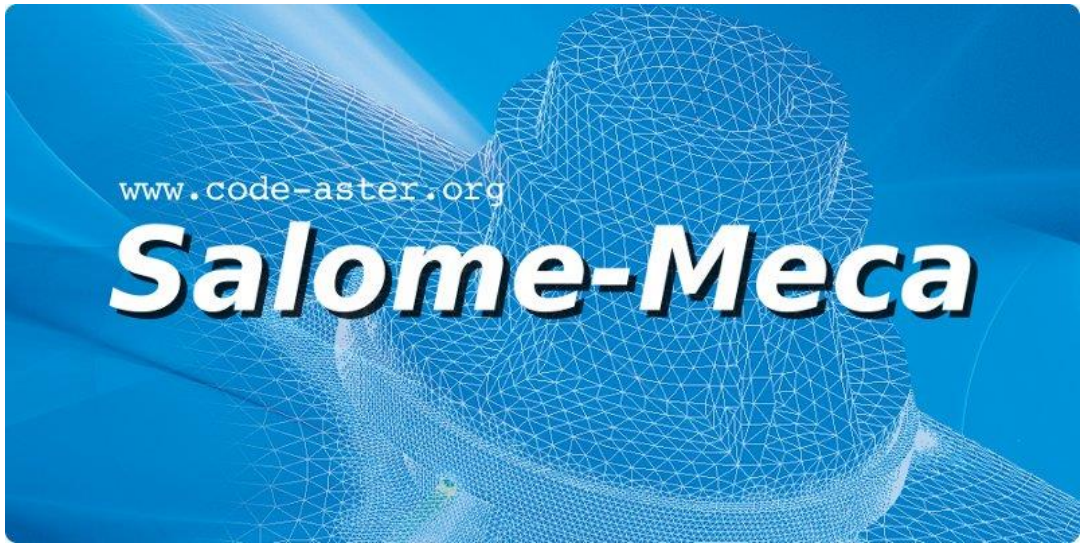
Use of Python in the command file



code_aster, Salome-Meca course material

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

```
>>> a[0] = 8 ; print(a)
[8, 7, 'text', 5.2, 'der']
```

- Dictionaries (`dict`)

  - Association of a value and a key

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

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 i, j in mydict.items():
        print(i, j)
static 1
dynamic 2
other 3
```

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

Simple use : `if`

```
# example:
# if a GIBI mesh is read, groups must be created
# if a MED mesh is read, groups are already created

# command file without python

DEBUT()

PRE_GIBI()

MA=LIRE_MAILLAGE()

# MA=LIRE_MAILLAGE(FORMAT='MED')

MA=DEFI_GROUP(reuse =MA,
              MAILLAGE=MA,
              CREA_GROUP_MA=_F(NOM= 'FACE',
              UNION=('FACE_1','FACE_2')))

MA=MODI_MAILLAGE(reuse =MA,
                 MAILLAGE=MA,
                 ORIE_PEAU_3D=_F(GROUP_MA='FACE'))

FIN()

#-----------------------------------------------------------------

DEBUT()

# PRE_GIBI()

# MA=LIRE_MAILLAGE()

MA=LIRE_MAILLAGE(FORMAT='MED')

# MA=DEFI_GROUP(reuse =MA,
#               MAILLAGE=MA,
#               CREA_GROUP_MA=_F(NOM= 'FACE',
#                                UNION=('FACE_1','FACE_2')))
#

MA=MODI_MAILLAGE(reuse =MA,
                 MAILLAGE=MA,
                 ORIE_PEAU_3D=_F(GROUP_MA='FACE'))

FIN()
```

- Replacing comments with conditional blocks

```
# example:
# if a GIBI mesh is read, groups must be created
# if a MED mesh is read, groups are already created

# command file with python

DEBUT()

type_mesh = 'med'
#type_mesh = 'gibi'

if type_mesh == 'gibi' :

   PRE_GIBI()

   MA=LIRE_MAILLAGE()

   MA=DEFI_GROUP(reuse =MA,
                 MAILLAGE=MA,
                 CREA_GROUP_MA=_F(NOM= 'FACE',
                 UNION=('FACE_1','FACE_2')))

elif type_mesh == 'med' :

   MA=LIRE_MAILLAGE(FORMAT='MED')


MA=MODI_MAILLAGE(reuse =MA,
                 MAILLAGE=MA,
                 ORIE_PEAU_3D=_F(GROUP_MA='FACE'))

FIN()
```

Simple use : `for`

- Replacing copy/paste by a loop

```
POURSUITE()

# several values of a keyword
# without python

G1=CALC_G(RESULTAT=RESU,
          COMP_ELAS=_F(RELATION='ELAS',),
          OPTION='CALC_G',
          THETA=_F(FOND_FISS=FF,
                   R_INF=2.,
                   R_SUP=4.,))

G2=CALC_G(RESULTAT=RESU,
          COMP_ELAS=_F(RELATION='ELAS',),
          OPTION='CALC_G',
          THETA=_F(FOND_FISS=FF,
                   R_INF=0.666.,
                   R_SUP=1.666.,))

G3=CALC_G(RESULTAT=RESU,
          COMP_ELAS=_F(RELATION='ELAS',),
          OPTION='CALC_G',
          THETA=_F(FOND_FISS=FF,
                   R_INF=1.,
                   R_SUP=2.,))

G4=CALC_G(RESULTAT=RESU,
          COMP_ELAS=_F(RELATION='ELAS',),
          OPTION='CALC_G',
          THETA=_F(FOND_FISS=FF,
                   R_INF=1.,
                   R_SUP=4.,))

IMPR_TABLE(TABLE=G1)
IMPR_TABLE(TABLE=G2)
IMPR_TABLE(TABLE=G3)
IMPR_TABLE(TABLE=G4)

FIN()
```

```
POURSUITE()

# several values of a keyword
# with python

RI=[2. , 0.666 , 1. , 1. ]
RS=[4. , 1.666 , 2. , 4. ]

nbc=len(RI)     # get length of the list RI
G = [None]*nbc # initialise list of concepts

for i in range(0,nbc) :

   G[i]=CALC_G(RESULTAT=RESU,
               COMP_ELAS=_F(RELATION='ELAS'),
               OPTION='CALC_G',
               THETA=_F(FOND_FISS=FF,
                        R_INF=RI[i],
                        R_SUP=RS[i]))

   IMPR_TABLE(TABLE=G[i])

FIN()
```

Created tables are: **`G_0`**, **`G_1`**, **`G_2`** and **`G_3`**

Warning: names limited to 8 characters

`TABLEG[i]` with **`i`** from 0 to 20 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')
```

- 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()
```

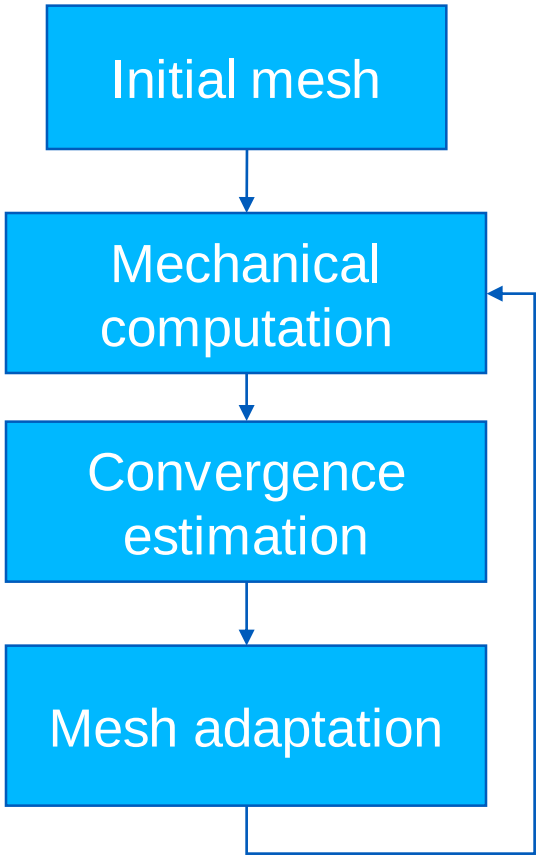
- Access to the content of a table (**PAR_LOT='NON'** is necessary)
  - Conversion in a Python table : `tab = tabres.EXTR_TABLE()`
  - Sort, extraction, filters, impression method : `help(tab)`
  - But many manipulations are possible with **CALC_TABLE**

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

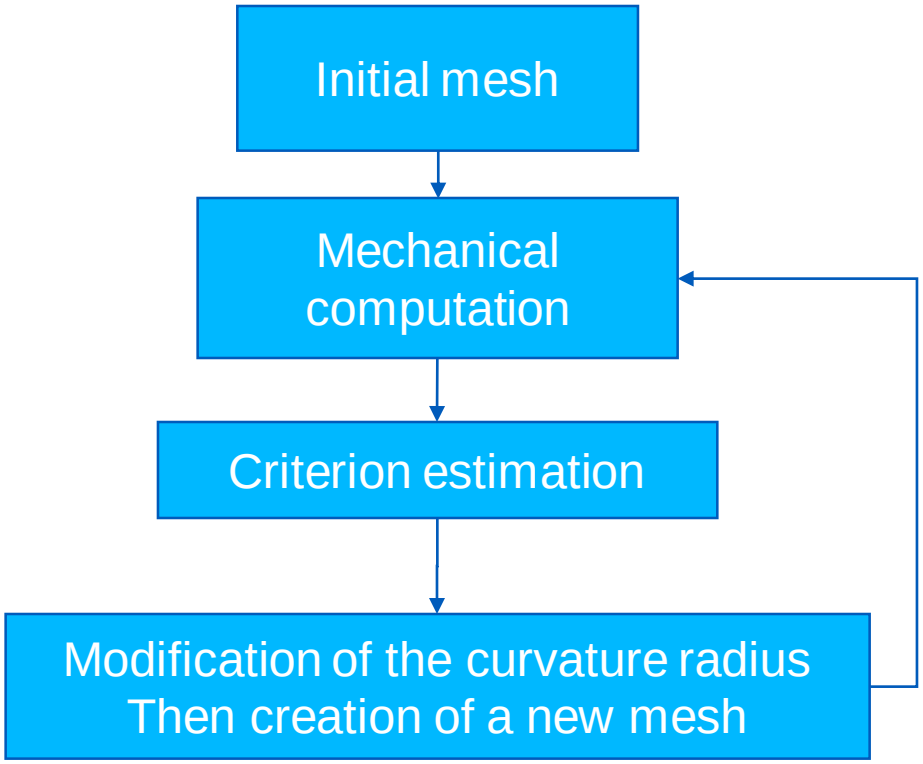
- Example 1: mesh adaptation, test case zzzz121a




```
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)



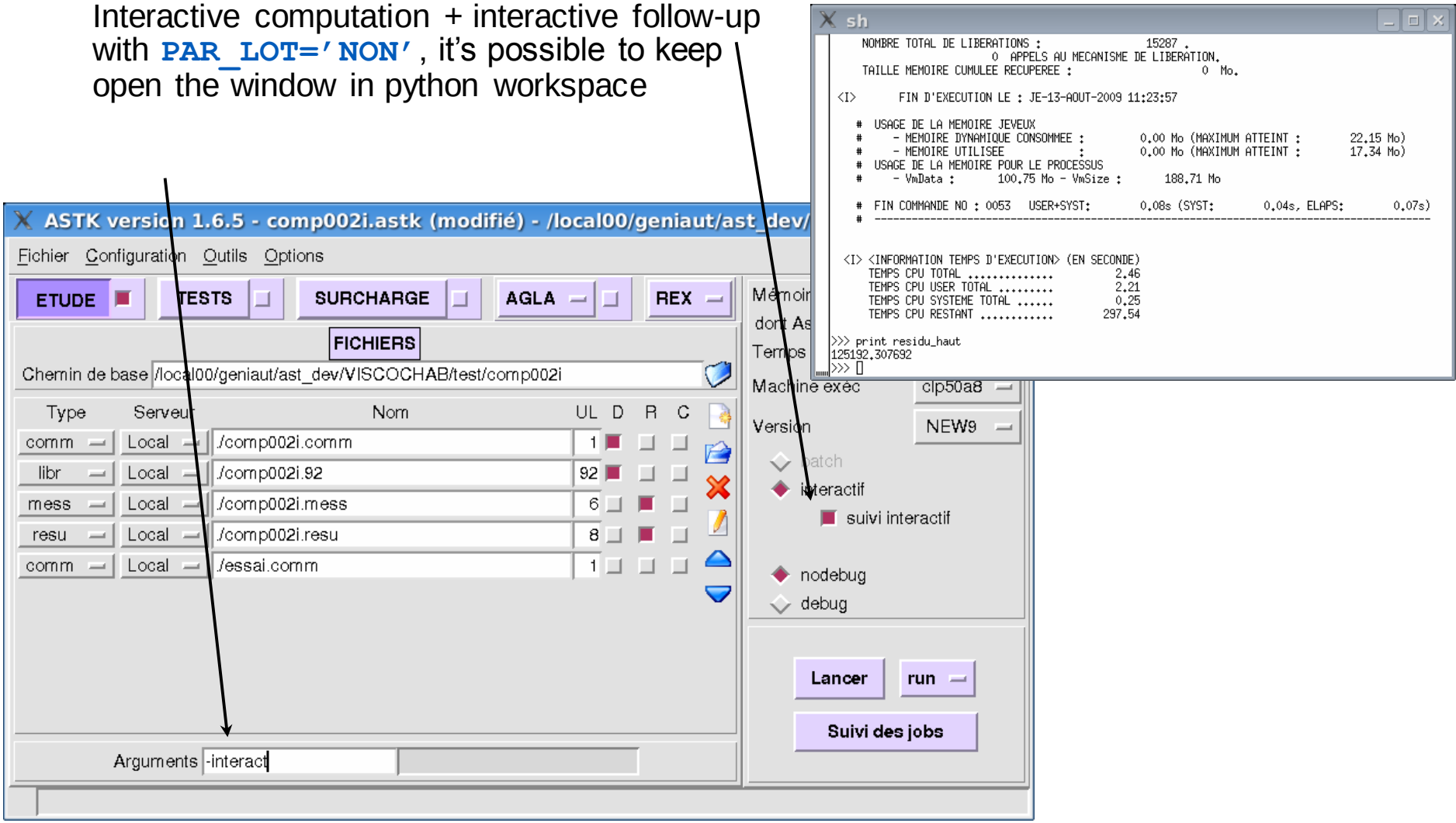
- Loop:
  - Modification of the geometry, mesh
  - Break criterion evaluate after mechanical computation
  - Interactive post-processing

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



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.