

Jython: Funkcie

Pri programovaní sa často stretnete so situáciou, keď sa určité časti programu dajú znova využiť na rôznych miestach, a aby ste ich nemuseli neustále opakovať, vznikla koncepcia podprogramu. Podprogram, ktorý vracia pri svojom ukončení nejaké hodnoty, sa nazýva funkcia.

Používateľsky definované funkcie

Najjednoduchšia forma definície funkcie má nasledujúci tvar:

```
-def menoFunkcia([ arg1, arg2, ... ]):
    blok
```

Kľúčové slovo `def` je nasledované názvom funkcie a nepovinným zoznamom argumentov, uzavretým do okrúhlych zátvoriek. Zátvorky sú súčasťou definície funkcie i v prípade, že funkcia nemá žiadne argumenty. Za zátvorkami nasleduje dvojbodka. Definícia funkcie neobsahuje špecifikáciu dátového typu hodnoty, ktorú bude vracieť. Funkcie v Jythone nielenže nedefinujú dátový typ návratovej hodnoty, ale ani nešpecifikujú, či vôbec nejakú hodnotu budú vracieť. V skutočnosti každá funkcia v Jythone vracia nejakú hodnotu. Pokiaľ táto hodnota nie je explicitne určená, funkcia vracia `None`.

Ak definícia funkcie obsahuje argumenty, tie sa takisto uvádzajú bez dátových typov. Tu je ukážka veľmi jednoduchej funkcie, ktorá vlastne vôbec nič nerobí:

```
>>> def prázdnaFunkcia():
...     pass
```

Definícia funkcie je pri jej volaní spracovaná nasledujúcim spôsobom. Určíte si pamätáte, že v Jythone je všetko objektom. Teda i funkcia je objekt, ktorý je vytvorený interpretom jazyka počas vykonávania programu a odkaz na tento objekt je uložený v premennej. Táto premenná má meno zhodné s názvom funkcie. Funkciu následne zavoláme tým spôsobom, že aplikujeme *operátor volania funkcie* () na túto premennú. Áno, vidíte dobre. Okrúhle zátvorky () sú operátor, ktorý je aplikovaný na premennú, ktorá obsahuje odkaz na objekt reprezentujúci funkciu. Je to zrejme z uvedeného príkladu:

```
>>> def jednoduchaFunkcia(retazec):
...     print 'Bol zadany retazec: ', retazec
... 
```

```
>>>
>>> # vytlačim obsah premennej jednoduchaFunkcia
>>> print jednoduchaFunkcia
<function jednoduchaFunkcia at 11403048>
>>>
>>> #zavolam funkciu a zadam vstupny parameter
>>> jednoduchaFunkcia('Som vam to hovoril, ze?:)')
Byl zadany retazec: Som vam to hovoril, ze?:)
```

Akkoľvek dôsledky uvedeného tvrdenia preberieme neskôr, až sa dozvieme viac o moduloch a menných priestoroch.

Návratovú hodnotu funkcie definujeme pomocou príkazu `return`. Príkaz `return` sa môže v definícii funkcie vyskytovať hocikrát. Kedykoľvek sa počas vykonávania kódu narazí na tento príkaz, ukončí sa vykonávanie volanej funkcie a vráti sa hodnota výrazu uvedeného za týmto príkazom. Keďže dátový typ vrátenej hodnoty je určený až v čase vykonávania funkcie, je možné, že funkcia môže vracieť hodnotu, ktorej dátový typ je závislý od vstupných parametrov:

```
>>> def novaFunkcia(akyTyp):
...     if akyTyp == 's':
...         return 'Vraciam retazec'
...     elif akyTyp == 'i':
...         return 123456
...     else:
...         return None
...
>>> novaFunkcia('nic')
>>>
>>> novaFunkcia('i')
123456
>>>
>>> novaFunkcia('s')
'Vraciam retazec'
>>>
```

Dokumentačný reťazec

Je dobrým zvykom, že zdrojový kód obsahuje komentáre, ktoré dokumentujú obraty a postupy použité v programoch. Komentáre sú viditeľné, ak máte tieto zdroje k dispozícii. Bolo by iste príjemné mať k dispozícii nástroj, ktorý by bol schopný v prípade, že neviete, na aký účel je vami volaná funkcia navrhnutá, resp. aké má argumenty a aký je ich význam, zobrazí tieto informácie.

V Jythone taký nástroj existuje a volá sa *dokumentačný reťazec*. Ak obsahuje funkcia *dokumentačný reťazec*, musí byť tento reťazec prvá vec, ktorá je v tele funkcie definovaná, t. j. musí nasledovať hneď za riadkom obsahujúcim príkaz `def`. Reťazec je ohraničený z oboch strán tromi úvodzovkami, ktoré dávajú interpretu Jythonu na vedomie, že všetko, čo je medzi týmito úvodzov-

kami, je *viacriadkový komentár* a má špeciálny význam. Môže to vyzeráť takto:

```
def novaFunkcia(akyTyp):
    """Funkcia vracia hodnoty rozneho datoveho typu.

    Vstupne parametre:
    's' ... vrati retazec
    'i' ... vrati cislo
    cokolvek ineho vrati 'None'"""

    if akyTyp == 's':
        return 'Vraciam retazec'
    elif akyTyp == 'i':
        return 123456
    else:
        return None
```

Ak uložíme uvedenú definíciu funkcie do súboru, napríklad *dokumkom.py*, môžeme výhody dokumentačného reťazca využiť takto:

```
D:\Jython\Source codes>jython -i dokumkom.py
>>>
>>> print novaFunkce.__doc__
Funkcia vracia hodnoty rozneho datoveho typu.

    Vstupne parametre:
    's' ... vrati retazec
    'i' ... vrati cislo
    cokolvek ineho vrati 'None'
>>>
>>> novaFunkcia('i')
123456
>>>
```

Kedykoľvek potrebujeme zobrazíť obsah dokumentačného reťazca nejakého objektu, použijeme obrat:

```
identifikatorObjektu.__doc__
```

kde `__doc__` je atribút objektu, obsahujúci dokumentačný reťazec. Objektom, na ktorý ukazuje premenná *identifikatorObjektu*, môže byť funkcia, modul alebo aj trieda.

Obdobne ako písanie názvov premenných i písanie dokumentačných reťazcov sa riadi určitými pravidlami. Prvý riadok by mal byť pokiaľ možno krátky, obsahujúci súhrnný opis účelu objektu. Pre stručnosť by nemal obsahovať ani názov objektu, ani typ objektu. To možno dosiahnuť i inými metódami. Tento riadok by sa mal začínať veľkým písmenom a končiť bodkou.



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

+ IT KONFERENCIE