IAEA 60 Years
Atoms for Peace and Development

About an ENDF parser with some applications

Georg Schnabel
g.schnabel [at] iaea.org

Nuclear Data Section
Division of Physical and Chemical Sciences NAPC
Department for Nuclear Sciences and Applications
IAEA, Vienna

TM on Nuclear Data Processing

1 December 2022

Acknowledgment

```
8.201440+5 2.500000+0 1.704000+2 7.200000-1 0.000000+0 0.000000+02631 2151   320
8.286320+5 2.500000+0 6.790000+2 7.200000-1 0.000000+0 0.000000+02631 2151   321
8.357980+5 2.500000+0 5.364000+2 7.200000-1 0.000000+0 0.000000+02631 2151   322
8.383400+5 2.500000+0 4.450000+1 7.200000-1 0.000000+0 0.000000+02631 2151   323
8.454520+5 2.500000+0 8.830000+2 7.200000-1 0.000000+0 0.000000+02631 2151   324
8.503770+5 2.500000+0 6.340000+1 7.200000-1 0.000000+0 0.000000+02631 2151   325
8.519200+5 2.500000+0 4.060000+1 7.200000-1 0.000000+0 0.000000+02631 2151   326
8.654000+5 2.500000+0 2.673330+2 7.200000-1 0.000000+0 0.000000+02631 2151   327
0.000000+0 0.000000+0           0           0           0          02631 2  099999
0.000000+0 0.000000+0           0           0           0          02631 0  0     0
2.605600+4 5.545440+1           0           0           0          02631 3  1     1
0.000000+0 0.000000+0           0           0           1       118862631 3  1     2
     11886           2                                              2631 3  1     3
1.000000-5 0.000000+0 8.500000+5 0.000000+0 8.500000+5 1.501370+02631 3  1     4
8.501600+5 1.621200+0 8.502720+5 1.830530+0 8.503280+5 1.991440+02631 3  1     5
8.503840+5 2.266760+0 8.504960+5 3.105180+0 8.505520+5 3.398490+02631 3  1     6
8.506080+5 3.240010+0 8.506640+5 2.706120+0 8.507200+5 2.269630+02631 3  1     7
8.508320+5 1.644860+0 8.508890+5 1.420670+0 8.509450+5 1.409280+02631 3  1     8
8.510010+5 1.224200+0 8.510570+5 1.257110+0 8.512250+5 1.190450+02631 3  1     9
8.515050+5 1.016710+0 8.516740+5 1.147350+0 8.517300+5 1.399960+02631 3  1    10
8.517860+5 1.710880+0 8.518420+5 2.202990+0 8.519540+5 3.723320+02631 3  1    11
8.520100+5 4.412230+0 8.520670+5 4.549740+0 8.521230+5 4.276150+02631 3  1    12
8.522350+5 3.192380+0 8.523470+5 2.571110+0 8.525160+5 1.867240+02631 3  1    13
8.525720+5 1.798760+0 8.526850+5 1.560780+0 8.528530+5 1.583620+02631 3  1    14
8.529100+5 1.417930+0 8.530220+5 1.282760+0 8.530780+5 1.381970+02631 3  1    15
8.531350+5 1.183290+0 8.533600+5 1.115840+0 8.536410+5 9.088520-12631 3  1    16
8.538660+5 1.015650+0 8.539230+5 8.782260-1 8.540350+5 8.654520-12631 3  1
8.540920+5 7.611950-1 8.543170+5 7.053160-1 8.543740+5 8.481200-12631 3  1
8.544860+5 8.733950-1 8.546560+5 6.495240-1 8.549940+5 6.797920-12631 3  1
8.550510+5 7.410150-1 8.551070+5 6.239980-1 8.552770+5 7.379170-12631 3  1
8.556150+5 8.505940-1 8.560110+5 1.205440+0 8.561240+5 1.366770+02631 3  1
8.561810+5 1.343680+0 8.564640+5 1.917050+0 8.568600+5 2.501140+02631 3  1
8.569170+5 2.523750+0 8.573130+5 3.187040+0 8.577670+5 3.452750+02631 3  1
8.581640+5 3.595140+0 8.586750+5 3.461960+0 8.589020+5 4.070410+02631 3  1
8.590730+5 4.755850+0 8.592430+5 4.043290+0 8.593000+5 3.663900+02631 3  1
8.593570+5 3.626010+0 8.594140+5 3.518320+0 8.598120+5 3.168320+02631 3  1
8.599260+5 3.156840+0 8.599830+5 3.040850+0 8.600400+5 3.161470+02631 3  1    27
8.600970+5 3.007280+0 8.602680+5 3.095120+0 8.603250+5 2.966430+02631 3  1    28
8.603820+5 3.057550+0 8.605530+5 2.975790+0 8.607810+5 2.898040+02631 3  1    29
8.612370+5 2.614040+0 8.613510+5 2.682170+0 8.614080+5 2.580580+02631 3  1    30
8.615800+5 2.506820+0 8.616370+5 2.566630+0 8.618650+5 2.417590+02631 3  1    31
8.620000+5 2.418840+0 8.621510+5 2.420220+0 8.622700+5 2.622580+02631 3  1    32
```

Daniel Lopez Aldama
(IAEA Consultant)

https://github.com/iaea-nds/acemaker

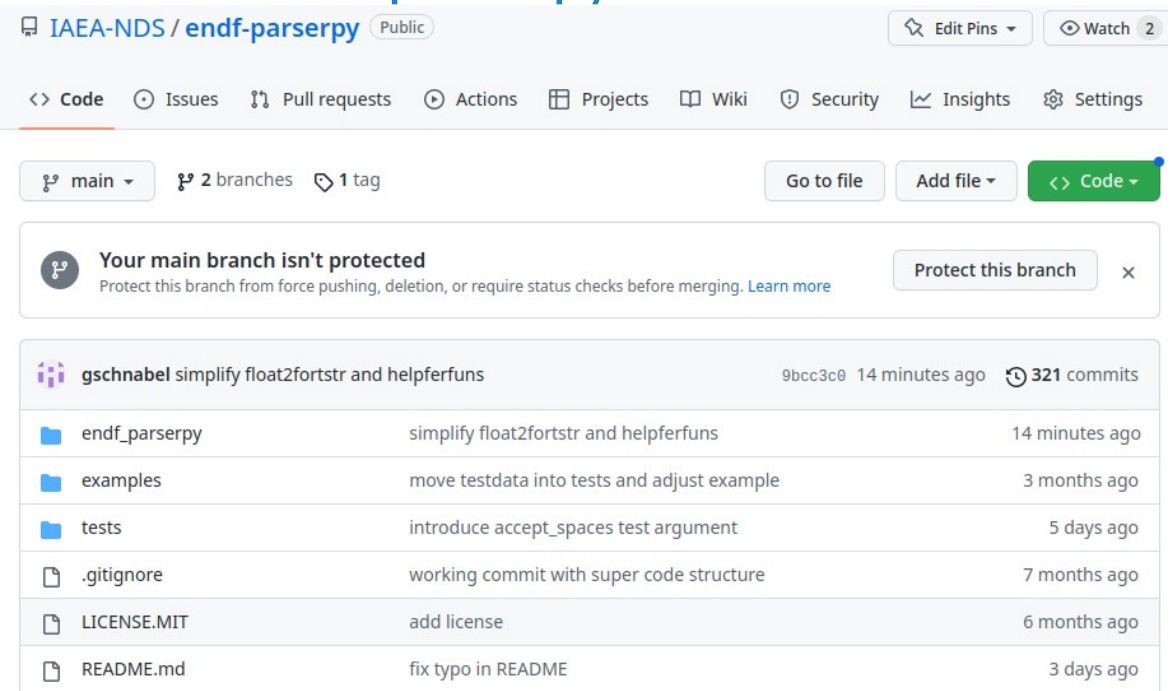
Outline

- What is endf-parserpy?
- How is it designed?
- Current status and features
- How it was already useful to me – some applications
  - Format verification of JEFF libraries
  - Extracting a Cf-252 (s.f) spectrum covariance matrix
  - FENDL provenance checking
  - Modifications of ENDF files
  - Bonus: Cu-63 checking

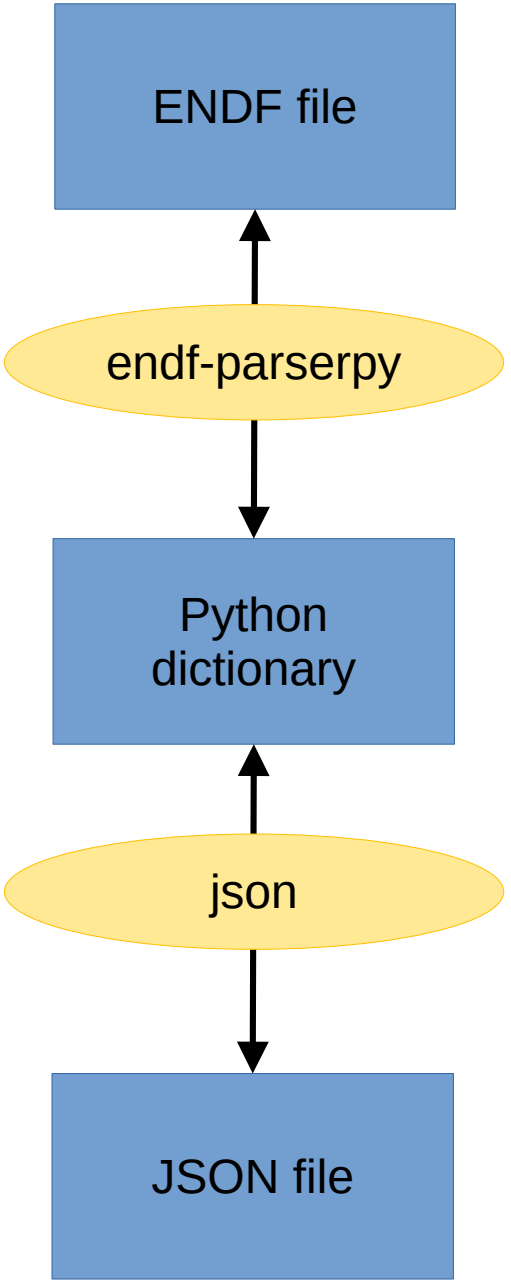
endf-parserpy

- Simplify writing, reading and validating ENDF-6 formatted files
- Translation of ENDF files to JSON files and back
- Complement to tools like ENDFtk, ENDF++, DeCE, SANDY, FUDGE, PREPRO and many more
- Open-source (MIT license) and available at

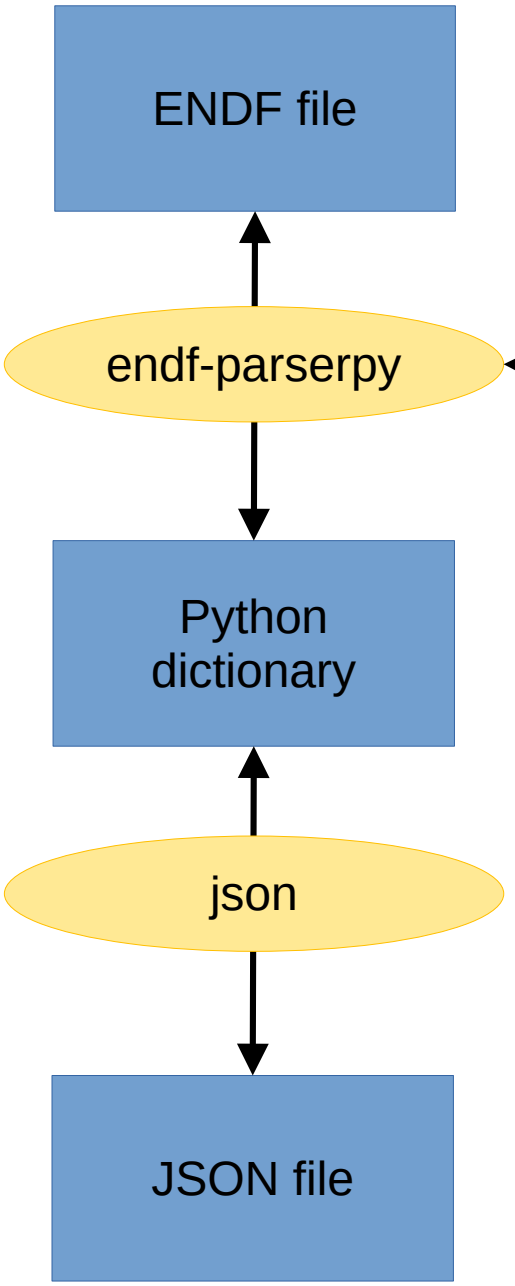
  https://github.com/iaea-nds/endf-parserpy

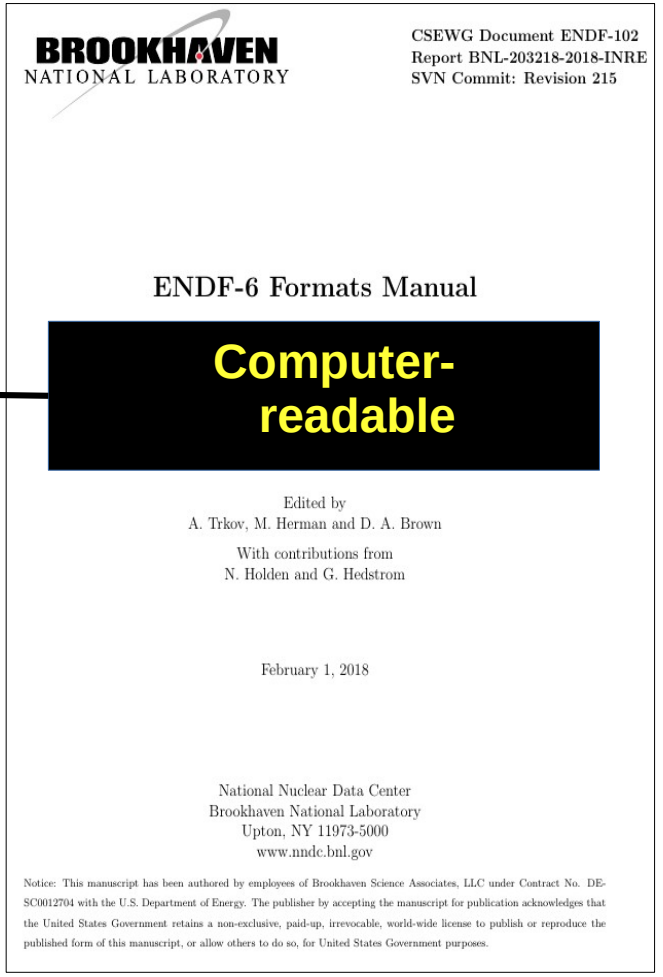


Basic design of endf-parserpy



Basic design of endf-parserpy




BROOKHAVEN NATIONAL LABORATORY

CSEWG Document ENDF-102
Report BNL-203218-2018-INRE
SVN Commit: Revision 215

ENDF-6 Formats Manual

Computer-readable

Edited by
A. Trkov, M. Herman and D. A. Brown

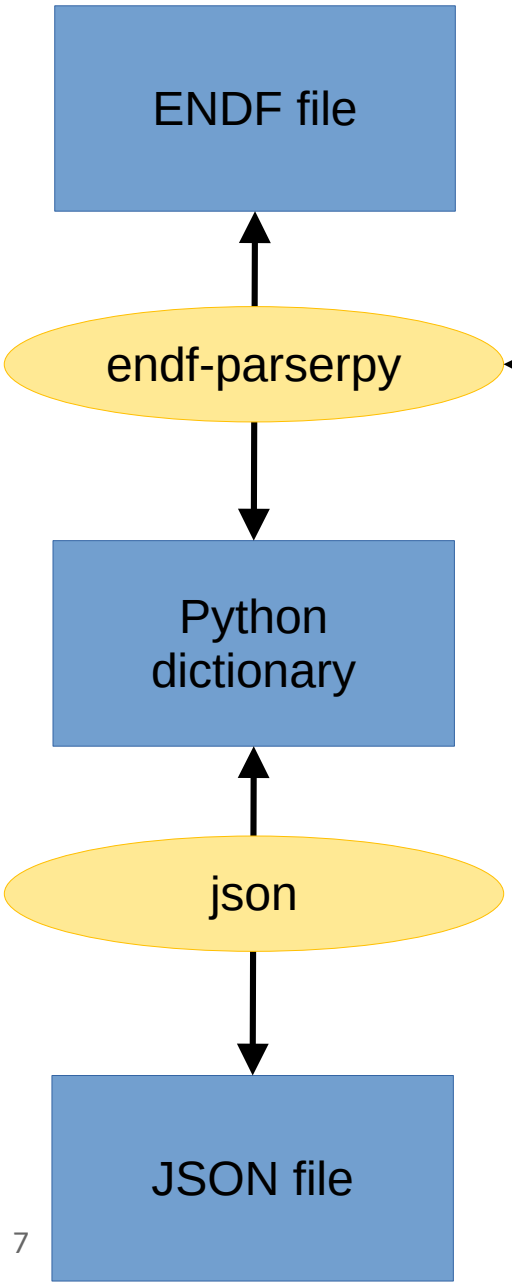
With contributions from
N. Holden and G. Hedstrom

February 1, 2018

National Nuclear Data Center
Brookhaven National Laboratory
Upton, NY 11973-5000
www.nndc.bnl.gov

Notice: This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

Basic design of endf-parserpy




```
[MAT, 4, MT/ ZA, AWR, 0, LTT, 0, 0]HEAD
if LTT==3 and LI==0 [lookahead=1]:
    [MAT, 4, MT/ 0.0, AWR?, LI, LCT, 0, NM]CONT
else:
    [MAT, 4, MT/ 0.0, AWR?, LI, LCT, 0, 0]CONT
endif

# Legendre coefficients
if LTT == 1 and LI == 0:
    [MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE/ Eint ]TAB2
    for i=1 to NE:
        [MAT, 4, MT/ T, E[i] , LT, 0, NL[i], 0/ {a[i,l]}{l=1 to NL[i]} ]LIST
    endfor

# Tabulated probability distributions
elif LTT==2 and LI==0:
    [MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE/ Eint ]TAB2 (energy_table)
    for i=1 to NE:
        [MAT, 4, MT/ T, E[i] , LT, 0, NR, NP/ mu / f]TAB1 (angtable[i])
    endfor

# Angular distributions over two energy ranges.
elif LTT==3 and LI==0:
    # lower range given by Legendre coefficients
    [MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE1/ Eint ]TAB2 (leg_int)
    for i=1 to NE1:
        [MAT, 4, MT/ T, E[i], LT, 0, NL[i], 0/
                    {al[i,j]}{j=1 to NL[i]} ]LIST
    endfor
    # higher range represented by probability distribution
    [MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE2/ Eint ]TAB2 (ang_int)
    for i=NE1 to NE1+NE2-1:
        [MAT, 4, MT/ T,  E[i] , LT, 0, NR, NP/ mu / f ]TAB1 (angtable[i])
    endfor
endif
SEND
```

Formalized ENDF-6 format specification language

Computer-readable ENDF-6 format specification: Language elements

- Record type specifications (HEAD, CONT, LIST, TAB1, TAB2, …)
- Scalars and arrays (1d and multi-dimensional)
- Two types of loops
- If (-elif-else) statements
- Sections (not MF/MT but subsection within MT sections)

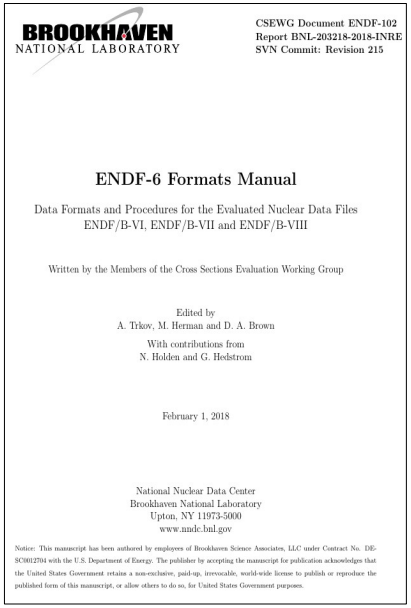
Record type specification

```
[MAT, 1,451/  TEMP,   0.0,  LDRV,      0,    NWD,    NXC]CONT
```

Exactly as done in the ENDF-6 formats manual

The fundamental nuisance

```
[MAT, 1,451/  TEMP,    0.0,  LDRV,      0,     NWD,     NXC]CONT
```



BROOKHAVEN NATIONAL LABORATORY

CSEWG Document ENDF-102
Report BNL-203218-2018-INRE
SVN Commit: Revision 215

ENDF-6 Formats Manual

Data Formats and Procedures for the Evaluated Nuclear Data Files
ENDF/B-VI, ENDF/B-VII and ENDF/B-VIII

Written by the Members of the Cross Sections Evaluation Working Group

Edited by
A. Trkov, M. Herman and D. A. Brown

With contributions from
N. Holden and G. Hedstrom

February 1, 2018

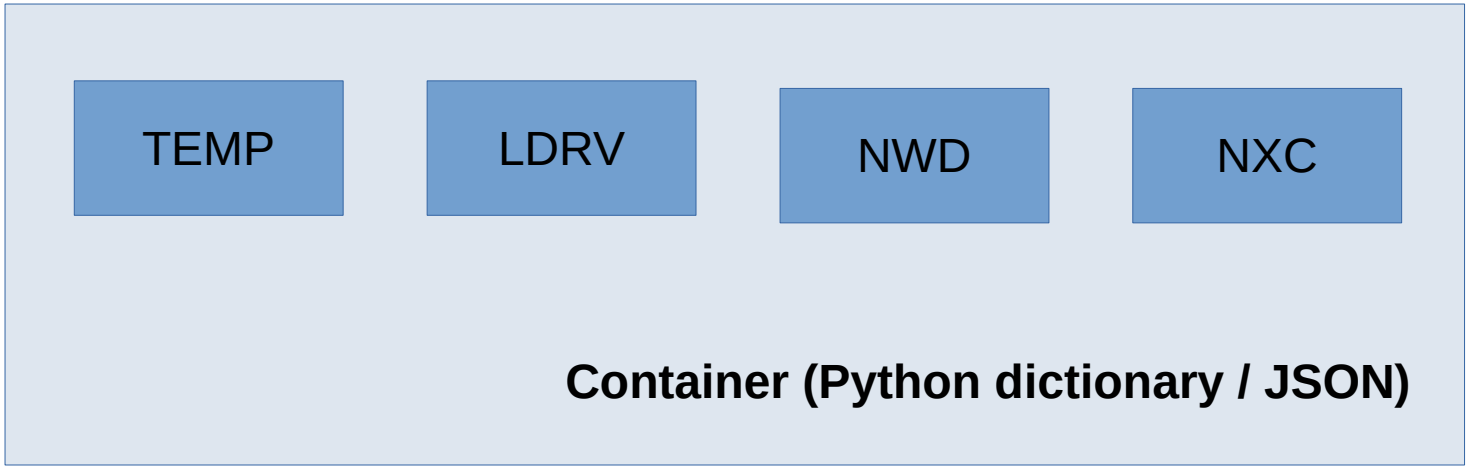
National Nuclear Data Center
Brookhaven National Laboratory
Upton, NY 11973-5000
www.nndc.bnl.gov

Notice: This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.


```
0.000000+0 0.000000+0          0                    481          1152925 1451    4
```

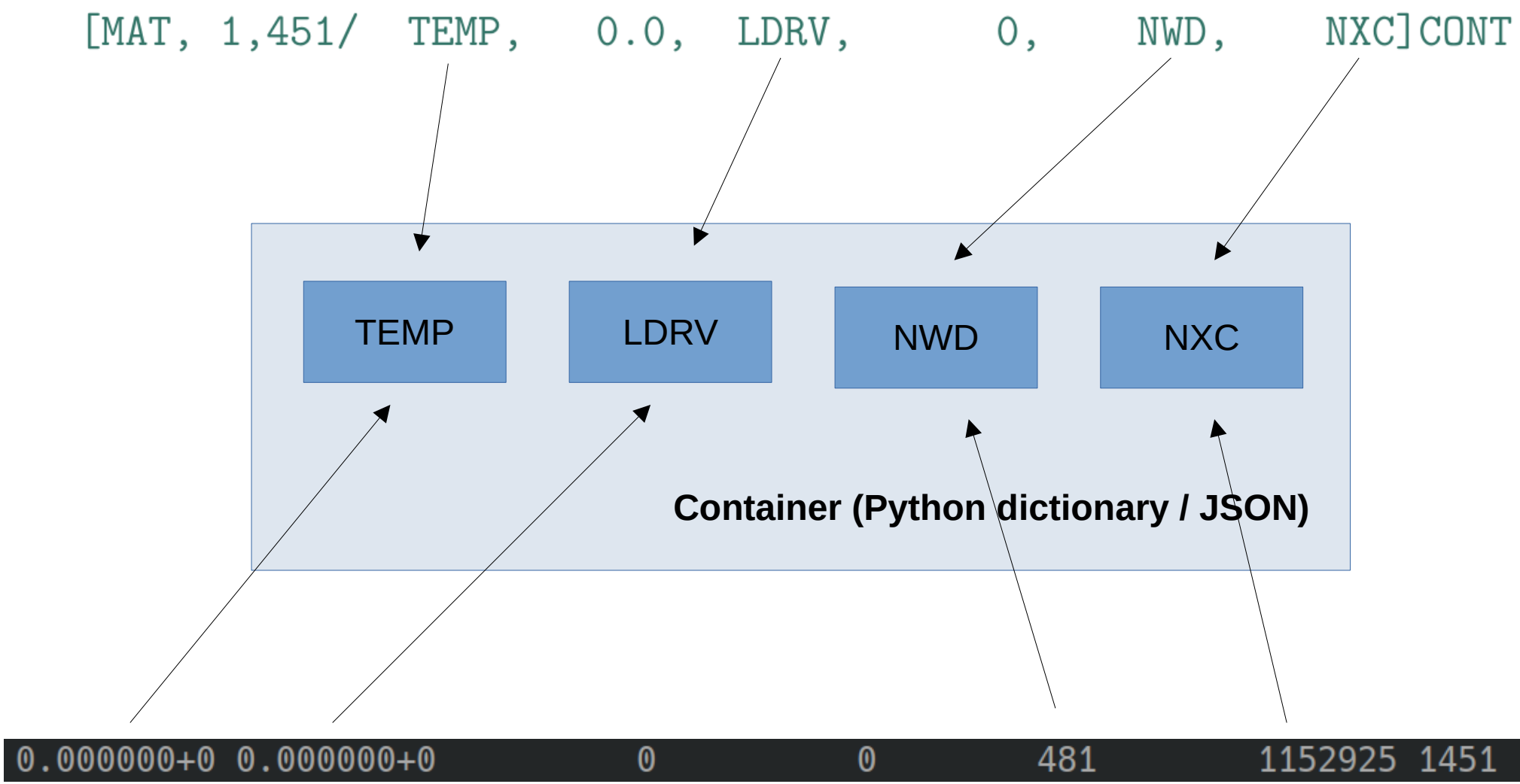
The better option:
Names and Numbers, unite!

```
[MAT, 1,451/  TEMP,    0.0,  LDRV,       0,     NWD,     NXC]CONT
```




```
0.000000+0 0.000000+0          0          0        481       1152925 1451    4
```

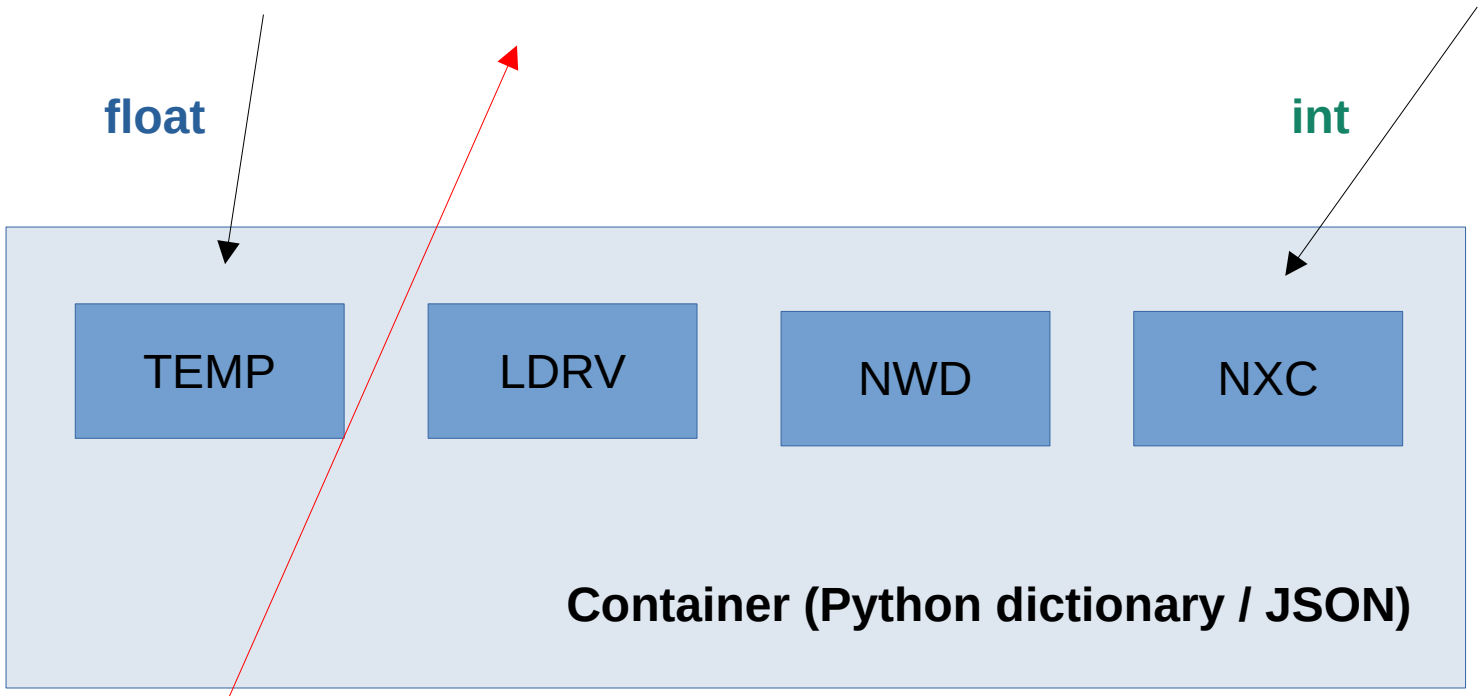
Record mappings
endf-parserpy can do the job




```
0.000000+0 0.000000+0          0          0        481       1152925 1451    4
```

endf-parserpy can do the job with number and type checking

```
[MAT, 1,451/  TEMP,    0.0,  LDRV,       0,     NWD,     NXC]CONT
```




```
0.000000+0 0.000000+0          0          0        481       1152925 1451    4
```

The human-readable loop

```
[MAT, 1,451/    ZA,    AWR,    LRP,     LFI,    NLIB,    NMOD]HEAD
[MAT, 1,451/  ELIS,    STA,    LIS,    LISO,       0,    NFOR]CONT
[MAT, 1,451/   AWI,   EMAX,   LREL,       0,    NSUB,    NVER]CONT
[MAT, 1,451/  TEMP,    0.0,   LDRV,       0,     NWD,     NXC]CONT
[MAT, 1,451/ZSYMAM,   ALAB, EDATE,  ...... AUTH ......    ]TEXT
[MAT, 1,451/ ... REF ... , DDATE,  RDATE, ENDATE, blank ]TEXT
[MAT, 1,451/  HSUB                                        ]TEXT
      ------------------------------------
      continue for the rest of the NWD descriptive records
      ------------------------------------
```

So we have a number of **NWD** TEXT records

The computer-readable loop

```
[MAT, 1,451/    ZA,    AWR,    LRP,     LFI,    NLIB,    NMOD]HEAD
[MAT, 1,451/  ELIS,    STA,    LIS,    LISO,       0,    NFOR]CONT
[MAT, 1,451/   AWI,   EMAX,   LREL,       0,    NSUB,    NVER]CONT
[MAT, 1,451/  TEMP,    0.0,   LDRV,       0,     NWD,     NXC]CONT
[MAT, 1,451/ZSYMAM,   ALAB, EDATE,  ...... AUTH ......    ]TEXT
[MAT, 1,451/ ... REF ... , DDATE,  RDATE, ENDATE, blank ]TEXT
[MAT, 1,451/  HSUB                                         ]TEXT
      ------------------------------------
      continue for the rest of the NWD descriptive records
      ------------------------------------
```

The formalized version understandable by endf-parserpy

```
[MAT, 1,451/ ZA, AWR, LRP, LFI, NLIB, NMOD]HEAD
[MAT, 1,451/ ELIS, STA, LIS, LISO, 0, NFOR]CONT
[MAT, 1,451/ AWI, EMAX, LREL, 0, NSUB, NVER]CONT
[MAT, 1,451/ TEMP, 0.0, LDRV, 0, NWD, NXC]CONT
for i=1 to NWD:
    [MAT, 1,451/ DESCRIPTION[i]]TEXT
endfor
```

Definition of arrays

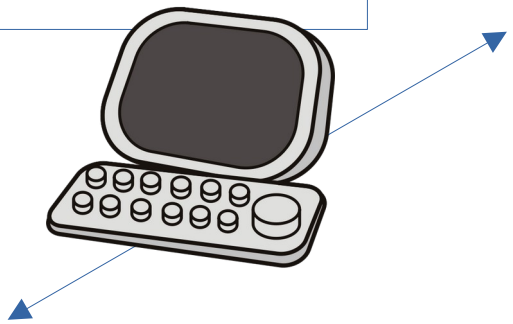
```
[MAT, 1,451/    ZA,    AWR,    LRP,     LFI,    NLIB,    NMOD]HEAD
[MAT, 1,451/  ELIS,    STA,    LIS,    LISO,       0,    NFOR]CONT
[MAT, 1,451/   AWI,   EMAX,   LREL,       0,    NSUB,    NVER]CONT
[MAT, 1,451/  TEMP,    0.0,   LDRV,       0,     NWD,     NXC]CONT
[MAT, 1,451/ZSYMAM,   ALAB, EDATE,  ...... AUTH ......    ]TEXT
[MAT, 1,451/ ... REF ... , DDATE,  RDATE, ENDATE, blank ]TEXT
[MAT, 1,451/  HSUB                                         ]TEXT
      ------------------------------------
      continue for the rest of the NWD descriptive records
      ------------------------------------
```

The formalized version understandable by endf-parserpy

```
[MAT, 1,451/ ZA, AWR, LRP, LFI, NLIB, NMOD]HEAD
[MAT, 1,451/ ELIS, STA, LIS, LISO, 0, NFOR]CONT
[MAT, 1,451/ AWI, EMAX, LREL, 0, NSUB, NVER]CONT
[MAT, 1,451/ TEMP, 0.0, LDRV, 0, NWD, NXC]CONT
for i=1 to NWD:
    [MAT, 1,451/ DESCRIPTION[i]]TEXT
endfor
```

MF1/MT451 recipe

```
[MAT, 1,451/ ZA, AWR, LRP, LFI, NLIB, NMOD]HEAD
[MAT, 1,451/ ELIS, STA, LIS, LISO, 0, NFOR]CONT
[MAT, 1,451/ AWI, EMAX, LREL, 0, NSUB, NVER]CONT
[MAT, 1,451/ TEMP, 0.0, LDRV, 0, NWD, NXC]CONT
for i=1 to NWD:
    [MAT, 1,451/ DESCRIPTION[i]]TEXT
endfor
```



JSON

- ZA
- AWR
- MF
- ...
- MT
- DESCRIPTION (array)

```
 2.906300+4 6.238900+1          1          0          0          52925 1451
 0.000000+0 0.000000+0          0          0          0          62925 1451
 1.000000+0 1.500000+8          8          0         10          72925 1451
 0.000000+0 0.000000+0          0          0        481        1152925 1451
 29-Cu- 63 LANL,ORNL  EVAL-FEB98 A.Koning,M.Chadwick,Hetrick      2925 1451
 CH98,CH99            DIST-DEC06 REV4-                 20011108   2925 1451
----ENDF/B-VII        MATERIAL 2925         REVISION 4            2925 1451
-----INCIDENT NEUTRON DATA                                       2925 1451
------ENDF-6 FORMAT                                             2925 1451
```

Loops in list bodies

2.2.1.1 SLBW and MLBW (LRF=1 or 2)

[MAT, 2,151/ AWRI, QX, L, LRX, 6*NRS, NRS
ER_1, AJ_1, GT_1, GN_1, GG_1, GF_1,
ER_2, AJ_2, GT_2, GN_2, GG_2, GF_2,
ER_{NRS}, AJ_{NRS}, GT_{NRS}, GN_{NRS}, GG_{NRS}, GF_{NRS}] LIST

Loops in list bodies

2.2.1.1 SLBW and MLBW (LRF=1 or 2)

[MAT, 2,151/ AWRI, QX, L, LRX, 6*NRS, NRS
ER_1, AJ_1, GT_1, GN_1, GG_1, GF_1,
ER_2, AJ_2, GT_2, GN_2, GG_2, GF_2,
ER_{NRS}, AJ_{NRS}, GT_{NRS}, GN_{NRS}, GG_{NRS}, GF_{NRS}] LIST

Formalized version

```
[MAT, 2,151/ AWRI, QX, L, LRX, 6*NRS, NRS /
{ER[k], AJ[k], GT[k], GN[k], GG[k], GF[k]}{k=1 to NRS} ]LIST
```

Conditionality: If statements

4.2.1 Legendre Polynomial Coefficients (LTT=1, LI=0)

When LTT=1 (angular distributions given in terms of Legendre polynomial coefficients), the structure of the section is:

```
[MAT, 4, MT/  ZA, AWR,  0, LTT,  0,  0]HEAD     (LTT=1)
[MAT, 4, MT/ 0.0, AWR, LI, LCT,  0,  0]CONT     (LI=0)
[MAT, 4, MT/ 0.0, 0.0,  0,   0, NR, NE/ E_int]TAB2
[MAT, 4, MT/   T,  E_1, LT,   0, NL,  0/ a_l(E_1)]LIST
[MAT, 4, MT/   T,  E_2, LT,   0, NL,  0/ a_l(E_2)]LIST
   ----------------------------------------
   ----------------------------------------
[MAT, 4, MT/   T,E_NE, LT,   0, NL,  0/ a_l(E_NE)]LIST
[MAT, 4,  0/ 0.0, 0.0,  0,   0,  0,  0]SEND
```

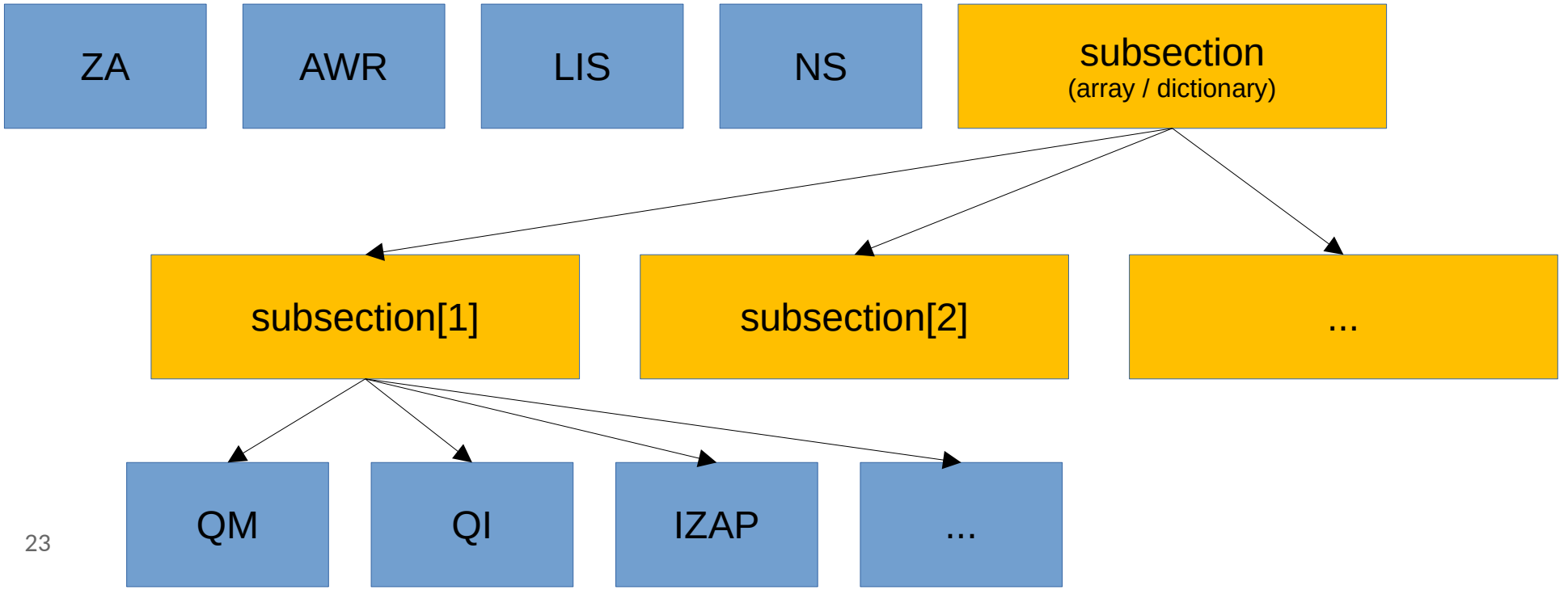
```
# Legendre coefficients
if LTT == 1 and LI == 0:
    [MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE/ Eint ]TAB2
    for i=1 to NE:
        [MAT, 4, MT/ T, E[i] , LT, 0, NL[i], 0/ {a[i,l]}{l=1 to NL[i]} ]LIST
    endfor
```

Allowing for nesting: Sections

```
[MAT, 10, MT/ ZA, AWR, LIS, 0, NS, 0]HEAD
for k=1 to NS:
(subsection[k])
    [MAT, 10, MT/ QM, QI, IZAP, LFS, NR, NP/ E / sigma ]TAB1
(/subsection[k])
endfor
SEND
```

Allowing for nesting: Sections

```
[MAT, 10, MT/ ZA, AWR, LIS, 0, NS, 0]HEAD
for k=1 to NS:
(subsection[k])
    [MAT, 10, MT/ QM, QI, IZAP, LFS, NR, NP/ E / sigma ]TAB1
(/subsection[k])
endfor
SEND
```



Current status and functionality of endf-parserpy

What ENDF-6 formats can already be read, written, validated and translated?

- **Already implemented:**

  MF 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 32, 33, 34, 35, 40

- **Still missing:**
  - MF 23, 26, 27, 28: Photo- and electro-atomic data
  - MF 30: Data covariance matrices obtained from parameter covariances
  - MF 31: Covariances of average number of neutrons per fission (nu-bar)
  - A few small gaps (e.g., MF1/MT460)

https://github.com/IAEA-NDS/endf-parserpy/tree/main/endf_parserpy/endf_recipes

Features

- Reading and writing
  - Fortran or C-style scientific notation
  - Flexible field width (possibly different from 11)
  - Tricks to increase precision for given width:
    - prefer_noexp: prefer “0.0123” over “1.2E-4” if more precise
    - abuse_signpos: “ 123.4567890” → “123.45678901”
    - skip_intzero: “0.123456789” → “.1234567890”
- Parsing
  - Accept/Ignore zero mismatches (0.0 in recipe vs != 0 in file)
  - Accept/Ignore inconsistent redundant variables specs
  - Accept/Ignore inconsistent non-zero number mismatches

How to use it for ENDF-6 format verification?

Format verification example: Debugging inconsistent variable specs

```
Failed: BasicEndfParser failed on file 18-Ar-40g.endf with exception
Here is the parser record log until failure:

-------- Line 0 -----------
Template:  [ MAT , 4 , MT / ZA , AWR , 0 , LTT , 0 , 0 ] HEAD
Line:     " 1.804000+4 3.961910+1          0          1          0          01837 4  2    1"

-------- Line 1 -----------
Template:  [ MAT , 4 , MT / 0.0 , AWR , LI , LCT , 0 , 0 ] CONT
Line:     " 0.000000+0 3.965640+1          0          2          0          01837 4  2    2"

Error message: Expected 39.6191 in the ENDF file but got 39.6564. The value was encountered
```

```
from endf_parserpy import BasicEndfParser
parser = BasicEndfParser()
endf_dic = parser.parsefile(endf_filepath)
```

Parser output for inconsistent variable specification

```
Failed: BasicEndfParser failed on file 18-Ar-40g.endf with exception
Here is the parser record log until failure:

-------- Line 0 -----------
Template:  [ MAT , 4 , MT / ZA , AWR , 0 , LTT , 0 , 0 ] HEAD
Line:     " 1.804000+4 3.961910+1          0          1          0          01837 4  2    1"

-------- Line 1 -----------
Template:  [ MAT , 4 , MT / 0.0 , AWR , LI , LCT , 0 , 0 ] CONT
Line:     " 0.000000+0 3.965640+1          0          2          0          01837 4  2    2"

Error message: Expected 39.6191 in the ENDF file but got 39.6564. The value was encountered
```

4.2.1 Legendre Polynomial Coefficients (LTT=1, LI=0)

When LTT=1 (angular distributions given in terms of Legendre polynomial coefficients), the structure of the section is:

[MAT, 4, MT/ ZA, AWR, 0, LTT, 0, 0]HEAD (LTT=1)
[MAT, 4, MT/ 0.0, AWR, LI, LCT, 0, 0]CONT (LI=0)
[MAT, 4, MT/ 0.0, 0.0, 0, 0, NR, NE/ E_{int}]TAB2
[MAT, 4, MT/ T, E_1, LT, 0, NL, 0/ $a_l(E_1)$]LIST
[MAT, 4, MT/ T, E_2, LT, 0, NL, 0/ $a_l(E_2)$]LIST

Inconsistent variable specification in mathematical expressions

```
-------- Line 11 -----------
Template:  [ MAT , 32 , 151 / AJ , PJ , 0 , 0 , 6 * NCH , NCH /
 { PPI [ k ] , L [ k ] , SCH [ k ] , BND [ k ] , APE [ k ] , APT [ k ] } { k = 1 to NCH } ] LIST
Line:     " 5.000000-1 0.000000+0          0          0         18          3 82532151   12"

-------- Line 15 -----------
Template:  [ MAT , 32 , 151 / 0.0 , 0.0 , 0 , NRSA , 12 * NX , NX ? /
 { ER [ k ] , { GAM [ p , k ] } { p = 1 to NCH } PADLINE ,
 DER [ k ] , { DGAM [ p , k ] } { p = 1 to NCH } PADLINE } { k = 1 to NRSA } ] LIST
Line:     " 0.000000+0 0.000000+0          0          4         48          8 82532151   16"

Error message: Expected 4 in the ENDF file but got 8. The value was encountered in a source field named N2
```

$$
\begin{aligned}
&[\mathrm{MAT},32,151/\ \mathrm{AJ},\quad \mathrm{PJ},\quad 0,\quad 0,\quad 6{*}\mathrm{NCH},\quad \mathrm{NCH}/\\
&\qquad \mathrm{PPI}_1,\quad \mathrm{L}_1,\quad \mathrm{SCH}_1,\quad \mathrm{BND}_1,\quad \mathrm{APE}_1,\quad \mathrm{APT}_1,\\
&\qquad \mathrm{PPI}_2,\quad \mathrm{L}_2,\quad \mathrm{SCH}_2,\quad \mathrm{BND}_2,\quad \mathrm{APE}_2,\quad \mathrm{APT}_2,\\
&\qquad\qquad \text{----------}\\
&\quad \mathrm{PPI}_{NCH},\quad \mathrm{L}_{NCH},\quad \mathrm{SCH}_{NCH},\quad \mathrm{BND}_{NCH},\quad \mathrm{APE}_{NCH},\quad \mathrm{APT}_{NCH}]\mathrm{LIST}\\
&\qquad\qquad \text{----------}\\
&[\mathrm{MAT},32,151/0.0,\quad 0.0,\quad 0,\quad \mathrm{NRSA},\quad 12{*}\mathrm{NX},\quad \mathrm{NX}/\\
&\qquad \mathrm{ER}_1,\quad \mathrm{GAM}_{1,1},\quad \mathrm{GAM}_{2,1},\quad \mathrm{GAM}_{3,1},\quad \mathrm{GAM}_{4,1},\quad \mathrm{GAM}_{5,1},\\
&\quad \mathrm{GAM}_{6,1},\ \ldots\quad \ldots\quad \ldots\quad \ldots\quad \ldots\quad \ldots\quad \ldots\quad .\ ,\quad \mathrm{GAM}_{NCH,1},
\end{aligned}
$$

Format verification example #2: Wrong sequence of records

```
Failed: BasicEndfParser failed on file 64-Gd-155g.endf with exception
Here is the parser record log until failure:

-------- Line 0 -----------
Template:  [ MAT , 4 , MT / ZA , AWR , 0 , LTT , 0 , 0 ] HEAD
Line:     " 6.415500+4 1.535920+2          1          1          0          06434 4  2    1"

-------- Line 1 -----------
Template:  [ MAT , 4 , MT / 0.0 , AWR , LI , LCT , 0 , 0 ] CONT
Line:     " 0.000000+0 1.535920+2          0          2        441        206434 4  2    2"

-------- Line 2 -----------
Template:  [ MAT , 4 , MT / 0.0 , 0.0 , 0 , 0 , NR , NE / Eint ] TAB2
Line:     " 1.000000+0 4.340500-3 8.478030-6-9.08196-19 0.000000+0 0.000000+06434 4  2    3"

Error message: invalid literal for int() with base 10: ' 8.478030-6'
```

Format verification example #2: Not a TAB2 record

```
Failed: BasicEndfParser failed on file 64-Gd-155g.endf with exception
Here is the parser record log until failure:

-------- Line 0 -----------
Template:  [ MAT , 4 , MT / ZA , AWR , 0 , LTT , 0 , 0 ] HEAD
Line:     " 6.415500+4 1.535920+2          1          1          0          06434 4  2    1"

-------- Line 1 -----------
Template:  [ MAT , 4 , MT / 0.0 , AWR , LI , LCT , 0 , 0 ] CONT
Line:     " 0.000000+0 1.535920+2          0          2        441        206434 4  2    2"

-------- Line 2 -----------
Template:  [ MAT , 4 , MT / 0.0 , 0.0 , 0 , 0 , NR , NE / Eint ] TAB2
Line:     " 1.000000+0 4.340500-3 8.478030-6 9.08196-19 0.000000+0 0.000000+06434 4  2    3"

Error message: invalid literal for int() with base 10: '8.478030-6'
```

4.2.1 Legendre Polynomial Coefficients (LTT=1, LI=0)

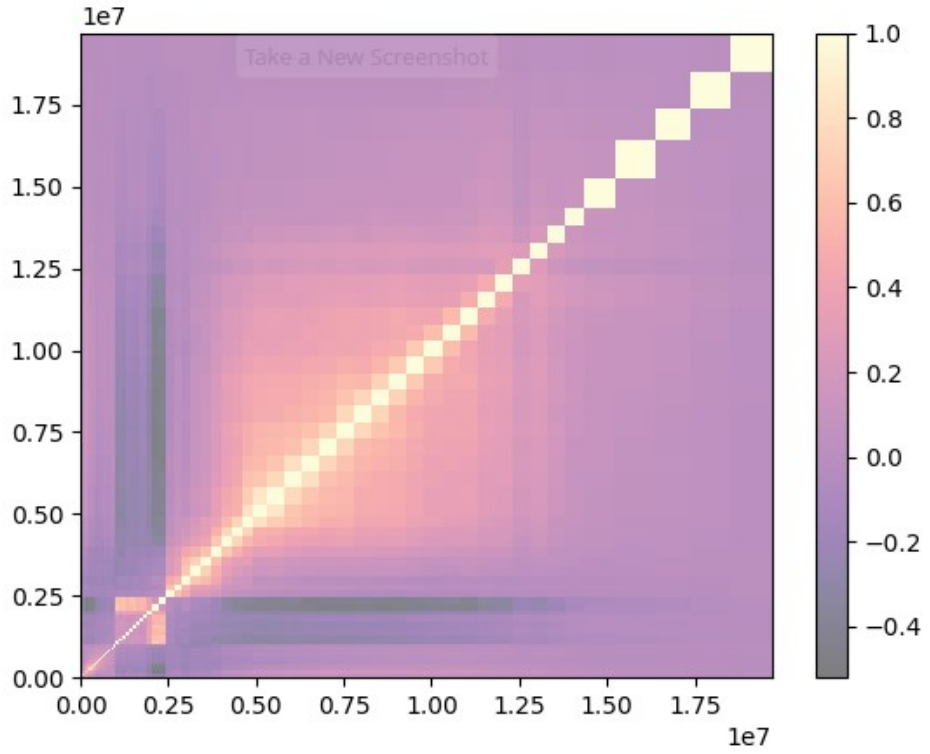
When LTT=1 (angular distributions given in terms of Legendre polynomial coefficients), the structure of the section is:

```
[MAT, 4, MT/  ZA, AWR,  0, LTT,  0,  0]HEAD      (LTT=1)
[MAT, 4, MT/ 0.0, AWR, LI, LCT,  0,  0]CONT      (LI=0)
[MAT, 4, MT/ 0.0, 0.0,  0,   0, NR, NE/ Eint]TAB2
[MAT, 4, MT/   T,  E1, LT,   0, NL,  0/ al(E1)]LIST
[MAT, 4, MT/   T,  E2, LT,   0, NL,  0/ al(E2)]LIST
```

How to use it for ENDF-6 file analysis, reading and writing?

Covariance retrieval example

```
parser = BasicEndfParser()
endf_file = os.path.join('auxiliary_data', 'IRDFF-II_Cf252_spectrum.endf')
cf252_endf = parser.parsefile(endf_file)
# get fission spectrum
mf3 = cf252_endf[3][261]
en = np.array(mf3['xstable']['E']) / 1e6
xs = np.array(mf3['xstable']['xs']) * 1e6
# get data to reconstruct covariance matrix
mf33 = cf252_endf[33][261]
E = mf33['subsection'][1]['ni_subsection'][1]['E']
F = mf33['subsection'][1]['ni_subsection'][1]['F']
# reconstruct covariance matrix
energies = np.array(tuple(E.values())) / 1e6
n = len(energies)-1
covmat = np.full((n, n), 0.0)
for i in F.keys():
    for j in F[i].keys():
        covmat[i-1, j-1] = F[i][j]
        if i != j:
            covmat[j-1, i-1] = F[i][j]
```



https://github.com/IAEA-NDS/neutron-standards-database/blob/main/codes/database_modifications/use_irdff_cf252_spectrum.py

Provenance checking: FENDL proton sublibrary

#	Material	Source	Emax [MeV]
1	1-H-1	JENDL-1	3000
2	1-H-2	ENDF/B-VII	150
3	1-H-3	ENDF/B-VII	20
4	2-He-3	ENDF/B-VII	20
5	3-Li-6	JENDL-4.0/HE	200
6	3-Li-7	JENDL-4.0/HE	200
7	4-Be-9	ENDF/B-VII	113
8	5-B-10	ENDF/B-VII	3
9	5-B-11	ENDF/B-?????	200

```
from endf_parserpy.endf_parser import BasicEndfParser
from endf_parserpy.debugging_utils import compare_objects
parser = BasicEndfParser()
fendl_endf = parser.parsefile(fendl_filename)
other_endf = parser.parsefile(other_endffile)
del fendl_endf[1][451]
del other_endf[1][451]
compare_objects(fendl_endf, other_endf, fail_on_diff=False)
```

Provenance checking: FENDL proton sublibrary

#	Material	Source	Emax [MeV]
1	1-H-1	JENDL-1	3000
2	1-H-2	ENDF/B-VII	150
3	1-H-3	ENDF/B-VII	20
4	2-He-3	ENDF/B-VII	20
5	3-Li-6	JENDL-4.0/HE	200
6	3-Li-7	JENDL-4.0/HE	200
7	4-Be-9	ENDF/B-VII	113
8	5-B-10	ENDF/B-VII	3
9	5-B-11	ENDF/B-?????	200

TENDL-2011

```
from endf_parserpy.endf_parser import BasicEndfParser
from endf_parserpy.debugging_utils import compare_objects
parser = BasicEndfParser()
fendl_endf = parser.parsefile(fendl_filename)
other_endf = parser.parsefile(other_endffile)
del fendl_endf[1][451]
del other_endf[1][451]
compare_objects(fendl_endf, other_endf, fail_on_diff=False)
```

```
---- difference for MAT 125 ------
at path .3: only obj2 contains {208, 209, 210}
at path .6: only obj2 contains {208, 209, 210}
```

JENDL-2007/HE

Provenance checking: FENDL proton sublibrary

#	Material	Source	Emax [MeV]
1	1-H-1	JENDL-1	3000
2	1-H-2	ENDF/B-VII	150
3	1-H-3	ENDF/B-VII	20
4	2-He-3	ENDF/B-VII	20
5	3-Li-6	JENDL-4.0/HE	200
6	3-Li-7	JENDL-4.0/HE	200
7	4-Be-9	ENDF/B-VII	113
8	5-B-10	ENDF/B-VII	3
9	5-B-11	ENDF/B-?????	200

```
from endf_parserpy.endf_parser import BasicEndfParser
from endf_parserpy.debugging_utils import compare_objects
parser = BasicEndfParser()
fendl_endf = parser.parsefile(fendl_filename)
other_endf = parser.parsefile(other_endffile)
del fendl_endf[1][451]
del other_endf[1][451]
compare_objects(fendl_endf, other_endf, fail_on_diff=False)
```

FENDL 3.2b = ENDF/B.VII.0

```
---- difference for MAT 131 ------
Value mismatch at .3.50.QI (-0.76387 vs -763870.0)
Value mismatch at .3.50.QM (-0.76387 vs -763870.0)
Value mismatch at .3.650.QI (-4.0329 vs -4032900.0)
Value mismatch at .3.650.QM (-4.0329 vs -4032900.0)
```

Comparison with ENDF/B-VII.1

Correcting and creating ENDF-6 formatted files

```
---- difference for MAT 131 ------
Value mismatch at .3.50.QI (-0.76387 vs -763870.0)
Value mismatch at .3.50.QM (-0.76387 vs -763870.0)
Value mismatch at .3.650.QI (-4.0329 vs -4032900.0)
Value mismatch at .3.650.QM (-4.0329 vs -4032900.0)
```

Update dictionaries and write out ENDF file

```
fendl_endf[3][50]['QI'] = other_endf[3][50]['QI']
fendl_endf[3][50]['QM'] = other_endf[3][50]['QM']
fendl_endf[3][650]['QI'] = other_endf[3][650]['QI']
fendl_endf[3][650]['QM'] = other_endf[3][650]['QM']
parser.writefile('corrected_file.endf', fendl_endf)
```

Bonus: Cu-63 checking

```
-------- Line 1205 -----------
Template:  [ MAT , 32 , 151 / 0.0 , 0.0 , 0 , NRSA , 12 * NX , NX ? /
 { ER [ k ] , { GAM [ p , k ] } { p = 1 to NCH } PADLINE ,
 DER [ k ] , { DGAM [ p , k ] } { p = 1 to NCH } PADLINE } { k = 1 to NRSA } ] LIST
Line:     " 0.000000+0 0.000000+0          0        156       1872        156292532151"

-------- Line 1518 -----------
Template:  [ MAT , 32 , 151 / AJ , PJ , 0 , 0 , 6 * NCH , NCH /
 { PPI [ k ] , L [ k ] , SCH [ k ] , BND [ k ] , APE [ k ] , APT [ k ] } { k = 1 to NCH } ] LIST
Line:     " 2.000000+0 0.000000+0          0          0         18          3292532151"

-------- Line 1522 -----------
Template:  [ MAT , 32 , 151 / 0.0 , 0.0 , 0 , NRSA , 12 * NX , NX ? /
 { ER [ k ] , { GAM [ p , k ] } { p = 1 to NCH } PADLINE ,
 DER [ k ] , { DGAM [ p , k ] } { p = 1 to NCH } PADLINE } { k = 1 to NRSA } ] LIST
Line:     " 0.000000+0 0.000000+0          0        148       1776        148292532151"

-------- Line 1819 -----------
Template:  [ MAT , 32 , 151 / AJ , PJ , 0 , 0 , 6 * NCH , NCH /
 { PPI [ k ] , L [ k ] , SCH [ k ] , BND [ k ] , APE [ k ] , APT [ k ] } { k = 1 to NCH } ] LIST
Line:     " 3.000000+0 0.000000+0          0          0         12          2292532151"

-------- Line 1822 -----------
Template:  [ MAT , 32 , 151 / 0.0 , 0.0 , 0 , NRSA , 12 * NX , NX ? /
 { ER [ k ] , { GAM [ p , k ] } { p = 1 to NCH } PADLINE ,
 DER [ k ] , { DGAM [ p , k ] } { p = 1 to NCH } PADLINE } { k = 1 to NRSA } ] LIST
Line:     " 0.000000+0 0.000000+0          0        204       2448        204292532151"

-------- Line 2231 -----------
Template:  [ MAT , 32 , 151 / 0.0 , 0.0 , NDIGIT , NNN , NM , 0 ] CONT
Line:     " 0.000000+0 0.000000+0          2       3598       4260          0292532151"

-------- Line 2232 -----------
Template:  [ MAT , 32 , 151 / II [ q ] , JJ [ q ] , KIJ [ q ] { NDIGIT } ] INTG
Line:     " 3.100000+1-9.810000+2 0.000000+0 0.000000+0 0.000000+0 0.000000+0292532151"

Error message: invalid literal for int() with base 10: ' 3.10'
```

Summary and conclusions

- New Python package endf-parserpy enables reading and writing of ENDF-6 formatted files as well as format verification and JSON translation
- Package relies on a formalized version of the ENDF-6 format specifications in the manual
- Already rather comprehensive coverage of the ENDF-6 format (photo-atomic MF numbers and MF 30 and 31 missing)
- Flexible reading and writing capabilities (flexible field width, Fortran or C-style scientific format, etc.)
- Easy manipulation of ENDF-6 files by leveraging Python dictionary manipulation capabilities (adding, removing, replacing MF/MT numbers are one-liners)
- Several examples of how to leverage this package (ENDF-6 format verification, ENDF file comparisons, modifications)
- Available on IAEA-NDS GitHub account under MIT license