validators Documentation

Release 0.11.2

Konsta Vesterinen

January 08, 2017

Contents

Python Data Validation for Humans™.

CHAPTER 1

Why?

Python has all kinds of validation tools, but every one of them requires defining a schema. I wanted to create a simple validation library where validating a simple value does not require defining a form or a schema. Apparently some other guys have felt the same way.

Often I've had for example a case where I just wanted to check if given string is an email. With `validators` this use case becomes as easy as:

```
>>> import validators

>>> validators.email('someone@example.com')
True
```

CHAPTER 2

Installation

You can install `validators` using pip:

```
pip install validators
```

Currently `validators` supports python versions 2.7, 3.3, 3.4, 3.5 and PyPy.

CHAPTER 3

Basic validators

Each validator in `validators` is a simple function that takes the value to validate and possibly some additional key-value arguments. Each function returns `True` when validation succeeds and *`ValidationFailure`* object when validation fails.

`ValidationFailure` class implements `__bool__` method so you can easily check if validation failed:

```
>>> if not validators.email('some_bogus_email@@@'):
...     # Do something here
...     pass
```

`ValidationFailure` object also holds all the arguments passed to original function:

```
>>> result = validators.between(3, min=5)
>>> result.value
3
>>> result.min
5
```

3.1 between

`validators.between.`**`between`**(*value*, *min=None*, *max=None*)

Validate that a number is between minimum and/or maximum value.

This will work with any comparable type, such as floats, decimals and dates not just integers.

This validator is originally based on WTForms NumberRange validator.

Examples:

```
>>> from datetime import datetime

>>> between(5, min=2)
True

>>> between(13.2, min=13, max=14)
True

>>> between(500, max=400)
ValidationFailure(func=between, args=...)

>>> between(
...     datetime(2000, 11, 11),
```

```
...     min=datetime(1999, 11, 11)
... )
True
```

Parameters

- **min** – The minimum required value of the number. If not provided, minimum value will not be checked.
- **max** – The maximum value of the number. If not provided, maximum value will not be checked.

New in version 0.2.

3.2 domain

validators.domain.**domain**(*value*)

Return whether or not given value is a valid domain.

If the value is valid domain name this function returns `True`, otherwise *ValidationFailure*.

Examples:

```
>>> domain('example.com')
True

>>> domain('example.com/')
ValidationFailure(func=domain, ...)
```

Supports IDN domains as well:

```
>>> domain('xn----gtbspbbmkef.xn--p1ai')
True
```

New in version 0.9.

Changed in version 0.10: Added support for internationalized domain name (IDN) validation.

Parameters **value** – domain string to validate

3.3 email

validators.email.**email**(*value*, *whitelist=None*)

Validate an email address.

This validator is based on Django's email validator. Returns `True` on success and *ValidationFailure* when validation fails.

Examples:

```
>>> email('someone@example.com')
True

>>> email('bogus@@')
ValidationFailure(func=email, ...)
```

New in version 0.1.

Parameters

- **value** – value to validate
- **whitelist** – domain names to whitelist

Copyright

3. Django Software Foundation and individual contributors.

License BSD

3.4 iban

validators.iban.**iban**(*value*)

Return whether or not given value is a valid IBAN code.

If the value is a valid IBAN this function returns `True`, otherwise *ValidationFailure*.

Examples:

```
>>> iban('DE29100500001061045672')
True

>>> iban('123456')
ValidationFailure(func=iban, ...)
```

New in version 0.8.

Parameters **value** – IBAN string to validate

3.5 ipv4

validators.ip_address.**ipv4**(*value*)

Return whether or not given value is a valid IP version 4 address.

This validator is based on WTForms IPAddress validator

Examples:

```
>>> ipv4('123.0.0.7')
True

>>> ipv4('900.80.70.11')
ValidationFailure(func=ipv4, args={'value': '900.80.70.11'})
```

New in version 0.2.

Parameters **value** – IP address string to validate

3.6 ipv6

validators.ip_address.**ipv6**(*value*)

Return whether or not given value is a valid IP version 6 address.

This validator is based on WTForms IPAddress validator.

Examples:

```
>>> ipv6('abcd:ef::42:1')
True

>>> ipv6('abc.0.0.1')
ValidationFailure(func=ipv6, args={'value': 'abc.0.0.1'})
```

New in version 0.2.

Parameters **value** – IP address string to validate

3.7 length

validators.length.**length**(*value*, *min=None*, *max=None*)

Return whether or not the length of given string is within a specified range.

Examples:

```
>>> length('something', min=2)
True

>>> length('something', min=9, max=9)
True

>>> length('something', max=5)
ValidationFailure(func=length, ...)
```

Parameters

- **value** – The string to validate.
- **min** – The minimum required length of the string. If not provided, minimum length will not be checked.
- **max** – The maximum length of the string. If not provided, maximum length will not be checked.

New in version 0.2.

3.8 mac_address

validators.mac_address.**mac_address**(*value*)

Return whether or not given value is a valid MAC address.

If the value is valid MAC address this function returns `True`, otherwise *`ValidationFailure`*.

This validator is based on WTForms MacAddress validator.

Examples:

```
>>> mac_address('01:23:45:67:ab:CD')
True

>>> mac_address('00:00:00:00:00')
ValidationFailure(func=mac_address, args={'value': '00:00:00:00:00'})
```

New in version 0.2.

Parameters **value** – Mac address string to validate

3.9 slug

validators.slug.**slug**(*value*)

Validate whether or not given value is valid slug.

Valid slug can contain only alphanumeric characters, hyphens and underscores.

Examples:

```
>>> slug('my.slug')
ValidationFailure(func=slug, args={'value': 'my.slug'})

>>> slug('my-slug-2134')
True
```

New in version 0.6.

Parameters **value** – value to validate

3.10 truthy

validators.truthy.**truthy**(*value*)

Validate that given value is not a falsey value.

This validator is based on WTForms DataRequired validator.

Examples:

```
>>> truthy(1)
True

>>> truthy('someone')
True

>>> truthy(0)
ValidationFailure(func=truthy, args={'value': 0})

>>> truthy('    ')
ValidationFailure(func=truthy, args={'value': '    '})

>>> truthy(False)
ValidationFailure(func=truthy, args={'value': False})

>>> truthy(None)
ValidationFailure(func=truthy, args={'value': None})
```

New in version 0.2.

3.11 url

validators.url.**url**(*value*, *public=False*)

Return whether or not given value is a valid URL.

If the value is valid URL this function returns `True`, otherwise *`ValidationFailure`*.

This validator is based on the wonderful URL validator of dperini.

Examples:

```
>>> url('http://foobar.dk')
True

>>> url('http://10.0.0.1')
True

>>> url('http://foobar.d')
ValidationFailure(func=url, ...)

>>> url('http://10.0.0.1', public=True)
ValidationFailure(func=url, ...)
```

New in version 0.2.

Changed in version 0.10.2: Added support for various exotic URLs and fixed various false positives.

Changed in version 0.10.3: Added `public` parameter.

Changed in version 0.10.4: Made the regular expression this function uses case insensitive.

Parameters

- **value** – URL address string to validate
- **public** – (default=False) Set True to only allow a public IP address

3.12 uuid

validators.uuid.**uuid**(*value*)

Return whether or not given value is a valid UUID.

If the value is valid UUID this function returns `True`, otherwise *`ValidationFailure`*.

This validator is based on WTForms UUID validator.

Examples:

```
>>> uuid('2bc1c94f-0deb-43e9-92a1-4775189ec9f8')
True

>>> uuid('2bc1c94f 0deb-43e9-92a1-4775189ec9f8')
ValidationFailure(func=uuid, ...)
```

New in version 0.2.

Parameters **value** – UUID string to validate

CHAPTER 4

i18n validators

4.1 Finnish

4.1.1 fi_business_id

validators.i18n.fi.**fi_business_id**(*business_id*)

Validate a Finnish Business ID.

Each company in Finland has a distinct business id. For more information see Finnish Trade Register

Examples:

```
>>> fi_business_id('0112038-9')  # Fast Monkeys Ltd
True

>>> fi_business_id('1234567-8')  # Bogus ID
ValidationFailure(func=fi_business_id, ...)
```

New in version 0.4.

Changed in version 0.5: Method renamed from `finnish_business_id` to `fi_business_id`

Parameters **business_id** – business_id to validate

4.1.2 fi_ssn

validators.i18n.fi.**fi_ssn**(*ssn*)

Validate a Finnish Social Security Number.

This validator is based on django-localflavor-fi.

Examples:

```
>>> fi_ssn('010101-0101')
True

>>> fi_ssn('101010-0102')
ValidationFailure(func=fi_ssn, args={'ssn': '101010-0102'})
```

New in version 0.5.

Parameters **ssn** – Social Security Number to validate

CHAPTER 5

Internals

5.1 validator

class validators.utils.**ValidationFailure**(*func*, *args*)

validators.utils.**validator**(*func*, **args*, ***kwargs*)

A decorator that makes given function validator.

Whenever the given function is called and returns `False` value this decorator returns *ValidationFailure* object.

Example:

```
>>> @validator
... def even(value):
...     return not (value % 2)

>>> even(4)
True

>>> even(5)
ValidationFailure(func=even, args={'value': 5})
```

Parameters

- **func** – function to decorate
- **args** – positional function arguments
- **kwargs** – key value function arguments

V