

Pozrime sa, aké operátory možno aplikovať na asociatívne pole. Na asociatívne pole nemožno aplikovať metódy a operátory využívajúce vlastnosť usporiadania zoznamu, obzvlášť operácie využívajúce koncept výrezov alebo spojovania zoznamov.

Pokiaľ potrebujeme zistiť, či sa v našom poli vyskytuje konkrétny kľúč, použijeme metódu `has_key()`:

```
>>> tel
{'kotelna': 111111, 'jirka': 123456, 'stefan': 789232}
>>> tel.has_key('kotelna')
1
>>> tel.has_key('lopata')
0
```

Na prácu s asociatívnymi poľami existujú tri metódy, ktoré umožňujú konvertovať asociatívne pole alebo jeho časti na sekvenciu hodnôt:

`keys()`

Vráti zoznam všetkých kľúčov vyskytujúcich sa v asociatívnom poli.

```
>>> tel
{'kotelna': 111111, 'jirka': 123456, 'stefan': 789232}
>>>
>>> tel.keys()
['kotelna', 'jirka', 'stefan']
values()
```

Vráti zoznam všetkých hodnôt uložených v asociatívnom poli.

```
>>> tel.values()
[111111, 123456, 789232]
```

`items()`

Vráti zoznam obsahujúci tuple zložené z dvojice *kľúč* : *hodnota* pre každú z týchto dvojíc.

```
>>> tel.items()
[('kotelna', 111111), ('jirka', 123456), ('stefan', 789232)]
```

Pokiaľ použijete metódy `asoc_pole.keys()` a `asoc_pole.values()` bez toho, aby bol obsah pole `asoc_pole` medzi následným volaním metód zmenený, je zaručená zhoda v indexoch oboch metód, t. j. platí, že dvojica `key[0] : value[0]` zodpovedá prvej dvojici *kľúč* : *hodnota*, *atd.*

```
>>> pole = {1: 'a', 2: 'b', 3: 'c'}
>>>
>>> key = pole.keys()
>>> value = pole.values()
>>>
>>> pole
{3: 'c', 2: 'b', 1: 'a'}
>>> key
[3, 2, 1]
>>> value
['c', 'b', 'a']
```

Súborový dátový typ: Jython Files

Obdobne ako väčšina skriptovacích jazykov po-

núka Jython priamu podporu na prístup k súborom prostredníctvom vstavanej funkcie `open` a vstavaného dátového typu. Pre základné vstupno-výstupné operácie vstavaná podpora úplne stačí. Neskôr si ukážeme, že v prípade potreby možno plne využívať knižničnú funkciu Javy `java.io`.

Funkcia `open` má nasledujúcu syntax:

```
subor = open(nazov_suboru, mod)
```

kde *mod* môže byť jeden z reťazcov: 'r', 'w', 'a' vo význame: čítanie, zápis a pripojenie ku koncu súboru. K reťazcom označujúcim spôsob otvorenia súboru možno pripojiť ako príponu, reťazec 't' alebo 'b' vo význame textový, prípadne binárny spôsob spracovania.

```
>>> subor = open('in.txt', 'r')
>>>
>>> subor
<file in.txt, mode r at 2572393>
```

Všetky vstupno-výstupné operácie so súbormi pracujú s reťazcami. V Jythone sú všetky reťazce interne spracované v kódovaní Unicode. Otázky týkajúce sa spôsobov kódovania budú prebrané v ďalších kapitolách.

Vstavaný dátový typ, reprezentujúci otvorený súbor, obsahuje nasledujúce metódy:

`close()`

Zavrie otvorený súbor.

`flush()`

Vyprázdni vstupno-výstupný prúd.

`read([počet_bajtov])`

Prečíta celý súbor. Ak je zadaný nepovinný argument [*velkosť*], prečíta sa príslušný počet bajtov zo súboru. Prečítané dáta sa vrátia ako reťazec.

`readline([počet_bajtov])`

Prečíta zo súboru celý riadok alebo príslušný počet bajtov z riadka. Ak je dosiahnutý koniec súboru, vráti sa prázdny reťazec. Vrátený reťazec obsahuje na konci znak '\n'.

`readlines([počet_bajtov])`

Prečíta celý súbor alebo príslušný počet bajtov. Vráti zoznam, v ktorom každý prvok zodpovedá jednému prečítanému riadku.

```
>>> subor = open('in.txt', 'r')
>>> subor.readlines()
['name: g1\n', 'mem: a, b, g2\n', '\n', 'name: g2\n', 'mem: a, d\n', '\n']
```

`seek(offset, počiatok)`

Nastaví bežnú pozíciu kurzora v súbore pre nasledujúce operácie. Defaultná hodnota 0 pre argument *počiatok* bude interpretovaná, že *offset*

bude počítaný oproti *počiatku* súboru. Ak je hodnota parametra *počiatok* == 1, je *offset* počítaný relatívne oproti existujúcej pozícii. Pre *počiatok* == 2 sa *offset* počíta relatívne ku koncu súboru.

`tell()`

Vráti bežnú pozíciu kurzora v súbore.

`truncate([počet_bajtov])`

Skráti súbor na danú veľkosť alebo, pokiaľ veľkosť nie je zadaná, k bežnej pozícii kurzora v súbore.

`write(reťazec)`

Zapíše obsah reťazca *reťazec* do súboru. Návratová hodnota je `None`.

`writelines(zoznam)`

Zapíše reťazce zo zoznamu *zoznam* do súboru. Do súboru nie sú automaticky vkladané žiadne znaky '\n', aby bola funkcia komplementárna k `readlines()`.

Nasleduje malá ukážka uvedených metód:

```
>>> f = open('novy.txt', 'w')
>>> f.write('Toto\n')
>>> f.writelines(['je maly\n', 'novy\n', 'subor\n'])
>>> f.close()
>>>
>>> f = open('novy.txt', 'r')
>>> f.readlines()
['Toto\n', 'je maly\n', 'novy\n', 'subor\n']
>>> f.close()
```

Operátory

Súhrnný prehľad všetkých operátorov vyskytujúcich sa v Jythone obsahuje tabuľka 2-3. Priorita operátorov je uvedená v klesajúcom poradí.

Operátory	Opis
{...}, [...], {...}, ...	Konštruktory pre - tuple/zoznam/asociatívne pole, konverzia na reťazec
X[i], X[i:], x.y, x(...)	Indexovanie, rezy (slicing), volanie atribútu (členské premenné) objektu, operátor volanie funkcie
-x, ~y, +x	Aritmetická negácia, bitový doplnok, identita
x*y, x/y, x%y	Násobenie/opakovanie, delenie, modulo/formátovaný výstup
x + y, x - y	Sčítanie/spájanie, odčítanie
x << y, x >> y	Bitový posun
x & y	Bitový and
x ^ y	Bitový XOR
x y	Bitový or
<, <=, >, ==, !=, <>	Porovnávanie
is, is not	Test identity
in, not in	Prvok zoznamu
not x	Logická negácia
x and y	Logický súčin
x or y, lambda arglist : expr	Logický súčet, anonymné funkcie

Tab. 2-3 Operátory a ich priorita



■ ŠTEFAN HAVLÍČEK,
Sales engineer, InterSystems B. V.