physics Documentation

Release 1.0.0

pyTeens

May 16, 2019

Contents

CHAPTER 1

physics

1.1 physics package

1.1.1 Submodules

1.1.2 physics.errors module

physics.errors

It contains the Errors class.

It could be used to do arithmetic operations using numbers and their errors on themselves.

class physics.errors.**Errors**(*float number*, ***settings*)

Bases: object

The Errors class is used to define a number with an absolute, relative or percentage error and do arithmetic operations with them.

__abs__

That function is used to return the absolute value of the chosen number.

__add__

That function is used to establish the result of an Addition, summing absolute errors and numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to add.

__eq__

That function is used to compare two numbers using "==".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__float__

That function is used to return the float of the chosen number.

__floordiv__
That function is used to establish the result of a Floor Division, summing relative errors and dividing numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to divide.

__ge__
That function is used to compare two numbers using ">=".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__gt__
That function is used to compare two numbers using ">".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__iadd__
That function is used to establish the result of an Inline Addition, summing absolute errors and numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to add.

__ifloordiv__
That function is used to establish the result of an Inline Floor Division, summing relative errors and dividing numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to divide.

__imod__
That function is used to establish the result of an Inline Modulo, summing relative errors and giving the remainder of the divided numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to get the modulo.

__imul__
That function is used to establish the result of an Inline Multiplication, summing relative errors and multiplying numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to multiply.

__init__
It initializes the object, checks if an absolute, relative or percentage is given and if not it generates an absolute error following the established rules during physics conventions.

Parameters

- **number** (*float or integer*) – The number you've chosen
- ****settings** (*dict*) – A dictionary of errors. It must include an absolute, relative or percentual error at all.

__int__
That function is used to return the integer of the chosen number.

__ipow__
That function is used to establish the result of an Inline Exponentiation, multiplying the first relative error for the second number and giving arithmetic power.

Parameters **second_number** (*integer, float or Errors*) – The number you want to get the power.

__isub__
That function is used to establish the result of an Inline Subtraction, summing absolute Errors and subtracting numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to substrate.

__itruediv__
That function is used to establish the result of an Inline True Division, summing relative errors and dividing numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to divide.

__le__
That function is used to compare two numbers using "<=".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__len__
That function is used to return the number of digits of the chosen number.

__lt__
That function is used to compare two numbers using "<".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__mod__
That function is used to establish the result of a Modulo, summing relative errors and giving the remainder of the divided numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to get the modulo.

__mul__
That function is used to establish the result of a Multiplication, summing relative errors and multiplying numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to multiply.

__ne__
That function is used to compare two numbers using "!=".

Parameters **second_number** (*integer, float or Errors*) – The number you want to compare.

__neg__
That function is used to return the negative value of the chosen number.

__new__()
Create and return a new object. See help(type) for accurate signature.

__pos__
That function is used to return the positive value of the chosen number.

__pow__
That function is used to establish the result of an Exponentiation, multiplying the first relative error for the second number and giving arithmetic power.

Parameters **second_number** (*integer, float or Errors*) – The number you want to get the power.

__pyx_vtable__ = <capsule object NULL>

__radd__
Return value+self.

__reduce__()
Errors.__reduce_cython__(self)

__repr__
Returns the representation of the object.

Returns The Representation

Return type str

__rfloordiv__
Return value//self.

__rmod__
Return value%self.

__rmul__
Return value*self.

__rpow__
Return pow(value, self, mod).

__rsub__
Return value-self.

__rtruediv__
Return value/self.

__setstate__()
Errors.__setstate_cython__(self, __pyx_state)

__str__
That function is used to return a string representation of the chosen number.

__sub__
That function is used to establish the result of a Subtraction, summing absolute Errors and subtracting numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to substrate.

__truediv__
That function is used to establish the result of a True Division, summing relative errors and dividing numbers.

Parameters **second_number** (*integer, float or Errors*) – The number you want to divide.

absolute_error

number

percentage_error

relative_error

1.1.3 physics.gravity module

physics.gravity

It contains the Gravity class.

It could be used to get the gravity force of some objects.

physics.gravity.**calculate_gravity**(*float mass*, *float second_mass=Earth*, *float distance=EarthRadius*) → float

Given two masses and their distance, it calculates the Gravity force between them.

Parameters

- **mass** (*float*) – The first mass.
- **second_mass** (*float*) – The second mass. By default, the earth mass is used.
- **distance** (*float*) – The distance between the two masses. By the default, the radius of the earth is used.

Returns The gravity force.

Return type float

1.1.4 physics.numbers module

physics.numbers

It contains the Numbers class.

It could be used to define numbers using significant digits.

class physics.numbers.**Numbers**(*float number*)

Bases: object

__abs__

That function is used to obtain the absolute value of the number.

Returns The Absolute Value.

Return type int or float

__add__

That function is used to establish the result of an Addition, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to add.

Returns The result

Return type Integer, float or *Numbers*

__eq__

That function is used to compare two numbers using "==".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__float__
That function is used to return the float of the chosen number.

Returns The Float.

Return type float

__floordiv__
That function is used to establish the result of a Floor Division, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to divide.

Returns The result

Return type Integer, float or *Numbers*

__ge__
That function is used to compare two numbers using ">=".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__gt__
That function is used to compare two numbers using ">".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__iadd__
That function is used to establish the result of an Inline Addition, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to add.

Returns The result

Return type Integer, float or *Numbers*

__ifloordiv__
That function is used to establish the result of an Inline Floor Division, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to divide.

Returns The result

Return type Integer, float or *Numbers*

__imul__
That function is used to establish the result of an Inline Multiplication, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to multiply.

Returns The result

Return type Integer, float or *Numbers*

__init__
It initializes the object and get the significant digits following the established rules during physics conventions.

Parameters **number** (*int or float*) – The number you've chosen.

__int__
That function is used to return the integer of the chosen number.

Returns The integer of the number.

Return type int

__invert__
That function is used to return the inverted value of the chosen number.

Returns The Inverted Value.

Return type int or float

__isub__
That function is used to establish the result of an Inline Subtraction, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to subtract.

Returns The result

Return type Integer, float or *Numbers*

__itruediv__
That function is used to establish the result of an Inline True Division, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to divide.

Returns The result

Return type Integer, float or *Numbers*

__le__
That function is used to compare two numbers using "<=".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__len__
That function is used to return the number of digits of the chosen number.

Returns The length.

Return type Integer

__lt__
That function is used to compare two numbers using "<".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__mul__
That function is used to establish the result of a Multiplication, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to multiply.

Returns The result

Return type Integer, float or *Numbers*

__ne__
That function is used to compare two numbers using "!=".

Parameters **other_number** (*integer, float or Numbers*) – The number you want to compare.

__neg__
That function is used to return the negative value of the chosen number.

Returns The Negative Value.

Return type int or float

__new__()
Create and return a new object. See help(type) for accurate signature.

__pos__
That function is used to return the positive value of the chosen number.

Returns The Positive Value.

Return type int or float

__radd__
Return value+self.

__reduce__()
Numbers.__reduce_cython__(self)

__repr__
That function is used to return the representation of the object

Returns The representation

Return type str

__rfloordiv__
Return value//self.

__rmul__
Return value*self.

__round__(*self*, *digits=0*) → float
That function is used to round the chosen number.

Returns The Rounded Value.

Return type float

__rsub__
Return value-self.

__rtruediv__
Return value/self.

__setstate__()
Numbers.__setstate_cython__(self, __pyx_state)

__str__
That function is used to return a string rappresentation of the chosen number.

Returns The String.

Return type str

__sub__
That function is used to establish the result of a Subtraction, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to subtract.

Returns The result

Return type Integer, float or *Numbers*

__truediv__
That function is used to establish the result of a True Division, using significant digits.

Parameters **other_number** (*integer, float or Numbers*) – The number you want to divide.

Returns The result

Return type Integer, float or *Numbers*

after_comma

number

significant_digits

1.1.5 physics.proportionality module

physics.proportionality

It contains the Proportionality class.

It could be used to define a proportionality relation between numbers.

exception physics.proportionality.**LessThanTwoNumbersError**
Bases: Exception

This exception is called when number of parameters are less than 2. 0 is not counted.

__init__
Initialize self. See help(type(self)) for accurate signature.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce_cython__(*self*)

__setstate_cython__(*self*, *__pyx_state*)

exception physics.proportionality.**MissingNeededParameters**
Bases: Exception

This exception is called when constant and proportionality aren't in the parameters and numbers is missing.

__init__
Initialize self. See help(type(self)) for accurate signature.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce_cython__(*self*)

__setstate_cython__(*self*, *__pyx_state*)

exception physics.proportionality.**NoRelationError**
Bases: Exception

This exception is called when there's no relation.

__init__
Initialize self. See help(type(self)) for accurate signature.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce_cython__(*self*)

__setstate_cython__(*self*, *__pyx_state*)

class physics.proportionality.**Proportionality**(***options*)
Bases: object

The proportionality class is used to calculate and use proportionality using numbers. percentage error and do arithmetic

__init__
It initializes the object and it checks options parameter (kwargs), and then get the constant of the proportionality.

Raises

- ***MissingNeededParameters*** – It throws an exception if some parameters are missing.
- ***NoRelationError*** – It throws an exception if there are no relations between numbers.
- ***LessThanTwoNumbersError*** – It throws an exception if there are less numbers than 2.

__new__()
Create and return a new object. See help(type) for accurate signature.

__pyx_vtable__ = <capsule object NULL>

__reduce__()
Proportionality.__reduce_cython__(self)

__repr__
Return the representation of the object.

Returns The representation

Return type str

__setstate__()
Proportionality.__setstate_cython__(self, __pyx_state)

__str__
Return the relation and the constant.

Returns The relation and its constant.

Return type str

calculate(*self*, *float x*) → float
Calculate the y using the formula created during proportionality check.

Parameters **x** (*float*) – The number you want to calculate.

constant

constant_formulas = {'direct': <cyfunction Proportionality.<lambda>>, 'inverse': <cy

formula

relation

1.1.6 Module contents

CHAPTER 2

Changelog

physics *Version:* 2.0.1

- Fix bugs

Main Contributors:

- Gabriel Hearot <gabriel@hearot.it>

CHAPTER 3

Welcome to physics!

physics is a simple **Educational library** written in Python. It could be used for your **school projects**.

Have you ever tried to define a number using errors? Calculating gravity? Get a proportionality relation? Now, that's possible and **simple**.

CHAPTER 4

Contents

4.1 Installation

4.2 From PyPi

Just use **pip**:

```
pip install physics
```

Or if you want to *upgrade* the package:

```
pip install --upgrade physics
```

4.3 From Github

4.3.1 Using Pip

Try using that piece of code:

```
pip install git+https://github.com/pyTeens/physics.git
```

Or if you want to *upgrade* the package

```
pip install --upgrade git+https://github.com/pyTeens/physics.git
```

4.3.2 Downloading files

In primis (from Latin, "firstable"), **clone** the repository:

```
git clone https://github.com/pyTeens/physics.git
```

Then, change directory:

```
cd physics
```

And finally, install the **package**:

```
sudo python3 setup.py install
```

4.4 Files

You'll find lots of *not understandable* **directory** and **files**, so here a list and definitions of them:

- **physics** - *Main directory*
  - **physics/__init__.pyx** - *Init file, it included all classes*
  - **physics/errors.pyx** - *Errors class*
  - **physics/gravity.pyx** - *Gravity class*
  - **physics/numbers.pyx** - *Numbers class*
  - **physics/proportionality.pyx** - *Proportionality class*

4.5 How to contribute

In primis ("firstable"), you **must** read the code of conducts and the contributing document, then ask hearot to enter the **organization** (pyTeens).

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

C

E

F

L

M

N

P

R

S