fuzzyfields Documentation

Release 2.0.0+dev.2b3e082

fuzzyfields Developers

2019-10-03

CONTENTS

TODO

CHAPTER

ONE

INSTALLATION

To install the package:

```
pip install fuzzyfields
```

or

```
conda install -c conda-forge fuzzyfields
```

1.1 Required dependencies

- Python 3.6 or later

1.2 Additional dependencies

- pandas (needed by *Timestamp*)

CHAPTER

TWO

TESTING

To run the test suite after installing fuzzyfields, install py.test (via pypi or conda) and run `py.test`.

CHAPTER

THREE

WHAT'S NEW

3.1 v2.0.0 (Unreleased)

Renamed and released to the Open Source community. Overhauled design; use as class properties.

3.2 v1.0.0 (2014-12-01)

Internal Legal & General release, called `landg.validators` and `landg.dictreadervalidator`

CHAPTER

FOUR

API REFERENCE

4.1 Base class

class fuzzyfields.**FuzzyField**(*, *required: bool = True*, *default: Any = None*, *description: str = None*, *unique: bool = False*)

Abstract base class.

Parameters

- **required** (*bool*) – If False, return default if value is "" or "N/A". If True, ensure that this field has a value,
- **default** – Default value to return in case required is False and value is None, NaN, NaT, empty string, "N/A", or similar (basically anything for which `pandas.isnull()` returns True, or that `pandas.read_csv()` interprets as a NaN)
- **description** (*str*) – Optional description for the specific field or property being validated. It should not contain the field name or settings.
- **unique** (*bool*) – Set to True to raise an error in case of duplicate values. When FuzzyField instances are used as class attributes, the uniqueness check is performed across all instances of the owner class and its subclasses.

__delete__(*instance*) → None

Delete the field value on an instance of the owner class.

__get__(*instance*, *owner*) → Any

Retrieve stored value of the property.

Returns Stored value, or self.default is the stored value is None. When invoked as a class property, return the FuzzyField object itself.

One may wish to postprocess the return value before it is returned. This can be particularly useful when one wants to alter the output of a field depending on the output of other attributes of the instance that may not be available when *`FuzzyField.validate()`* is executed. This can be achieved by overriding this method as follows:

```
>>> from fuzzyfields import String

>>> class Dog(String):
...     def __get__(self, instance, owner):
...         value = super().__get__(instance, owner)
...         if value is self:
...             return self
...
...         return f"{value}, {instance.name}'s dog!'"
```

(continues on next page)

(continued from previous page)

```
>>> class Owner:
...     name = String()
...     dog = Dog()

>>> human = Owner()
>>> human.dog = 'Lassie'
>>> human.name = 'Bob'
>>> human.dog
"Lassie, Bob's dog!"
```

__init__(*, *required: bool = True*, *default: Any = None*, *description: str = None*, *unique: bool = False*)
Initialize self. See help(type(self)) for accurate signature.

__repr__() → str
Fancy print the description of the fuzzyfield and all the relevant settings. Used when building the docstring of the owner class.

Internally invokes *FuzzyField.sphinxdoc()*.

__set__(*instance*, *value*) → None
Store value of the property for parent object. Can be used in two ways:

- with a regular value to be validated
- with a new instance of another FuzzyField. This way one can override settings with instance-specific ones.

__set_name__(*owner*, *name: str*) → None
Called at the time the owner class is created. The descriptor has been assigned to name.

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

copy()
Shallow copy of self. The seen_values set is recreated as an empty set.

name = None
Name of the field being validated. This is set automatically:

- when FuzzyField instances are used as class attributes, by *FuzzyField.__set_name__()*, or by *FuzzyField.__set__()* for instance-specific fuzzyfields
- when FuzzyField instances are used within the *DictReader* framework, by *DictReader.__init__()*

owner = None
The class to which the FuzzyField is attached to as a descriptor. None when used within the *DictReader* framework.

parse(*value: Any*) → Any
On-the fly parsing and validation for a local variable.

This is a wrapper around *preprocess()* -> *validate()* -> *postprocess()*.

Parameters **value** – Raw value to be preprocessed and validated

Returns Fully preprocessed value, or self.default if the value is null-like and required=False

postprocess(*value: Any*) → Any
Post-process the value after validating it and before storing it. This method is invoked after *FuzzyField.validate()* and tests the `required` and `unique` flags.

Raises

- ***MissingFieldError*** – if self.required is True and value is None
- ***DuplicateError*** – if self.unique is True and value is not None and already found

static preprocess(*value: Any*) → Any
Perform initial cleanup of a raw input value. This method is automatically invoked before *FuzzyField.validate()*.

Parameters **value** – raw input value

Returns the argument, stripped of leading and trailing whitespace and carriage returns if it is a string. If the argument is null, return None. Otherwise return the argument unaltered.

seen_values = None
Record of already encountered values. This attribute only exists if unique=True.

property sphinxdoc
Virtual property - to be overridden. Automated documentation that will appear in Sphinx. It should not include the name, owner, required, default, unique, or description attributes.

validate(*value: Any*) → Any
Virtual method - to be overridden. Validate and reformat value. This method is invoked when processing a new value, after *FuzzyField.preprocess()* and before *FuzzyField.postprocess()*, but only if the value is not None after preprocess.

Parameters **value** – Input data, already preprocessed by *FuzzyField.preprocess()*. Object type could be anything and should be either tested or carefully handled through duck-typing.

Returns

Reformatted value, or None if default is to be used.

Note: Do not return self.default. This is left to *postprocess()*. Instead, for any value that equates to null/blank, always return None.

Raises ***MalformedFieldError***, ***FieldTypeError*** – if the value is not valid

4.2 Fields

class fuzzyfields.**String**(**, required: bool = True, default: Any = None, description: str = None, unique: bool = False*)
Any string value

class fuzzyfields.**RegEx**(*pattern: str, **kwargs*)
Validate an input string against a regular expression

Parameters

- **pattern** (*str*) – regular expression pattern string
- **kwargs** – parameters to be passed to *FuzzyField*

class fuzzyfields.**ISOCodeAlpha**(*chars: int = 3*, ***kwargs*)
Letters-only ISO code, e.g. for country or currency. Case insensitive (it will be converted to uppercase).

Parameters

- **chars** (*int*) – Number of characters of the code (default: 3)
- **kwargs** – parameters to be passed to *FuzzyField*

class fuzzyfields.**Boolean**(**, required: bool = True, default: Any = None, description: str = None, unique: bool = False*)
A boolean, any string representation of false/true or no/yes, or 0/1.

class fuzzyfields.**Domain**(*choices: Iterable, *, case_sensitive: bool = True, passthrough: bool = False, **kwargs*)
A field which can only accept a specific set of values

Parameters

- **choices** – collection of acceptable values. The default needs not be included.
- **case_sensitive** (*bool*) – ignore case when validating string input. The output will be converted to the case listed in choices.
- **passthrough** (*bool*) – If True, store the choices object by reference and assume it will change after this class has been initialised. The change will be reflected in the next parsed value.

  Example:

  ```
  v1 = String("ID", unique=True)
  v2 = Domain("CrossRef", domain=v1.seen_values, passthrough=True)
  ```

  In the above example, the field 'CrossRef' must be one of the values that already appeared for the field 'ID'.

  passthrough comes with a performance cost; set it to False (the default) to allow for optimisations. This assumes that neither the choices collection nor the objects it contains will change in the future.
- **kwargs** – extra parameters for *FuzzyField*

class fuzzyfields.**Float**(**, min_value: Union[int, float] = -inf, max_value: Union[int, float] = inf, allow_min: bool = True, allow_max: bool = True, allow_zero: bool = True, default: Any = nan, **kwargs*)
Convert a string representing a number, an int, or other numeric types (e.g. *numpy.float64*) to float.

Parameters

- **default** – Default value. Unlike in all other FuzzyFields, if omitted it is NaN instead of None.
- **min_value** – Minimum allowable value. Omit for no minimum.
- **max_value** – Maximum allowable value. Omit for no maximum.
- **allow_min** (*bool*) – If True, test that value >= min_value, otherwise value > min_value
- **allow_max** (*bool*) – If True, test that value <= max_value, otherwise value < max_value
- **allow_zero** (*bool*) – If False, test that value != 0
- **kwargs** (*dict*) – parameters to be passed to *FuzzyField*

class fuzzyfields.**Decimal**(*, *default: Any = Decimal('NaN')*, ***kwargs*)

Convert a number or a string representation of a number to `Decimal`, which is much much slower and heavier than float but avoids converting 3.1 to 3.0999999.

class fuzzyfields.**Integer**(*, *min_value: Union[int, float] = -inf*, *max_value: Union[int, float] = inf*, *allow_min: bool = True*, *allow_max: bool = True*, *allow_zero: bool = True*, *default: Any = nan*, ***kwargs*)

Whole number.

Valid values are:

- anything that is parsed by the *int* constructor.
- floats with strictly trailing zeros (e.g. 1.0000)
- scientific format as long as there are no digits below 10^0 (1.23e2)

Note: inf and -inf are valid inputs, but in these cases the output will be of type float. To disable them you can use

- `min_value=-math.inf, allow_min=False`
- `max_value=math.inf, allow_max=False`

NaN is treated as an empty cell, so it is accepted if required=False; in that case the validation will return whatever is set for default, which is math.nan unless overridden, which makes it a third case where the output value won't be int but float.

Raises ***MalformedFieldError*** – if the number can't be cast to int without losing precision

class fuzzyfields.**Percentage**(*, *min_value: Union[int, float] = -inf*, *max_value: Union[int, float] = inf*, *allow_min: bool = True*, *allow_max: bool = True*, *allow_zero: bool = True*, *default: Any = nan*, ***kwargs*)

Percentage, e.g. 5% or .05

Warning: There's nothing stopping somebody from writing "35" where it should have been either "35%" or "0.35". If this field receives "35", it will return 3500.0. You should use the min_value and max_value parameters of *`Float`* to prevent this kind of incidents. Still, nothing will ever protect you from a "1", which will be converted to 1.00 but the author of the input may have wanted to say 0.01.

class fuzzyfields.**Timestamp**(*, *output: str = 'pandas'*, *required: bool = True*, *default=None*, *description: str = None*, *unique: bool = False*, ***kwargs*)

Parse and check various date and time formats

Note: This field requires pandas.

Parameters

- **output** (*`str`*) – Format of the output value. Possible values are:

  'pandas' (default) return type is `pandas.Timestamp`

> **Warning:** This format is limited to the period between 1677-09-22 and 2262-04-11, see pandas documentation. Timestamps outside of this range will be automatically coerced to its edges.

'datetime' return type is `datetime.datetime`

'numpy' return type is *numpy.datetime64*

any other string anything else will be interpreted as a format string for `pandas.Timestamp.strftime()`; e.g. `%Y/%m/%d` will produce a string YYYY/MM/DD.

- **required** (*bool*) – See *FuzzyField*
- **default** – See *FuzzyField*
- **description** (*str*) – See *FuzzyField*
- **unique** (*bool*) – See *FuzzyField*
- **kwargs** – Parameters to be passed to `pandas.to_datetime()`.

> **Note:** The default is to set dayfirst=True, meaning that in case of ambiguity this function will choose the European format DD/MM/YYYY, whereas the default for `pandas.to_datetime()` is dayfirst=False (American format MM/DD/YYYY).

4.3 DictReader

class fuzzyfields.**DictReader**(*iterable: Iterable*, *fields: Dict[str, fuzzyfields.fuzzyfield.FuzzyField] = None*, ***, *errors: Union[str, Callable[Exception, Any]] = None*, *name_map: Dict[str, str] = None*)

Generic iterable that acquires an iterable of dicts in input, e.g. `csv.DictReader`, and for every input line it yields a line that is filtered, validated and processed depending on the input parameters.

Parameters

- **iterable** – an iterable object, e.g. `csv.DictReader`, that yields dicts of `{field : value}`.
- **fields** – dict of instance-specific *FuzzyField* objects. You should *not* use this parameter to set any fields that are known at the time of writing the code, which is the most common use case. Instead, you should create a subclass of DictReader and override the DictReader.fields class attribute.
- **errors** – One of:

  'raise' (default) raise a *ValidationError* on the first line

  'critical', 'error', 'warning', 'info', 'debug' log the error with the matching functions in `logging` and continue

  callable(*ValidationError*) invoke a custom callable and continue (unless it itself raises an Exception)

  In case errors != 'raise' and a FuzzyField raises an exception,

  – if the field is required, the entire line is discarded

  – otherwise, the field is replaced with its default value

Alternatively to passing this parameter, you may create a subclass of DictReader and override the DictReader.errors class attribute.

- **name_map** (*dict*) – optional dict of `{from name: to name}` renames, where each pair performs a key replacement.

  Alternatively to passing this parameter, you may create a subclass of DictReader and override the DictReader.name_map class attribute.

__init__(*iterable: Iterable*, *fields: Dict[str, fuzzyfields.fuzzyfield.FuzzyField] = None*, ***, *errors: Union[str, Callable[Exception, Any]] = None*, *name_map: Dict[str, str] = None*)
Build new object

classmethod __init_subclass__()
Executed after all subclasses of the current class are defined. Set FuzzyField.name and enrich the docstring of the subclass with the documentation of the fields.

__iter__()
Draw dicts from the underlying iterable and yield dicts of `{field name : parsed value}`.

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

errors = 'raise'
Class level error handling system. Can be overridden with an instance-specific value through the matching `__init__` parameter.

fields = {}
Class-level map of `{field name: FuzzyField}`. Overriding this dict is the preferential way to add fields, as they will dynamically build Sphinx documentation. You may add instance-specific fields with the matching `__init__` parameter. Override with a `OrderedDict` if you need the fields to be parsed in order (this is generally only necessary when one field defines the domain of another).

property line_num
Return line number of underlying file.

Raises **AttributeError** – if the underlying iterator is not a `csv.reader()`, `csv.DictReader`, or another duck-type compatible class

name_map = {}
Class-level map of field renames. The keys in this dict must be a subset of the keys in the fields dict. You can add to this dict in an instance-specific way by setting the matching `__init__` parameter.

postprocess_row(*row: Dict[str, Any]*) → Dict[str, Any]
Give child classes an opportunity to post-process every row after it's been parsed by the FuzzyFields. This allows handling special cases and performing cross-field validation.

Parameters **row** – The row as composed by the fields, after name mapping

Returns Modified row, or None if the row should be skipped

preprocess_row(*row: Any*) → Dict[str, Any]
Give child classes an opportunity to pre-process every row before feeding it to the FuzzyFields. This allows handling special cases.

You must use this method to manipulate the row if the underlying iterator does not natively yields dicts, e.g. a `csv.reader()` object.

Parameters **row** – The row as read by self.iterable, with all names and before name mapping

Returns modified row, or None if the row should be skipped

record_num = None
Current record (counting from 0), or -1 if the iteration hasn't started yet.

4.4 Exceptions

class fuzzyfields.**ValidationError**(*name: Optional[str] = None*)
Common ancestor of all landg.validators exceptions

Parameters **name** (*str*) – Field name, or None if the FuzzyField is used neither as a class property nor within a *DictReader*

class fuzzyfields.**MalformedFieldError**(*name: Optional[str], value: Any = None, expect: Any = None*)
Parsed malformed field

class fuzzyfields.**FieldTypeError**(*name: Optional[str], value: Any = None, expect: Any = None*)
Parsed field of invalid type

class fuzzyfields.**DuplicateError**(*name: Optional[str], value: Any = None*)
The same value appeared twice for the same field and the unique parameter is set to True.

class fuzzyfields.**DomainError**(*name: Optional[str], value: Any = None, choices: Any = None*)
Value is not among the permissible ones

class fuzzyfields.**MissingFieldError**(*name: Optional[str] = None*)
Field is null and required is True, or a dict key (typically a column header) is missing from the value returned by the input *DictReader*

CHAPTER

FIVE

LICENSE

fuzzyfields is available under the open source Apache License.

INDEX

Symbols

B

C

D

E

F

I

L

M

N

O

P

R

S

T

V