Budapest University of Technology and Economics
Department of Applied Mechanics

ABAQUS Python scripting

Barnabás PIRI

September 16, 2025

Contents

1. Prerequisites

To run PYTHON scripts in ABAQUS, you will need the following things:

- ABAQUS installation: this tutorial is written for the ABAQUS 2022 version.
- PYTHON installation: provides syntax highlighting, test the codes.
- `pyabaqus` package: provides syntax highlighting for ABAQUS commands, install via:

```
pip install pyabaqus
```

- Text editor: your preferred editor to write the program codes.

We can make sure if the PYTHON interpreter for ABAQUS is working correctly, by running this command in a CMD:

```
abaqus python --version
```

This will print out the version number. By entering

```
abaqus python
```

we can step into the environment, then entering

```
help('modules')
```

will print out the available modules (packages) in the environment. In ABAQUS 2022, PYTHON 2.7.15 comes as a bundled package.

2. General layout of a Python script in ABAQUS

2.1. Set up the parameters

The parameters and the working directory is set up as list in a different PYTHON file. The following code provides an example definition of the parameters. The parameters are defined in a list. **Important:** integer values also have to be defined as floats, so 3 has to be explicitly written as 3.0 due to ABAQUS number handling. The working directory has to exist, and free of spaces, accents and special characters. We will call this file as `parameters.py`.

```
parameter1_range = [20.0, 21.0, 22.0]
parameter2_range = [[15.0, -3.2], [18.0, 2.4]]
parameter3_range = ["point1", "point2"]

RUNDIR = "C:\\work\\work_abaqus\\"
```

2.2. Import section

The first section is dedicated for importing the required packages. When running a PYTHON script, ABAQUS will use the PYTHON installation, which comes with the software. Packages installed for the global PYTHON interpreter can't be used (and this is true for virtual environments as well). The following packages are available in the used PYTHON installation:

- numpy,
- os.

More, ABAQUS specific packages are available, which will be used for the scripts. Important to note the `from parameters import RUNDIR` line, which imports the path of the working directory, then `os.chdir(RUNDIR)` sets it. Make sure that the script is placed in the same folder as `parameters.py`. You can rename the files, just make sure that you rename them in the referenced imports sections also.

```
from abaqusConstants import *
from abaqus import *
from re import M
import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
import numpy as np
import os

# Import the parameters
from parameters import *

# Import and set the path of the working directory
from parameters import RUNDIR
os.chdir(RUNDIR)
```

2.3. Function section

This section is dedicated for the code which is responsible to build and run the FE model. It is embedded into a function, due to easier parameter handling. Important to set the job name to the desired combination of the parameters. Generate strings from the floats, use roundings,

etc. The job name cannot contain ".", so every "." is changed onto "x" symbols automatically by the second line during the name set commands.

```
def myModel(parameter1, parameter2, parameter3):
    # envelope everything in a try loop, so if a parameter set fails, and
       model can't be built, it skips and steps into the next parameter set
    try:
        # set the job name
        jobName = "myModel_" + "param1_" + str(round(parameter1,1)) + "
           _param2_" + str(round(parameter2,1)) + "_param3_" + str(
           parameter3)
        jobName = jobName.replace('.', 'x')

        # do something in this loop with the parameters
        # and use Abaqus commands to perform software actions

        # ...

        # after the model has been built, every definition was set, job was
           set up, and ready to run

        # submit the job
        mdb.jobs[jobName].submit(consistencyChecking=OFF)

        # wait for the job to complete
        mdb.jobs[jobName].waitForCompletion()

        # close the model database
        Mdb()

    # the except loop of the try function
    except:
        # close the model database
        Mdb()
```

2.4. Executive section

This section is dedicated to execute the above defined function, which runs the model. We are using a `for` loop for this purpose, which goes through the defined parameter sets.

```
for par1 in parameter1_range:
    for par2 in parameter2_range:
        for par3 in parameter3_range:
            myModel(par1, par2, par3)
```

3. Remarks

Of course, the parameters can be also set in the same file. But it's often easier to separate the script which you have to modify from those which you don't. These scripts will be running from the CMD, thus no graphical representation of the models will be available for you. You can also create scripts, which can be run from the graphical user interface, by emitting the

`parameters.py` file, and set up a single set of parameters inside the script file, which is the used to call the function of the model.

4. Running the script

The script can be run via CMD:

```
abaqus cae noGUI=<scriptname>.py
```

where `<scriptname>.py` is the actual name of your script. You have to be in the same folder, to access the file without defining the whole path of the PYTHON file. If you want to run a script file through the GUI, start ABAQUS, and choose the script via:

`File` → `Run Script...` → `Select the script`

5. Recording a set of commands from ABAQUS

The easiest way to write a script is to record the session from ABAQUS, and then do everything in the graphical user interface. This will record every step you did into a file you select.

- Open the Macro Manager: `File` → `Macro Manager...`
- Create a new macro: `Create...`
- Set the name of the macro.
- Select the directory, where the macro should be placed. Recommended: `Work`
- The macro is recording now, do the set of steps you want to include in the macro.
- Navigate to the path of the recorded path, you will find `abaqusMacro.py`.
- Open it, you will find the function with the name you set previously. The PYTHON commands will be listed in this function, which can be run to reproduce the same steps you did in the CAE environment.

Appendix

A sample ABAQUS script is accessible in the following link:
https://github.com/barnabaspiri/ABAQUS_Python_scripting.