UncValue

Release 0.1.1

Sep 20, 2020

Contents

Simple python class to evaluate the uncertainty for complex or very long calculations given the initial values together with its uncertainty.

CHAPTER 1

Installation

Install the library with pip:

```
$ pip install uncvalue
```

CHAPTER 2

Examples

Imagine you measured a rectangular table whose sides measure `1.5m` and `80cm` with ruler with precision up to `1mm`. Then, you can initialize the `Value` as

- Python:

```python
from uncvalue import Value
L1 = Value(1.5, 1e-3)
L2 = Value(0.8, 1e-3)
```

- Julia:

```julia
using UncValue
L1 = Value(1.5, 1e-3)
L2 = Value(0.8, 1e-3)
```

You now want to calculate the area of the table, so you multiply both lengths

- Python:

```python
A = L1 * L2
print(A)
```

- Julia:

```julia
A = L1 * L2
println(A)
```

and obtain as outcome `(1200.0 ± 1.7)·10^-3`.

Of course, it is possible to perform more complex operations like

- Python (`numpy` required, functions from python `math` will only compute the value)

```python
import numpy as np
print(L1 ** L2)  # power -> (13831.6 ± 9.3)·10^-4
```

(continues on next page)

(continued from previous page)

```
print(np.sin(L1))  # sinus -> (99749.5 ± 7.1)·10^-5
print(np.exp(L2))  # exponential -> (2225.5 ± 2.2)·10^-3
```

- Julia

```
println(L1^L2)  # power -> (13831.6 ± 9.3)·10^-4
println(sin(L1))  # sinus -> (99749.5 ± 7.1)·10^-5
println(exp(L2))  # exponential -> (2225.5 ± 2.2)·10^-3
```

You can also take a look at the sample floder containing real use and more complex examples.

CHAPTER 3

API Reference

class uncvalue.**Value**(*x*, *ux*)

A class to keep track of the uncertainty of measured values.

Parameters

x [int or float or tuple or list or numpy.ndarray] the value or list of values

ux [int or float or tuple or list or numpy.ndarray] the uncertainty of the value, if *x* is a list and *ux* a single number, an array filled with *ux* will be created to match the dimensions of *x*, if both are list then the dimension must match

Raises

ValueError if any uncertainty is negative, if any value passed is complex, if dimensions of *x* and *ux* do not match when both are arrays

Attributes

unc returns the uncertainty of the given instance

val returns the value of the given instance

Methods

arccos()	computes the inverse cosinus of the value
arccosh()	computes the inverse hyperbolic cosinus of the value
arcsin()	computes the inverse sinus of the value
arcsinh()	computes the inverse hyperbolic sinus of the value
arctan()	computes the inverse tangent of the value
arctanh()	computes the inverse hyperbolic tangent of the value
cbrt()	computes the cubic root of the value
copy()	returns a new instance with the same value and uncertainty.

Continued on next page

Table 1 – continued from previous page

cos()	computes the cosinus of the value
cosh()	computes the hyperbolic cosinus of the value
exp()	computes the exponentiation of the value
exp2()	computes the base two power of the value
expm1()	computes the exponentiation minus 1 of the value
*is_*(y)	Extension of *is* keyword.
is_not(y)	Extension of *is not* keyword.
log()	computes the natural logarithm of the value
log10()	computes the decimal logarithm of the value
log1p()	computes the natural logarithm plus 1 of the value
log2()	computes the binary logarithm of the value
max(**kwargs)	returns the maximum value in the range covered by the uncertainty
min(**kwargs)	returns the minimum value in the range covered by the uncertainty
precision()	Gives the position of the least significant digit, counting as powers of 10.
sin()	computes the sinus of the value
sinh()	computes the hyperbolic sinus of the value
sqrt()	computes the square root of the value
tan()	computes the tangent of the value
tanh()	computes the hyperbolic tangent of the value

__abs__()
absolute value of the Value

Returns

Value |*x*|

__add__(*y*)
addition (+) of the Value and a number or another Value

Returns

Value *self* + *y*

__bool__()
returns true if the value is different from 0 and false otherwise

__ceil__()
ceils the number to match the precision dictated by the uncertainty

Returns

float or numpy.ndarray the value ceiled to 2 significant digits of precision

See also:

numpy.ceil

__complex__()
returns the complex representation of the value without uncertainty

__contains__(*y*)
Extension of *in* keyword. Checks if the *y* value is within the uncertainty of this value

__div__(*y*)
division (/) of the Value and a number or another Value

Deprecated since version 3.0: *__div__* has been removed from Python > 2.7 and replaced by *__truediv__*

Returns

Value *self* / *y*

See also:

uncvalue.Value.__truediv__

__eq__(*y*)
Equal than comparison between *self* and *y*

__float__()
returns the float representation of the value without uncertainty

__floor__()
floors the number to match the precision dictated by the uncertainty

Returns

float or numpy.ndarray the value floored to 2 significant digits of precision

See also:

numpy.floor

__floordiv__(*y*)
integer division (//) of the Value and a number or another Value

Returns

Value *self* // *y*

__ge__(*y*)
Greater or equal than comparison between *self* and *y*

__gt__(*y*)
Greater than comparison between *self* and *y*

__iadd__(*y*)
self addition (+=) of the number and a float or another Value

Returns

Value *self* += *y*

__idiv__(*y*)
self division (/=) of the Value and a number or another Value

Deprecated since version 3.0: *__idiv__* has been removed from Python > 2.7 and replaced by *__itruediv__*

Returns

Value *self* /= *y*

See also:

uncvalue.Value.__itruediv__

__ifloordiv__(*y*)
self integer division (//=) of the Value and a number or another Value

Returns

Value *self //= y*

__imul__(*y*)
self product (*=) of the number and a float or another Value

Returns

Value *self *= y*

__init__(*x*, *ux*)
Initialize self. See help(type(self)) for accurate signature.

__int__()
returns the int representation of the value without uncertainty

__invert__()
invert value with respect to the multiplication of the Value

Returns

Value *1 / x*

__ipow__(*y*)
self power (**=) of the Value and a number or another Value

Returns

Value *self **= y*

__isub__(*y*)
self substraction (-=) of the number and a float or another Value

Returns

Value *self -= y*

__itruediv__(*y*)
self true division (/=) of the Value and a number or another Value

Returns

Value *self /= y*

__le__(*y*)
Smaller or equal than comparison between *self* and *y*

__lt__(*y*)
Smaller than comparison between *self* and *y*

__mul__(*y*)
product (*) of the Value and a number or another Value

Returns

Value *self * y*

__ne__(*y*)
Not equal to comparison between *self* and *y*

__neg__()
negation of the Value

Returns

Value *-x*

__pow__ (*y*)
power (**) of the Value and a number or another Value

Returns

Value *self ** y*

__radd__ (*y*)
addition (+) of a number and the Value

Returns

Value *y + self*

__rdiv__ (*y*)
right division (/) of a number and the Value

Deprecated since version 3.0: *__rdiv__* has been removed from Python > 2.7 and replaced by *__rtruediv__*

Returns

Value *y / self*

See also:

uncvalue.Value.__rtruediv__

__repr__ ()
Return repr(self).

__rfloordiv__ (*y*)
right integer division (//) of a number and the Value

Returns

Value *y // self*

__rmul__ (*y*)
product (*) of a number and the Value

Returns

Value *y * self*

__round__ (***kwargs*)
rounds the number to match the precision dictated by the uncertainty

Returns

float or numpy.ndarray the value rounded to 2 significant digits of precision

See also:

numpy.around

__rpow__ (*y*)
power (**) of a number and the Value

Returns

Value *y ** self*

__rsub__ (*y*)
substraction (-) of a number and the Value

Returns

Value *y - self*

__rtruediv__ (*y*)
right true division (/) of a number and the Value

Returns

Value *y / self*

__str__ ()
String represrntation of a number with its uncertainty.

The number is rounded as to match the precision given by the two most significant digits of the uncertainty

__sub__ (*y*)
substraction (-) of the Value and a number or another Value

Returns

Value *self - y*

__truediv__ (*y*)
true division (/) of the Value and a number or another Value

Returns

Value *self / y*

See also:

operator.__truediv__

__trunc__ ()
truncates the number to match the precision dictated by the uncertainty

Returns

float or numpy.ndarray the value truncated to 2 significant digits of precision

See also:

numpy.trunc

__weakref__
list of weak references to the object (if defined)

arccos ()
computes the inverse cosinus of the value

arccosh ()
computes the inverse hyperbolic cosinus of the value

arcsin ()
computes the inverse sinus of the value

arcsinh ()
computes the inverse hyperbolic sinus of the value

arctan ()
computes the inverse tangent of the value

arctanh()
computes the inverse hyperbolic tangent of the value

cbrt()
computes the cubic root of the value

copy()
returns a new instance with the same value and uncertainty.

cos()
computes the cosinus of the value

cosh()
computes the hyperbolic cosinus of the value

exp()
computes the exponentiation of the value

exp2()
computes the base two power of the value

expm1()
computes the exponentiation minus 1 of the value

is_(*y*)
Extension of *is* keyword. Returns true if *self* is equal to *y* in the value

is_not(*y*)
Extension of *is not* keyword. Returns true if *self* is not equal to *y* in the value

log()
computes the natural logarithm of the value

log10()
computes the decimal logarithm of the value

log1p()
computes the natural logarithm plus 1 of the value

log2()
computes the binary logarithm of the value

max(***kwargs*)
returns the maximum value in the range covered by the uncertainty

Returns

float or numpy.ndarray *x + ux*

min(***kwargs*)
returns the minimum value in the range covered by the uncertainty

Returns

float or numpy.ndarray *x - ux*

precision()
Gives the position of the least significant digit, counting as powers of 10.

Returns

int or numpy.ndarray of ints the precision of the number

Examples

Here are some examples of the output given by this function:

Uncertainty	Precision
1,45e-04	-7
0,001456	-3
0,0006666	-4
0,123	-1
1	0
22,22	1
7684,65	3
17,8e8	9

sin()
computes the sinus of the value

sinh()
computes the hyperbolic sinus of the value

sqrt()
computes the square root of the value

tan()
computes the tangent of the value

tanh()
computes the hyperbolic tangent of the value

unc
returns the uncertainty of the given instance

See also:

uncvalue.unc

val
returns the value of the given instance

See also:

uncvalue.val

uncvalue.**val**(*b*)
Returns the values of input parameter without the uncertainty as pure Python numbers.

Parameters

b: int or float or tuple or list or numpy.ndarray or Value the parameter to extract the value from

Returns

value: int or float or numpy.ndarray the value without the uncertainty, if the input parameter is an array it will return an array of floats with the same shape

See also:

uncvalue.unc

Examples

```
>>> from uncvalue import Value, val
>>> val(Value(3.14, 0.25))
3.14
>>> val([Value(3.14, 0.25), Value(2.17, 0.03)])
array([3.14, 2.17])
>>> val(Value([3.14, 2.17], [0.25, 0.03]))
array([3.14, 2.17])
```

uncvalue.**unc**(*b*)

Returns the uncertainty of input parameter without the value as pure Python numbers.

Parameters

b: int or float or tuple or list or numpy.ndarray or Value the parameter to extract the uncertainty from

Returns

value: int or float or numpy.ndarray the uncertainty without the value, if the input parameter is already a pure python number it return 0, if the input parameter is an array it will return an array of floats with the same shape

See also:

uncvalue.val

Examples

```
>>> from uncvalue import Value, unc
>>> unc(Value(3.14, 0.25))
0.25
>>> unc([Value(3.14, 0.25), Value(2.17, 0.03)])
array([0.25, 0.03])
>>> unc(Value([3.14, 2.17], [0.25, 0.03]))
array([0.25, 0.03])
```

uncvalue.**set_unc**(*x*, *ux*)

Set the given uncertainty to the specified value.

Parameters

x [int or float or tuple or list or numpy.ndarray or Value] the parameter to set the uncertainty to

ux [int or float or tuple or list or numpy.ndarray] the uncertainty to assciate to the given value, if *ux* is a list or similar it must have the same shape as *x*

Returns

value [Value or numpy.ndarray] the value with the uncertainty, if the input parameter is an array it will return an array of Values with the same shape

Notes

For a single value this is the same as initialising a new instance of Value, i.e. *set_unc(3.14, 0.25)* and *Value(3.14, 0.25)* return the same result.

Examples

```
>>> from uncvalue import Value, set_unc
>>> set_unc(3.14, 0.25)
(31.4 ± 2.5)·10^-1
>>> set_unc([3.14, 2.17], [0.25, 0.03])
array([(31.4 ± 2.5)·10^-1, (217.0 ± 3.0)·10^-2], dtype=object)
```

CHAPTER 4

Contact

You may open an issue in the GitHub page.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index

Symbols

A

C

E

I

L

M

P

S

T

U

V