

---

# UncValue

*Release 0.1.1*

Sep 20, 2020



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Examples</b>	<b>5</b>
<b>3</b>	<b>API Reference</b>	<b>7</b>
<b>4</b>	<b>Contact</b>	<b>17</b>
<b>5</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



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:

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

- 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:

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

- Julia:

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

and obtain as outcome  $(1200.0 \pm 1.7) \cdot 10^{-3}$ .

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

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

```
import numpy as np
print(L1 ** L2) # power ->  $(13831.6 \pm 9.3) \cdot 10^{-4}$ 
```

(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 folder](#) containing real use and more complex examples.

**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

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

Continued on next page

Table 1 – continued from previous page

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





**u**

uncvalue, [7](#)



## Symbols

[\\_\\_abs\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_add\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_bool\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_ceil\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_complex\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_contains\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_div\\_\\_\(\)](#) (*uncvalue.Value method*), 8  
[\\_\\_eq\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_float\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_floor\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_floordiv\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_ge\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_gt\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_iadd\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_idiv\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_ifloordiv\\_\\_\(\)](#) (*uncvalue.Value method*), 9  
[\\_\\_imul\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_init\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_int\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_invert\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_ipow\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_isub\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_itrueidiv\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_le\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_lt\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_mul\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_ne\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_neg\\_\\_\(\)](#) (*uncvalue.Value method*), 10  
[\\_\\_pow\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_radd\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rdiv\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_repr\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rfloordiv\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rmul\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_round\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rpow\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rsub\\_\\_\(\)](#) (*uncvalue.Value method*), 11  
[\\_\\_rtrueidiv\\_\\_\(\)](#) (*uncvalue.Value method*), 12

[\\_\\_str\\_\\_\(\)](#) (*uncvalue.Value method*), 12  
[\\_\\_sub\\_\\_\(\)](#) (*uncvalue.Value method*), 12  
[\\_\\_trueidiv\\_\\_\(\)](#) (*uncvalue.Value method*), 12  
[\\_\\_trunc\\_\\_\(\)](#) (*uncvalue.Value method*), 12  
[\\_\\_weakref\\_\\_](#) (*uncvalue.Value attribute*), 12

## A

[arccos\(\)](#) (*uncvalue.Value method*), 12  
[arccosh\(\)](#) (*uncvalue.Value method*), 12  
[arcsin\(\)](#) (*uncvalue.Value method*), 12  
[arcsinh\(\)](#) (*uncvalue.Value method*), 12  
[arctan\(\)](#) (*uncvalue.Value method*), 12  
[arctanh\(\)](#) (*uncvalue.Value method*), 12

## C

[cbrt\(\)](#) (*uncvalue.Value method*), 13  
[copy\(\)](#) (*uncvalue.Value method*), 13  
[cos\(\)](#) (*uncvalue.Value method*), 13  
[cosh\(\)](#) (*uncvalue.Value method*), 13

## E

[exp\(\)](#) (*uncvalue.Value method*), 13  
[exp2\(\)](#) (*uncvalue.Value method*), 13  
[expm1\(\)](#) (*uncvalue.Value method*), 13

## I

[is\\_\(\)](#) (*uncvalue.Value method*), 13  
[is\\_not\(\)](#) (*uncvalue.Value method*), 13

## L

[log\(\)](#) (*uncvalue.Value method*), 13  
[log10\(\)](#) (*uncvalue.Value method*), 13  
[log1p\(\)](#) (*uncvalue.Value method*), 13  
[log2\(\)](#) (*uncvalue.Value method*), 13

## M

[max\(\)](#) (*uncvalue.Value method*), 13  
[min\(\)](#) (*uncvalue.Value method*), 13

## P

`precision()` (*uncvalue.Value* method), [13](#)

## S

`set_unc()` (*in module uncvalue*), [15](#)

`sin()` (*uncvalue.Value* method), [14](#)

`sinh()` (*uncvalue.Value* method), [14](#)

`sqrt()` (*uncvalue.Value* method), [14](#)

## T

`tan()` (*uncvalue.Value* method), [14](#)

`tanh()` (*uncvalue.Value* method), [14](#)

## U

`unc` (*uncvalue.Value* attribute), [14](#)

`unc()` (*in module uncvalue*), [15](#)

`uncvalue` (*module*), [7](#)

## V

`val` (*uncvalue.Value* attribute), [14](#)

`val()` (*in module uncvalue*), [14](#)

`Value` (*class in uncvalue*), [7](#)