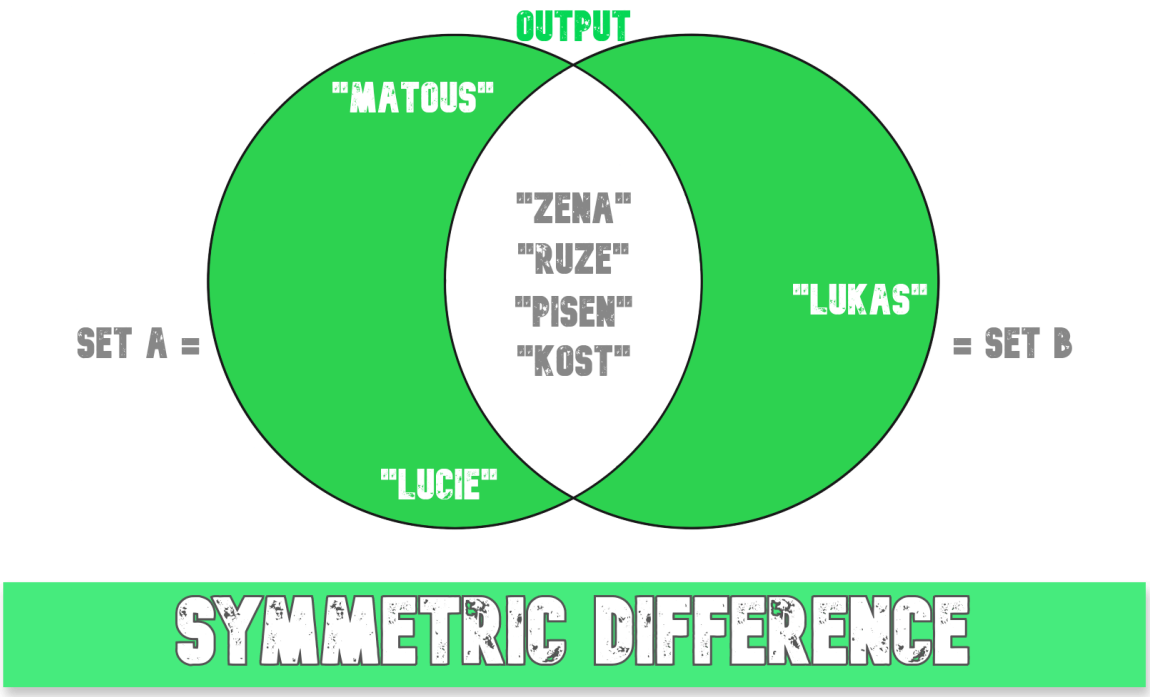

```
print(muj_set_A.difference(muj_set_B))
```

```
{'Matous', 'Lucie'}
```




```
print(muj_set_B.difference(muj_set_A))
```

```
{'Lukas'}
```




```
print(muj_set_B.symmetric_difference(muj_set_A))
```

```
{'Lucie', 'Matous', 'Lukas'}
```

```
print(muj_set_A.symmetric_difference(muj_set_B))
```

```
{'Lukas', 'Lucie', 'Matous'}
```

```
muj_set_A ^ muj_set_B
```

```
{'Lucie', 'Lukas', 'Matous'}
```

Další metody

```
prikl_1 = {"jedna", "dve", "tri"}
```

```
prikl_2 = {"ctyri", "pet"}
```

isdisjoint

Vrací `True` pokud žádný prvek není společný pro obě množiny. Jinak vrací `False`.

```
print(prikl_1.isdisjoint(prikl_2))
```

```
True
```

```
print(muj_set_B)
```

```
{'kost', 'pisen', 'zena', 'ruze', 'Lukas'}
```

```
print(muj_set_A)
```

```
{'kost', 'Matous', 'Lucie', 'zena', 'ruze', 'pisen'}
```

```
muj_set_A.isdisjoint(muj_set_B)
```

```
False
```

issubset

Vrací `True` pokud všechny hodnoty v setu existují i ve specifikovaném setu. Jinak vrací `False`.

```
print(muj_set_A.issubset(muj_set_B))
```

```
False
```

```
prikl_1 = {"jedna", "dve", "tri"}
```

```
prikl_2 = {"jedna", "dve"}
```

```
print(prikl_2.issubset(prikl_1))
```

```
True
```

```
print(prikl_1.issubset(prikl_2))
```

```
False
```

Frozenset

```
frozenset({'ruze', 'zena', 'Jan', 'pisen', 'kost'})
```

```
frozenset({'Jan', 'kost', 'pisen', 'ruze', 'zena'})
```

Pomocí funkce `frozenset` máme jako uživatelé Pythonu možnost vytvořit **nezměnitelný** `set` (princip podobný jako u `tuple`).

```
muj_set = frozenset()
```

```
type(muj_set)
```

```
frozenset
```

Nový frozenset

```
muj_set = {"zena", "ruze", "pisen", "kost"}
```

```
print(type(muj_set))
```

```
muj_nezm_set = frozenset(muj_set.copy())
```

```
print(type(muj_nezm_set))
```

```
print(type(muj_set))
```

Metody frozensetu