Generic Reduction Procedure

1) READ INPUT:
   User input: Runnumbers (REF, DB, WATER, INST BACK)
   Metadata: slits, detector angle, sample angle, MONITOR, TIME …
   User input data: foreground range, background range, wavelength range, grouping,
   Detector: 2D, line, 1D
2) Determine Measurement Type
   A: TOF (POL or UPOL)
   B: MONO (POL or UPOL)
3) Determine if kinetic / streaming
4) Determine XYZ coordinate of axis
   IF detector XY: Integrate loose collimation direction
   IF MONO: determine scan axis, stack data on scan axis direction
5) Sort or group measurements to measurement type
   NOTE: different measurements can be stored in 5^{th} dimension:
   1: x-axis, 2: y-axis, 3: intensity, 4: polarization, 5: time/streaming/temperature/fields/etc…
   IF MONO: sort scan axis, sort polarizations, sort measurements in sequence
   IF TOF: sort polarizations, sort angles, sort measurements in sequence
   IF kinetic: loop through slices
   IF streaming: sort cyclic data and average similar input
6) Calculate errors and propagate appropriately through following steps
7) Determine foreground (ROI) and background (back)
   IF no DB: DB=monitor OR DB=1
   NOTE: Here a binning can take place, but that requires calculating lambda, theta and the resolutions before.
8) Normalization:
   slits, water, monitor/time
9) Subtract instrument background from REF and DB, subtract background from REF and DB (averaged or fitted)
10) Average data at similar XY coordinate
    IF MONO: average same theta/2theta values
    IF TOF: probably no further averaging
11) CASE: Gravity correction for horizontal reflectometer
12) Calculate missing axis theta, 2theta, lambda,
    CASE: Reflection UP or DOWN in horizontal reflectometer
    CASE: Coherent or incoherent analysis
13) Calculate angular width on detector of REF and DB => sample waviness for coherent
14) IF DB supplied: Integrate DB foreground
    IF TOF: => 1D DB(lambda)
    IF MONO: => 1D DB(scan axis)
15) IF POL: Correct 1D DB for polarization efficiency

16) Calculate 1D reflectivity:
    CASE: incoherent
        Integrate at constant lambda over 2theta => 1D REF
        Divide 1D REF by 1D DB => 1D REF/DB + E
    CASE: coherent
        Divide 2D REF by 1D DB column wise
        Regroup data within new wavelength limits onto a given 2theta line
            CASE: bent sample
            CASE: divergent beam
            => 1D REF/DB + E
17) IF TOF POL: loop to get all the polarizations to correct for efficiencies the 1D REF/DB as a function of lambda
    IF POL MONO: loop to get all polarizations to correct for polarization efficiency at fixed lambda
18) Calculate resolutions in theta, lambda
    CASE: incoherent
    CASE: coherent
19) Calculate Q
20) Group to a fraction of the Q resolution
    NOTE: for the incoherent method it is possible here to use 1D REF(rebinned)/1D DB (rebinned)
21) Calculate 2D reflectivity in requested coordinates: QX/QZ, pi/pf/ theta/2theta
    Divide 2D REF by 1D DB column wise
22) Perform polarization efficiency correction on all spin channels column wise
23) Update storage with direct beam
24) IF kinetic: loop over slices
25) IF multiple datasets: loop over 5th dimension
26) Perform final normalization corrections
27) Join data corresponding to 1 measurement
28) Store 1D and 2D outputs in readable format with metadata

```
    PHIa1   = 0.0114
    PHIk1   = -0.0005
    PHIk2   = 0.0014
    PHIk3   = 0.0087
    PHIk4   = 0.0121
    PHIk5   = 0.0196
    PHIxi1  = 8.1299
    PHIxi2  = 13.0000
    PHIxi3  = 16.1075
    PHIxi4  = 20.0963
    PHIyi1 = PHIa1+PHIk1*PHIxi1
    PHIyi2 = PHIyi1+PHIk2*(PHIxi2-PHIxi1)
    PHIyi3 = PHIyi2+PHIk3*(PHIxi3-PHIxi2)
    PHIyi4 = PHIyi3+PHIk4*(PHIxi4-PHIxi3)



xin=xinA[j]
        ;The efficiency at the lambda point:
        ;;**F1
        if xin le F1xi1 then F1L[j] = F1a1+ F1k1*xin else $
        if xin le F1xi2 then F1L[j] = F1yi1+ F1k2*(xin-F1xi1) else $
        if xin le F1xi3 then F1L[j] = F1yi2+ F1k3*(xin-F1xi2) else $
        if xin le F1xi4 then F1L[j] = F1yi3+ F1k4*(xin-F1xi3) else F1L[j] =
F1yi4+ F1k5*(xin-F1xi4)
        ;;**F2
        if xin le F2xi1 then F2L[j] = F2a1+ F2k1*xin else $
        if xin le F2xi2 then F2L[j] = F2yi1+ F2k2*(xin-F2xi1) else $
        if xin le F2xi3 then F2L[j] = F2yi2+ F2k3*(xin-F2xi2) else $
        if xin le F2xi4 then F2L[j] = F2yi3+ F2k4*(xin-F2xi3) else F2L[j] =
F2yi4+ F2k5*(xin-F2xi4)
        ;;**F2
        if xin le PHIxi1 then PHIL[j] = PHIa1+ PHIk1*xin else $
        if xin le PHIxi2 then PHIL[j] = PHIyi1+ PHIk2*(xin-PHIxi1) else $
        if xin le PHIxi3 then PHIL[j] = PHIyi2+ PHIk3*(xin-PHIxi2) else $
        if xin le PHIxi4 then PHIL[j] = PHIyi3+ PHIk4*(xin-PHIxi3) else
PHIL[j] = PHIyi4+ PHIk5*(xin-PHIxi4)
        eF1L[j]=0.001*F1L[j];
        eF2L[j]=0.001*F2L[j];
        ePHIL[j]=0.001*PHIL[j];
```