

NAME

ciaaw - library for isotopic abundances and standard atomic weights

LIBRARY

ciaaw (-libciaaw, -lciaaw)

SYNOPSIS

```
use ciaaw
include "ciaaw.h"
import pyciaaw
```

DESCRIPTION

ciaaw is a Fortran library providing the standard and abridged atomic weights, the isotopic abundance and the isotopes' standard atomic weights. The data are taken from <http://ciaaw.org>. C API allows usage from C, or can be used as a basis for other wrappers. Python wrapper allows easy usage from Python.

What have been implemented:

- o **SAW** Standard Atomic Weights.
- o **ICE** Isotopic Composition of the Element
- o **NAW** Nuclides Atomic Weight.

The latest standard atomic weights were released in 2021 by the *ciaaw* (<https://www.ciaaw.org>). All the values for the atomic weights are provided as double precision reals. The standard atomic weights (or relative atomic mass), $A_r(E)$, are extracted from table 1 from Prohaska et al. (IUPAC Technical Report **94**(5):573–600). For the elements that feature an interval for the standard atomic weight, the mean value and the uncertainty are computed using formulas defined in [2]. The standard atomic weights are a dimensionless quantity and thus they need to be multiplied by the molar mass constant Mu in order to get the value in g.mol-1. See codata for physical constants.

The latest isotopic compositions were released in 2013 by the *ciaaw* (<https://www.ciaaw.org>). All the values for the compositions are provided as double precision reals. The isotopic compositions of the element, are extracted from table 1 from Meija et al. (IUPAC Technical Report. **88**(3):293–306).

The latest atomic weights for nuclides were released in 2020 by *ciaaw* <https://www.ciaaw.org> from Huang et al. (AME 2020 **45**(3):030002). All the values for the nuclide atomic weights are provided as double precision reals.

Fortran API

- o **function get_version(result(fptr))**
Get the version
- o **character(len=:), pointer :: fptr**
Fortran pointer to a string indicating the version..
- o **function is_in_pt(z,result(res))**
Check if the atomic number z is in the periodic table
- o **integer(int32), intent(in) :: z**
Atomic number
- o **logical :: res**
True or False
- o **function get_z_by_symbol(s,result(res))**
Get the atomic number z of the element defined by the symbol s.
- o **character(len=*), intent(in) :: s**
Element symbol

o integer(int32) :: res
 >0 if found and -1 if not found.

o subroutine print_periodic_table()
 Print periodic table.

o function get_saw(s, abridged, uncertainty)result(res)
 Get the standard atomic weight for the element s.

o character(len=*), intent(in) :: s
 Element symbol.

o logical, intent(in), optional :: abridged
 Set to False if the abridged value is not desired. Default to TRUE.

o logical, intent(in), optional :: uncertainty
 Set to True if the uncertainty is desired. Default to FALSE.

o real(dp) :: res
 NaN if the provided element is incorrect or -1 if the element does not have a SAW.

o function get_ice(s, A, uncertainty)result(res)
 Get the isotopic composition of the element s for the mass number A.

o character(len=*), intent(in) :: s
 Element symbol.

o integer(int32), intent(in) :: A
 Mass number.

o logical, intent(in), optional :: uncertainty
 Set to True if the uncertainty is desired. Default to FALSE.

o real(dp) :: res
 NaN if the provided element or the mass number A are incorrect or -1 if the element does not have an ICE.

o function get_nice(s)result(res)
 Get the number of isotopes in ICE of the element s.

o character(len=*), intent(in) :: s
 Element symbol.

o integer(int32) :: res
 >0 if found or -1 if not found.

o ! function get_ice_values(s)result(res)
o character(len=*), intent(in) :: s Element symbol.

o function get_naw(s, A, uncertainty)result(res)
 Get the atomic weight of the nuclide s for the mass number A.

o character(len=*), intent(in) :: s
 Element symbol.

o integer(int32), intent(in) :: A
 Mass number.

o logical, intent(in), optional :: uncertainty
 Flag for returning the uncertainty instead of the value. Default to FALSE.

o real(dp) :: res
 NaN if the provided element or A are incorrect or -1 if the element does not have an NAW.

o function get_nnaw(s)result(res)
Get the number of nuclides in NAW of the element s.

o character(len=*) intent(in) :: s
Element symbol.

o integer(int32) :: res
>0 if found or -1 if not found.

C API

- `char* ciaaw_get_version(void)`
- `double ciaaw_get_saw(char *s, int n, bool abridged, bool uncertainty)`
- `double ciaaw_get_ice(char *s, int n, int A, bool uncertainty)`
- `int ciaaw_get_nice(char *s, int n)`
- `double ciaaw_get_naw(char *s, int n, int A, bool uncertainty)`
- `int ciaaw_get_nnaw(char *s, int n)`

Python wrappers

- `get_saw(s: str, abridged: bool=True, uncertainty: bool=False)->float`
- `get_ice(s:str, A:int, uncertainty: bool=False)->float`
- `get_nice(s:str)->int`
- `get_naw(s:str, A:int, uncertainty: bool=False)->float`
- `get_nnaw(s:str)->int`

NOTES

To use `ciaaw` within your fpm project, add the following to your `fpm.toml` file:

```
[dependencies]
iapws = { git="https://github.com/MilanSkocic/ciaaw.git" }
```

`dp` stands for double precision and it is an alias to `real64` from the `iso_fortran_env` module.

EXAMPLES

Example in Fortran:

```
program example_in_f
use ciaaw
implicit none

character(len=8) :: s

! ASAW = Abridged Standard Atomic Weight
! SAW = Standard Atomic Weight
! ICE = Isotopic Composition of the Element
! NAW = Nuclide Atomic Weight
! U = Uncertainty

print '(A)', '# ##### CIAAW VERSION #####'
print *, "version ", get_version()

print '(A)', '# ##### CIAAW SAW #####'
print '(A10, F10.5)', 'ASAW H = ', get_saw("H", abridged=.true.)
print '(A10, F10.5)', 'U ASAW H = ', get_saw("H", uncertainty=.true.)
print '(A10, F10.5)', 'SAW H = ', get_saw("H", abridged = .false.)
print '(A10, F10.5)', 'U SAW H = ', get_saw("H", abridged = .false., un
```

```

print '(A10, F10.5)', 'ASAW T      = ', get_saw("Tc", abridged=.true.)

print '(A)', '##### CIAAW ICE #####'
print '(A, I3)',      'N ICE H     = ', get_nice("H")
print '(A, F12.6)',   'ICE H 1     = ', get_ice("H", A=1)
print '(A, ES23.16)', 'U ICE H 1    = ', get_ice("H", A=1, uncertainty=.true.)
print '(A, F12.6)',   'ICE H 2     = ', get_ice("H", A=2)
print '(A, ES23.16)', 'U ICE H 2    = ', get_ice("H", A=2, uncertainty=.true.)
print '(A, I3)',      'N ICE Tc    = ', get_nice("Tc")
print '(A, I3)',      'N ICE C     = ', get_nice("C")

print '(A)', '##### CIAAW NAW #####'
print '(A, ES23.16)', 'NAW H 2     = ', get_naw("H", A=2)
print '(A, ES23.16)', 'U NAW H 2    = ', get_naw("H", A=2, uncertainty=.true.)
print '(A, I3)',      'N NAW Tc    = ', get_nnaw("Tc")
end program

```

Example in C:

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "ciaaw.h"

// ASAW = Abridged Standard Atomic Weight
// SAW  = Standard Atomic Weight
// ICE  = Isotopic Composition of the Element
// NAW  = Nuclide Atomic Weight
// U    = Uncertainty

int main(void){
    printf("%s0, ##### CIAAW VERSION #####");
    printf("version %s0, ciaaw_get_version());

    printf("%s0, ##### CIAAW SAW #####");
    printf("%s %10.5f0, "ASAW H     = ", ciaaw_get_saw("H", 1, true, false))
    printf("%s %10.5f0, "U ASAW H   = ", ciaaw_get_saw("H", 1, true, true))
    printf("%s %10.5f0, "SAW H     = ", ciaaw_get_saw("H", 1, false, false))
    printf("%s %10.5f0, "U SAW H   = ", ciaaw_get_saw("H", 1, false, true))
    printf("%s %10.5f0, "ASAW Tc   = ", ciaaw_get_saw("Tc", 2, true, false))

    printf("%s0, ##### CIAAW ICE #####");
    printf("%s %d0,      "N ICE H     = ", ciaaw_get_nice("H", 1));
    printf("%s %12.6f0,   "ICE H 1     = ", ciaaw_get_ice("H", 1, 1, false))
    printf("%s %23.16e0, "U ICE H 1    = ", ciaaw_get_ice("H", 1, 1, true))
    printf("%s %12.6f0,   "ICE H 2     = ", ciaaw_get_ice("H", 1, 2, false))
    printf("%s %23.16e0, "U ICE H 2    = ", ciaaw_get_ice("H", 1, 2, true))
    printf("%s %d0,      "N ICE Tc    = ", ciaaw_get_nice("Tc", 2));
    printf("%s %d0,      "N ICE C     = ", ciaaw_get_nice("C", 1));

    printf("%s0, ##### CIAAW NAW #####");
    printf("%s %23.16f0, "NAW H 2     = ", ciaaw_get_naw("H", 1, 2, false))
    printf("%s %23.16e0, "U NAW H 2    = ", ciaaw_get_naw("H", 1, 2, true))

```

```

    printf("%s %d0,      "N NAW Tc      = " , ciaaw_get_nnaw("Tc", 2));
    return EXIT_SUCCESS;
}

```

Example in Python:

```

import pyciaaw

# ASA = Abridged Standard Atomic Weight
# SAW = Standard Atomic Weight
# ICE = Isotopic Composition of the Element
# NAW = Nuclide Atomic Weight
# U   = Uncertainty

print("##### CIAAW VERSION #####")
print("version ", pyciaaw.__version__)

print("##### CIAAW SAW #####")
print("ASA H = ", pyciaaw.get_saw("H"))
print("U ASA H = ", pyciaaw.get_saw("H", uncertainty=True))
print("SAW H = ", pyciaaw.get_saw("H", abridged=False, uncertainty=False))
print("U SAW H = ", pyciaaw.get_saw("H", abridged=False, uncertainty=True))
print("ASA Tc = ", pyciaaw.get_saw("Tc"))

print("##### CIAAW ICE #####")
print("N ICE H = ", pyciaaw.get_nice("H"))
print('ICE H 1 = ', pyciaaw.get_ice("H", A=1))
print('U ICE H 1 = ', pyciaaw.get_ice("H", A=1, uncertainty=True))
print('ICE H 2 = ', pyciaaw.get_ice("H", A=2))
print('U ICE H 2 = ', pyciaaw.get_ice("H", A=2, uncertainty=True))
print("N ICE Tc = ", pyciaaw.get_nice("Tc"))
print("N ICE C = ", pyciaaw.get_nice("C"))

print("##### CIAAW NAW #####")
print('NAW H 2 = ', pyciaaw.get_naw("H", A=2))
print('U NAW H 2 = ', pyciaaw.get_naw("H", A=2, uncertainty=True))
print("N NAW Tc = ", pyciaaw.get_nnaw("Tc"))

```

SEE ALSO

[gsl\(3\)](#), [codata\(3\)](#)