```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""    Ihor Mirzov, February 2020
Distributed under GNU General Public License v3.0

Converts CalculiX .frd resutls file to ASCII .vtk or XML .vtu format:
python3 ccx2paraview.py ./tests/other/Ihor_Mirzov_baffle_2D.frd vtk
python3 ccx2paraview.py ./tests/other/Ihor_Mirzov_baffle_2D.frd vtu

TODO It would be a killer feature if Paraview could
visualize gauss point results from the dat file...
https://public.kitware.com/pipermail/paraview/2013-January/027121.html

TODO Parse DAT files - there are lots of results

TODO XDMF format:
https://github.com/calculix/ccx2paraview/issues/6

"""

import os
import sys
import logging
import argparse

import FRDParser
import VTKWriter
import VTUWriter
import PVDWriter
import clean


class Converter:

    def __init__(self, file_name, fmt):
        self.file_name = file_name
        self.fmt = fmt

    def run(self):

        # Parse FRD-file
        relpath = os.path.relpath(self.file_name, start=os.path.dirname(
            __file__))
        logging.info('Parsing ' + relpath)
        p = FRDParser.Parse(self.file_name)

        # If file contains mesh data
        if p.node_block and p.elem_block:
            times = sorted(set([b.value for b in p.result_blocks]))
            l = len(times)
```

```
51              if l:
52                  logging.info('{} time increment{}'.format(l, 's'*min(1, l
                        -1)))
53
54                  """ If model has many time steps - many output files
55                  will be created. Each output file's name should contain
56                  increment number padded with zero """
57                  counter = 1
58                  times_names = {} # {increment time: file name, ...}
59                  for t in sorted(times):
60                      if l > 1:
61                          ext = '.{:0{width}}.{}'.format(counter, self.fmt,
                                width=len(str(l)))
62                          file_name = self.file_name.replace('.frd', ext)
63                      else:
64                          ext = '.{}'.format(self.fmt)
65                          file_name = self.file_name.replace('.frd', ext)
66                      times_names[t] = file_name
67                      counter += 1
68
69                  # For each time increment generate separate .vt* file
70                  # Output file name will be the same as input
71                  for t, file_name in times_names.items():
72                      relpath = os.path.relpath(file_name, start=os.path.
                            dirname(__file__))
73                      logging.info('Writing {}'.format(relpath))
74                      if self.fmt == 'vtk':
75                          VTKWriter.writeVTK(p, file_name, t)
76                      if self.fmt == 'vtu':
77                          VTUWriter.writeVTU(p, file_name, t)
78
79                  # Write ParaView Data (PVD) for series of VTU files.
80                  if l > 1 and self.fmt == 'vtu':
81                      PVDWriter.writePVD(self.file_name.replace('.frd', '.pvd
                            '), times_names)
82
83              else:
84                  logging.warning('No time increments!')
85                  file_name = self.file_name[:-3] + self.fmt
86                  if self.fmt == 'vtk':
87                      VTKWriter.writeVTK(p, file_name, None)
88                  if self.fmt == 'vtu':
89                      VTUWriter.writeVTU(p, file_name, None)
90          else:
91              logging.warning('File is empty!')
92
93
94  if __name__ == '__main__':
95
96      # Configure logging
97      logging.basicConfig(level=logging.INFO,
```

```
98                          format='%(levelname)s: %(message)s')
99
100     # Command line parameters
101     parser = argparse.ArgumentParser()
102     parser.add_argument('filename', type=str,
103                         help='FRD file name with extension')
104     parser.add_argument('format', type=str,
105                         help='output format: vtu or vtk')
106     args = parser.parse_args()
107
108     # Create converter and run it
109     if args.format in ['vtk', 'vtu']:
110         ccx2paraview = Converter(args.filename, args.format)
111         ccx2paraview.run()
112     else:
113         msg = 'ERROR! Wrong format \"{}\". '.format(args.format) \
114             + 'Choose one of: vtk, vtu.'
115         print(msg)
116
117     # Delete cached files
118     clean.cache()
```