



OSNOVA

< PŘEHLED KURU

49% hotovo z Lekce 3

Vyhledávat



ÚVOD DO TŘETÍ LEKCE

Obsah

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / ÚVOD DO TŘETÍ LEKCE / OBSAH

V této lekci tě čeká následující:

1. Datový typ **slovník** a jeho metody,
2. datový typ **množina** a její metody,
3. datový typ **frozenset** a jeho metody,
4. opakovací cvičení,
5. úloha *Filmový slovník*.

SLOVNÍKY

Úvod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / ÚVOD

Slovník je datový typ, který je tvořený různým počtem **párů dat**. Pár se vždy skládá ze **jména klíče** a jeho **hodnoty**.

Dále si určitě všimni **dvojtečky** mezi *klíčem a hodnotou*.

```
>>> muj_slovnik = {"jmeno": "Matous", "registrovany": True, "vek": 100}
```

V příkladu si můžeme všimnout těchto **charakteristických rysů** pro *slovník*:

1. **Složené závorky** na začátku a na konci slovníku,
2. **čárek**, které oddělují jednotlivé páry,
3. **klíče** jsou stringy (`"jmeno"`, `"registrovany"`, `"vek"`),
4. **hodnoty** jsou různého datového typu (`str`, `bool`, `int`),
5. **proměnná**, do které si nově napsaný slovník schováme (`muj_slovnik`).

Vyzkoušej si funkci `type` na nově vytvořený slovník. Uvidíš, že výstup bude obsahovat zkratku `dict`, tedy **dictionary** což znamená *slovník*.

```
1 # Spusť si ukázku a sleduj výstup
2 muj_slovnik = {"jmeno": "Matous", "registrovany": True, "vek": 100}
3 print(type(muj_slovnik)) # výstupem bude "<class 'dict'>"
```

SPUSTIT KÓD

Zde se zobrazí výstup po spuštění kódu.

Vytvoření nového slovníku

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / VYTVOŘENÍ NOVÉHO SLOVNÍKU

Prázdný slovník

Jelikož je slovník změnitelný (*~mutable*) datový typ, vytvoříme nejprve prázdný slovník:

1. Pomocí **prázdných složených závorek**,

```
>>> prvni_slovnik = {}
>>> type(prvni_slovnik)
<class 'dict'>
```

2. Pomocí zabudované **funkce `dict()`**.

```
>>> druhyslovnik = dict()
>>> type(druhy_slovnik)
<class 'dict'>
```

Funkce **`type`** nám současně potvrdí, že nově vytvořený objekt skutečně odpovídá datovému typu **`dict`**.

Slovník s hodnotami

Nyní vytvoříme slovník s dvěma klíči, opět dvěma různými způsoby:

1. Pomocí **složených závorek**,

```
>>> treti_slovnik = {"jmeno": "Marek", "vek": 20}
>>> print(treti_slovnik)
{'jmeno': 'Marek', 'vek': 20}
```

2. Pomocí zabudované **funkce `dict()`**.

```
>>> ctvrty_slovnik = dict(jmeno="Lukas", vek=25)
>>> print(ctvrty_slovnik)
{'jmeno': 'Lukas', 'vek': 25}
```

Obecně častěji se používá první způsob, pomocí složených závorek.

Jak najdu hodnotu

Se slovníkem pracuješ tak, že pomocí **jména klíče** vyhledáš jeho **hodnotu**.

Tento proces se označuje jako **mapování**. Jméno klíče potom zapisujeme do hranatých závorek, podobně jako u *indexování*.

```
>>> treti_slovnik = {"jmeno": "Marek", "vek": 20}
>>> treti_slovnik["jmeno"]
'Marek'
```

Mapování ale není to stejné, co **indexování**. Indexování je založené na pořadí údajů v objektu. Klíče ve slovnících **nemají pořadí**.

Jak probíhá zápis klíče

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / JAK PROBÍHÁ ZÁPIS KLÍČE

Pravidla zápisu klíče

Podobně jako u vytvoření jména proměnné, existují pravidla pro vytvoření klíče ve slovníku. Tentokrát jich ale nebude tolik.

Při tvorbě **nového klíče**, pomáhá na pozadí funkce `hash`, která převádí tebou zadaný údaj na celé číslo:

```
>>> hash("jmeno")
6241779474799247831
>>> hash(11234)
11234
>>> hash(True)
1
```

Všimni si, že funkce `hash` nepracuje s každým datovým typem:

```
>>> hash(["a", "b"])
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Pokud funkci `hash` zadáš **změnitelný datový typ** (jako `list` v ukázce), vypíše ti chybu. Umí pracovat pouze s **nezměnitelnými datovými typy**, které dovede převést na celé číslo.

Jestli si tedy nebudeš jistý, můžeš pomocí funkce `hash` vyzkoušet, jestli je tebou vybraný klíč správný.

Uložení klíčů a hodnot

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / ULOŽENÍ KLÍČŮ A HODNOT

Unikátní klíče

Pojmenovat klíče už umíš. Dále budeš potřebovat, tebou vybrané jméno klíče, uložit do slovníku. V jednom slovníku nemohou být dva stejné klíče.

Dávej pozor, ať nezapisuješ žádné jméno klíče ve slovníku dvakrát, jinak si přepíšeš původní hodnotu.

V ukázce niže vytvoříme slovník `muj_slovnik` s klíčem `jmeno`. Tento klíč má nejprve hodnotu `"Matous"`. Dále budeme chtít vytvořit druhý klíč `jmeno` s hodnotou `"Lukas"`.

```
>>> muj_slovnik = {}
>>> muj_slovnik["jmeno"] = "Matous"
>>> print(muj_slovnik)
{'jmeno': 'Matous'}
>>> muj_slovnik["jmeno"] = "Lukas"
>>> print(muj_slovnik)
{'jmeno': 'Lukas'}
```

Funkce `hash` totiž vytvoří v obou případech stejně celé číslo pro klíč `jmeno`, takže ve výsledku jenom přepíšeme původní klíč s novou hodnotou `Lukas`.

Zapisování hodnot

Ohledně zapisování hodnot ke klíčům není slovník tak striktní. Vložit můžeš prakticky jakoukoliv hodnotu:

```
>>> muj_slovnik = {}
>>> muj_slovnik["jmeno"] = "Matous"
>>> muj_slovnik["vek"] = 100
>>> muj_slovnik["ridicske opravneni"] = True
>>> muj_slovnik["kontaktni_udaje"] = ("+420582582582", "matous@matous.cz")
```

K hodnotám potom můžeš přistupovat pomocí jména klíče:

```
>>> muj_slovník["jmeno"]
'Matous'
>>> muj_slovník["vek"]
100
>>> muj_slovník["kontaktní_udaje"]
('+420582582582', 'matous@matous.cz')
```

Jako hodnotu tedy můžeš použít všechny uvedené datové typy výše.

```
1 # Vytvoříme nový prázdný slovník
2 muj_slovník = {}
3
4 # Tady vyzkoušej vložit libovolné klíče a hodnoty
5 muj_slovník['povidla'] = 'knedliky'
6
7 # Vypiš obsah tvého slovníku po změnách
8 print(muj_slovník)
```

SPUSTIT ZNOVU

```
Traceback (most recent call last):
  File <string> line 5, in <module>
NameError: name 'povidla' is not defined
```

Vnořená datová struktura

Možná tě při pohledu na předchozí ukázkou napadne, že klíč `kontaktní_udaje` není příliš praktický.

Pokud by chtěl kdokoliv získat z kontaktních údajů jen telefonní číslo, musel by vybírat z klíče `kontaktní_udaje` kontaktní číslo nebo email pomocí indexů. V opačném případě výpisu klíče `kontakt` všechny hodnoty.

Určitě by bylo pohodlnější vybrat slovník s kontaktními údaji a pomocí označení klíče `mobil` nebo `email` získat příslušnou hodnotu.

Jako hodnotu klíče ve slovníku můžeš zapsat i samotný slovník. Celému procesu se říká **nestování** (~vnoření). Nejprve vytvoříme nový slovník s kontaktními údaji:

```
>>> kontakty = {"mobil": "+420582582582", "email": "matous@matous.cz" }
```

Přepíšeme původní `tuple` v klíci `kontakt` s novým slovníkem:

```
>>> muj_slovník["kontakt"] = kontakty
```

Všimni si, jak je klíč `"kontakt"` zapsaný v uvozovkách (jde tedy o `str`), zatímco na pravé straně od `=` je proměnná `kontakty` (`dict` vytvořený o pár řádků dříve).

Celý zápis si můžeš sám vyzkoušet:

```
1 # Vytvoříme nový prázdný slovník a vložíme do něj kontakty
2 muj_slovník = {}
3 kontakty = {"mobil": "+420582582582", "email": "matous@matous.cz" }
4 muj_slovník["kontakt"] = kontakty
5
6 # Zobrazím všechny kontakty
7 print(muj_slovník["kontakt"])
8
9 print(muj_slovník)
10
11 # Zobrazím mobil
12 print(muj_slovník["kontakt"]["mobil"])
13
14 # Zobrazím email
```

```
15 print(muj_slovník["kontakt"]["email"])

SPUSTIT ZNOVU
```

```
{'mobil': '+420582582582', 'email': 'matous@matous.cz'}
{'kontakt': {'mobil': '+420582582582', 'email': 'matous@matous.cz'}}
+420582582582
matous@matous.cz
```

Metody slovníků

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / METODY SLOVNÍKŮ

Často budeš potřebovat při práci se slovníky pomoci. V takovém momentě je vhodné zkontrolovat, jestli nenajdeš nějakou šikovnou existující metodu.

Metoda	Použití
<code>clear</code>	odstraní všechny údaje ze slovníku
<code>copy</code>	vráti kopii slovníku
<code>fromkeys</code>	vráti slovník z předepsaných klíčů a hodnot
<code>get</code>	vráti hodnotu, pokud existuje zadaný klíč
<code>items</code>	vráti objekt obsahující <code>tuple</code> za každý pár (klíč a hodnota)
<code>keys</code>	vráti objekt obsahující všechny klíče slovníku
<code>pop</code>	pomocí zadaného jména klíče odstraň tento klíč a jeho hodnotu
<code>popitem</code>	odstraň poslední přidaný klíč a jeho hodnotu
<code>setdefault</code>	vráti hodnotu zadaného klíče, pokud neexistuje vytvoří jej
<code>update</code>	aktualizuje slovník s novými páry (klíče i hodnoty)
<code>values</code>	vráti objekt obsahující všechny hodnoty

Ukázky metod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SLOVNÍKY / UKÁZKY METOD

Metoda `.items()`

Pokud budeš potřebovat pracovat jak s klíči, tak s hodnotami ve slovníku současně, můžeš si pomocí metodu `.items()`.

```
1 # Metoda 'items' vráci speciální objekt obsahující tuple pro každý klíč a hodnotu
2 muj_slovník = {"jmeno": "Lukas", "email": "lukas@gmail.com"}
3 print(muj_slovník.items())
```

SPUSTIT ZNOVU

```
dict_items([('jmeno', 'Lukas'), ('email', 'lukas@gmail.com')])
```

Metoda .get()

Může se stát, že potřebuješ získat ze slovníku jistý klíč. Současně ale jistě nevíš, jestli tento klíč vůbec existuje.

Kvůli takovým situacím se ti může hodit metoda `.get()`.

```
1 # Metoda 'get' vrací hodnotu zadaného klíče, pokud existuje
2 muj_slovnik = {"jmeno": "Lukas", "email": "lukas@gmail.com"}
3 print(muj_slovnik["jmeno"])
4 print(muj_slovnik.get("adresa"))
5 print(muj_slovnik.get("vek", "Klic neexistuje!"))
```

SPUSTIT ZNOVU

```
Lukas
None
Klic neexistuje!
```

Metoda .pop()

Pokud budeš potřebovat explicitně odstranit existující klíč, můžeš použít metodu `.pop()`.

```
1 # Metoda 'pop' odstraní podle zadaného jména klíče samotný klíč i jeho hodnotu
2 muj_slovnik = {"jmeno": "Lukas", "email": "lukas@gmail.com"}
3 print(muj_slovnik.pop("email"))
4 print(muj_slovnik)
```

SPUSTIT KÓD

Zde se zobrazí výstup po spuštění kódu.

Kvíz

 Quiz finished!

You scored 2 / 4

SETY

Úvod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SETY / ÚVOD

set (~anglicky, česky *množina*) je datový typ, který je určený pro **množinové operace**.

Na první pohled se od **slovníku** liší jen velice málo. Jeho praktické využití je však naprosto odlišné.

Zatímco u **slovníku** pomocí klíčů vyhledáváš hodnoty, u **setů** pracuješ především s množinovými operacemi.

Třeba pokud potřebuješ najít *společné*, případně *rozdílné* členy ve dvou množinách a více množinách. Současně se chceš vyhnout duplicitám mezi hodnotami.

Pojďme se nejprve zaměřit na to, jak se vůbec **set** vypadá:

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

V příkladu si můžeme všimnout těchto **charakteristických rysů** pro *set*:

1. **Složené závorky** na začátku a na konci setu,
2. jednotlivé hodnoty jsou datového typu **str** (`"zena"`, `"ruze"`, `"pisen"`, `"kost"`),
3. **čárky**, které oddělují jednotlivé údaje,
4. **proměnná**, do které si nově napsaný slovník schováme (`muj_set`).

Vyzkoušej si funkci **type** na nově vytvořený set. Uvidíš, že výstup bude obsahovat **set**, což znamená *množina*.

```
1 muj_set = {"zena", "ruze", "pisen", "kost"}
2 print(type(muj_set))
```

```
<class 'set'>
```

Vytvoření nového setu

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SETY / VYTVOŘENÍ NOVÉHO SETU

Prázdný set

Pokud potřebuješ vytvořit nový a prázdný `set`, musíš použít k tomuto účelu připravenou funkci (tentokrát nelze napsat prázdné složené závorky, ty tvoří nový slovník):

```
>>> první_set = set()  
>>> type(první_set)  
<class 'set'>
```

Funkce `type` nám současně potvrdí, že nově vytvořený objekt skutečně odpovídá datovému typu `set`.

Set, stejně jako slovník můžeš po vytvoření změnit (*mutable* ~ změnitelný). Jak hodnoty přidávat a odebírat si ukážeme za chvíli u **metod setu**.

Set s hodnotami

Hodnoty, které budeš chtít schovat do setu prostě odděliš čárkami a vložíš do složených závorek.

```
>>> druhy_set = {"predseda", "soudce"}  
>>> type(druhy_set)
```

Při tvorbě neprázdného setu můžeš použít již tobě známe datové typy jako `list` a `tuple`:

```
>>> muj_list = ["mesto", "more", "kure", "staveni"]  
>>> muj_tupl = ["pan", "hrad", "muz", "stroj"]  
>>> treti_set = set(muj_list)  
>>> ctvrty_set = set(muj_tupl)  
>>> type(treti_set)  
<class 'set'>  
>>> type(ctvrty_set)  
<class 'set'>
```

Protože účelem **setu** není práce s jednotlivými hodnotami uloženými uvnitř, pracuješ s ním nejčastěji pomocí metod.

Metody setů

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SETY / METODY SETŮ

Metoda	Použití
<code>add</code>	přidá údaj do setu
<code>clear</code>	odstraní všechny údaje ze setu
<code>copy</code>	vytvoří kopii setu
<code>difference</code>	vytvoří kopii setu, která obsahuje rozdílné údaje ze dvou a více setů
<code>difference_update</code>	odstraní hodnoty z výchozího setu, které jsou současně v zadaném setu
<code>discard</code>	odstraní zadanou hodnotu
<code>intersection</code>	vytvoří kopii setu, která obsahuje hodnoty společné pro výchozí set a zadáný set
<code>intersection_update</code>	odstraní hodnoty, které nejsou současně ve výchozímu setu a v zadaném setu
<code>isdisjoint</code>	vrátí <code>True</code> , pokud mají dva sety průnik hodnot. Jinak vrátí <code>False</code>
<code>issubset</code>	vrátí <code>True</code> , pokud je výchozí set součástí zadaného setu. Jinak

<code>issubset</code>	vrátí <code>False</code>
<code>issuperset</code>	vrátí <code>True</code> , pokud je zadaný set součástí výchozího setu. Jinak vrátí <code>False</code>
<code>pop</code>	odstraní zadanou hodnotu ze setu
<code>remove</code>	odstraní zadanou hodnotu ze setu
<code>symmetric_difference</code>	vytvoří kopii setu, která je tvořena symetrickým rozdílem hodnot dvou setů
<code>symmetric_difference_update</code>	vytvoří set ze symetrického rozdílu hodnot dvou setů
<code>union</code>	vytvoří kopii setu obsahující hodnoty dvou různých setů
<code>update</code>	vloží do setu jiný set, nebo objekt jako list či tuple

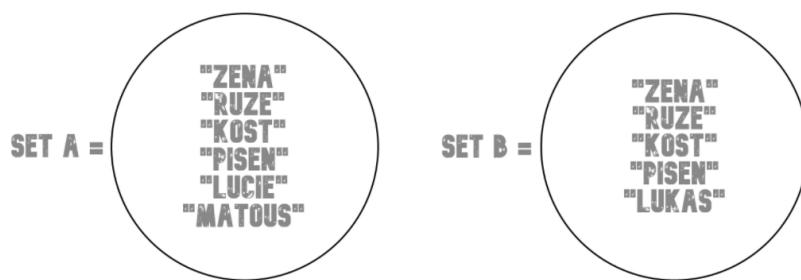
Ukázky metod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SETY / UKÁZKY METOD

Úvod

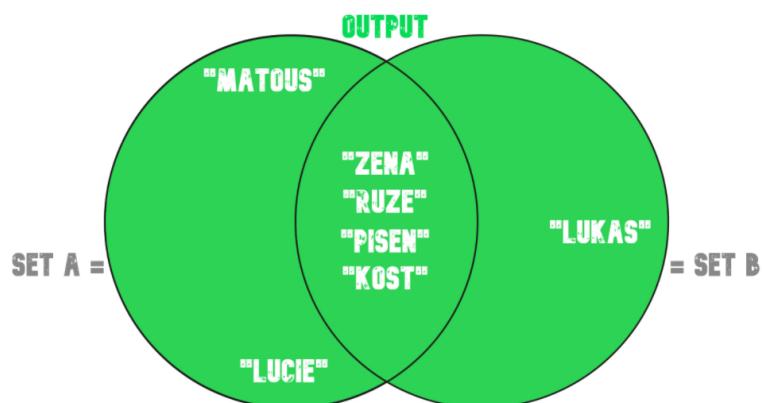
Ukážeme si práci se setovými operacemi na dvou obecných setech:

```
set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
```



OPERACE SE SETY

Sjednocení setů



```

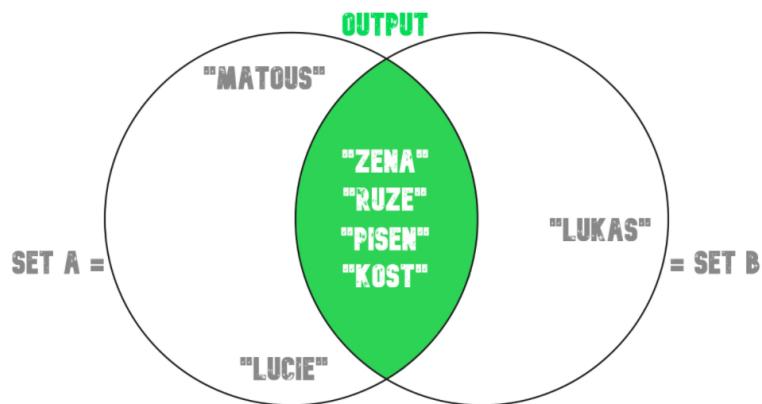
1 # Sjednocení setu A a setu B
2 set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
3 set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
4 print(set_a.union(set_b))

```

SPUSTIT ZNOVU

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

Průnik setů



```

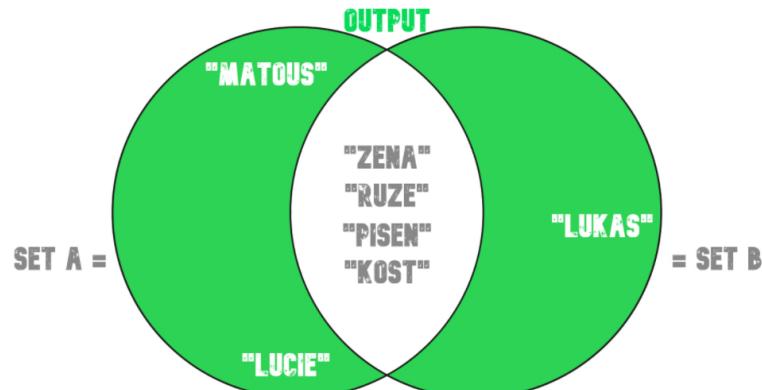
1 # Průnik setu A se setem B
2 set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
3 set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
4 print(set_a.intersection(set_b))

```

SPUSTIT ZNOVU

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

Symetrický rozdíl setů



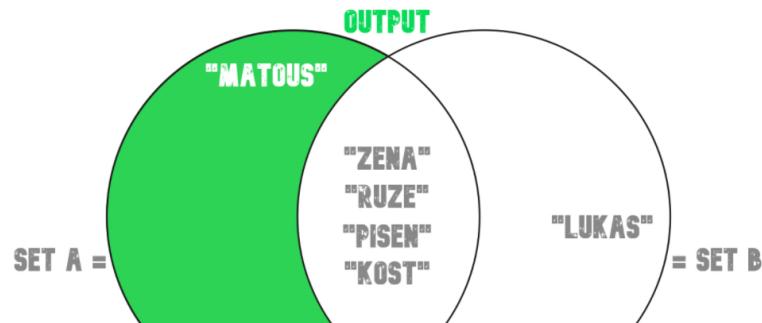
SYMMETRIC DIFFERENCE

```
1 # Symetrický rozdíl setu A a setu B
2 set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
3 set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
4 print(set_a.symmetric_difference(set_b))
```

SPUSTIT ZNOVU

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

Rozdíl setu A od setu B



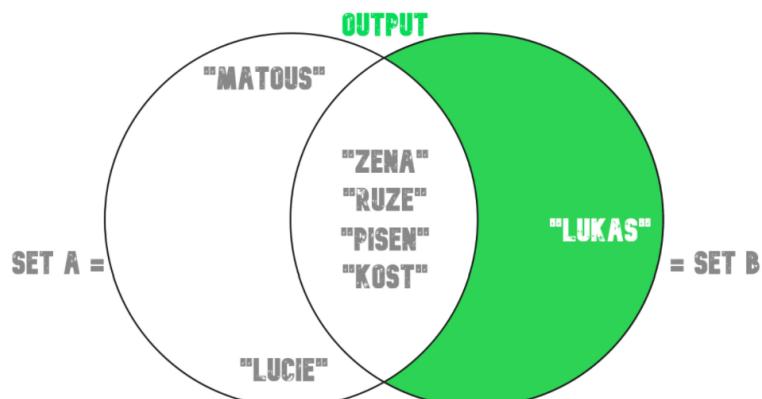
DIFFERENCE (SET A)

```
1 # Rozdíl setu A od setu B
2 set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
3 set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
4 print(set_a.difference(set_b))
```

SPUSTIT ZNOVU

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

Rozdíl setu B od setu A



DIFFERENCE (SET B)

```
1 # Rozdíl setu B od setu A
2 set_a = {"zena", "ruze", "kost", "pisen", "Lucie", "Matous"}
3 set_b = {"zena", "ruze", "kost", "pisen", "Lukas"}
4 print(set_b.difference(set_a))
```

SPUSTIT ZNOVU

```
{'Lukas'}
```

Přidávání a odebírání hodnot

Do sestu můžeš vložit hodnotu pomocí metody `add`, pokud chceš přidat jednu hodnotu.

Pokud chceš vložit několik hodnot současně, můžeš použít metodu `update`. Pamatuj, že metoda `update` může pracovat jenom s jednou hodnotou, takže musíš více údajů schovat do tuplu nebo listu (vyzkoušej si příklad níže).

```
1 # Přidávání pomocí metod 'add' a 'update'
2 set_a = {"zena", "ruze", "kost"}
3 set_a.add("pisen")
4 set_a.update(("Matous", "Lucie"))
5 print(set_a)
```

SPUSTIT ZNOVU

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

Pokud chceš hodnotu odstranit můžeš si vybrat mezi metodami `discard` a `pop`. `discard` ti umožňuje vybrat hodnotu, kterou chceš odstranit. Metoda `pop` ti na druhou stranu vybere sama hodnotu, kterou odstraní.

```
1 # Odebírání hodnot pomocí metody 'discard' a 'pop'
2 set_a = {"zena", "ruze", "kost"}
3 set_a.add("pisen")
4 set_a.update(("Matous", "Lucie"))
5
6 set_a.discard("Matous")
7 print(set_a) # chybí 'Matous'
8 set_a.pop() # metoda 'pop' si vybere udaj sama
9 print(set_a)
```

SPUSTIT ZNOVU

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

Kvíz

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / SETY / KVÍZ

6/6

✓ Quiz finished!

You scored 5 / 6

FROZENSET

Úvod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / FROZENSET / ÚVOD

Frozenset je téměř identický jako obyčejný **set**. Jediný rozdíl je, že jej nelze změnit, poté co jej vytvoříme. Jistě si pamatuješ, že je tomu tak i v případě **tuple** v porovnání s **list**.

Opět je hlavním důvodem existence tohoto datového typu možnost vytvořit takový objekt, který se v průběhu tvého programu nemůže změnit.

Vytvořením nového frozensetu

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / FROZENSET / VYTVOŘENÍM NOVÉHO FROZENSETU

Prázdný frozenset

Jelikož je **frozenset** nezměnitelný, nemá význam vytvořit prázdnou hodnotu:

```
>>> muj_prvni_fs = frozenset()  
>>> print(muj_prvni_fs)  
<class 'frozenset'>
```

Protože bys do takové proměnné nemohl přidat jiné hodnoty.

```
1 # Vyzkoušej si vytvořit prázdný 'frozenset'  
2 muj_prvni_fs = frozenset()  
3 print(type(muj_prvni_fs))
```

SPUSTIT ZNOVU

```
<class 'frozenset'>
```

Frozenset ze stringu

K vytvoření nové hodnoty `frozenset` můžeš použít `str`:

```
>>> muj_druhy_fs = frozenset("abc")
>>> print(muj_druhy_fs)
frozenset({'a', 'c', 'b'})
```

Aby v tom byl pořádek, interpret před výsledný datový typ připíše, že se jedná o `frozenset`.

```
1 # Vyzkoušej si vytvořit nový 'frozenset' ze 'str'
2 muj_druhy_fs = frozenset("abc")
3 print(muj_druhy_fs)
```

SPUSTIT ZNOVU

```
frozenset({'a', 'b', 'c'})
```

Frozenset ze stringu

K vytvoření nové hodnoty `frozenset` můžeš použít `str`:

```
>>> muj_druhy_fs = frozenset("abc")
>>> print(muj_druhy_fs)
frozenset({'a', 'c', 'b'})
```

Aby v tom byl pořádek, interpret před výsledný datový typ připíše, že se jedná o `frozenset`.

```
1 # Vyzkoušej si vytvořit 'frozenset' z 'list', 'tuple', 'set'
2 muj_treti_fs = frozenset(["a", "b", "c"]) # list
3 muj_ctvrty_fs = frozenset(("a", "b", "c")) # tuple
4 muj_paty_fs = frozenset({"a", "b", "c"}) # set
5 print(muj_treti_fs)
6 print(muj_ctvrty_fs)
7 print(muj_paty_fs)
```

SPUSTIT ZNOVU

```
frozenset({'a', 'b', 'c'})  
frozenset({'a', 'b', 'c'})  
frozenset({'a', 'b', 'c'})
```

Metody frozensetu

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / FROZENSET / METODY FROZENSETU

Metoda	Použití
<code>copy</code>	vytvoří kopii setu
<code>difference</code>	vytvoří kopii setu, která obsahuje rozdílné údaje ze dvou a více setů
<code>intersection</code>	vytvoří kopii setu, která obsahuje hodnoty společné pro výchozí set a zadaný set
<code>isdisjoint</code>	vrátí <code>True</code> , pokud mají dva sety průnik hodnot. Jinak vrátí <code>False</code>
<code>issubset</code>	vrátí <code>True</code> , pokud je výchozí set součástí zadaného setu. Jinak vrátí <code>False</code>
<code>issuperset</code>	vrátí <code>True</code> , pokud je zadaný set součástí výchozího setu. Jinak vrátí <code>False</code>
<code>symmetric_difference</code>	vytvoří kopii setu, která je tvořena symetrickým rozdílem hodnot dvou setů
<code>union</code>	vytvoří kopii setu obsahující hodnoty dvou různých setů

Ukázky metod

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / FROZENSET / UKÁZKY METOD

Úvod

Metody pro datový typ `frozenset` jsou velmi podobné metodám pro `set`.

Jediným rozdílem, jsou metody, které nějakým způsobem upravují obsah frozensetu. Ty použít nelze.

```
fr_set_a = frozenset({"zena", "ruze", "kost", "pisen", "Lucie", "Matous"})  
fr_set_b = frozenset({"zena", "ruze", "kost", "pisen", "Lukas"})
```

Průnik setů

Klasické množinové operace se chovají stejně:

```
1 # Průnik setu A se setem B  
2 fr_set_a = frozenset({"zena", "ruze", "kost", "pisen", "Lucie", "Matous"})  
3 fr_set_b = frozenset({"zena", "ruze", "kost", "pisen", "Lukas"})  
4 print(fr_set_a.intersection(fr_set_b))
```

SPUSTIT ZNOVU

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

Podmnožiny

```
1 # Je set A podmnožinou setu B
2 fr_set_a = frozenset({"zena", "ruze", "kost", "pisen", "Lucie", "Matous"})
3 fr_set_b = frozenset({"zena", "ruze", "kost", "pisen", "Lukas"})
4 print(fr_set_a.isdisjoint(fr_set_b))
```

SPUSTIT ZNOVU

```
False
```

Přidávání hodnot do setů

Jakmile budeš chtít nějak změnit obsah, dostaneš výjimku **AttributeError**, protože zkoušíš spustit takovou metodu, která pro daný datový typ není dostupná.

```
1 # Zkusíme přidat do setu A string "Petr"
2 fr_set_a = frozenset({"zena", "ruze", "kost", "pisen", "Lucie", "Matous"})
3 fr_set_a.add("Petr")
4 print(fr_set_a)
```

SPUSTIT ZNOVU

```
Traceback (most recent call last):
  File <string> line 3, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
```

Kvíz

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / FROZENSET / KVÍZ

4/4

✓ Quiz finished!

You scored 4 / 4

OPAKOVÁNÍ

Úkol 15: Update

SPUSTIT ÚKOL

Procič si slovníkovou metodu update.

⌚ 15 min. ••• střední

Úkol 16: Ověření hesla

SPUSTIT ÚKOL

V této úloze budeš ověřovat, jestli uživatel zadá heslo patřící k jeho účtu.

⌚ 15 min. ••• těžší

Úkol 17: Metody slovníku I.

SPUSTIT ÚKOL

Procičuj metody na slovnících.

⌚ 20 min. ••• těžší

Úkol 18: Metody slovníku II.

SPUSTIT ÚKOL

Procičuj metody na slovnících.

⌚ 20 min. •••

Úkol 19: Kombinace setů

SPUSTIT ÚKOL

Nyní si pojďme procvičit, co jsme se naučili o setech.

⌚ 10 min. ••• střední

ÚLOHA

Filmový slovník

ONLINE PYTHON AKADEMIE / SLOVNÍKY A MNOŽINY / ÚLOHA / FILMOVÝ SLOVNÍK

Úkol 20: Filmový slovník

[SPUSTIT ÚKOL](#)

Pojďme si zopakovat slovníky a sety na praktické úloze [Filmový slovník](#).

① 40 min. ••• těžší

[DALŠÍ LEKCE](#)

[» Terminál](#)

