Množica

Osnove programiranja

Nejc Ilc

Lastnosti množice

Python temu pravi `set`. Podpira osnovne matematične operacije nad množicami, kot so unija, razlika in presek.

- Novo množico ustvarimo tako:
  - prazna množica: `set()` ali
  - elementi v zavitih oklepajih:

    `{'a', 'b', 'c'}` ali ekvivalentno

    `set(['a', 'b', 'c'])`
  - `{}` ne pomeni prazne množice, ampak prazen *slovar*, ki ga bomo spoznali naslednjič.
- Elementi množice so edinstveni (unikatni) elementi. Torej se elementi v množici ne ponavljajo: `{1, 1, 2}` ➞ `{1, 2}`.
- Množica je spremenljiv objekt (*plastelin*), elemente lahko dodajamo in brišemo.
- Elementi množice so lahko samo nespremenljivi objekti (*kamen*). Množica ne more vsebovati seznama, slovarja ali druge množice.
- Elementi v množici niso urejeni, njihov vrstni red ni pomemben.

```
>>> {'b', 'a', 'c', 'd'}
{'c', 'd', 'a', 'b'}
```

- Elementov v množici ne označujemo z indeksi. Recimo, da imamo `a = {1, 2}`. Naslavljanje `a[0]` sproži napako. Prav tako rezanje `a[1:]`.

Ustvarjanje množice

Lahko ustvarimo prazno ali neprazno množico ali pa v množico pretvorimo zaporedje.

Kot smo že omenili, je edini način, da ustvarimo prazno množico ta:

```
>>> set()
set()
```

Množico z definiranimi elementi ustvarimo tako:

```
>>> {'sliva', 'češnja', 'jabolko'}
{'sliva', 'češnja', 'jabolko'}
```

ali z uporabo *konstruktorja* objektov:

```
>>> set(['sliva', 'češnja', 'jabolko'])
{'sliva', 'češnja', 'jabolko'}
```

Konstruktor je posebna funkcija, ki izdela nov objekt.

V prejšnjem primeru smo v množico pretvorili seznam. V splošnem lahko v množico pretvorimo poljubno drugo zaporedje (seznam, terko, niz, obseg `range`, slovar):

```
>>> set(('sliva', 'češnja', 'jabolko'))
{'sliva', 'češnja', 'jabolko'}
>>> set('jabolko')
{'o', 'j', 'a', 'b', 'k', 'l'}
>>> set(range(1,4))
{1, 2, 3}
```

Pretvarjanje zaporedja v množico ima pomembno posledico: s tem odstranimo podvojene vrednosti:

```
>>> set(('sliva', 'češnja', 'jabolko', 'sliva'))
{'sliva', 'češnja', 'jabolko'}
```

Funkcije nad množico

Stari znanci

Kot pri seznamih in terkah, lahko nad množico kličemo funkcije:

- `len()`: moč množice (število elementov)
- `sum()`: vsota elementov množice; dobimo napako, če elementi množice niso števila
- `min()` in `max()`: podobno kot `sum()`, sta tudi ti dve funkciji definirani za številčne elemente množice

```
>>> m = {1, 2, 3}
>>> len(m)
3
>>> sum(m) - min(m) + max(m)
8
```

Sprehod čez množico

Elementov množice ne moremo indeksirati

Za sprehod preko elementov množice lahko uporabimo zanko `for`:

```
sadje = {'češnja', 'sliva' , 'jabolko'}
for sadez in sadje:
    print(sadez)
```

Če želimo oštevilčiti elemente, lahko uporabimo `enumerate()`:

```
for i, sadez in enumerate(sadje):
    print(i, sadez)
```

Izpis:

```
0 sliva
1 češnja
2 jabolko
```

Operatorji nad množico

Definirani so operatorji vsebovanosti in primerjanja

Vsebovanost

Z operatorjem `in` (ali `not in`) lahko preverimo pripadnost elementa množici. Torej, ali element x pripada množici M: $x \in M$. Lahko trdimo tudi, da element množici ne pripada: $x \notin M$.

```
>>> M = {'Merkur', 'Venera', 'Zemlja', 'Mars'}
>>> x = 'Pluton'
>>> x in M
False
>>> x not in M
True
```

Primerjanje množic

- (Ne)enakost

```
>>> {1, 2} == {2, 1}
True
>>> {1, 2, 3} != {2, 1, 1}
True
```

- Relacija podmnožice: $A \subset B$

  Množica A je podmnožica množice B, če so vsi elementi množice A vključeni tudi v množico B.

```
>>> {1, 2} < {0, 1, 2, 3} # {1, 2} ⊂ {0, 1, 2, 3}
True
>>> {1, 2} < {0, 1}       # {1, 2} ⊄ {0, 1}
False
>>> {1, 2} <= {0, 1}
False
```

Metode množice

Najprej si oglejmo metode, ki dodajo ali odstranijo element.

`mnozica.add(x)`

V množico `mnozica` doda element `x`. Če v tej množici že obstaja element z vrednostjo `x`, metoda `add()` nima učinka.

```
>>> jedilnik = set()
>>> jedilnik.add('juha')
>>> jedilnik.add('glavna jed')
>>> jedilnik
{'glavna jed', 'juha'}
>>> jedilnik.add('juha')
>>> jedilnik
{'glavna jed', 'juha'}
```

`mnozica.remove(x)`

Iz množice `mnozica` izbriše element `x`. Če ga ni, javi napako.

```
>>> jedilnik.remove('juha')
>>> jedilnik
{'glavna jed'}
>>> jedilnik.remove('sladica')
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'sladica'
```

`mnozica.discard(x)`

Iz množice izbriše element `x`. Če ga ni, je tiho.

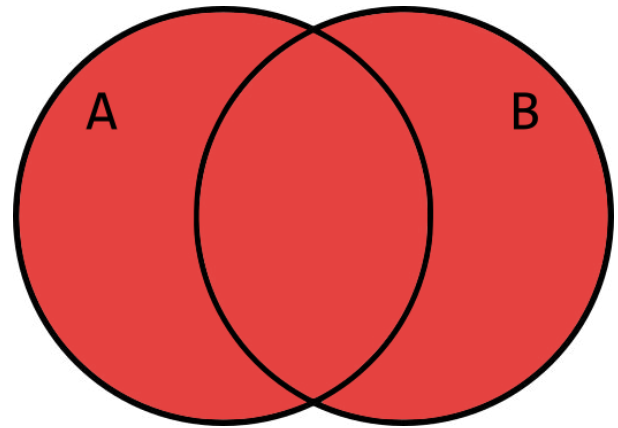
Metode množice: operacije nad množicami

Tako kot pri matematiki

Unija

Unija množic A in B je množica, ki vsebuje elemente iz obeh. Matematično to zapišemo kot $A \cup B$, v Pythonu pa:

- z uporabo operatorja `A | B` ali
- s klicem metode `A.union(B)`.



$A \cup B$

`A | B`

Izračunamo lahko tudi unijo več množic, denimo:

- `A | B | C`
- `A.union(B, C)`

Metode množice: operacije nad množicami

Tako kot pri matematiki

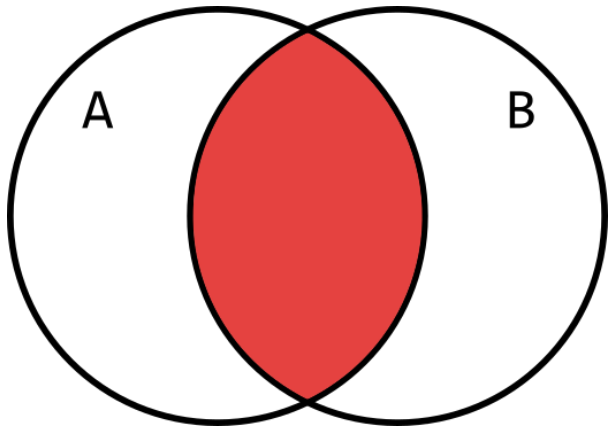
Presek

Presek množic A in B je množica elementov, ki so hkrati v A in v B. Matematično to zapišemo kot $A \cap B$, v Pythonu pa:

- z uporabo operatorja `A & B` ali
- s klicem metode `A.intersection(B)`.

Izračunamo lahko tudi presek med več množicami, recimo:

- `A & B & C`
- `A.intersection(B, C)`



$A \cap B$

`A & B`

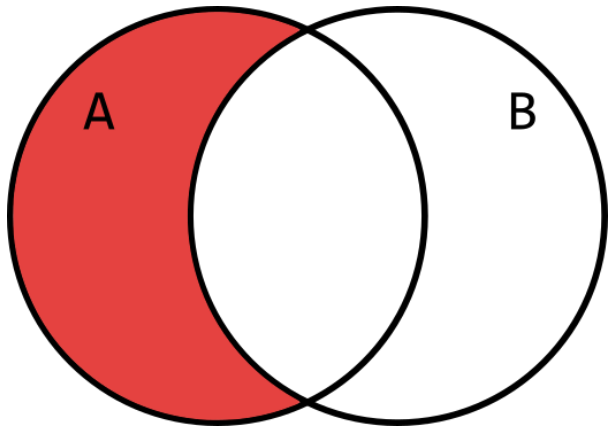
Metode množice: operacije nad množicami

Tako kot pri matematiki

Razlika

Razlika množic A in B je množica elementov, ki so v A in niso v B. Matematično to zapišemo kot $A \setminus B$, v Pythonu pa:

- z uporabo operatorja `A - B` ali
- s klicem metode `A.difference(B)`.



$A \setminus B$

`A - B`

Izračunamo lahko tudi razliko več množic, npr.:

- `A - B - C`
- `A.difference(B, C)`

Pozor: $A - B$ ni enako kot $B - A$.

Metode množice: operacije nad množicami

Tako kot pri matematiki

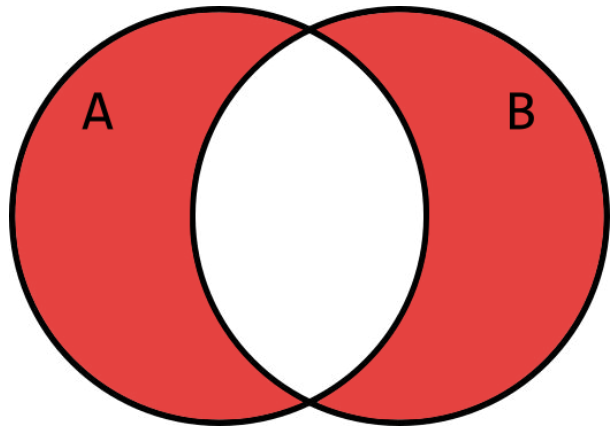
Simetrična razlika

Simetrična razlika množic A in B je množica elementov, ki so v njuni uniji, a niso v njunem preseku. Matematično to zapišemo kot $A \triangle B$, v Pythonu pa:

- z uporabo operatorja `A ^ B` ali
- s klicem metode `A.symmetric_difference(B)`.

Izračunamo lahko tudi simetrično razliko več množic, naprimer:

- `A ^ B ^ C = (A ^ B) ^ C`



$$A \triangle B = (A \cup B) \setminus (A \cap B)$$

`A ^ B`

Operacije nad množicami: primeri

"O okusih se ne razpravlja." - latinski pregovor

```
>>> Maja = {'The Weeknd', 'Shakira', 'Joker Out'}
>>> Lara = {'Shakira', 'Vlado Kreslin', 'Joker Out'}
>>> Tomi = {'Newsboys', 'The Weeknd', 'Joker Out'}

# Kakšen okus imata dekleti?
>>> Maja | Lara
{'Vlado Kreslin', 'Shakira', 'Joker Out', 'The Weeknd'}

# Vsi radi poslušajo ...
>>> Maja & Lara & Tomi
{'Joker Out'}

# Kateri izvajalec je skupen samo Tomiju in Lari?
>>> Tomi & Lara - Maja
set()

# Kateri izvajalci niso v nobenem preseku?
>>> (Tomi ^ Lara ^ Maja) - (Maja & Lara & Tomi)
{'Newsboys', 'Vlado Kreslin'}
```