ciaaw

Table of Contents

NAME

ciaaw – library for isotopic abundances and standard atomic weights

LIBRARY

ciaaw library (*libciaaw*, *–lciaaw*)

SYNOPSIS

Fortran **use ciaaw**

C **#include "ciaaw.h"**

Python **import pyciaaw**

DESCRIPTION

ciaaw is a Fortran library providing the standard and abridged atomic weights, the isotopic abundance and the isotopesâ standard atomic weights:

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

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.

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 realtive atomic mass), A_r(E), are extracted from table 1 from Prohaska et al. (IUPAC Technical Report 94(5):573â600). For the elements that feature an interval for the standard atomic weight, the mean value and the uncertainty are computed using formulas defined in IUPAC Technical Report 93(5):629â646. 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):293â306).

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.

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. The definitions of the acronyms:

ASAW Abridged Standard Atomic Weight

SAW	Standard Atomic Weight
ICE	Isotopic Composition of the Element
NAW	Nuclide Atomic Weight

The definitions of the abbreviations:

s	Element
Z	Atomic number
A	Mass number
u	Uncertainty
ab	Abridged
res	Result

References

[1] J. Meija et al., "Isotopic Compositions of the Elements 2013 (IUPAC Technical Report)," vol. 88, no. 3, pp. 293â306, 2016, doi: 10.1515/pac-2015-0503.

[2] W. J. Huang, M. Wang, F. G. Kondev, G. Audi, and S. Naimi, "The AME 2020 Atomic Mass Evaluation (I). Evaluation of Input Data, and Adjustment Procedures," Chinese Physics C, vol. 45, no. 3, p. 30002, Mar. 2021, doi: 10.1088/1674-1137/abddb0.

[3] A. M. H. van der Veen, J. Meija, A. Possolo, and D. B. Hibbert, "Interpretation and Use of Standard Atomic Weights (IUPAC Technical Report)," Pure and Applied Chemistry, vol. 93, no. 5, pp. 629-646, 2021.

[4] T. Prohaska et al., "Standard Atomic Weights of the Elements 2021 (IUPAC Technical Report)," vol. 94, no. 5, pp. 573-600, 2022, doi: 10.1515/pac-2019-0603.

EXAMPLES

Example in Fortran:

```
! EXAMPLE IN FORTRAN
program example_in_f
use ciaaw, only: saw, ice, naw, nice, nnaw, version
implicit none(type,external)
print *, "version ", version()

print '(A)', '########### CIAAW SAW ##########'
print '(A10, F10.5)', 'ASAW H = ', saw("H", ab=.true.)
print '(A10, F10.5)', 'U ASAW H = ', saw("H", u=.true.)
print '(A10, F10.5)', 'SAW H = ', saw("H", ab=.false.)
print '(A10, F10.5)', 'U SAW H = ', saw("H", ab=.false., u=.true.)
print '(A10, F10.5)', 'ASAW T = ', saw("Tc", ab=.true.)

print '(A)', '########### CIAAW ICE ##########'
print '(A, I3)', 'N ICE H = ', nice("H")
print '(A, F12.6)', 'ICE H 1 = ', ice("H", A=1)
print '(A, ES23.16)', 'U ICE H 1 = ', ice("H", A=1, u=.true.)
print '(A, F12.6)', 'ICE H 2 = ', ice("H", A=2)
print '(A, ES23.16)', 'U ICE H 2 = ', ice("H", A=2, u=.true.)
print '(A, I3)', 'N ICE Tc = ', nice("Tc")
print '(A, I3)', 'N ICE C = ', nice("C")
```

```
print '(A)', '########### CIAAW NAW ##########'
print '(A, ES23.16)', 'NAW H 2 = ', naw("H", A=2)
print '(A, ES23.16)', 'U NAW H 2 = ', naw("H", A=2, u=.true.)
print '(A, I3)', 'N NAW Tc = ', nnaw("Tc")

end program example_in_f
```

Example in C

```
/* EXAMPLE IN C */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "ciaaw.h"

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

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

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

printf("%s0, "########## CIAAW NAW ##########");
printf("%s %23.16f0, "NAW H 2      = ", ciaaw_naw("H", 1, 2, false));
printf("%s %23.16e0, "U NAW H 2    = ", ciaaw_naw("H", 1, 2, true));
printf("%s %d0,      "N NAW Tc     = ", ciaaw_nnaw("Tc", 2));
return EXIT_SUCCESS;
}
```

Example in Python

```
#!/usr/bin/env python
r"""EXAMPLE IN PYTHON"""
import sys
sys.path.insert(0, "../py/src/")
import pyciaaw

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

print("########## CIAAW SAW  ##########")
print("ASAW H   = ", pyciaaw.saw("H"))
```

```
print("U ASAW H = ", pyciaaw.saw("H", u=True))
print("SAW H    = ", pyciaaw.saw("H", ab=False, u=False))
print("U SAW H  = ", pyciaaw.saw("H", ab=False, u=True))
print("ASAW Tc  = ", pyciaaw.saw("Tc"))

print("########## CIAAW ICE  ##########")
print("N ICE H   = ", pyciaaw.nice("H"))
print('ICE H 1   = ', pyciaaw.ice("H", A=1))
print('U ICE H 1 = ', pyciaaw.ice("H", A=1, u=True))
print('ICE H 2   = ', pyciaaw.ice("H", A=2))
print('U ICE H 2 = ', pyciaaw.ice("H", A=2, u=True))
print("N ICE Tc  = ", pyciaaw.nice("Tc"))
print("N ICE C   = ", pyciaaw.nice("C"))

print("########## CIAAW NAW  ##########")
print('NAW H 2   = ', pyciaaw.naw("H", A=2))
print('U NAW H 2 = ', pyciaaw.naw("H", A=2, u=True))
print("N NAW Tc  = ", pyciaaw.nnaw("Tc"))
```

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

ciaaw − CLI for ciaaw

SYNOPSIS

ciaaw [*OPTION*]... [*ELEMENT*]...

DESCRIPTION

ciaaw is a command line interface which provides the atomic weights, the isotopic compositions and the nuclides atomic weights. If no element is provided the full periodic table is displayed.

OPTIONS

-s, --saw
: Get the standard atomic weight.

-i, --ice Get the isotopic composition.

-n, --naw
: Get the nuclide atomic weight.

-m, --mu
: Get the molar masses in g/mol by multiplying the atomic weights by the molar mass contant Mu.

-c, --colnames
: Show the headers in the outputs.

−u, −−usage
: Show usage text and exit.

−v,−−version
: Show version information and exit.

−h, −−help
: Show help text and exit.

NOTES

You may replace the default options from a file if your first options begin with @file. Initial options will then be read from the "response file" "file.rsp" in the current directory.

If "file" does not exist or cannot be read, then an error occurs and the program stops. Each line of the file is prefixed with "options" and interpreted as a separate argument. The file itself may not contain @file arguments. That is, it is not processed recursively.

For more information on response files see https://urbanjost.github.io/M_CLI2/set_args.3m_cli2.html

EXAMPLE

Minimal example

```
ciaaw
```

```
ciaaw H C B O Zr Nb --saw --ice --naw --colnames
ciaaw H C B O Zr Nb -sinc
```

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

version − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function version()result(***fptr***)**

character(len=:), pointer :: *fptr*

C **char* ciaaw_version()**

Python **str:** *pyciaaw.__version__*

DESCRIPTION

Get the library version.

RETURN VALUE

pointer Pointer to the version string.

EXAMPLE

Fortran:

```
use ciaaw
print *, "version = ", version()
```

C:

```
include <stdio.h>
include "ciaaw.h"
printf("version = %s\n", ciaaw_version());
```

Python:

```
import pyciaaw
print(f"version = {pyciaaw.__version__}")
```

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

saw − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function saw(***s***,***ab***,***u***)result(res)**

character(len=*), intent(in) :: *s*

logical, intent(in), optional :: *ab*

logical, intent(in), optional :: *u*

real(dp) :: *res*

C **double ciaaw_saw(char ***s,***int** *n,***bool** *ab,***bool** *u***);**

Python **saw(***s:***str,***abridged:***bool=True,***uncertainty:***bool=False)->float:**

DESCRIPTION

Get the standard atomic weight for the element *s*. The default behavior is to provide the abridged value,*ab is TRUE*, recommended by the CIAAW. The uncertainty can be retrieved by setting *u*=TRUE.

Arguments:

s	Element symbol.
ab	Set to False if the abridged value is not desired. Default to TRUE.
u	Set to True if the uncertainty is desired. Default to FALSE.

RETURN VALUE

NaN	If the provided element is incorrect.
-1	If the element does not have a SAW.

EXAMPLE

Fortran:

```
use ciaaw
saw("H", ab=.false., u=.true.)
```

C:

```
include <stdio.h>
include "ciaaw.h"
ciaaw_saw("H", 1, false, true)
```

Python:

```
import pyciaaw
pyciaaw.saw("H", ab=False, u=True)
```

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

ice − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function ice(***s***,***A***,***u***)result(res)**

character(len=*), intent(in) :: *s*

integer(int32), intent(in) :: *A*

logical, intent(in), optional :: *u*

real(dp) :: *res*

C **double ciaaw_ice(char *** *s*,**int** *n*,**int** *A*,**bool** *u*);

Python **ice(***s*:**str,***A*:**int,***u*:**bool=False)->float:**

DESCRIPTION

Get the isotopic composition of the element *s* for the mass number *A*. The uncertainty can be retrieved by setting *u*=TRUE.

Arguments:

s	Element symbol.
A	Mass number.
u	Set to True if the uncertainty is desired. Default to FALSE.

RETURN VALUE

NaN	If the provided element or the mass number are incorrect.
-1	If the element does not have an ICE.

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

naw − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function naw(***s***,***A***,***u***)result(res)**

character(len=*), intent(in) :: *s*

integer(int32), intent(in) :: *A*

logical, intent(in), optional :: *u*

real(dp) :: *res*

C **double ciaaw_naw(char *** *s*,**int** *n*,**int** *A*,**bool** *u*);

Python **naw(***s:***str,***A:***int,***u:***bool=False)->float:**

DESCRIPTION

Get the atomic weight of the nuclide *s* for the mass number *A*. The uncertainty can be retrieved by setting *u*=TRUE.

Arguments:

s	Element symbol.
A	Mass number.
u	Set to True if the uncertainty is desired. Default to FALSE.

RETURN VALUE

NaN	If the provided element or the mass number are incorrect.
-1	If the element does not have a NAW.

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

nice − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function nice(***s***)result(***res***)**

character(len=*), intent(in) :: *s*

integer(int32) :: *res*

C **int ciaaw_nice(char ***s*,**int** *n***);**

Python **nice(***s:***str)->int**

DESCRIPTION

Get the number of isotopes in ICE of the element *s*.

Arguments:

s	Element symbol.
n	Size of the string *s*.

RETURN VALUE

-1	If the provided element is incorrect.
>0	If the element has ICE.

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)

NAME

nnaw − function

LIBRARY

ciaaw library (*libciaaw*, *−lciaaw*)

SYNOPSIS

Fortran **function nnaw(***s***)result(***res***)**

character(len=*), intent(in) :: *s*

integer(int32) :: *res*

C **int ciaaw_nnaw(char *****s*,**int** *n***);**

Python **nnaw(***s:***str)->int**

DESCRIPTION

Get the number of nuclides in NAW of the element *s*.

Arguments:

s	Element symbol.
n	Size of the string *s*.

RETURN VALUE

-1	If the provided element is incorrect.
>0	If the element has NAW.

SEE ALSO

ciaaw(3), **ciaaw_cli**(1), **ciaaw_version**(3), **ciaaw_saw**(3), **ciaaw_ice**(3), **ciaaw_nice**(3), **ciaaw_naw**(3), **ciaaw_nnaw**(3)