Python

Tomáš Kroupa

20. května 2014

Abstrakt

Python je, podobně jako Matlab, interpretační jazyk, proto se nehodí na věci jako super-rychlé řešení obrovských soustav lineárních rovnic. Na podobné super-rychlé výpočty se hodí spíše programy napsané v programovacích jazycích jako Fortran nebo C. Ovšem díky obrovskému množství doplňků a celkem jednoduché syntaxi a volné šiřitelnosti, je to velice vhodný jazyk pro psaní čehokoliv obslužného a pomocného.

Jedna drobnost stojí za zmínku už v abstraktu. Python je napsaný tak, že pole atd. indexuje od 0 a pokud chcete poslední prvek v poli stačí zadat index -1 (předposlední -2 atd). Pro řadu matematických problémů je toto skoro až překvapivě šikovná věc!

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Obsah

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Důležité informace 1: Poznámka o Pythonu na začátek.

Jakmile nainstalujete Python, lze využívat příkazový řádek (python command line), což je obdoba command window v Matlabu. Pokud něco zkoušíte a nějak to nefunguje, ořežte problém na „dřeň“ a vyzkoušejte si to v příkazovém řádku. V 90% případů to pomůže[a]!

[a]Zejména autorovi tohoto textu a to tím, že ho nebudete navštěvovat 5×denně kvůli řešitelným drobnostem;-)!

1 Instalace, šikovná rozšíření a rozběhnutí s Abaqusem

Zde si řekneme několik obecných informací o instalaci Pythonu a několik obecných poznámek o používání tohoto programovacího jazyka se softwary jako Abaqus a MSC.Marc. Vše ukážeme zejména na příkladu s verzí Abaqus 6.11.

S verzí Abaqus 6.11 dobře spolupracuje Python 2.7. Stáhnout instalační balíky lze z `http://python.org/`. Dobře spolupracuje znamená, že to funguje alespoň na autorově PC a že je na něm možné používat v rámci skriptů pro Abaqus i další rozšíření Pythonu jako například

- Numpy - Maticové a vektorové výpočty (`http://numpy.scipy.org/`).
- Scipy - Další vědecké výpočty (`http://www.scipy.org/`).
- Mathplotlib - Vykreslování grafů v Matlab stylu (`http://matplotlib.sourceforge.net/`). Bohužel tohle rozšíření asi nebude fungovat přímo ve skriptu, který je určený pro Abaqus. Některé knihovny se „nemají rády“ a nechtějí spolupracovat.

Postup v důležitých informacích 2 ukazuje, jak zprovoznit rozšíření Pythonu pro skriptování v Abaqusu. Podobně lze postupovat i pro MSC.Marc.

Důležité informace 2: Instalace pythonu a jeho rozšíření a rozběhnutí všeho v rámci Abaqusu pro Windows 7.

1. Nejprve nainstalujte Python, ideálně do defaultního adresáře.

   `c:\Python27\`

2. Poté nainstalujte rozšíření

   Numpy a SciPy `http://www.lfd.uci.edu/~gohlke/pythonlibs/`

   Mathplotlib `http://matplotlib.sourceforge.net/`)

   Další vychytávky instalujte dle libosti.

3. Dále vše z adresáře rozšíření, pravděpodobně:

   `c:\Python27\Lib\site-packages\`

   nakopírujte do adresáře rozšíření Abaqusího Pythonu, ve verzi 6.11-1 je to

   `c:\SIMULIA\Abaqus\6.11-1\Python\Lib\site-packages\`

   nebo ve verzi 6.12-3

   `c:\SIMULIA\Abaqus\6.12-3\tools\SMApy\Lib\site-packages\`

   (NumPy by měla verze tohoto pythonu pro Abaqus obsahovat defaultne - zdroj L.Bek :-)) Adresář site-packages nemusí být v instalačním adresáři Abaqusu, proto je třeba ho během kopírování vytvořit.

4. Hotovo, jak je vidět, pokud se ví jak na to, není to žádná věda :-).

2 Struktura scriptu

V této kapitole bude ukázáno, jak vypadají skripty v Pythonu. Opět se speciálním zřetelem na Abaqus 6.11.

Důležité informace 3: Nutné vědět o Pythonu.

V Pythonu se nepoužívá žádný „end“. Struktura programu/skriptu se vytváří odsazením textu. Proto používejte šikovný editor (například PSpad).

2.1 Obecná struktura

Obecně lze strukturu programu v Pythonu rozdělit na dvě části

1. Import balíků
2. Samotný program

Ovšem nemusí to být přímo takto, importovat balíky lze i na jiných místech. Lze importovat i části vlastních jiných skriptů. Dále lze napsat program tak, že začíná jednou funkcí __main__ na konci a z ní volat vše ostatní, nebo tak, že běží od začátku do konce jako skript. Je vidět, že v Pythonu se příliš nepředepisují pravidla!

2.2 Jednoduše jako skript

Ukažme malou ukázku (zdrojová data 1) toho, jak může vypadat „Abaqusí“ skript v Pythonu. Lze ho napsat jako jednoduchý skript bez funkcí a složité struktury a funguje krásně. Nejprve se naimportují jednotlivé moduly (části rozšíření). To se provede příkazy ve stylu „from *něco* import *něco*“, nebo „import *něco*“. V následující ukázce jsou importovány výhradně moduly z Abaqusích balíků (package). Za znakem „#“ lze zapsat poznámky a komentáře. Můžete si všimnout databázové struktury, ve které je uložený celý model v Abaqusu, k tomuto se vrátíme v kapitole 3.

2.3 Jednoduše s funkcemi

Jednoduchý skript je fajn, ale pokud chce člověk vytvořit trochu složitější model a v něm například na více místech používat jednu část skriptu, hned se hodí si tuto část napsat jako funkci. Přidejme opět malou ukázku (zdrojová data 2). Zejména si všimněte, že volání funkce `vytiskni_cas()` je až za místem kde je funkce definovaná. Funkce se označuje klíčovým slovem `def`, pak následuje název funkce a v závorce mohou být vstupní parametry a na závěr je nutná dvojtečka. Následující příkazy ve funkci jsou odsazeny a na závšr funkce lze pomocí příkazu `return prvni_vystupni_velicina,druha_vystupni_velicina` definovat výstupní veličiny/parametry/datové struktury.

2.4 Komplikovaně a vychytrale jako program

Spustit program `vytiskni_cas.py` ukázaný ve zdrojových datech 3, který je uložený v adresáři, ze kterého ho chceme spustit, lze spustit například příkazem uvedeným ve zdrojových datech 4.

Zdrojová data 1: Ukázka skriptu v Pythonu. Nejprve import balíků a pak vytvoření jedné desky.

```
#Import baliku
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

#Samotny skript, vytvoreni skorepiny - jen CAD model jedne desky
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=20.0)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(-5.
    0, -1.0),point2=(5.0, 1.0))
mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR, name='Part
    -1', type=DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseShell(sketch=mdb.models['
    Model-1'].sketches['__profile__'])
```

Zdrojová data 2: Skript na vytisknutí datumu a času.

```
from datetime import datetime

def vytiskni_cas():

    print datetime.now()

vytiskni_cas()
```

Zdrojová data 3: Ukázka programové struktury v Pythonu. Nejprve import balíků, pak funkce a na závěr začátek programu. Celý program jen vytiskne datum a čas na dvou místech skriptu.

```
from datetime import datetime

def vytiskni_cas():

    print datetime.now()

if __name__ == '__main__':

    print 'Ted vytiskneme cas poprve!'
    vytiskni_cas()

    print 'A na jinem miste programu budeme chtit vytisknout cas
        znova, tak pouzijeme stejnou funkci.'
    vytiskni_cas()
```

Zdrojová data 4: Spuštění pythonu z příkazové řádky windows. Toto můžete také použít jako zvýrazňovač v PSPadu, abyste mohli spouštět skripty pomocí „čistého“ Pythonu rovnou z PSPadu.

```
"C:\Python27\python.exe" "vytiskni_cas.py"
```

Další informace ohledně skrtiptování v Pythonu lze získat zde `http://www.kme.zcu.cz/kmet/tutorials/01_25_marc_python.php` a v [1].

3 Abaqusí rozšíření a databáze ...

... aneb tohle musíte vědět a naučit se používat.

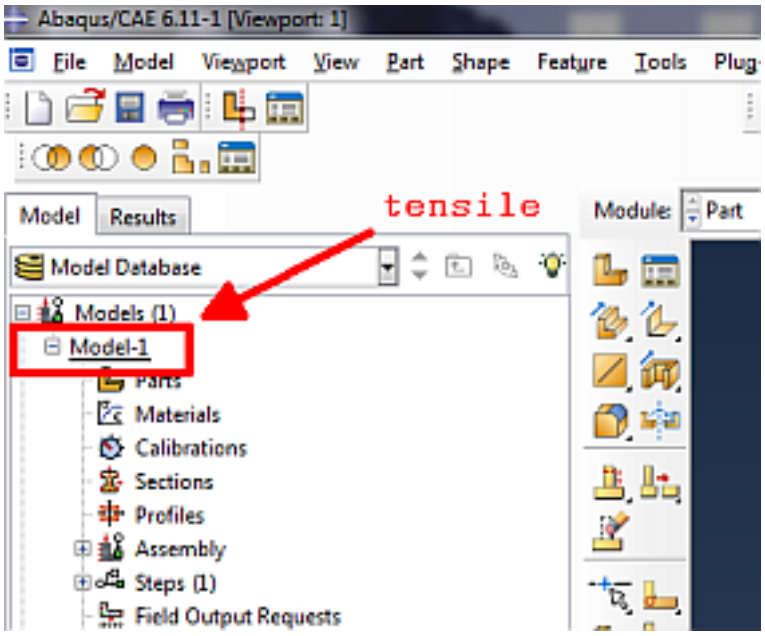
V Abaqusu je celý model databáze. Když začnete vytvářet model, doporučuji pracovat následujícícm stylem

1. Naťukám část modelu myší v Abaqus/CAE.
2. Kouknu se do souboru **název_modelu.rpy**, jaké příkazy jsem vlastně naťukal.
3. Zkopíruji právě vytištěné příkazy do svého skriptu.
4. Upravím skript podle svého.

Jak se pracuje s databází je poměrně intuitivní, komplikovanější je zjistit, častěji spíše vytušit, co se přesně ukrývá v jednotlivých datových strukturách, aby člověk nemusel dlouze prohledávat manuál (*Abaqus scripting manual*). Ve zdrojových datech 5 uveďme jednoduchou ukázku, a to příkaz který změní jméno modelu. Kde lze tento příkaz provést myší je ukázáno na obrázku 1.

Zdrojová data 5: Změna jména modelu pomocí příkazy v Pythonu.

```
#Vytvoreni databaze
mdb= Mdb()
#Zmena jmena modelu
mdb.models.changeKey(fromName='Model-1', toName='tensile')
```



Obrázek 1: Kde změní příkaz jméno modelu.

Další informace, která Vám usnadní život je, zadávání například matic vytvořených pomocí NumPy do Abaqusu pomocí Pythonu. Toto je to občas svízelné, zejména v případě zadávání velké tabulky. Protože vývojáři chtějí zachovat jako nutné moduly k používaní v rámci Abaqusu jen ty svoje a to tak, aby jich bylo co nejméně, všechno číselné se ukládá do tzv. tuple. Tuple je obecný datový typ, ve kterém může být uloženo cokoliv, asi jako struktura v Matlabu. Základní rozdíl je v tom, že **tuple nelze indexovat**. Čili v běžné praxi je dobré pracovat s vektory a maticemi reprezentopvanými polemi v Numpy a před zadáním do Abaqusu z těchto polí vytvořit tuple a to pak uložit do databáze abaqusího modelu.

Tabulka 1: Zadání obecné datové struktury (Python vs. Matlab).

Matlab (struktura)	Python (tuple)
data={'Dromedar' [10 20]}	data=('Dromedar' [10,20])

Ukázku jak vložit čísla do datového typu tuple, které obsahuje dva „sloupce“ můžete vidět ve zdrojovych datech 6. Pro snažší zorientování zde uveďme klíčové řádky. Klíčová je čárka v části ((syield,eqp),), která je zde podtržená. Bez ní vkládání v cyklu nebude fungovat, protože tato čárka říká, že se jedná o tuple! Symbol „plus“ v tomto případě není sčítání, ale provede přidání prvku do tuple.

Další zmínku zaslouží datový typ float (tedy real). Vždy, když chcete aby proměnná byla číslo s plovoucí desetinnou čárkou, vždy ve zdrojovém kódu této proměnné přiřazujte číslo ve tvaru 1.0 nebo 1e0, nikdy ne jako 1. Desetinná tečka nebo symbol e jsou ukazatele, že se jedná o datový typ float.

Zdrojová data 6: Vytvoření tabulky pro křivku plasticity a zadání do modelu. V datové struktuře d jsou jednotlivé parametry křivky zpevnění. Například počáteční mez kluzu je d.SY0.

```
no_eqp  = 1000
eqp_max = 10.0
for i in range(0,no_eqp+1):

    eqp   = (float(i)/float(no_eqp))*eqp_max

    AA1     = (d.FZT0*eqp)/(d.FES0+d.FES1*( 1+(tanh(d.FM*(eqp-d.
        FEP0))) ));
    AA1pow  = pow(AA1,d.FN)
    AA1pow2 = pow((1.0+AA1pow),(1.0/d.FN))
    syield  = d.SY0 + (d.FZT0*eqp)/AA1pow2;

    if i==0:

        table_plasticity  = ((syield,eqp),)

    else:

        table_plasticity  = table_plasticity +((syield,eqp),)

mdb.models[d.name].materials['material_no_umat'].Plastic(
    table=table_plasticity )
```

Vše ostatní musí přejít do krve během práce a seznamování se s Pythonem.

Snad jen dodejme, že v datovém poli Numpy[1] se vše čísluje od nuly a ne od jedničky jako v Matlabu (pokud si nenainstalujete nějaký modul, který umí indexovat od jedničky alà Matlab). Dále je šikovné, že záporné indexy indexují od zadu v poli (viz. Zdrojová data 7).

Zdrojová data 7: Indexování Numpy polí.

```
from numpy import *

#Toto je zadani Numpy array, Matlabovsky ekvivalent je a=[1 2 3 4]
a = array([1, 2, 3, 4])

#Toto vytiskne jednicku
print a[0]

#Toto vytiskne ctyrku
print a[-1]
```

Dále se jistě bude hodit vědět, jak tisknout textové soubory. Ukázku tisku textových souborů lze opět najít výše. Čtení textových souborů funguje podobným způsobem a detaily lze snadno vyčíst na stránkách dokumentace Pythonu [2].

Zdrojová data 8: Tisk textových souborů. Pro čtení se použije parametr „r“. Pro další detaily ohledně čtení a tisknutí souborů odkažme na [2]

```
#Otevreni souboru k zapisu
f = open(d.name+'_data.abqp', 'w')

#Zapsani dvou radek se znakem \n, ktery ukoncuje radku
f.write('*no_of_strips\n')
f.write(str(len(d.theta))+'\n')
```

Reference

[1] Gautam, P.: Python Scripts for Abaqus, Learn by Example, `http://www.abaquspython.com/`

[2] `http://www.python.org/`

[1] Numpy array

Tato prezentace je spolufinancována Evropským sociálním fondem a státním rozpočtem České republiky v rámci projektu č. **CZ.1.07/2.2.00/28.0206 „Inovace výuky podpořená praxí“.**

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.