

Use of Python in the command file



code_aster, salome_meca course material

GNU FDL licence (<http://www.gnu.org/copyleft/fdl.html>)

Outline

- ▶ Python basis
- ▶ Simple use of Python in the `code_aster` command file
- ▶ Advanced use of Python

Python basis

▶ Some words about Python

- Simple to use
- Interpreted programming language, portable
- Object Oriented
- Free and Open Source
- Widely used for scientific computing (many modules available)

Python basis

► Structure of a Python program

- No type declaration. Variables directly affected by the sign =

```
b = 12.2
a = 4. * b + 2.
```

- Code blocks indicated by indentation uniform (important)

```
i = 0
while i <10:
    print i
    i = i + 1
print('finished')
```

- Case sensitive

Mesh is different from **MESH**

- Comments preceded by #

- Reserved keywords

```
and, assert, break, class, continue, def, del, elif, else,
except, exec, finally, for, from, global, if, import, in, is,
lambda, not, or, pass, print, raise, return, try, while
```

Python basis

► Numbers

- Integers and division

```
-1, 0, 4
```

```
>>> print(1/2) #return 0 with python2, 0.5 with python3
```

- Real, complex

```
0., 1.618, 6e23, 5.39e-44, 1+2j
```

- Operations

```
+, -, *, /, **
```

► Strings

- Examples

```
>>> text = 'hello world'
```

- Concatenation

```
>>> text = 'he' + 'llo'
```

- Conversions number/text and text/number

```
>>> str(3)
```

```
>>> int('537') ; float('3.14')
```

Python basis

► Lists (`list`)

```
>>> a = [4, 7, 'text', 5.2]
>>> b = [3, -1., [9, 8], 'abc']
```

■ Concatenation

```
>>> a.extend(b); print(a)
[4, 7, 'text', 5.2, 3, -1., [9, 8], 'abc']
```

■ Adding an item

```
>>> a.append('der')
[4, 7, 'text', 5.2, 'der']
```

■ Length

```
>>> len(b)
4
```

■ Item manipulation

```
>>> print(a, a[0], a[-1], a[3:])
[8, 7, 'text', 5.2, 'der'] 4 der [5.2, 'der']
```

Python basis

► Dictionaries (`dict`)

- Association of a value and a key

```
>>> a = {'static': 1, 'dynamic': 2} ; a['other'] = 3
>>> print(a)
{'dynamic': 2, 'static': 1, 'other': 3}
>>> print(a.keys())
['dynamic', 'static', 'other']
>>> print(a.values())
[2, 1, 3]
```

Python basis

▶ Control structures

■ if

```
if x == 2:
    print('A')
elif x == 'hello' or y <= 4:
    q = q + 1
else:
    h = h**2
```

■ while

```
while a < b:
    a += 1
    print(a)
```

■ for

```
>>> for key, val in a.items():
...     print(key, val)
...
dynamic 2
static 1
other 3
```

- **break** to stop a loop, **continue** to jump to the next iteration

Simple use : if

- ▶ Replacing comments with conditional blocks

```
#-----  
#           CHARGEMENT  
#-----  
if fonction == 0 :  
    CHPRES=AFPE_CHAR_MECA(MODELE=MODELEK,  
                           PRES_REP=_F(GROUP_MA=('SURFSUP', 'SURFINF',),  
                                       PRES=PRES, ),  
                           );  
elif fonction == 1 :  
    CHPRES=AFPE_CHAR_MECA_F(MODELE=MODELEK,  
                             PRES_REP=_F(GROUP_MA=('SURFSUP', 'SURFINF',),  
                                           PRES=FCTPRES, ),  
                             );
```

Simple use : for

► Replacing copy/paste by a loop

```
FLUN=[None]*6
# valeur de la fonction FOE
for i, coef in enumerate([0.87, 1.6, 1.32, 1.4, 1.5, 1.6]):

    FLUN[i] = CREA_CHAMP(TYPE_CHAM = 'ELEM_FLUN_R',
                        OPERATION = 'AFFE',
                        MODELE     = MO,
                        PROL_ZERO='OUI',
                        INFO=2,
                        AFFE       = (_F(
                                MAILLE = 'M134',
                                NOM_CMP=('FLUN',),
                                VALE=(-1.2*coef,),
                                ),
                                _F(
                                GROUP_MA = 'GMA16',
                                NOM_CMP=('FLUN',),
                                VALE=(-0.8*coef,),
                                ),
                                ),
                        )
```

Created fields are named: **FLUN_0**, **FLUN_1**,... **FLUN_5**

Warning: names are limited to 8 characters.

TABLEG[i] with **i > 9** is not possible.

Simple use : definition, evaluation

► Define a formula via a Python function

```
def heaviside(x):  
    if x < 0.:  
        return 0.  
    else:  
        return 1.  
  
H = FORMULE(VALE='heaviside(X)', NOM_PARA='X',  
            heaviside=heaviside)
```

► Evaluate a function or a formula

■ Example with a function

```
f = DEFI_FONCTION( VALE=(0.,2.e11,  
                        20.,2.5e11),  
                  NOM_PARA='INST')  
  
mat = DEFI_MATERIAU(E=f(15.), ...)  
print(f(15.))          # 2.375e11
```

■ Example with a formula

```
g = FORMULE(VALE='Y**2+ X', NOM_PARA=('X', 'Y',))  
print(g(1., 2.))      # return 5  
print(H(-2), heaviside(6.)) # return 0., 1.
```

Distinction between `PAR_LOT= 'OUI' / 'NON'`

- ▶ `DEBUT (PAR_LOT= 'NON')` necessary for accessing Aster objects content
 - Allows various manipulations
 - Allows modifications of the Aster object content (but requires to know the data structures)
 - Complex command files: no quality insurance

- ▶ Various post-processing features are available with `CALC_FONCTION, CALC_TABLE`
 - Take advantage of them as much as possible
 - If possible, isolate `PAR_LOT= 'NON'` in `POURSUITE`

Simple use : access to values

▶ Access to the content of a list or a function (`PAR_LOT='NON'` is necessary)

- Example with a list

```
listr = DEFI_LIST_REEL(DEBUT=0.,  
                      INTERVALLE=_F(JUSQU_A = 10.,NOMBRE = 2,))  
lst = listr.Valeurs()  
print lst # return [0.0, 5.0, 10.0]
```

- Example with a function

```
func = DEFI_FONCTION( VALE=(0.,2.e11,  
                          20.,2.5e11),  
                    NOM_PARA='INST')  
temp = func.Absc() # temp contains [0., 20.]  
val = func.Ordo() # val contains [2.e11, 2.5e11]  
list_x, list_y = func.Valeurs()
```

▶ Access to the content of a table (`PAR_LOT='NON'` is necessary)

- Conversion in a Python table : `tab = tabres.EXTR_TABLE()`
- Sort, extraction, filters, impression method : `help(tab)`
- But many manipulations are possible with `CALC_TABLE`

Advanced use: access to complex objects

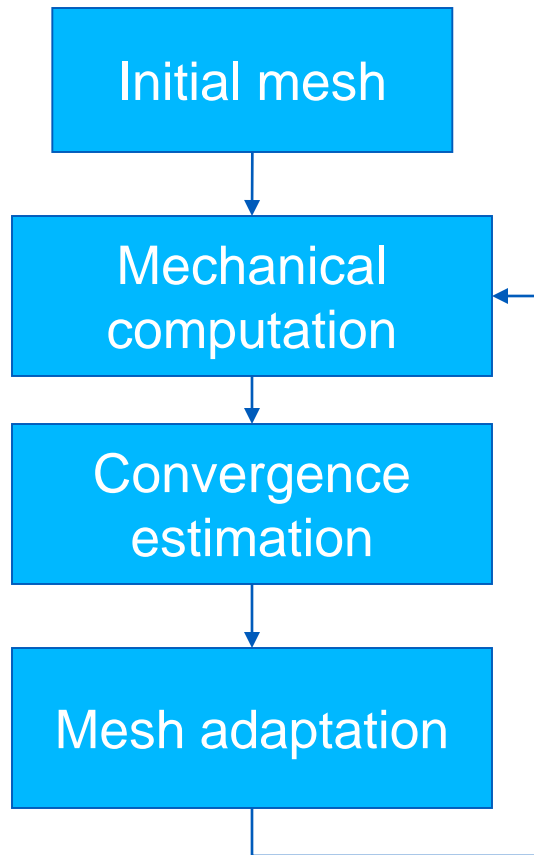
▶ Access methods to object content

- Meshes
 - List of element groups
 - Python objects representing a mesh
- Fields
 - Method `EXTR_COMP`
- Any Fortran/Jeveux object
 - `getvectjev, getcolljev` (+ methods `put`)
 - `res = aster.getvectjev("MA .COORDO .VALE")`
- Prefer use the datastructure description
 - `res = MA.sdj.COORDO.VALE.get()`

▶ Advanced use, restricted to macros development

Advanced use

► Example 1: mesh adaptation, test case zzzz121a



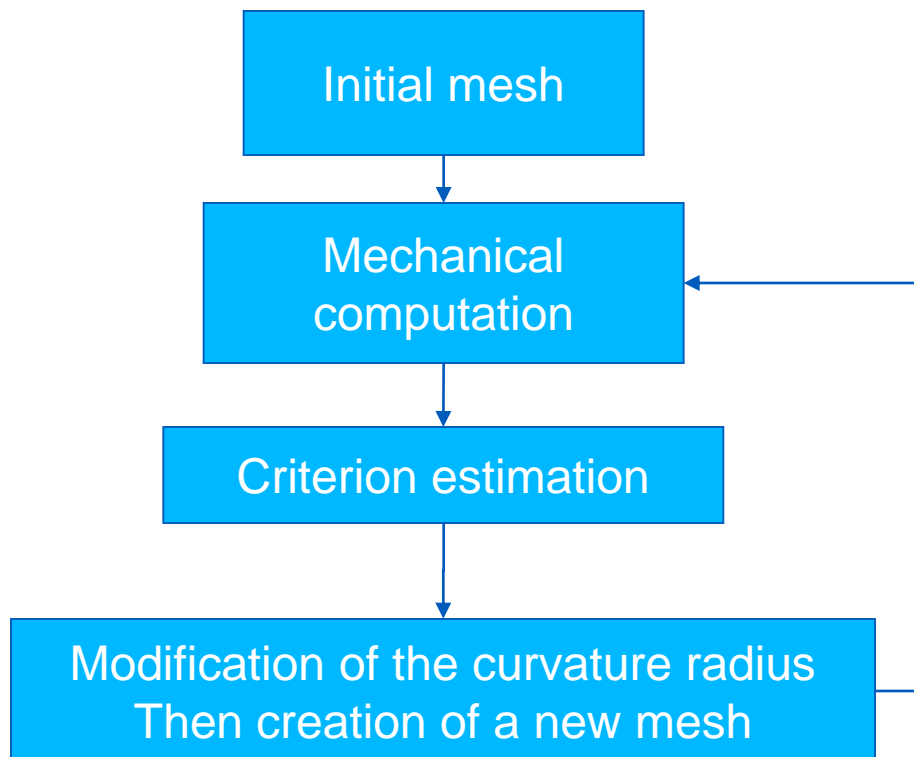
```
M[1] = LIRE_MAILLAGE()  
for k in range(1,10):  
    MODE[k] = AFFE_MODELE(...)  
    MATE[k] = AFFE_MATERIAU(...)  
    CHAR[k] = AFFE_CHAR_MECA(...)  
    RESU[k] = MECA_STATIQUE(...)  
    RELV[k] = POST_RELEVE_T(...)  
    pom = pop  
    pop = RELV[k]['DX',1]  
    if abs(pop - pom) < 1.0e-6:  
        break  
    MACR_ADAP_MAIL(...)
```

Advanced use

▶ Example 2: optimisation of the curvature radius of a curved pipe (test case demo006a)

■ Loop:

- Modification of the geometry, mesh
- Break criterion evaluate after mechanical computation
- Interactive post-processing



Advanced use

▶ Mathematical computation

- `import numpy`
- Advanced mathematical functions
- Array manipulations
- Used in `CALC_FONCTION`, `INFO_FONCTION...`
- Always available

▶ Module for statistical computation, signal processing

- Can use external modules
- Depends on the Python installation used by `code_aster`

▶ GUI modules

- Interactive plotting, visualization

Advice

Interactive computation + interactive follow-up with `PAR_LOT='NON'`, it's possible to keep open the window in python workspace

The screenshot shows the ASTK version 1.6.5 interface. The 'FICHIERS' tab is active, displaying a list of files:

Type	Serveur	Nom	UL	D	R	C
comm	Local	/comp002i.comm	1			
libr	Local	/comp002i.92	92			
mess	Local	/comp002i.mess	6			
resu	Local	/comp002i.resu	8			
comm	Local	/essai.comm	1			

The 'Arguments' field at the bottom contains the text `-interact`. The interface also features tabs for 'ETUDE', 'TESTS', 'SURCHARGE', 'AGLA', and 'REX'.

```
sh
NOMBRE TOTAL DE LIBERATIONS : 15287
0 APPELS AU MECANISME DE LIBERATION.
TAILLE MEMOIRE CUMULEE RECUPEREE : 0 Mo.
<I> FIN D'EXECUTION LE : JE-13-AOUT-2009 11:23:57
# USAGE DE LA MEMOIRE JEVEUX
# - MEMOIRE DYNAMIQUE CONSOMMEE : 0,00 Mo (MAXIMUM ATTEINT : 22,15 Mo)
# - MEMOIRE UTILISEE : 0,00 Mo (MAXIMUM ATTEINT : 17,34 Mo)
# USAGE DE LA MEMOIRE POUR LE PROCESSUS
# - VmData : 100,75 Mo - VmSize : 188,71 Mo
# FIN COMMANDE NO : 0053 USER+SYST: 0,08s (SYST: 0,04s, ELAPS: 0,07s)
<I> <INFORMATION TEMPS D'EXECUTION> (EN SECONDE)
TEMPS CPU TOTAL ..... 2,46
TEMPS CPU USER TOTAL ..... 2,21
TEMPS CPU SYSTEME TOTAL ..... 0,25
TEMPS CPU RESTANT ..... 297,54
>>> print residu_haut
125192,307692
>>> 
```

Useful documentation

▶ *code_aster* documentation

- U1.03.01: Supervisor and command language
- U1.03.02: Python methods for access to the Aster objects

▶ Examples

■ Test cases:

ssnp125: exception, non-convergence

zzzz121: loop

▶ Other references

- www.python.org

End of presentation

Is something missing or unclear in this document?
Or feeling happy to have read such a clear tutorial?

Please, we welcome any feedbacks about code_aster training materials.
Do not hesitate to share with us your comments on the code_aster forum
[dedicated thread](#).