VALIMAT

Python Interface

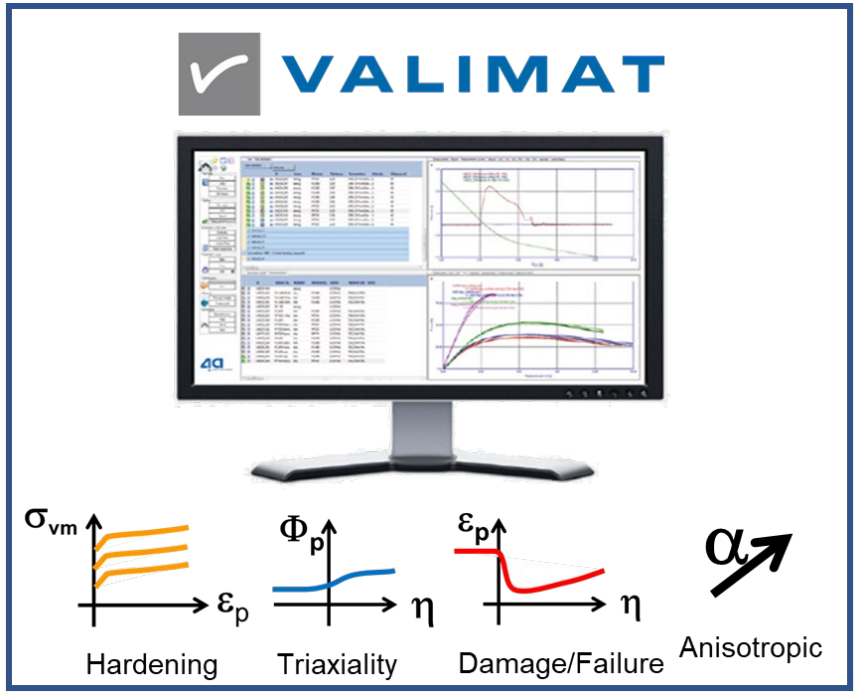
4a ENGINEERING GmbH

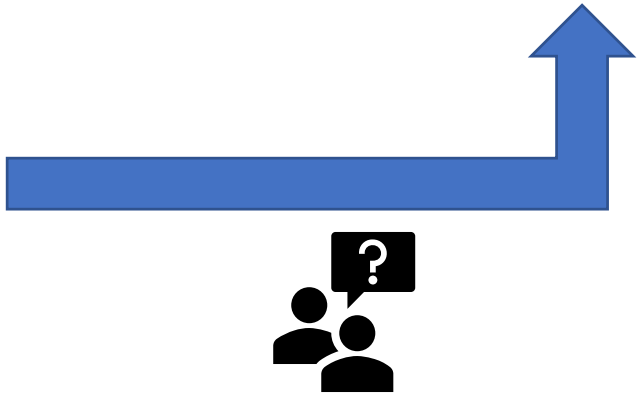
content

- What is possible with the Python interface
- Setting up python in VALIMAT®
- What is available?
- General structure
- Live demo

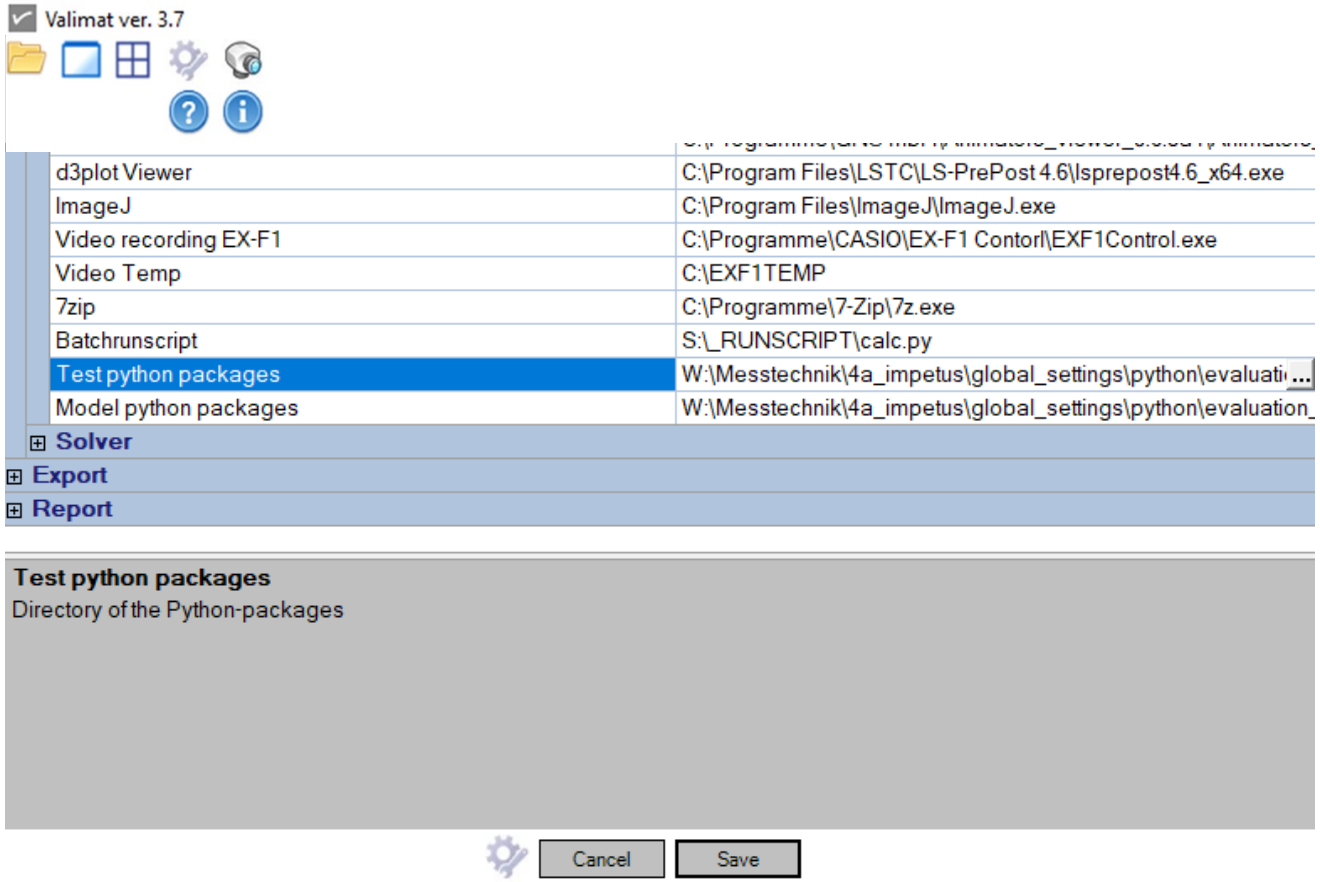
What is possible with the Python interface

- Access all values that are stored in the Valimat database (read-only)
- Read the raw measurement data for custom evaluation
- Read the evaluated data
- Read the simulation results
- Add custom packages that are not distributed with Valimat





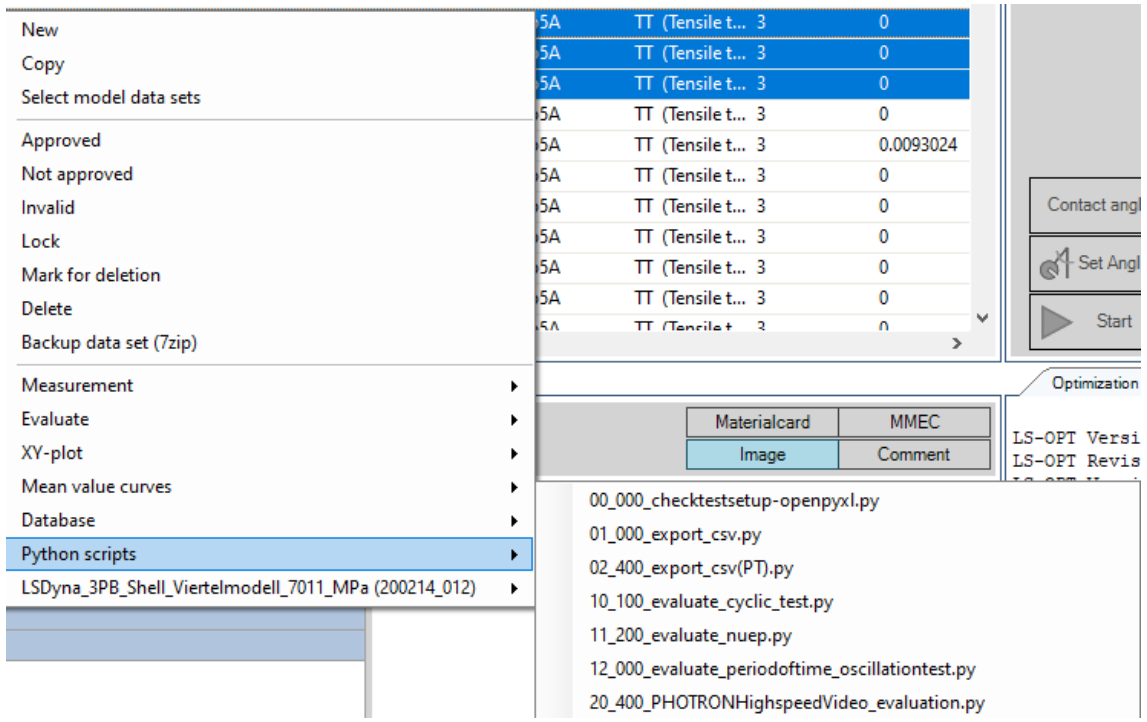
Setting up python in VALIMAT®



Überschrift

- Test python packages:
  - All *.py files in that directory will be available in the context menu of the “Test database” Tab
  - It defaults to a subdirectory of the Valimat installation
- Model python packages:
  - Same as above for the “Model database” Tab
- The Python interpreter that is delivered with Valimat and doesn’t need an install. It will be used to call the scripts

Setting up python in VALIMAT®



Überschrift

- Select the Tests or Models of interest
- Open the context menu
- The available Python files can be found at “Python scripts”

VALIMAT® database access (new)

The new VALIMAT® module allows read only access to the database

```
from VALIMAT import all

def main(base_path, table_type, table_name, work_dir, db_path, ids):
    Database.init_instance(db_path)
    test_data=Database.get_instance().getData('TESTS', [test_id])
    filter_velo=test_data[0]['T_LOADING_VELOCITY']
```

General structure

- Create a copy of the prototype file (_prototype.py_)
- Edit the function main according to your needs
- The part that should be altered is enclosed in a block of hash letters

Import → MAIN → Parameter parsing

```
import os
import platform
import inspect
import sys
import math
import subprocess
```

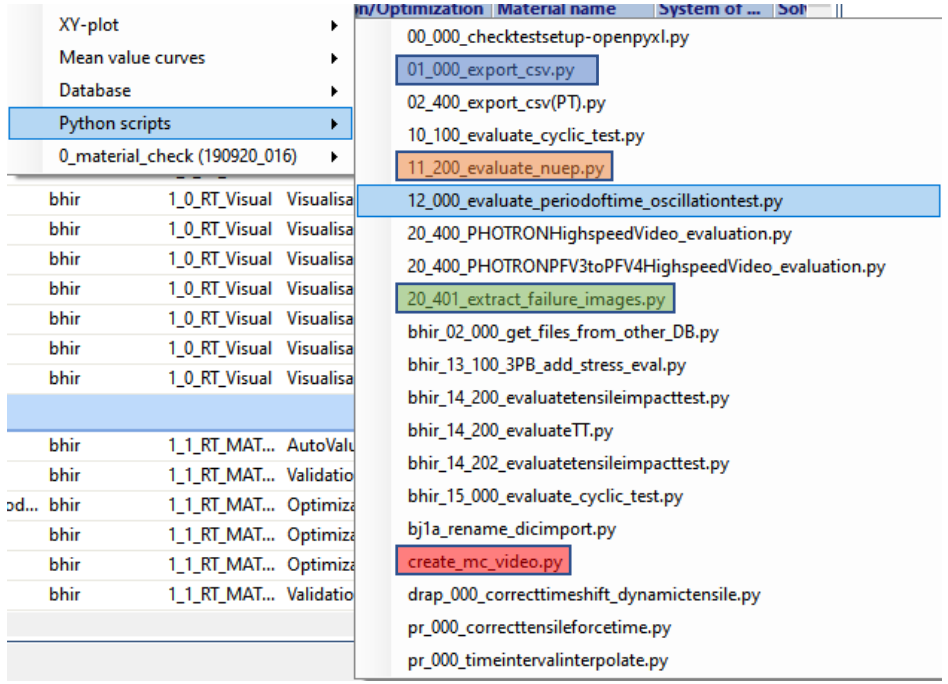
```
##############################################################################
def main(base_path, table_type, table_name, work_dir, db_path, ids):

    # Please remove this part if you do not need any Database values
    extract_call = [db_extract, db_path, base_path, table_type]
    extract_call.extend(ids)
    subprocess.call(extract_call)
    rows = readDBData(base_path, table_name, ids)

    if table_name == "ANALYSIS":
        for row in rows:
            print row.id
            print row.density
            print row.name
    elif table_name == "TESTS":
        for row in rows:
            print row.id
            print row.ambient_moisture
            print row.ambient_temp
```

What is available?

- Tests
  - export data to a csv-file
  - evaluation of plastic poisson's ratio
  - extract images at the failuretime
  - create highspeed videos with measurement data side by side



What is available?

- Models
  - extract results not available by default
  - create customized plots from simulation results

```
01_000_export_csv.py
02_000_preSimToTest.py
10_000_extract_extended_results.py
11_000_reextraxt_LSDYNA.py
12_000_extract_ForceDispl.py
13_000_extract_triax_LSDYNA.py
14_000_FailExtract.py
15_000_MMMFailExtract.py
20_200_overlay_SIM_tensiletest.py
30_000_CombineMaxFailData.py
31_000_CombineFailData.py
bhir_00_000_pathtest.py
bhir_02_000_get_files_from_other_DB.py
bhir_02_000_getDYNALicenseInfo.py
bhir_03_000_energycomparison.py
bhir_03_000_extract_cputime.py
bhir_03_000_failurerewrite.py
bhir_10_000_evaluate_detail_info.py
bhir_12_200_createAutoFit.py
bhir_15_000_extract_failelement.py
bhir_16_000_PlotLocalisation.py
bhir_21_200_TTstresscorr.py
bhir_get_auto_opt_startvalues.py
```

Things to do as software engineer

- The main function:

```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
```

- base_path: location of VALIMAT MDB
- table_type: either curvestore or model
- table_name: VALIMAT database name “4a_impetus.mdb”
- work_dir: base_path+table_type
- db_path: base_path+table_name
- ids: list of test or model ids ([‘190508_013’, ‘190508_014’, ‘190508_015’])

Live demo

- Example export_csv.py

Live demo

- Example create_mc_video.py (maybe eval_triax_on_notched_specimen.py)

Things to do as software engineer

- VALIMAT database structure
- database
- curvestore

- model
  - input files
  - material cards for optimization with lsopt
  - average curves for optimization (oavg_”casename”.”specifier”, in simulation units)
  - 4a_inpetus_sampling directory (lsopt results)
  - case_” casename” (lsopt results)
    - StageResults
    - directories containing the simulation models
      - .xy simulation curve files, in simulation units

VALIMAT database access (old)

Be sure to have the following executable in your script directory:

- extract_values_from_db.exe

- In your script do the following:

```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
    extract_call = [db_extract, db_path, base_path, table_type]
    extract_call.extend(ids)
    subprocess.call(extract_call)
    data = readDBData(base_path, table_name, ids)

    for curr_test in data:
        sssr=curr_test.stressstrainstrainrate
```

Summary

- Python can be used to extend Valimat
- Almost no limit for the creativity of the user
- 2 Examples are delivered with Valimat
  - export_csv.py
  - create_mc_video.py
- Prototype file for a rapid start into the development
- Easy to start the script out of Valimat