django-localflavor Documentation

Release 4.0

Django Software Foundation and individual contributors

Apr 22, 2023

Contents

django-localflavor is a collection of assorted pieces of code that are useful for particular countries or cultures. These are called the "local flavor" add-ons and live in the `localflavor` package.

Inside that package, country- or culture-specific code is organized into subpackages, named using ISO 3166 country codes.

Most of the `localflavor` add-ons are localized form components deriving from the forms framework – for example, a `USStateField` that knows how to validate U.S. state abbreviations, and a `FISocialSecurityNumber` that knows how to validate Finnish social security numbers.

- *Romania (ro)*
- *Russia (ru)*
- *Sweden (se)*
- *Slovenia (si)*
- *Slovakia (sk)*
- *Tunisia (tn)*
- *Turkey (tr)*
- *United States of America (us)*
- *Uruguay (uy)*
- *South Africa (za)*

To use one of these localized components, just import the relevant subpackage. For example, here's how you can create a form with a field representing a Greek postal code:

```
from django import forms
from localflavor.gr.forms import GRPostalCodeField

class MyForm(forms.Form):
    my_greek_postal_code = GRPostalCodeField()
```

The `localflavor` package also includes a *generic* subpackage, containing useful code that is not specific to one particular country or culture. This package defines date, datetime and split datetime input fields based on those from the forms, but with non-US default formats. Here's an example of how to use them:

```
from django import forms
from localflavor import generic

class MyForm(forms.Form):
    my_date_field = generic.forms.DateField()
```

The `localflavor` generic package also has IBAN and BIC model and form fields. Here's an example of how to use the IBAN and BIC form fields:

```
from django import forms
from localflavor.generic.forms import BICFormField, IBANFormField

class MyForm(forms.Form):
    iban = IBANFormField()
    bic = BICFormField()
```

CHAPTER 1

Installation

To install django-localflavor use your favorite packaging tool, e.g.pip:

```
pip install django-localflavor
```

Or download the source distribution from PyPI at https://pypi.python.org/pypi/django-localflavor, decompress the file and run `python setup.py install` in the unpacked directory.

Then add `'localflavor'` to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
    # ...
    'localflavor',
)
```

Note: Adding `'localflavor'` to your `INSTALLED_APPS` setting is required for migrations and translations to work. Using django-localflavor without adding it to your `INSTALLED_APPS` setting is not recommended.

CHAPTER 2

Internationalization

Local flavor has its own catalog of translations, in the directory `localflavor/locale`, and it's not loaded automatically like Django's general catalog in `django/conf/locale`. If you want local flavor's texts to be translated, like form fields error messages, you must include *`localflavor`* in the `INSTALLED_APPS` setting, so the internationalization system can find the catalog, as explained in How Django discovers translations.

CHAPTER 3

Adding flavors

We'd love to add more of these, so please create an issue or pull request with any code you'd like to contribute. See any of the existing flavors for examples.

See the contributing documentation for how to run the tests while working on a local flavor.

If you consider adding a new localflavor for country here are some examples that you might consider implementing:

- form fields and form widgets
  - ID verification
  - tax or social security number validator
  - car registration
  - postal code validation
  - country area selects, e.g. cities, counties, states, provinces
- model fields, e.g. for storing any of the above form fields' values
- local translations of English area names. Join your language team at Transifex: https://www.transifex.com/projects/p/django-localflavor/

Note: django-localflavor does not accept contributions of country specific phone number fields. The django-phonenumber-field package has excellent support for validating phone numbers in many countries and we recommend this package.

CHAPTER 4

Releases

django-localflavor releases follow semver with the major version number matching the major version number of Django (from Django 2.0 and above). A compatible version of django-localflavor will be released within one month of each Django release. django-localflavor may have additional releases if there are enough changes in between Django versions to justify a new version of django-localflavor. This means that the minor version number for django-localflavor may not match the minor version of Django itself. See the documentation about Django's release process for more information.

4.1 Deprecation Policy

When non-internal parts of the project are deprecated a *DeprecationWarning* or a *PendingDeprecationWarning* will be thrown upon use until the next major version is released. The warning will explain how to safely update your code, and which version the functionality will be removed in. Deprecated code will be removed in releases with a new major version number (e.g. x.0 releases).

CHAPTER 5

Backwards compatibility

We will always attempt to make *localflavor* reflect the officially gazetted policies of the appropriate local government authority. For example, if a government body makes a change to add, alter, or remove a province (or state, or county), that change will be reflected in localflavor in the next release.

When a backwards-incompatible change is made (for example, the removal or renaming of a province) the localflavor in question will raise a warning when that localflavor is imported. This provides a run-time indication that something may require attention.

However, once you have addressed the backwards compatibility (for example, auditing your code to see if any data migration is required), the warning serves no purpose. The warning can then be suppressed. For example, to suppress the warnings raised by the Indonesian localflavor you would use the following code:

```
import warnings
warnings.filterwarnings('ignore',
                        category=RuntimeWarning,
                        module='localflavor.id')
from localflavor.id import forms as id_forms
```

CHAPTER 6

Indices and tables

6.1 Authors

- Aaron Boman
- Adam Taylor
- Adnane Belmadiaf
- Adonys Alea Boffill
- Adrian Holovaty
- Adrián Matejov
- Agustín Scaramuzza
- Ahmad Zolfaghari
- Ahmed Shahwan
- Alex Butum
- Alexey Kotlyarov
- Alex Gaynor
- Alex Hill
- Alex Zhang
- Alix Martineau
- Alonisser
- Andreas Pelme
- André Niero Setti
- André Ramos
- Andres Torres Marroquin

- Andrew Godwin
- Anton Zhyltsou
- Arthur de Jong
- Aymeric Augustin
- babastienne
- baffolobill
- Baptiste Darthenay
- Ben Davis
- Ben Konrath
- Bruno M. Custódio
- Burhan Khalid
- Claude Paroz
- Daniel Ampuero
- Daniela Ponader
- Danielle Madeley
- David Smith
- Daniel Roschka
- Didier 'OdyX' Raboud
- Diederik van der Boor
- d.merc
- Dmitry Dygalo
- Dominick Rivard
- Douglas Miranda
- Elliott Fawcett
- Erik Romijn
- Flavio Curella
- Florian Apolloner
- François Constant
- Gary Wilson Jr
- Gerardo Orozco
- Ghassen Telmoudi
- Gil Obradors
- Grzes Furga
- Hamad AlGhanim
- Héctor Asencio
- Henrik Ossipoff Hansen

- Heri Priyatno
- Honza Král
- Horst Gutmann
- Igor Támara
- Illia Volochii
- Ivan Fisun
- Jaap Roes
- Jacob Kaplan-Moss
- James Bennett
- Jannis Leidel
- Jan Pieter Waagmeester
- Jarmo van Lenthe
- Jérémie Ferry
- Jocelyn Delalande
- Johannes Hoppe
- Johnny Lee Othon
- Jonas Ghyllebert
- Joseph Kocherhans
- Josh Crompton
- Julien Phalip
- Justin Bronn
- Karen Tracey
- Kevin Ramirez Zavalza
- László Ratskó
- Lefteris Nikoltsios
- Luis Alberto Santana
- Łukasz Langa
- Luke Benstead
- luyikei
- Malcolm Tredinnick
- Marco Beri
- Martin Ogden
- Marti Raudsepp
- Matias Dinota
- Michał Sałaban
- Mike Lissner

- Morgane Alonso
- Naglis Jonaitis
- Nishit Shah
- Olivier Sels
- Olle Vidner
- Paul Cunnane
- Paul Donohue
- Paulo Poiati
- Peter J. Farrell
- Rael Max
- Ramiro Morales
- Raphael Michel
- Rolf Erik Lekang
- Russell Keith-Magee
- Sandeep N
- Serafeim Papastefanos
- Sergio Oliveira
- Simonas Kazlauskas
- Simon Charette
- Stefan Kjartansson
- Syafiq Termizi
- tadeo
- Thiago Avelino
- Thor K. Høgås
- Tino de Bruijn
- Tom Forbes
- Trey Hunner
- Tyler Ball
- Vaclav Rehak
- Venelin Stoykov
- Vishal Pandey
- Vladimir Nani
- Abhineet Tamrakar

6.2 Changelog

6.2.1 4.0 (2023-04-22)

New flavors:

- Nepal LocalFlavor: Support for Nepal added (gh-451).
- Belarus localflavor (gh-422, gh-442).
- Ghana localflavor (gh-460).

New fields for existing flavors:

- Added *fr.forms.FRRNAField* models field (gh-443).
- Added permanent account number(PAN) field in Indian flavor. (gh-457).
- Added the Canadian Models fields. (gh-465).

Modifications to existing flavors:

- Properly validate IBANs using BBAN to ensure invalid IBANs cannot be entered, updated IBAN_SEPA_COUNTRIES and IBAN_COUNTRY_CODE_LENGTH to latest data (gh-486).
- Fix typo in Marijampolė county name in LTCountySelect (gh-480).
- Add support for new Finnish identity codes (gh-478).
- CIF spanish starting with 'U' bug resolved (gh-469).
- Fix error code for BRPostalCodeValidator (gh-448).
- Fix spelling of the India state of Chhattisgarh (gh-444).
- Fix CURP regex for MX flavor (gh-449).
- Change text based fields that inherited from *django.forms.Field* to inherit from *django.forms.CharField*. The following fields have been updated (gh-446):
  - *at.forms.ATSocialSecurityNumberField*
  - *br.forms.BRStateChoiceField*
  - *ca.forms.CAProvinceField*
  - *ca.forms.CASocialInsuranceNumberField*
  - *ch.forms.CHIdentityCardNumberField*
  - *cu.forms.CUProvinceField*
  - *cu.forms.CURegionField*
  - *cz.forms.CZBirthNumberField*
  - *cz.forms.CZICNumberField*
  - *de.forms.DEIdentityCardNumberField*
  - *ee.forms.EEBusinessRegistryCode*
  - *ee.forms.EEPersonalIdentificationCode*
  - *fi.forms.FISocialSecurityNumber*
  - *gr.forms.GRTaxNumberCodeField*

  - *hr.forms.HRJMBAGField*
  - *hr.forms.HRJMBGField*
  - *hr.forms.HRLicensePlateField*
  - *hr.forms.HRPostalCodeField*
  - *id_.forms.IDLicensePlateField*
  - *id_.forms.IDNationalIdentityNumberField*
  - *id_.forms.IDPostCodeField*
  - *il.forms.ILIDNumberField*
  - *in_.forms.INAadhaarNumberField*
  - *in_.forms.INStateField*
  - *ir.forms.IRIDNumberField*
  - *it.forms.ITVatNumberField*
  - *lt.forms.LTPostalCodeField*
  - *lv.forms.LVPersonalCodeField*
  - *lv.forms.LVPostalCodeField*
  - *no.forms.NOSocialSecurityNumber*
  - *nz.forms.NZBankAccountNumberField*
  - *pt.forms.PTCitizenCardNumberField*
  - *pt.forms.PTSocialSecurityNumberField*
  - *ro.forms.ROCountyField*
  - *tr.forms.TRIdentificationNumberField*
  - *us.forms.USStateField*

- Removed inconvenient word VACA from CURP_INCONVENIENT_WORDS for MX flavor

Other changes:

- Use 'return value' when value is in the empty_values list (gh-461).
- Dropped support for Django 2.2, 3.0 and 3.1.
- Dropped support for Python 3.5.
- Added support for Python 3.10 and 3.11.

6.2.2 3.1 (2021-05-28)

Breaking data changes:

A schema and data migration are required for users of *mx.models.MXStateField* and *mx.forms.MXStateSelect*. The following steps are required:

- run *manage.py makemigrations* to generate a schema migration
- migrate *DIF* to *CDMX* with a data migration

A data migration is required for users of *in_.models.INStateField* and *in_.forms.INStateSelect*. The following data migrations are required:

- Migrate *CG* to *CT* for Chattisgarh
- Migrate *UA* to *UT* for Uttarakhand
- Migrate *DD* and *DN* to *DH* for Dadra and Nagar Haveli and Daman and Diu

A warning message will be displayed when *mx.models.MXStateField*, *mx.forms.MXStateSelect*, *in_.models.INStateField* or *in_.forms.INStateSelect* are used. See the localflavor online docs for instructions on how to suppress this warning once the migration has been completed.

New flavors:

- None

New fields for existing flavors:

- None

Modifications to existing flavors:

- Fix *fr.forms.FRNationalIdentificationNumber* validation for people born overseas (gh-415).
- Breaking data change: Updated Indian states and union territories names and code as per iso 3166 (https://www.iso.org/obp/ui/#iso:code:3166:IN). The key for Chattisgarh has been changed from CG to CT, the key for Uttarakhand has been changed from UA to UT, and the keys DD (Dadra and Nagar Haveli) and DN (Daman and Diu) have been removed and combined into DH (Dadra and Nagar Haveli and Daman and Diu). Ladakh (LA) is the new addition in the Union Territories. There are also a few modifications in the States and Union Territories names: Orissa (OR) is now Odisha (OR), Pondicherry (PY) is now Puducherry (PY) Andaman and Nicobar (AN) is now Andaman and Nicobar Islands (AN). (gh-427).
- Correct sorting of US_STATES to sort by full name rather than code (gh-424 gh-428).
- Added new region for CL (gh-432, gh-433).
- Updated IBAN validation for changes in IBAN Registry release 89, March 2021 (gh-436).
- Breaking data change: *mx.mx_states.STATE_CHOICES* has been updated to change DIF/Distrito Federal to CDMX/Ciudad de México, the legal name for this state as of 29 January 2016 (gh-235, gh-400, gh-438).

Other changes:

- Extended validation of BICs to match official SEPA regulations (gh-418).
- Removed positional arguments (**args*) from form fields that inherit from Django's *forms.CharField* and *forms.Field*. Positional arguments are not supported in the the parent form and did not work gh-421).
- Added error codes to all ValidationError's as recommended by 'Django's form validation documentation (gh-440).
- Renamed zh_CN and zh_TW locales to zh_Hans and zh_Hant respectively to match the Django locale names.

6.2.3 3.0 (2020-02-19)

Breaking changes:

Dropped support for Django < 2.2.

The deprecated *generic.checksums.luhn* and *generic.checksums.ean* functions have been removed in this release. Please use python-stdnum instead.

Some Icelandic postcodes in *IS_POSTALCODES* have had their spelling updated, and some entries have been removed entirely. A warning message will be displayed when *is_.forms.ISPostalCodeSelect* is used. See the localflavor online docs for instructions on how to suppress this warning once any incompatibilities have been dealt with.

A data migration is required for users of *it.forms.ITRegionProvinceSelect*. The *CI*, *VS*, *OG*, and *OT* keys need to be migrated to *SU* to account for the 2016 Italian provincial changes. Users wishing to maintain compatibility with the old provincial structure will need to create a custom version of *it.forms.ITRegionProvinceSelect*. A warning message will be displayed when *it.forms.ITRegionProvinceSelect* is used. See the localflavor online docs for instructions on how to suppress this warning once the migration has been completed.

Using positional arguments with fields that inherit from Django's *forms.RegexField* previously only worked with Django 1.11 but were ignored with Django >= 2.0. Positional arguments have now been removed from all fields that inherit from Django's *forms.RegexField*. Any options needed on the parent *forms.RegexField*, *forms.CharField* or *forms.Field* must now be set with keyword arguments.

New flavors:

- Egypt local flavor
- Malaysia local flavor

New fields for existing flavors:

- None

Modifications to existing flavors:

- Extended Danish *DK_POSTALCODES* with small Danish islands getting independent post code since 2017 (gh-380).
- Switched incorrect *ar.forms.ARCBUField* implementation to use python-stdnum instead (gh-391).
- Use set value of *strip* in fields that inherit from *django.forms.CharField* (gh-392):
  - *gb.forms.GBPostcodeField*
  - *si.forms.SIEMSOField*
  - *si.forms.SITaxNumberField*
  - *za.forms.ZAIDField*
- Updated Icelandic *IS_POSTALCODES* with missing entries, updated spelling of entries, and removed non-existing ones. See breaking changes notice above (gh-394).
- Add Kalimantan Utara in *PROVINCE_CHOICES* for Indonesia local flavor (gh-385).
- Add validation for women National identity number for Indonesia localflavor (gh-386).
- Updated *ITRegionProvinceSelect* for 2016 Italian provincial changes. See breaking changes notice above (gh-378, gh-402).
- Use the value returned by clean() in the following fields (gh-401, gh-403):
  - *ca.forms.CAProvinceField*
  - *ca.forms.CASocialInsuranceNumberField*
  - *ch.forms.CHIdentityCardNumberField*
  - *cl.forms.CLRutField*
  - *cn.forms.CNIDCardField*
  - *cu.forms.CURegionField*
  - *cu.forms.CUProvinceField*
  - *cz.forms.CZBirthNumberField*
  - *cz.forms.CZICNumberField*

- *de.forms.DEIdentityCardNumberField*
- *ee.forms.EEPersonalIdentificationCode*
- *eg.forms.EGNationalIDNumberField*
- *es.forms.ESIdentityCardNumberField*
- *es.forms.ESCCCField*
- *fi.forms.FISocialSecurityNumber*
- *fr.forms.FRNationalIdentificationNumber*
- *fr.forms.FRSIRENField*
- *fr.forms.FRSIRETField*
- *gr.forms.GRTaxNumberCodeField*
- *gr.forms.GRSocialSecurityNumberCodeField*
- *hr.forms.HRJMBGField*
- *hr.forms.HROIBField*
- *hr.forms.HRLicensePlateField*
- *hr.forms.HRPostalCodeField*
- *hr.forms.HRJMBAGField*
- *id.forms.IDPostCodeField*
- *id.forms.IDLicensePlateField*
- *id.forms.IDNationalIdentityNumberField*
- *kw.forms.KWCivilIDNumberField*
- *lt.forms.LTIDCodeField*
- *lv.forms.LVPersonalCodeField*
- *no.forms.NOSocialSecurityNumber*
- *nz.forms.NZBankAccountNumberField*
- *pl.forms.PLPESELField*
- *pl.forms.PLNationalIDCardNumberField*
- *pl.forms.PLNIPField*
- *pl.forms.PLREGONField*
- *pt.forms.PTCitizenCardNumberField*
- *pt.forms.PTSocialSecurityNumberField*
- *ro.forms.ROCountyField*
- *sg.forms.SGNRICFINField*
- *si.forms.SIEMSOField*
- *si.forms.SITaxNumberField*
- *tr.forms.TRIdentificationNumberField*
- *us.forms.USSocialSecurityNumberField*

  - *us.forms.USStateField*
  - *za.forms.ZAIDField*

- Removed unused positional arguments from fields that inherit from *forms.RegexField* (gh-405).

Other changes:

- Removed deprecated *generic.checksums.luhn* and *generic.checksums.ean* functions (gh-379).

6.2.4 2.2 (2019-05-07)

All deprecated code will be removed in the next release (3.0). Please run you project's tests using *python -Wd* so that deprecation warnings appear and can be addressed.

New flavors:

- Added local flavor for Iran (gh-359).

New fields for existing flavors:

- Added *BRPostalCodeField*, *BRCPFField* and *BRCNPJField* models fields (gh-365).
- Added *EircodeField* in IE flavor (gh-360) (gh-366).
- Added Models for Spain (*ESPostalCodeField* and *ESIdentityCardNumberField*) (gh-357) (gh-372).

Modifications to existing flavors:

- Deprecated *generic.checksums.luhn* and *generic.checksums.ean*. Please use the python-stdnum library instead. (gh-370).

Other changes:

- Added dependency on python-stdnum which is currently used for Luhn and EAN validation in several local-flavors (gh-370).
- Added support for Vatican IBAN (gh-355).
- Extended validation of BICs to check for the correct character set (gh-364).
- Run tests for Django 2.2 and Python 3.5, 3.6 and 3.7 (gh-368).
- Run tests for Django 2.0 and Python 3.7 (gh-368).

6.2.5 2.1 (2018-08-24)

New flavors:

- Added local flavor for Moldova (gh-309).

New fields for existing flavors:

- *NLLicensePlateField* in NL flavor (gh-327).
- *GRSocialSecurityNumberField* (AMKA) in GR flavor (gh-337).

Modifications to existing flavors:

- Allowed invalid message to be overridden in ESIdentityCardNumberField (gh-339).
- Fix COFA validation for USStateField (gh-303)

Other changes:

- Added VAT identification number validator for all EU locales (gh-324).

- Fix EAN validation when intermediate checksum is 10 (gh-331).
- Confirmed support for Django 2.1.
- Added 34 as a valid CUIT prefix value for *ARCUITField* (gh-342).

6.2.6 2.0 (2017-12-30)

All deprecated code has been removed in this release. Specifically, all of the phone number fields have been removed and we recommend that you use django-phonenumber-field instead. If you need to use django-phonenumber-field with Django 2.0, you will need to use the version from the Django 2.0 support pull request until this pull request is merged.

A full list of the removed classes and functions is the "Other changes" section below.

New flavors:

- None

New fields for existing flavors:

- None

Modifications to existing flavors:

- Changed RUT to NIT in CONITField form field error message.
- Fixed validation of Czech birth numbers for birth dates after 1st January 1954 (gh-315).

Other changes:

- Added support for Django 2.0 and dropped support for Django < 1.11 (gh-310).
- Fixed README and changelog documentation about dropping Python 2 and Django 1.11.
- Removed all deprecated classes, functions and associated data / regular expressions. These are the classes and functions that have been removed (gh-321):
  - *au.forms.AUPhoneNumberField*
  - *au.models.AUPhoneNumberField*
  - *be.forms.BEPhoneNumberField*
  - *br.forms.BRPhoneNumberField*
  - *br.forms.DV_maker*
  - *ca.forms.CAPhoneNumberField*
  - *ch.forms.CHPhoneNumberField*
  - *cn.forms.CNPhoneNumberField*
  - *cn.forms.CNCellNumberField*
  - *dk.forms.DKPhoneNumberField*
  - *es.forms.ESPhoneNumberField*
  - *fr.forms.FRPhoneNumberField*
  - *gr.forms.GRPhoneNumberField*
  - *gr.forms.GRMobilePhoneNumberField*
  - *hk.forms.HKPhoneNumberField* (*localflavor.hk* has been removed because it only contained this field)

- *hr.forms.HRPhoneNumberField*
- *hr.forms.HRPhoneNumberPrefixSelect*
- *id_.forms.IDPhoneNumberField*
- *il.forms.ILMobilePhoneNumberField*
- *in.forms.INPhoneNumberField*
- *is_.forms.ISPhoneNumberField*
- *it.forms.ITPhoneNumberField*
- *lt.forms.LTPhoneField*
- *nl.forms.NLPhoneNumberField*
- *nl.forms.NLSoFiNumberField*
- *nl.models.NLBankAccountNumberField*
- *nl.models.NLPhoneNumberField*
- *nl.models.NLSoFiNumberField*
- *nl.validators.NLBankAccountNumberFieldValidator*
- *nl.validators.NLPhoneNumberFieldValidator*
- *nl.validators.NLSoFiNumberFieldValidator*
- *no.forms.NOPhoneNumberField*
- *nz.forms.NZPhoneNumberField*
- *pk.forms.PKPhoneNumberField*
- *pk.models.PKPhoneNumberField*
- *pt.forms.PTPhoneNumberField*
- *ro.forms.ROIBANField*
- *ro.forms.ROPhoneNumberField*
- *sg.forms.SGPhoneNumberField*
- *sg.forms.SGNRIC_FINField*
- *si.forms.SIPhoneNumberField*
- *tr.forms.TRPhoneNumberField*
- *us.forms.USPhoneNumberField*
- *us.models.PhoneNumberField*

6.2.7 1.6 (2017-11-22)

All deprecated code will be removed in the next release. Please run you project's tests using *python -Wd* so that deprecation warnings appear and can be addressed.

New flavors:

- Added local flavor for Cuba (gh-292).

New fields for existing flavors:

- Added KWAreaSelect form field (gh-296).
- Added CONITField form field (gh-145).
- Added *nl.models.NLBSNField*, *nl.forms.NLBSNFormField* and *nl.validators.NLBSNFieldValidator* (gh-314).

Modifications to existing flavors:

- Fixed crash with USZipCodeField form validation when null=True is allowed (gh-295).
- Deprecated br.forms.DV_maker, sg.forms.SGNRIC_FINField, lt.forms.LTPhoneField and ro.forms.ROIBANField (gh-305).
- Added support for Swedish interim personal identity numbers (gh-308).
- Deprecated *nl.models.NLBankAccountNumberField* (gh-307).
- Updated IBANField to support the latest additions to the IBAN Registry (version 78 / August 2017).
- Deprecated *nl.models.NLSoFiNumberField*, *nl.forms.NLSoFiNumberField* and *nl.validators.NLSoFiNumberFieldValidator* (gh-314).
- Fixes issue with *no.forms.NOBankAccountNumber* unclean data (gh-311).

Other changes:

- Added support for empty_value kwarg in Django >= 1.11 (gh-298).
- Dropped support for Python 3.2.

6.2.8 1.5 (2017-05-26)

New flavors:

- Added local flavor for Ukraine (gh-273).

New fields for existing flavors:

- Added NOBankAccountNumber form field (gh-275).
- Added AUCompanyNumberField model and form field (gh-278).

Modifications to existing flavors:

- Added normalized versions of COFA state names for US (gh-277).
- Fixed Dutch NLZipCodeField field not to store empty value as a single space (gh-280).
- Fixed validation for old Australian tax file numbers (gh-284).

Other changes:

- None

6.2.9 1.4 (2017-01-03)

New flavors:

- Added local flavor for Venezuela (gh-245).
- Added local flavor for Morocco (gh-270).

New fields for existing flavors:

- Added MXCLABEField model and form fields (gh-227).

- Added AUTaxFileNumberField model and form fields (gh-238).
- Added KWGovernorateSelect field to easily select Kuwait governorates. (gh-231).
- Added FRRegion2016Select field to stick to current legislation (gh-260). and (gh-268).

Modifications to existing flavors:

- Enhancements of localflavor.br.forms.BRCNPJField (gh-240 gh-254).
- Fixed century bug with Kuwait Civil ID verification localflavor.kw.forms (gh-195).
- Allow passing field name as first positional argument of IBANField (gh-236).
- Fixed French FRNationalIdentificationNumber bug with imaginary birth month values (gh-242).
- Fixed French FRNationalIdentificationNumber bug with corsican people born after 2000 (gh-242).
- Fixed the translation for US state 'Georgia' from colliding with the country 'Georgia' (gh-250).
- Fixed the styling errors and enabled prospector (gh-259).
- Allow AU ABN value with spaces to validate (gh-266 gh-267).

Other changes:

- Drop support for Django 1.7 (gh-218).
- Ensure the migration framework generates schema migrations for model fields that change the max_length (gh-257). Users will need to generate migrations for any model fields they use with 'makemigrations'.
- Lazily generate US_STATES, STATE_CHOICES, and USPS_CHOICES (gh-203 gh-272).
- Deprecated Phone Number fields (gh-262).
- Bumped versions of requirements for testing (gh-274).

6.2.10 1.3 (2016-05-06)

New flavors:

- Added local flavor for Bulgaria (gh-191).
- Added local flavor for Tunisia (gh-141).
- Added local flavor for Hungary (gh-213).

New fields for existing flavors:

- Added ARCBUField form field. (gh-151).
- Added NLZipCodeField, NLProvinceField, NLSoFiNumberField, NLPhoneNumberField model fields (gh-152).
- Added AUBusinessNumberField model and form fields (gh-63).

Modifications to existing flavors:

- Moved Dutch validators from localflavor.nl.forms to localflavor.nl.validators (gh-152).
- Fix check for promotional social security numbers in USSocialSecurityNumberField (gh-157).
- Updated IBANField to support the latest additions to the IBAN Registry (version 64 / March 2016).
- Fix bug with MXRFCField where some incorrect values would validate correctly. (gh-204).
- Fixed bug with IBANFormField validation. (gh-215).
- Update regex in DEZipCodeField to prohibit invalid postal codes. (gh-216).

- Added deconstructor methods to validators. (gh-220).
- Fix bug in ESIdentityCardNumberField where some valid values for NIE numbers were not validating (gh-217).
- Add deconstruct method to all model fields (gh-162 gh-224).

Other changes:

- Drop support for Django 1.5, Django 1.6 and Python 2.6 (gh-170).

6.2.11 1.2 (2015-11-27)

New flavors:

- None

New fields for existing flavors:

- Added form field for Estonian business registration codes (gh-135).
- Added model field for Ecuadorian provinces (gh-138).
- Added form field for Swiss Social Security numbers ((gh-155).
- Added form field for Brazilian Legal Process numbers (Processo) (gh-163).

Modifications to existing flavors:

- Fixed misspelled Polish administrative unit names (gh-136).
- Added Kosovo and Timor-Leste to list of IBAN countries (gh-139).
- Fixed error in Romanian fiscal identity code (CIF) field when value has a trailing slash (gh-146).
- Updated validation in Swiss postal code field to only accept values in the range 1000 - 9000 (gh-154).
- Added validator for International Article Number (EAN) to the generic module (gh-156).
- Updated Italian social security number field to use 'tax code' in error message (gh-167).
- Fixed error in Greek tax number code field when value has only alpha characters (gh-171).
- Added stricter validation in the Brazilian Cadastro de Pessoas Físicas (CPF) field (gh-172).
- Corrected Romanian counties choice names to use ș and ț (comma below) (gh-175).
- Updated Brazilian postal code field to also accept values with XX.XXX-XXX and XXXXXXXX formats (gh-177).
- Marked US state names for translation (gh-178).
- Fixed French national identification number validation for people born before 1976 in Corsica (gh-186).

6.2.12 1.1 (2014-12-10)

New flavors:

- Added local flavor for Denmark (gh-83)
- Added local flavor for Estonia (gh-70)
- Added local flavor for Latvia (gh-68)
- Added local flavor for Malta (gh-88)
- Added local flavor for Pakistan (gh-41)

- Added local flavor for Singapore (gh-119)

New fields for existing flavors:

- Added model and form fields for French SIREN/SIRET numbers (gh-123)
- Added model field for states of Brazil (gh-22)
- Added form field for Indian Aadhaar numbers (gh-23)
- Added model field for states of India (gh-23)
- Added form field for Lithuanian phone numbers
- Added model field for Dutch bank accounts (gh-42)
- Added form field for Italian phone numbers (gh-74)
- Added form field for French National Identification Number (gh-75)
- Added IBAN model and form fields (gh-86)
- Added BIC model and form fields (gh-125)
- Added SSN model field for US (gh-96)
- Added ZIP code model field for US (gh-55)

Other modifications to existing flavors:

- *backward incompatible* Updated the region lists of Great Britain (gh-43, gh-126)
- Added Ceuta and Mellila to regions of Spain (gh-8)
- Added support entities in Italian SSN form field (gh-20)
- Added Japanese prefecture codes and fix prefecture order (gh-27)
- Added normalization for Lithuanian postal code field (gh-69)
- Added whitespace stripping whitespace from US ZIP code field (gh-77)
- Added an option for customizing French form field labels (gh-102)
- Added mapping between provinces and regions for Italy (gh-105)
- Added Telengana to states of India (gh-107)
- Added support for 14X and 17X Chinese cell numbers (gh-17, gh-120)
- Allowed spaces in CPF numbers for Brazil (gh-32)
- Fixed CIF validation for Spain (gh-78)
- Fixed armed forces “states” for US (gh-8)
- Fixed REGON number validation for Poland (gh-62)
- Rejected US SSN starting with 9 (gh-35)
- Rejected Brazilian CPF number when all numbers all numbers are equal (gh-103)
- Added ‘Y’ to the NIE number validation for Spain (gh-127)
- Updated Argentina’s CUIT number validation to support legal types 24 and 33 (gh-121)
- Added ‘R’, ‘V’ and ‘W’ to the Spanish identity card number validation (gh-132)

Other changes:

- Added checksums module (from Django) providing a Luhn validator (gh-122)

6.2.13 1.0 (2013-07-29)

Initial release

- genindex
- modindex
- search

6.3 Argentina (`ar`)

AR-specific Form helpers.

class localflavor.ar.forms.**ARCBUField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

This field validates a CBU (Clave Bancaria Uniforme).

A CBU is a 22-digits long number. The first 8 digits denote bank and branch number, plus a verifying digit. The remaining 14 digits denote an account number, plus a verifying digit.

More info: https://es.wikipedia.org/wiki/Clave_Bancaria_Uniforme

New in version 1.3.

Changed in version 3.0.

clean(*value*)

Value must be a 22 digits long number.

class localflavor.ar.forms.**ARCUITField**(***kwargs*)

This field validates a CUIT (Código Único de Identificación Tributaria).

A CUIT is of the form XX-XXXXXXXX-V. The last digit is a check digit.

More info: http://es.wikipedia.org/wiki/Clave_%C3%9Anica_de_Identificaci%C3%B3n_Tributaria

Info in English: http://www.justlanded.com/english/Argentina/Argentina-Guide/Visas-Permits/Other-Legal-Documents

Changed in version 2.1: `ARCUITField` now also accepts CUIT with prefix 34.

clean(*value*)

Value can be either a string in the format XX-XXXXXXXX-X or an 11-digit number.

class localflavor.ar.forms.**ARDNIField**(*max_length=10*, *min_length=7*, ***kwargs*)

A field that validates 'Documento Nacional de Identidad' (DNI) numbers.

clean(*value*)

Value can be a string either in the [X]X.XXX.XXX or [X]XXXXXXX formats.

class localflavor.ar.forms.**ARPostalCodeField**(*max_length=8*, *min_length=4*, ***kwargs*)

A field that accepts a 'classic' NNNN Postal Code or a CPA.

See:

- http://www.correoargentino.com.ar/cpa/que_es
- http://www.correoargentino.com.ar/cpa/como_escribirlo

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ar.forms.**ARProvinceSelect**(*attrs=None*)
A Select widget that uses a list of Argentinean provinces/autonomous cities as its choices.

localflavor.ar.ar_provinces.**PROVINCE_CHOICES = (('B', 'Buenos Aires'), ('K', 'Catamarca'),**
A list of Argentinean provinces and autonomous cities as *choices* in a formfield. From http://www.argentina.gov.ar/argentina/portal/paginas.dhtml?pagina=425

6.4 Austria (at)

AT-specific Form helpers.

class localflavor.at.forms.**ATSocialSecurityNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
Austrian Social Security numbers are composed of a 4 digits and 6 digits field.

The latter represents in most cases the person's birthdate while the first 4 digits represent a 3-digits counter and a one-digit checksum.

The 6-digits field can also differ from the person's birthdate if the 3-digits counter suffered an overflow.

This code is based on information available on http://de.wikipedia.org/wiki/Sozialversicherungsnummer#.C3.96sterreich

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.at.forms.**ATStateSelect**(*attrs=None*)
A `Select` widget that uses a list of AT states as its choices.

class localflavor.at.forms.**ATZipCodeField**(***kwargs*)
A form field that validates its input is an Austrian postcode.

Accepts 4 digits (first digit must be greater than 0).

localflavor.at.at_states.**STATE_CHOICES = (('BL', 'Burgenland'), ('KA', 'Carinthia'), ('NO',**
A list of Austrian states according to https://en.wikipedia.org/wiki/States_of_Austria

6.5 Australia (au)

6.5.1 Forms

Australian-specific Form helpers.

class localflavor.au.forms.**AUBusinessNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates input as an Australian Business Number (ABN).

New in version 1.3.

Changed in version 1.4.

prepare_value(*value*)
Format the value for display.

to_python(*value*)
Return a string.

class localflavor.au.forms.**AUCompanyNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates input as an Australian Company Number (ACN).

New in version 1.5.

prepare_value(*value*)
Format the value for display.

to_python(*value*)
Return a string.

class localflavor.au.forms.**AUPostCodeField**(*max_length=4*, ***kwargs*)
Australian post code field.

Assumed to be 4 digits. Northern Territory 3-digit postcodes should have leading zero.

class localflavor.au.forms.**AUStateSelect**(*attrs=None*)
A Select widget that uses a list of Australian states/territories as its choices.

class localflavor.au.forms.**AUTaxFileNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates input as an Australian Tax File Number (TFN).

New in version 1.4.

prepare_value(*value*)
Format the value for display.

6.5.2 Models

class localflavor.au.models.**AUBusinessNumberField**(**args*, ***kwargs*)
A model field that checks that the value is a valid Australian Business Number (ABN).

New in version 1.3.

description = 'Australian Business Number'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

to_python(*value*)
Ensure the ABN is stored without spaces.

validators = [<localflavor.au.validators.AUBusinessNumberFieldValidator object>, <djan

class localflavor.au.models.**AUCompanyNumberField**(**args*, ***kwargs*)
A model field that checks that the value is a valid Australian Company Number (ACN).

New in version 1.5.

description = 'Australian Company Number'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

to_python(*value*)
Ensure the ACN is stored without spaces.

validators = [<localflavor.au.validators.AUCompanyNumberFieldValidator object>, <django

class localflavor.au.models.**AUPostCodeField**(**args*, ***kwargs*)
A model field that stores the four-digit Australian postcode in the database.

This field is represented by forms as a *AUPostCodeField* field.

description = 'Australian Postcode'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.au.models.**AUStateField**(**args*, ***kwargs*)
A model field that stores the three-letter Australian state abbreviation in the database.

It is represented with STATE_CHOICES` choices.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

description = 'Australian State'

class localflavor.au.models.**AUTaxFileNumberField**(**args*, ***kwargs*)
A model field that checks that the value is a valid Tax File Number (TFN).

A TFN is a number issued to a person by the Commissioner of Taxation and is used to verify client identity and establish their income levels. It is a eight or nine digit number without any embedded meaning.

New in version 1.4.

description = 'Australian Tax File Number'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

to_python(*value*)
Ensure the TFN is stored without spaces.

```
validators = [<localflavor.au.validators.AUTaxFileNumberFieldValidator object>, <django
```

6.5.3 Data

localflavor.au.au_states.**STATE_CHOICES = (('ACT', 'Australian Capital Territory'), ('NSW',**
An alphabetical list of states for use as *choices* in a formfield.

6.6 Belgium (be)

6.6.1 Forms

Belgium-specific Form helpers.

class localflavor.be.forms.**BEPostalCodeField**(***kwargs*)
A form field that validates its input as a belgium postal code.

Belgium postal code is a 4 digits string. The first digit indicates the province (except for the 3ddd numbers that are shared by the eastern part of Flemish Brabant and Limburg and the and 1ddd that are shared by the Brussels Capital Region, the western part of Flemish Brabant and Walloon Brabant)

class localflavor.be.forms.**BEProvinceSelect**(*attrs=None*)
A Select widget that uses a list of belgium provinces as its choices.

class localflavor.be.forms.**BERegionSelect**(*attrs=None*)
A Select widget that uses a list of belgium regions as its choices.

6.6.2 Data

localflavor.be.be_provinces.**PROVINCE_CHOICES = (('VAN', 'Antwerp'), ('BRU', 'Brussels'), (**
ISO codes

localflavor.be.be_regions.**REGION_CHOICES = (('BRU', 'Brussels Capital Region'), ('VLG', 'Fl**
ISO codes

6.7 Bulgaria (bg)

6.7.1 Validators

class localflavor.bg.validators.**EGNValidator**
Check Bulgarian unique citizenship number (EGN) for validity.

More details https://en.wikipedia.org/wiki/Unique_citizenship_number Full information in Bulgarian about algorithm is available here http://www.grao.bg/esgraon.html#section2

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

class localflavor.bg.validators.**EIKValidator**
Check Bulgarian EIK/BULSTAT codes for validity.

Full information in Bulgarian about algorithm is available here http://bulstat.registryagency.bg/About.html

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

6.7.2 Model Fields

class localflavor.bg.models.**BGEGNField**(**args*, ***kwargs*)
Field that stores Bulgarian unique citizenship number (EGN).

This is shortcut for:

```
models.CharField(max_length=10, validators=[localflavor.bg.validators.egn_
↪validator])
```

class localflavor.bg.models.**BGEIKField**(**args*, ***kwargs*)
Field that stores Bulgarian EIK/BULSTAT codes.

This is shortcut for:

```
models.CharField(max_length=13, validators=[localflavor.bg.validators.eik_
↪validator])
```

6.7.3 Utils

localflavor.bg.utils.**get_egn_birth_date**(*egn*)
Extract birth date from Bulgarian unique citizenship number (EGN).

More details https://en.wikipedia.org/wiki/Unique_citizenship_number Information in Bulgarian for this can be found here http://www.grao.bg/esgraon.html#section2

6.8 Brazil (br)

6.8.1 Forms

BR-specific Form helpers.

class localflavor.br.forms.**BRCNPJField**(*min_length=14*, *max_length=18*, ***kwargs*)
A form field that validates input as Brazilian CNPJ.

Input can either be of the format XX.XXX.XXX/XXXX-XX or be a group of 14 digits.

If you want to use the long format only, you can specify: brcnpj_field = BRCNPJField(min_length=16)

If you want to use the short format, you can specify: brcnpj_field = BRCNPJField(max_length=14)

Otherwise both formats will be valid.

Changed in version 1.4.

Changed in version 2.2: Use BRCNPJValidator to centralize validation logic and share with equivalent model field. More details at: https://github.com/django/django-localflavor/issues/334

class localflavor.br.forms.**BRCPFField**(*max_length=14*, *min_length=11*, ***kwargs*)
A form field that validates a CPF number or a CPF string.

A CPF number is compounded by XXX.XXX.XXX-VD. The two last digits are check digits.

More information: http://en.wikipedia.org/wiki/Cadastro_de_Pessoas_F%C3%ADsicas

Changed in version 2.2: Use BRCPFValidator to centralize validation logic and share with equivalent model field. More details at: https://github.com/django/django-localflavor/issues/334

class localflavor.br.forms.**BRProcessoField**(*max_length=25*, *min_length=20*, ***kwargs*)
A form field that validates a Legal Process(Processo) number or a Legal Process string.

A Processo number is compounded by NNNNNNN-DD.AAAA.J.TR.OOOO. The two DD digits are check digits. More information: http://www.cnj.jus.br/busca-atos-adm?documento=2748

New in version 1.2.

clean(*value*)
Value can be either a string in the format NNNNNNN-DD.AAAA.J.TR.OOOO or an 20-digit number.

class localflavor.br.forms.**BRStateChoiceField**(***kwargs*)
A choice field that uses a list of Brazilian states as its choices.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.br.forms.**BRStateSelect**(*attrs=None*)
A Select widget that uses a list of Brazilian states/territories as its choices.

class localflavor.br.forms.**BRZipCodeField**(***kwargs*)
A form field that validates input as a Brazilian zip code, with the format 00000-000.

Changed in version 2.2: Use BRPostalCodeValidator to centralize validation logic and share with equivalent model field. More details at: https://github.com/django/django-localflavor/issues/334

6.8.2 Models

class localflavor.br.models.**BRCNPJField**(**args*, ***kwargs*)
A model field for the brazilian document named of CNPJ (Cadastro Nacional de Pessoa Jurídica)

New in version 2.2.

class localflavor.br.models.**BRCPFField**(**args*, ***kwargs*)
A model field for the brazilian document named of CPF (Cadastro de Pessoa Física)

New in version 2.2.

class localflavor.br.models.**BRPostalCodeField**(**args*, ***kwargs*)
A model field for the brazilian zip code

New in version 2.2.

class localflavor.br.models.**BRStateField**(**args*, ***kwargs*)
A model field for states of Brazil.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

6.8.3 Data

localflavor.br.br_states.**STATE_CHOICES = (('AC', 'Acre'), ('AL', 'Alagoas'), ('AP', 'Amapá**
An alphabetical list of Brazilian states for use as *choices* in a formfield

6.9 Canada (ca)

6.9.1 Forms

Canada-specific Form helpers.

class localflavor.ca.forms.**CAPostalCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

Canadian postal code form field.

Validates against known invalid characters: D, F, I, O, Q, U Additionally the first character cannot be Z or W. For more info see: http://www.canadapost.ca/tools/pg/manual/PGaddress-e.asp#1402170

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ca.forms.**CAProvinceField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field that validates its input is a Canadian province name or abbreviation.

It normalizes the input to the standard two-leter postal service abbreviation for the given province.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ca.forms.**CAProvinceSelect**(*attrs=None*)

A Select widget that uses a list of Canadian provinces and territories as its choices.

class localflavor.ca.forms.**CASocialInsuranceNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A Canadian Social Insurance Number (SIN).

Checks the following rules to determine whether the number is valid:

- Conforms to the XXX-XXX-XXX format.
- **Passes the check digit process "Luhn Algorithm"** See: http://en.wikipedia.org/wiki/Social_Insurance_Number

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.9.2 Models

class localflavor.ca.models.**CAPostalCodeField**(**args*, ***kwargs*)
A model field that stores the Canadian Postal code in the database.

Forms represent it as a *CAPostalCodeField* field.

New in version 4.0.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.ca.models.**CAProvinceField**(**args*, ***kwargs*)
A model field that stores the two-letter Canadian province abbreviation in the database.

Forms represent it as a `forms.CAProvinceField` field.

New in version 4.0.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.ca.models.**CASocialInsuranceNumberField**(**args*, ***kwargs*)
A model field that stores a Canadian Social Insurance Number (SIN) in the format XXX-XXX-XXX.

Forms represent it as forms.CASocialInsuranceNumberField field.

New in version 4.0.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

6.9.3 Data

localflavor.ca.ca_provinces.**PROVINCE_CHOICES = (('AB', 'Alberta'), ('BC', 'British Columbia**
An alphabetical list of provinces and territories for use as *choices* in a formfield. Source: http://www.canada.gc.ca/othergov/prov_e.html

localflavor.ca.ca_provinces.**PROVINCES_NORMALIZED = {'ab': 'AB', 'alberta': 'AB', 'b.c.':**
a mapping of province misspellings/abbreviations to normalized abbreviations

6.10 Switzerland (ch)

6.10.1 Forms

Swiss-specific Form helpers.

class localflavor.ch.forms.**CHIdentityCardNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A Swiss identity card number.

Checks the following rules to determine whether the number is valid:

- Conforms to the X1234567<0 or 1234567890 format.
- Included checksums match calculated checksums

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ch.forms.**CHSocialSecurityNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A Swiss Social Security number (also known as the new AHV Number).

Checks the following rules to determine whether the number is valid:

- Conforms to the 756.XXXX.XXXX.XX
- Included checksums match calculated checksums

See: http://de.wikipedia.org/wiki/Sozialversicherungsnummer#Versichertennummer

New in version 1.2.

class localflavor.ch.forms.**CHStateSelect**(*attrs=None*)
A Select widget that uses a list of CH states as its choices.

class localflavor.ch.forms.**CHZipCodeField**(***kwargs*)
A form field that validates input as a Swiss zip code.

Valid codes consist of four digits ranging from 1XXX to 9XXX.

See: http://en.wikipedia.org/wiki/Postal_codes_in_Switzerland_and_Liechtenstein

6.10.2 Data

localflavor.ch.ch_states.**STATE_CHOICES = (('AG', 'Aargau'), ('AI', 'Appenzell Innerrhoden')**
An alphabetical list of states

6.11 Chile (cl)

6.11.1 Forms

Chile specific form helpers.

class localflavor.cl.forms.**CLRegionSelect**(*attrs=None*)
A Select widget that uses a list of Chilean Regions (Regiones) as its choices.

class localflavor.cl.forms.**CLRutField**(***kwargs*)
Chilean "Rol Unico Tributario" (RUT) field.

This is the Chilean national identification number.

Samples for testing are available from https://palena.sii.cl/cvc/dte/ee_empresas_emisoras.html

clean(*value*)
Check and clean the Chilean RUT.

6.11.2 Data

localflavor.cl.cl_regions.**REGION_CHOICES = (('RM', 'Región Metropolitana de Santiago'), ('**
A list of Chilean regions as *choices* in a formfield.

6.12 China (cn)

6.12.1 Forms

China(mainland)-specific Form helpers.

class localflavor.cn.forms.**CNProvinceSelect**(*attrs=None*)
A select widget providing the list of provinces and districts in People's Republic of China as choices.

class localflavor.cn.forms.**CNPostCodeField**(***kwargs*)
A form field that validates input as postal codes in mainland China.

Valid codes are in the format of XXXXXX where X is a digit.

class localflavor.cn.forms.**CNIDCardField**(*max_length=18*, *min_length=15*, ***kwargs*)
A form field that validates input as a Resident Identity Card (PRC) number.

This field would check the following restrictions:

- the length could only be 15 or 18;
- if the length is 18, the last character can be x or X;
- has a valid checksum (only for those with a length of 18);
- has a valid date of birth;
- has a valid province.

The checksum algorithm is described in GB11643-1999. See: http://en.wikipedia.org/wiki/Resident_Identity_Card#Identity_card_number

clean(*value*)
Check whether the input is a valid ID Card Number.

has_valid_birthday(*value*)
This method grabs the date of birth from the ID card number and test whether it is a valid date.

has_valid_checksum(*value*)
This method checks if the last letter/digit is valid according to GB11643-1999.

has_valid_location(*value*)
This method checks if the first two digits in the ID Card are valid province code.

6.12.2 Data

localflavor.cn.cn_provinces.**CN_PROVINCE_CHOICES = (('anhui', ''), ('beijing', ''), ('chongc**
An alphabetical list of provinces for use as *choices* in a formfield. http://en.wikipedia.org/wiki/ISO_3166-2:CN http://en.wikipedia.org/wiki/Province_%28China%29 http://en.wikipedia.org/wiki/Direct-controlled_municipality http://en.wikipedia.org/wiki/Autonomous_regions_of_China

6.13 Colombia (co)

6.13.1 Forms

Colombian-specific form helpers.

class localflavor.co.forms.**CODepartmentSelect**(*attrs=None*)
A Select widget that uses a list of Colombian states as its choices.

class localflavor.co.forms.**CONITField**(***kwargs*)
This field validates a NIT (NUmero de IdentificaciOn Tributaria). A NIT is of the form XXXXXXXXXX-V. The last digit is a check digit. This field can be used for people and companies.

More info: http://es.wikipedia.org/wiki/N%C3%BAmero_de_Identificaci%C3%B3n_Tributaria

clean(*value*)
Value can be either a string in the format XXXXXXXXXX-Y or XXXXXXXXXXY.

6.13.2 Data

localflavor.co.co_departments.**DEPARTMENT_CHOICES = (('AMA', 'Amazonas'), ('ANT', 'Antioquia**
A list of Colombian departaments as *choices* in a formfield.

6.14 Czech Republic (cz)

6.14.1 Forms

Czech-specific form helpers.

class localflavor.cz.forms.**CZBirthNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

Czech birth number form field.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.cz.forms.**CZICNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

Czech IC number form field.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.cz.forms.**CZPostalCodeField**(***kwargs*)

A form field that validates its input as Czech postal code.

Valid form is XXXXX or XXX XX, where X represents integer.

clean(*value*)

Validates the input and returns a string that contains only numbers.

Returns an empty string for empty values.

class localflavor.cz.forms.**CZRegionSelect**(*attrs=None*)

A select widget widget with list of Czech regions as choices.

6.14.2 Data

localflavor.cz.cz_regions.**REGION_CHOICES = (('PR', 'Prague'), ('CE', 'Central Bohemian Reg**

Czech regions, translations get from http://www.crwflags.com/fotw/Flags/cz-re.html

6.15 Germany (de)

6.15.1 Forms

DE-specific Form helpers.

class localflavor.de.forms.**DEIdentityCardNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A German identity card number.

Checks the following rules to determine whether the number is valid:

- Conforms to the XXXXXXXXXXX-XXXXXXX-XXXXXXX-X format.
- No group consists entirely of zeroes.
- Included checksums match calculated checksums

Algorithm is documented at http://de.wikipedia.org/wiki/Personalausweis

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.de.forms.**DEStateSelect**(*attrs=None*)
A Select widget that uses a list of DE states as its choices.

class localflavor.de.forms.**DEZipCodeField**(***kwargs*)
A form field that validates input as a German zip code.

Valid zip codes consist of five digits.

6.15.2 Data

localflavor.de.de_states.**STATE_CHOICES = (('BW', 'Baden-Wuerttemberg'), ('BY', 'Bavaria'),**
An alphabetical list of states

6.16 Denmark (dk)

New in version 1.1.

6.16.1 Forms

Denmark specific Form helpers.

class localflavor.dk.forms.**DKMunicipalitySelect**(*attrs=None*)
A Select widget that uses a list of Danish municipalities (kommuner) as its choices.

class localflavor.dk.forms.**DKPostalCodeField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
An Input widget that uses a list of Danish postal codes as valid input.

6.16.2 Data

localflavor.dk.dk_postalcodes.**DK_POSTALCODES = (('0555', 'Scanning'), ('0783', 'Facility'),**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

localflavor.dk.dk_municipalities.**REGION_HOVEDSTADEN = [('albertslund', 'Albertslund'), ('al**
A list of municipalities in the Danish region Hovedstaden as *choices* in a formfield.

localflavor.dk.dk_municipalities.**REGION_MIDTJYLLAND = [('favrskov', 'Favrskov'), ('hedenste**
A list of municipalities in the Danish region Midtjylland as *choices* in a formfield.

localflavor.dk.dk_municipalities.**REGION_NORDJYLLAND = [('broenderslev', 'Brønderslev'), ('**
A list of municipalities in the Danish region Nordjylland as *choices* in a formfield.

localflavor.dk.dk_municipalities.**REGION_SJAELLAND = [('faxe', 'Faxe'), ('greve', 'Greve'),**
A list of municipalities in the Danish region Sjælland as *choices* in a formfield.

localflavor.dk.dk_municipalities.**REGION_SYDDANMARK = [('assens', 'Assens'), ('billund', 'Bi**
A list of municipalities in the Danish region Syddanmark as *choices* in a formfield.

localflavor.dk.dk_municipalities.**DK_MUNICIPALITIES = [('Region Hovedstaden', [('albertslund**
A list of Danish municipalities grouped by region.

6.17 Ecuador (ec)

6.17.1 Forms

Ecuador-specific form helpers.

class localflavor.ec.forms.**ECProvinceSelect**(*attrs=None*)
A Select widget that uses a list of Ecuador provinces as its choices.

6.17.2 Data

localflavor.ec.ec_provinces.**PROVINCE_CHOICES = (('A', 'Azuay'), ('B', 'Bolívar'), ('F', 'Ca**
A list of Ecuador provinces as *choices* in a formfield.

6.18 Estonia (ee)

New in version 1.1.

6.18.1 Forms

class localflavor.ee.forms.**EEBusinessRegistryCode**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field that validates input as an Estonian business registration code.

New in version 1.2.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ee.forms.**EECountySelect**(*attrs=None*)
A Select widget that uses a list of Estonian counties as its choices.

class localflavor.ee.forms.**EEPersonalIdentificationCode**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field that validates input as an Estonian personal identification code.

See: https://www.riigiteataja.ee/akt/106032012004

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

static ee_checksum(*value*)
Takes a string of digits as input, returns check digit.

class localflavor.ee.forms.**EEZipCodeField**(***kwargs*)
A form field that validates input as a Estonian zip code.

Valid codes consist of five digits; first digit cannot be 0.

6.18.2 Data

localflavor.ee.ee_counties.**COUNTY_CHOICES = (('37', 'Harju County'), ('39', 'Hiiu County'),**
A list of Estonian counties as *choices* in a formfield. Identifiers based on ISO 3166-2:EE. https://en.wikipedia.org/wiki/ISO_3166-2:EE

6.19 Egypt (eg)

6.19.1 Forms

Egypt-specific Form helpers.

class localflavor.eg.forms.**EGGovernorateSelect**(*attrs=None*)
A Select widget that uses a list of Egypt governorates as its choices.

New in version 3.0.

class localflavor.eg.forms.**EGNationalIDNumberField**(*max_length=14*, *min_length=14*, ***kwargs*)
Egypt ID numbers are 14 digits, second to seventh digits represents the person's birthdate.

Checks the following rules to determine the validity of the number:

- The number consist of 14 digits.
- The century number is valid.
- The birthdate of the person is a valid date.
- The governorate code is valid.

New in version 3.0.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.20 Spain (es)

6.20.1 Forms

Spanish-specific Form helpers.

class localflavor.es.forms.**ESCCCField**(***kwargs*)
A form field that validates its input as a Spanish bank account or CCC (Codigo Cuenta Cliente).

Spanish CCC is in format EEEE-OOOO-CC-AAAAAAAAAA where:

E = entity O = office C = checksum A = account

It's also valid to use a space as delimiter, or to use no delimiter.

First checksum digit validates entity and office, and last one validates account. Validation is done multiplying every digit of 10 digit value (with leading 0 if necessary) by number in its position in string 1, 2, 4, 8, 5, 10, 9, 7, 3, 6. Sum resulting numbers and extract it from 11. Result is checksum except when 10 then is 1, or when 11 then is 0.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.es.forms.**ESIdentityCardNumberField**(*only_nif=False*, ***kwargs*)
Spanish NIF/NIE/CIF (Fiscal Identification Number) code.

Validates three different formats:

NIF (individuals): 12345678A CIF (companies): A12345678 NIE (foreigners): X12345678A

according to a couple of simple checksum algorithms.

Value can include a space or hyphen separator between number and letters. Number length is not checked for NIF (or NIE), old values start with a 1, and future values can contain digits greater than 8. The CIF control digit can be a number or a letter depending on company type. Algorithm is not public, and different authors have different opinions on which ones allows letters, so both validations are assumed true for all types.

http://es.wikipedia.org/wiki/N%C3%BAmero_de_identificaci%C3%B3n_fiscal

Changed in version 1.1.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.es.forms.**ESPostalCodeField**(***kwargs*)
A form field that validates its input as a spanish postal code.

Spanish postal code is a five digits string, with two first digits between 01 and 52, assigned to provinces code.

class localflavor.es.forms.**ESProvinceSelect**(*attrs=None*)
A Select widget that uses a list of spanish provinces as its choices.

class localflavor.es.forms.**ESRegionSelect**(*attrs=None*)
A Select widget that uses a list of spanish regions as its choices.

6.20.2 Models

class localflavor.es.models.**ESIdentityCardNumberField**(**args*, ***kwargs*)
A model field that stores Spanish NIF/NIE/CIF in format `XXXXXXXXX`

Forms represent it as `form.ESIdentityCardNumberField` field.

New in version 2.2.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

to_python(*value*)
Convert the input value into the expected Python data type, raising django.core.exceptions.ValidationError if the data can't be converted. Return the converted value. Subclasses should override this.

class localflavor.es.models.**ESPostalCodeField**(**args*, ***kwargs*)
A model field that stores the five numbers (XXXXX) of Spain Postal Codes

Forms represent it as `form.ESPostalCodeField`

New in version 2.2.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

6.20.3 Data

localflavor.es.es_provinces.**PROVINCE_CHOICES = (('01', 'Araba'), ('02', 'Albacete'), ('03',**
An alphabetical list of provinces

localflavor.es.es_regions.**REGION_CHOICES = (('AN', 'Andalusia'), ('AR', 'Aragon'), ('O', '**
An alphabetical list of regions

6.21 Finland (`fi`)

6.21.1 Forms

FI-specific Form helpers.

class localflavor.fi.forms.**FIMunicipalitySelect**(*attrs=None*)
A Select widget that uses a list of Finnish municipalities as its choices.

class localflavor.fi.forms.**FISocialSecurityNumber**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates input as a Finnish social security number.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.fi.forms.**FIZipCodeField**(***kwargs*)
A form field that validates input as a Finnish zip code.

Valid codes consist of five digits.

6.21.2 Data

localflavor.fi.fi_municipalities.**MUNICIPALITY_CHOICES = (('akaa', 'Akaa'), ('alajarvi', 'Al**
An alphabetical list of Finnish municipalities for use as *choices* in a formfield.

6.22 France (`fr`)

6.22.1 Forms

FR-specific Form helpers

class localflavor.fr.forms.**FRDepartmentField**(***kwargs*)
A Select Field that uses a FRDepartmentSelect widget.

widget
alias of *FRDepartmentSelect*

class localflavor.fr.forms.**FRDepartmentSelect**(*attrs=None*)
A Select widget that uses a list of FR departments as its choices.

class localflavor.fr.forms.**FRNationalIdentificationNumber**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
Validates input as a French National Identification number.

Validation of the Number, and checksum calculation is detailed at http://en.wikipedia.org/wiki/INSEE_code

Complete spec of the codification is detailed here:

- https://fr.scribd.com/document/456848429/INSEE-Guide-Identification
- https://fr.scribd.com/document/456848431/INSEE-Codes-Pays

New in version 1.1.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.fr.forms.**FRRNAField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
RNA Stands for "Répertoire National des Associations"

It's under the authority of the French Minister of the Interior. See https://fr.wikipedia.org/wiki/R%C3%A9pertoire_national_des_associations for more information.

New in version 4.0.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.fr.forms.**FRRegion2016Select**(*attrs=None*)
A Select widget that uses a list of France's New Regions as its choices.

class localflavor.fr.forms.**FRRegionField**(***kwargs*)
A Select Field that uses a FRRegionSelect widget.

widget
alias of *FRRegionSelect*

class localflavor.fr.forms.**FRRegionSelect**(*attrs=None*)
A Select widget that uses a list of FR Regions as its choices.

class localflavor.fr.forms.**FRSIRENField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
SIREN stands for "Système d'identification du répertoire des entreprises".

It's under authority of the INSEE. See http://fr.wikipedia.org/wiki/Système_d'identification_du_répertoire_des_entreprises for more information.

New in version 1.1.

class localflavor.fr.forms.**FRSIRETField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
SIRET stands for "Système d'identification du répertoire des établissements".

It's under authority of the INSEE. See http://fr.wikipedia.org/wiki/Système_d'identification_du_répertoire_des_établissements for more information.

New in version 1.1.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.fr.forms.**FRZipCodeField**(***kwargs*)
Validate local French zip code.

The correct format is 'XXXXX'.

6.22.2 Data

localflavor.fr.fr_department.**DEPARTMENT_CHOICES_PER_REGION = (('01', 'Ain', '82'), ('02',**
See the "Code officiel géographique" on the INSEE website <www.insee.fr>.

localflavor.fr.fr_department.**DEPARTMENT_CHOICES = (('01', '01 - Ain'), ('02', '02 - Aisne')**
A list of departments

localflavor.fr.fr_region.**REGION_CHOICES = (('01', 'Guadeloupe'), ('02', 'Martinique'), ('0**
See the "Code officiel géographique" on the INSEE website <www.insee.fr>.

localflavor.fr.fr_region.**REGION_2016_CHOICES = (('01', 'Guadeloupe'), ('02', 'Martinique'),**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

6.23 Great Britain (gb)

6.23.1 Forms

GB-specific Form helpers.

class localflavor.gb.forms.**GBCountySelect**(*attrs=None*)
A Select widget that uses a list of UK Counties/Regions as its choices.

class localflavor.gb.forms.**GBNationSelect**(*attrs=None*)
A Select widget that uses a list of UK Nations as its choices.

class localflavor.gb.forms.**GBPostcodeField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field that validates its input is a UK postcode.

The regular expression used is sourced from the schema for British Standard BS7666 address types: https://data.gov.uk/education-standards/sites/default/files/CL-Address-Line-Type-v3-0.pdf

The value is uppercased and a space added in the correct place, if required.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.23.2 Data

This module contains various lists of regions and subdivisions in Great Britain.

Since subdivisions aren't clear this is supposed to be the most pragmatic collection of lists as possible. Your mileage may vary.

See https://github.com/django/django-localflavor/pull/43 for a long discussion about it.

Changed in version 1.1.

`localflavor.gb.gb_regions.`**`ENGLAND_REGION_CHOICES = (('Bath and North East Somerset', 'Bath`**
Metropolitan and non-metropolitan counties of England (not the ceremonial counties) Retrieved 10th Nov 2014 from http://en.wikipedia.org/wiki/Metropolitan_and_non-metropolitan_counties_of_England

`localflavor.gb.gb_regions.`**`GB_NATIONS_CHOICES = (('England', 'England'), ('Northern Ireland`**
Nations of the United Kingdom

`localflavor.gb.gb_regions.`**`GB_REGION_CHOICES = (('Bath and North East Somerset', 'Bath and`**
All regions of United Kingdom

`localflavor.gb.gb_regions.`**`NORTHERN_IRELAND_REGION_CHOICES = (('Antrim', 'Antrim'), ('Armagh`**
Counties of Northern Ireland (not the more recent, but less well-known, districts of Northern Ireland) Retrieved 10th Nov 2014 from http://en.wikipedia.org/wiki/Counties_of_Northern_Ireland

`localflavor.gb.gb_regions.`**`SCOTTISH_REGION_CHOICES = (('Aberdeen City', 'Aberdeen City'), (`**
Council areas of Scotland Retrieved 10th Nov 2014 from http://en.wikipedia.org/wiki/Subdivisions_of_Scotland#Council_areas

`localflavor.gb.gb_regions.`**`WALES_REGION_CHOICES = (('Blaenau Gwent', 'Blaenau Gwent'), ('Bri`**
Principal areas of Wales (not the preserved or historic counties) Retrieved 10th Nov 2014 from http://en.wikipedia.org/wiki/Local_government_in_Wales

6.24 Ghana (gh)

6.24.1 Forms

Ghana specific form helpers.

class `localflavor.gh.forms.`**`GHRegionSelect`**(*attrs=None*)
A Select widget with option to select a region from list of all regions of Ghana.

6.24.2 Data

`localflavor.gh.gh_regions.`**`REGIONS = [('ahafo', 'Ahafo'), ('ashanti', 'Ashanti'), ('bono_eas`**
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

6.25 Greece (gr)

6.25.1 Forms

Greek-specific forms helpers.

class localflavor.gr.forms.**GRPostalCodeField**(***kwargs*)
Greek Postal code field.

Format: XXXXX, where X is any digit, and first digit is not 0 or 9.

class localflavor.gr.forms.**GRSocialSecurityNumberCodeField**(*allow_test_value=False*, ***kwargs*)
Greek social security number (AMKA) field.

The allow_test_value option can be used to enable the usage of the non valid 00000000000 (11 zeros) value for testing and development

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.gr.forms.**GRTaxNumberCodeField**(*allow_test_value=False*, ***kwargs*)
Greek tax number field.

The allow_test_value option can be used to enable the usage of the non valid 000000000 value for testing and development

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.26 Croatia (hr)

6.26.1 Forms

HR-specific Form helpers.

class localflavor.hr.forms.**HRCountySelect**(*attrs=None*)
A Select widget that uses a list of counties of Croatia as its choices.

class localflavor.hr.forms.**HRJMBAGField**(**, max_length=None, min_length=None, strip=True, empty_value=''*, ***kwargs*)
Unique Master Academic Citizen Number of Croatia (JMBAG) field.

This number is used by college students and professors in Croatia.

http://www.cap.srce.hr/IzgledX.aspx

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.hr.forms.**HRJMBGField**(**, max_length=None, min_length=None, strip=True, empty_value=''*, ***kwargs*)
Unique Master Citizen Number (JMBG) field.

The number is still in use in Croatia, but it is being replaced by OIB.

Source: http://en.wikipedia.org/wiki/Unique_Master_Citizen_Number

For who might be reimplementing: The "area" regular expression group is used to calculate the region where a person was registered. Additional validation can be implemented in accordance with it, however this could result in exclusion of legit immigrated citizens. Therefore, this field works for any ex-Yugoslavia country.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.hr.forms.**HRLicensePlateField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
Vehicle license plate of Croatia field.

Normalizes to the specific format below. Suffix is constructed from the shared letters of the Croatian and English alphabets.

Format examples: SB 123-A (but also supports more characters) ZG 1234-AA

Used for standardized license plates only.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.hr.forms.**HRLicensePlatePrefixSelect**(*attrs=None*)
A Select widget that uses a list of vehicle license plate prefixes of Croatia as its choices.

class localflavor.hr.forms.**HROIBField**(*min_length=11*, *max_length=11*, ***kwargs*)
Personal Identification Number of Croatia (OIB) field.

http://www.oib.hr/

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.hr.forms.**HRPostalCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
Postal code of Croatia field.

It consists of exactly five digits ranging from 10000 to possibly less than 60000.

http://www.posta.hr/main.aspx?id=66

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.26.2 Data

localflavor.hr.hr_choices.**HR_COUNTY_CHOICES = (('GZG', 'Grad Zagreb'), ('BBŽ', 'Bjelovarsk**
Croatian Counties: http://en.wikipedia.org/wiki/ISO_3166-2:HR Croatia doesn't have official abbreviations for counties. The ones provided are in common use.

localflavor.hr.hr_choices.**HR_LICENSE_PLATE_PREFIX_CHOICES = (('BJ', 'BJ'), ('BM', 'BM'), (**
Only common license plate prefixes are provided. Special cases and obsolete prefixes are omitted. http://hr.wikipedia.org/wiki/Dodatak:Popis_registracijskih_oznaka_za_cestovna_vozila_u_Hrvatskoj

6.27 Hungary (hu)

6.27.1 Forms

HU-specific Form helpers

class localflavor.hu.forms.**HUCountySelect**(*attrs=None*)
A Select widget that uses a list of Hungarian Counties as its choices.

New in version 1.3.

6.27.2 Data

localflavor.hu.hu_counties.**HU_COUNTY_CHOICES = (('bacs_kiskun', 'Bács-Kiskun'), ('baranya',**
Hungarian counties: https://en.wikipedia.org/wiki/Counties_of_Hungary

6.28 Indonesia (id)

6.28.1 Forms

ID-specific Form helpers.

class localflavor.id_.forms.**IDLicensePlateField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
An Indonesian vehicle license plate field.

http://id.wikipedia.org/wiki/Tanda_Nomor_Kendaraan_Bermotor

Plus: "B 12345 12"

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.id_.forms.**IDLicensePlatePrefixSelect**(*attrs=None*)
A Select widget that uses a list of vehicle license plate prefix code of Indonesia as its choices.

http://id.wikipedia.org/wiki/Tanda_Nomor_Kendaraan_Bermotor

class localflavor.id_.forms.**IDNationalIdentityNumberField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
An Indonesian national identity number (NIK/KTP#) field.

http://id.wikipedia.org/wiki/Nomor_Induk_Kependudukan

xx.xxxx.ddmmyy.xxxx - 16 digits (excl. dots) notes: for women dd + 40

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.id_.forms.**IDPostCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

An Indonesian post code field.

http://id.wikipedia.org/wiki/Kode_pos

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.id_.forms.**IDProvinceSelect**(*attrs=None*)

A Select widget that uses a list of provinces of Indonesia as its choices.

6.28.2 Data

localflavor.id_.id_choices.**PROVINCE_CHOICES = (('ACE', 'Aceh'), ('BLI', 'Bali'), ('BTN', 'I**

Indonesia does not have an official Province code standard. I decided to use unambiguous and consistent (some are common) 3-letter codes. http://id.wikipedia.org/wiki/Daftar_provinsi_Indonesia

localflavor.id_.id_choices.**LICENSE_PLATE_PREFIX_CHOICES = (('A', 'Banten'), ('AA', 'Magelar**

License plate prefixes

6.29 Ireland (ie)

6.29.1 Forms

IE-specific Form helpers.

class localflavor.ie.forms.**EircodeField**(***kwargs*)

A form field that validates its input is a valid Eircode (Irish postcode).

The value is uppercased and has the internal space removed, if any.

See page 12 for the validation syntax: https://www.eircode.ie/docs/default-source/Common/prepareyourbusinessforeircode-edition3published.pdf?sfvrsn=2

New in version 2.2.

to_python(*value*)

Return a string.

class localflavor.ie.forms.**IECountySelect**(*attrs=None*)

A Select widget that uses a list of Irish Counties as its choices.

6.29.2 Data

localflavor.ie.ie_counties.**IE_COUNTY_CHOICES = (('carlow', 'Carlow'), ('cavan', 'Cavan'),**

Irish Counties: http://en.wikipedia.org/wiki/Counties_of_Ireland

6.30 Israel (il)

6.30.1 Forms

Israeli-specific form helpers.

class localflavor.il.forms.**ILIDNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field that validates its input as an Israeli identification number.

Valid form is per the Israeli ID specification.

Israeli ID numbers consist of up to 8 digits followed by a checksum digit. Numbers which are shorter than 8 digits are effectively left-zero-padded. The checksum digit is occasionally separated from the number by a hyphen, and is calculated using the luhn algorithm.

Relevant references (in Hewbrew):

http://he.wikipedia.org/wiki/%D7%9E%D7%A1%D7%A4%D7%A8_%D7%96%D7%94%D7%95%D7%AA_(%D7%99%D7%A9%D7%A8%D7%90%D7%9C) http://he.wikipedia.org/wiki/%D7%A1%D7%A4%D7%A8%D7%AA_%D7%91%D7%99%D7%A7%D7%95%D7%A8%D7%AA http://he.wikipedia.org/wiki/%D7%A7%D7%99%D7%93%D7%95%D7%9E%D7%AA_%D7%98%D7%9C%D7%A4%D7%95%D7%9F_%D7%91%D7%99%D7%A9%D7%A8%D7%90%D7%9C#.D7.A7.D7.99.D7.93.D7.95.D7.9E.D7.95.D7.AA_.D7.91.D7.99.D7.A9.D7.A8.D7.90.D7.9C_.D7.9C.D7.A4.D7.99_.D7.9E.D7.A4.D7.A2.D7.99.D7.9C.D7.99.D7.9D_.D7.95.D7.97.D7.9C.D7.95.D7.A7.D7.94_.D7.92.D7.90.D7.95.D7.92.D7.A8.D7.A4.D7.99.D7.AA

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.il.forms.**ILPostalCodeField**(***kwargs*)

A form field that validates its input as an Israeli postal code.

Valid form is XXXXX where X represents integer.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.31 India (in)

6.31.1 Forms

India-specific Form helpers.

class localflavor.in_.forms.**INAadhaarNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field for Aadhaar number issued by Unique Identification Authority of India (UIDAI).

Checks the following rules to determine whether the number is valid:

- Conforms to the XXXX XXXX XXXX format.
- No group consists entirely of zeroes.

Important information:

- Aadhaar number is a proof of identity but not of citizenship.
- Aadhaar number is issued to every resident of India including foreign citizens.
- Aadhaar number is not mandatory.

More information can be found at http://uidai.gov.in/what-is-aadhaar-number.html

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.in_.forms.**INPANCardNumberFormField**(***kwargs*)
A form field that accepts Indian Permanent account number(PAN) Card Number.

Rules:

1. It should be ten characters long.
2. The first three characters must be any upper case alphabets.
3. **The fourth character of PAN must be one of the following characters.** A — Association of persons (AOP) B — Body of individuals (BOI) C — Company F — Firm G — Government H — HUF (Hindu undivided family) L — Local authority J — Artificial juridical person P — Person (Individual) T — Trust (AOP)
4. **The fifth character is first letter of lastname of the PAN Card holder in case of Individual** or the first letter of first name in case of non-individual.
5. The next four-characters must be any number from 0000 to 9999.
6. The last(tenth) character which is a check-sum character must be any upper case alphabet.

Note:

1. **The validation of the fifth character must be done by the developer themselves,** as this validation is out of the scope of this project.
2. **The validation for the last digit (i.e check-sum character) is not available** in public domain, hence it is not implemented.

More Information at:

1. https://en.wikipedia.org/wiki/Permanent_account_number
2. https://www.incometaxindia.gov.in/tutorials/1.permanent%20account%20number%20(pan).pdf

New in version 4.0.

class localflavor.in_.forms.**INStateField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates its input is a Indian state name or abbreviation.

It normalizes the input to the standard two-letter vehicle registration abbreviation for the given state or union territory

Changed in version 1.1: Added Telangana to list of states. More details at https://en.wikipedia.org/wiki/Telangana#Bifurcation_of_Andhra_Pradesh

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.in_.forms.**INStateSelect**(*attrs=None*)
A Select widget that uses a list of Indian states/territories as its choices.

Changed in version 1.1: Added Telangana to list of states. More details at https://en.wikipedia.org/wiki/Telangana#Bifurcation_of_Andhra_Pradesh

Changed in version 3.1: Updated Indian states and union territories names and code as per iso 3166 (https://www.iso.org/obp/ui/#iso:code:3166:IN)

class localflavor.in_.forms.**INZipCodeField**(***kwargs*)
A form field that validates input as an Indian zip code, with the format XXXXXXX.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.31.2 Data

localflavor.in_.in_states.**STATE_CHOICES = (('KA', 'Karnataka'), ('AP', 'Andhra Pradesh'),**
A list of states

localflavor.in_.in_states.**STATES_NORMALIZED = {'an': 'AN', 'andaman and nicobar': 'AN',**
Normalized state names

6.32 Iran (ir)

6.32.1 Forms

Iranian-specific form helpers.

class localflavor.ir.forms.**IRIDNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates its input as an Iranian identification number.

Valid form is per the Iranian ID specification.

Persian documentation : http://www.aliarash.com/article/codemeli/codemeli.htm

There isn't good English documentation available for the Iranian identification number. Non-Persian speakers will need to use an online translation service to read this documentation.

New in version 2.2.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ir.forms.**IRPostalCodeField**(***kwargs*)
A form field that validates its input as an Iran postal code.

Valid form is XXXXXXXXXX where X represents integer.

Validate code:

- don't use 0 in first 5 digit
- don't use 2 in postal code
- First 4 digit is not the same
- The 5th digit cannot be 5
- all digits aren't the same

New in version 2.2.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ir.forms.**IRProvinceSelect**(*attrs=None*)
A Select widget that uses a list of Iran provinces cities as its choices.

New in version 2.2.

localflavor.ir.ir_provinces.**PROVINCE_CHOICES = (('AL', 'Alborz'), ('AR', 'Ardabil'), ('AE',**
A list of Iran provinces and autonomous cities as *choices* in a formfield. From https://en.wikipedia.org/wiki/Provinces_of_Iran

6.33 Iceland (is)

6.33.1 Forms

Iceland specific form helpers.

class localflavor.is_.forms.**ISIdNumberField**(*max_length=11*, *min_length=10*, ***kwargs*)
Icelandic identification number (kennitala).

This is a number every citizen of Iceland has.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.is_.forms.**ISPostalCodeSelect**(*attrs=None*)
A Select widget that uses a list of Icelandic postal codes as its choices.

6.33.2 Data

localflavor.is_.is_postalcodes.**IS_POSTALCODES = (('101', '101 Reykjavík'), ('102', '102 Rey**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

6.34 Italy (it)

6.34.1 Forms

IT-specific Form helpers.

class localflavor.it.forms.**ITProvinceSelect**(*attrs=None*)
A Select widget that uses a list of IT provinces as its choices.

class localflavor.it.forms.**ITRegionProvinceSelect**(*attrs=None*)
A Select widget that uses a named group list of IT regions mapped to regions as its choices.

class localflavor.it.forms.**ITRegionSelect**(*attrs=None*)
A Select widget that uses a list of IT regions as its choices.

class localflavor.it.forms.**ITSocialSecurityNumberField**(***kwargs*)
A form field that validates Italian Tax code (codice fiscale) for both persons and entities.

For reference see http://www.agenziaentrate.it/ and search for:

- 'Informazioni sulla codificazione delle persone fisiche' for persons' SSN
- 'Codice fiscale Modello AA5/6' for entities' SSN

Changed in version 1.1.

The `ITSocialSecurityNumberField` now also accepts SSN values for entities (numeric-only form).

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.it.forms.**ITVatNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates Italian VAT numbers (partita IVA).

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.it.forms.**ITZipCodeField**(***kwargs*)
A form field that validates input as an Italian zip code.

Valid codes must have five digits.

6.34.2 Utilities

localflavor.it.util.**ssn_check_digit**(*value*)
Calculate Italian social security number check digit.

localflavor.it.util.**ssn_validation**(*ssn_value*)
Validate Italian SSN for persons

`ValueError` is raised if validation fails.

localflavor.it.util.**vat_number_check_digit**(*vat_number*)
Calculate Italian VAT number check digit.

localflavor.it.util.**vat_number_validation**(*vat_number*)
Validate Italian VAT number. Used also for entities SSN validation.

`ValueError` is raised if validation fails.

6.34.3 Data

localflavor.it.it_province.**PROVINCE_CHOICES = (('AG', 'Agrigento'), ('AL', 'Alessandria'),**
An alphabetical list of provinces

localflavor.it.it_province.**PROVINCE_REGIONS = {'AG': 'SIC', 'AL': 'PMN', 'AN': 'MAR', 'AO'**
A dictionary of provinces mapped to regions

New in version 1.1.

localflavor.it.it_region.**REGION_CHOICES = (('ABR', 'Abruzzo'), ('BAS', 'Basilicata'), ('CA**
An alphabetical list of regions

localflavor.it.it_region.**REGION_PROVINCES = {'ABR': ['AQ', 'CH', 'PE', 'TE'], 'BAS': ['MT',**
A dictionary of regions mapped to provinces

New in version 1.1.

localflavor.it.it_region.**REGION_PROVINCE_CHOICES = [('Abruzzo', (('CH', 'Chieti'), ('AQ',**
A alphabetical list of provinces mapped to regions

New in version 1.1.

6.35 Japan (jp)

6.35.1 Forms

JP-specific Form helpers.

class localflavor.jp.forms.**JPPostalCodeField**(***kwargs*)
A form field that validates its input is a Japanese postcode.

Accepts 7 digits, with or without a hyphen.

clean(*value*)
Validates the input and returns a string that contains only numbers.

class localflavor.jp.forms.**JPPrefectureCodeSelect**(*attrs=None*)
A Select widget for Japanese prefecture codes.

It uses a list of Japanese prefectures as its choices and the prefectures code as the post value.

class localflavor.jp.forms.**JPPrefectureSelect**(*attrs=None*)
A Select widget that uses a list of Japanese prefectures as its choices.

6.35.2 Data

localflavor.jp.jp_prefectures.**JP_PREFECTURES = (('hokkaido', 'Hokkaido'), ('aomori', 'Aomo**
A list of prefectures

6.36 Kuwait (kw)

6.36.1 Forms

Kuwait-specific Form helpers.

class localflavor.kw.forms.**KWAreaSelect**(*attrs=None*)
A Select widget that uses a list of Kuwait areas as its choices.

New in version 1.6.

class localflavor.kw.forms.**KWCivilIDNumberField**(*max_length=12*, *min_length=12*, ***kwargs*)
Kuwaiti Civil ID numbers are 12 digits, second to seventh digits represents the person's birthdate.

Checks the following rules to determine the validity of the number:

- The number consist of 12 digits.
- The birthdate of the person is a valid date.
- The calculated checksum equals to the last digit of the Civil ID.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.kw.forms.**KWGovernorateSelect**(*attrs=None*)
A Select widget that uses a list of Kuwait governorates as its choices.

6.37 Lithuania (lt)

6.37.1 Forms

class localflavor.lt.forms.**LTCountySelect**(*attrs=None*)
A select field with the Lithuanian counties as choices

class localflavor.lt.forms.**LTIDCodeField**(***kwargs*)
A form field that validates as Lithuanian ID Code.

Checks:

- Made of exactly 11 decimal numbers.
- Checksum is correct.
- ID contains valid date.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

valid_date(*value*)
Check if date in ID code is valid.

We won't check for dates in future as it would become too restrictive.

class localflavor.lt.forms.**LTMunicipalitySelect**(*attrs=None*)
A select field with the Lithuanian municipalities as choices

class localflavor.lt.forms.**LTPostalCodeField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field that validates and normalizes Lithuanian postal codes.

Lithuanian postal codes in following forms accepted:

- XXXXX
- LT-XXXXX

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.37.2 Data

localflavor.lt.lt_choices.**COUNTY_CHOICES = (('alytus', 'Alytus'), ('kaunas', 'Kaunas'), ('**
Alphabetically sorted list of Lithuanian counties.

localflavor.lt.lt_choices.**MUNICIPALITY_CHOICES = (('akmene', 'Akmenė district'), ('alytus_**
Alphabetically sorted lists of Lithuanian municipalities.

6.38 Latvia (lv)

New in version 1.1.

6.38.1 Forms

class localflavor.lv.forms.**LVMunicipalitySelect**(*attrs=None*)
A select field of Latvian municipalities.

class localflavor.lv.forms.**LVPersonalCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates input as a Latvian personal code.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

static lv_checksum(*value*)
Takes a string of 10 digits as input, returns check digit.

class localflavor.lv.forms.**LVPostalCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
A form field that validates and normalizes Latvian postal codes.

Latvian postal codes in following forms accepted:

- XXXX
- LV-XXXX

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.38.2 Data

localflavor.lv.lv_choices.**MUNICIPALITY_CHOICES = (('DGV', 'Daugavpils'), ('JEL', 'Jelgava')**
A list of Latvian municipalities and republican cities. Identifiers based on ISO 3166-2:LV. https://en.wikipedia.org/wiki/ISO_3166-2:LV

6.39 Moldova (md)

6.39.1 Forms

class localflavor.md.forms.**MDCompanyTypesSelect**(*attrs=None*)
A Select widget that uses a list of Moldavian company types as its choices.

New in version 2.1.

class localflavor.md.forms.**MDIDNOField**(***kwargs*)
A form field for the Moldavian company identification number (IDNO).

New in version 2.1.

class localflavor.md.forms.**MDLicensePlateField**(***kwargs*)
A form field for the Moldavian license plate number.

New in version 2.1.

class localflavor.md.forms.**MDRegionSelect**(*attrs=None*)
A Select widget that uses a list of Moldavian regions as its choices.

New in version 2.1.

6.39.2 Models

class localflavor.md.models.**MDCompanyTypeField**(**args*, ***kwargs*)
A model field for the Moldavian company type abbreviation.

New in version 2.1.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.md.models.**MDIDNOField**(**args*, ***kwargs*)
A model field for the Moldavian company identification number (IDNO).

New in version 2.1.

class localflavor.md.models.**MDLicensePlateField**(**args*, ***kwargs*)
A model field for the Moldavian license plate number.

New in version 2.1.

6.39.3 Validators

class localflavor.md.validators.**MDIDNOFieldValidator**(*regex=None*, *message=None*, *code=None*, *inverse_match=None*, *flags=None*)

Validation for Moldavian IDNO.

New in version 2.1.

class localflavor.md.validators.**MDLicensePlateValidator**(*regex=None*, *message=None*, *code=None*, *inverse_match=None*, *flags=None*)

Validation for Moldavian License Plates.

New in version 2.1.

6.39.4 Data

localflavor.md.choices.**COMPANY_TYPES_CHOICES = (('II', 'Întreprindere Individuală'), ('SA',**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

localflavor.md.choices.**LICENSE_PLATE_DIPLOMATIC = (('CD', 'Diplomatic Corps'), ('TS', 'Serv**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

localflavor.md.choices.**LICENSE_PLATE_GOVERNMENT_TYPE = (('P', 'Parliament'), ('G', 'Governm**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

localflavor.md.choices.**LICENSE_PLATE_POLICE = (('MAI', 'Ministry of Internal Affairs'), ('M**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

6.40 Macedonia (mk)

6.40.1 Forms

class localflavor.mk.forms.**MKIdentityCardNumberField**(***kwargs*)

A Macedonian ID card number.

Accepts both old and new format.

class localflavor.mk.forms.**MKMunicipalitySelect**(*attrs=None*)
A form `Select` widget that uses a list of Macedonian municipalities as choices.

The label is the name of the municipality and the value is a 2 character code for the municipality.

class localflavor.mk.forms.**UMCNField**(***kwargs*)
A form field that validates input as a unique master citizen number.

The format of the unique master citizen number has been kept the same from Yugoslavia. It is still in use in other countries as well, it is not applicable solely in Macedonia. For more information see: https://secure.wikimedia.org/wikipedia/en/wiki/Unique_Master_Citizen_Number

A value will pass validation if it complies to the following rules:

- Consists of exactly 13 digits
- The first 7 digits represent a valid past date in the format DDMMYYY
- The last digit of the UMCN passes a checksum test

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.40.2 Models

class localflavor.mk.models.**MKIdentityCardNumberField**(**args*, ***kwargs*)
A form field that validates input as a Macedonian identity card number.

Both old and new identity card numbers are supported.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.mk.models.**MKMunicipalityField**(**args*, ***kwargs*)
A form field that validates input as a Macedonian identity card number.

Both old and new identity card numbers are supported.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.mk.models.**UMCNField**(**args*, ***kwargs*)
A form field that validates input as a unique master citizen number.

The format of the unique master citizen number is not unique to Macedonia. For more information see: https://secure.wikimedia.org/wikipedia/en/wiki/Unique_Master_Citizen_Number

A value will pass validation if it complies to the following rules:

- Consists of exactly 13 digits
- The first 7 digits represent a valid past date in the format DDMMYYY
- The last digit of the UMCN passes a checksum test

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

6.40.3 Data

localflavor.mk.mk_choices.**MK_MUNICIPALITIES = (('AD', 'Aerodrom'), ('AR', 'Aračinovo'), ('**
Macedonian municipalities per the reorganization from 2004.

6.41 Malta (mt)

New in version 1.1.

6.41.1 Forms

Maltese-specific Form helpers.

6.42 Mexico (mx)

6.42.1 Forms

Mexican-specific form helpers.

localflavor.mx.forms.**CURP_INCONVENIENT_WORDS = ['BACA', 'BAKA', 'BUEI', 'BUEY', 'CACA', 'C**
This is the list of inconvenient words according to the *Anexo 2* of the document described in the next link: http://portal.veracruz.gob.mx/pls/portal/url/ITEM/444112558A57C6E0E040A8C02E00695C

class localflavor.mx.forms.**MXCLABEField**(*min_length=18*, *max_length=18*, ***kwargs*)
This field validates a CLABE (Clave Bancaria Estandarizada).

A CLABE is a 18-digits long number. The first 6 digits denote bank and branch number. The remaining 12 digits denote an account number, plus a verifying digit.

More info: https://en.wikipedia.org/wiki/CLABE

New in version 1.4.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.mx.forms.**MXCURPField**(*min_length=18*, *max_length=18*, ***kwargs*)
A field that validates a Mexican Clave Única de Registro de Población.

The CURP is integrated by a juxtaposition of characters following the next pattern:

Index	Format	Accepted Characters
1	X	Any letter
2	X	Any vowel
3-4	XX	Any letter
5-10	YYMMDD	Any valid date
11	X	Either *H* or *M*, depending on the person's gender
12-13	XX	Any valid acronym for a state in Mexico
14-16	XXX	Any consonant
17	X	Any number between 0 and 9 or any letter
18	X	Any number between 0 and 9

More info about this: http://www.condusef.gob.mx/index.php/clave-unica-de-registro-de-poblacion-curp

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.mx.forms.**MXRFCField**(*min_length=12*, *max_length=13*, ***kwargs*)
A form field that validates a Mexican *Registro Federal de Contribuyentes*.

Validates either *Persona física* or *Persona moral*. The Persona física RFC string is integrated by a juxtaposition of characters following the next pattern:

Index	Format	Accepted Characters
1	X	Any letter
2	X	Any vowel
3-4	XX	Any letter
5-10	YYMMDD	Any valid date
11-12	XX	Any letter or number between 0 and 9
13	X	Any digit between 0 and 9 or the letter *A*

The Persona moral RFC string is integrated by a juxtaposition of characters following the next pattern:

Index	Format	Accepted Characters
1-3	XXX	Any letter including *&* and *Ñ* chars
4-9	YYMMDD	Any valid date
10-11	XX	Any letter or number between 0 and 9
12	X	Any number between 0 and 9 or the letter *A*

More info about this: http://es.wikipedia.org/wiki/Registro_Federal_de_Contribuyentes_(M%C3%A9xico)

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.mx.forms.**MXSocialSecurityNumberField**(*min_length=11*, *max_length=11*, ***kwargs*)
A field that validates a Mexican Social Security Number.

The Social Security Number is integrated by a juxtaposition of digits following the next pattern:

Index	Required numbers
1-2	The number of the branch office where the Social Security Number was designated.
3-4	The year of inscription to the Social Security.
5-6	The year of birth of the Social Security Number owner.
7-10	The progressive number of procedure for the IMSS. (This digit is provided exclusively by the Institute as it regards the Folio number of such procedure).
11	The verification digit.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.mx.forms.**MXStateSelect**(*attrs=None*)
A Select widget that uses a list of Mexican states as its choices.

class localflavor.mx.forms.**MXZipCodeField**(***kwargs*)
A form field that accepts a Mexican Zip Code.

More info about this: http://en.wikipedia.org/wiki/List_of_postal_codes_in_Mexico

localflavor.mx.forms.**RFC_INCONVENIENT_WORDS = ['BUEI', 'BUEY', 'CACA', 'CACO', 'CAGA', 'CA**
This is the list of inconvenient words according to the *Anexo IV* of the document described in the next link: http://www.sisi.org.mx/jspsi/documentos/2005/seguimiento/06101/0610100162005_065.doc

6.42.2 Models

class localflavor.mx.models.**MXCLABEField**(**args*, ***kwargs*)
A model field that forms represent as a forms.MXCURPField field and stores the value of a valid Mexican CLABE.

New in version 1.4.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.mx.models.**MXCURPField**(**args*, ***kwargs*)
A model field that forms represent as a forms.MXCURPField field and stores the value of a valid Mexican CURP.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.mx.models.**MXRFCField**(**args*, ***kwargs*)
A model field that forms represent as a forms.MXRFCField field and stores the value of a valid Mexican RFC.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.mx.models.**MXSocialSecurityNumberField**(**args*, ***kwargs*)
A model field that forms represent as a forms.MXSocialSecurityNumberField field.

It stores the value of a valid Mexican Social Security Number.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.mx.models.**MXStateField**(**args*, ***kwargs*)
A model field that stores the three or four letter Mexican state abbreviation in the database.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.mx.models.**MXZipCodeField**(**args*, ***kwargs*)
A model field that forms represent as a forms.MXZipCodeField field and stores the five-digit Mexican zip code.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

6.42.3 Data

localflavor.mx.mx_states.**STATE_CHOICES = (('AGU', 'Aguascalientes'), ('BCN', 'Baja Califor**
All 31 states, plus the *Ciudad de México*.

6.43 Malaysia (my)

6.43.1 Forms

class localflavor.my.forms.**MyKadFormField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field that validates input as a Malaysia MyKad number.

Conforms to the YYMMDD-PB-###G or YYMMDDPB###G format More info: https://en.wikipedia.org/wiki/Malaysian_identity_card

New in version 3.0.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

to_python(*value*)

Return a string.

6.44 The Netherlands (nl)

6.44.1 Forms

NL-specific Form helpers.

class localflavor.nl.forms.**NLBSNFormField**(***kwargs*)

A Dutch social security number (BSN) field.

https://nl.wikipedia.org/wiki/Burgerservicenummer

Note that you may only process the BSN if you have a legal basis to do so!

New in version 1.6.

class localflavor.nl.forms.**NLLicensePlateFormField**(***kwargs*)

A Dutch license plate field.

https://www.rdw.nl/ https://nl.wikipedia.org/wiki/Nederlands_kenteken

New in version 2.1.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.nl.forms.**NLProvinceSelect**(*attrs=None*)

A Select widget that uses a list of provinces of the Netherlands as it's choices.

class localflavor.nl.forms.**NLZipCodeField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A Dutch zip code field.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.44.2 Models

class localflavor.nl.models.**NLBSNField**(**args*, ***kwargs*)
A Dutch social security number (BSN).

This model field uses `validators.NLBSNFieldValidator` for validation.

New in version 1.6.

description = 'Dutch social security number (BSN)'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

validators = [<localflavor.nl.validators.NLBSNFieldValidator object>]

class localflavor.nl.models.**NLLicensePlateField**(**args*, ***kwargs*)
A Dutch license plate.

This model field uses `validators.NLLicensePlateFieldValidator` for validation.

New in version 2.1.

default_form_field
alias of *localflavor.nl.forms.NLLicensePlateFormField*

description = 'Dutch license plate'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

validators = [<localflavor.nl.validators.NLLicensePlateFieldValidator object>]

class localflavor.nl.models.**NLProvinceField**(**args*, ***kwargs*)
A Dutch Province field.

New in version 1.3.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

description = 'Dutch province'

class localflavor.nl.models.**NLZipCodeField**(**args*, ***kwargs*)

A Dutch zip code model field.

This model field uses `validators.NLZipCodeFieldValidator` for validation.

New in version 1.3.

description = 'Dutch zipcode'

formfield(***kwargs*)

Return a django.forms.Field instance for this field.

to_python(*value*)

Convert the input value into the expected Python data type, raising django.core.exceptions.ValidationError if the data can't be converted. Return the converted value. Subclasses should override this.

validators = [<localflavor.nl.validators.NLZipCodeFieldValidator object>]

6.44.3 Data

localflavor.nl.nl_provinces.**PROVINCE_CHOICES = (('DR', 'Drenthe'), ('FL', 'Flevoland'), ('**

An alphabetical list of provinces

6.45 Norway (no)

6.45.1 Forms

Norwegian-specific Form helpers.

class localflavor.no.forms.**NOBankAccountNumber**(**, max_length=None, min_length=None, strip=True, empty_value=''*, ***kwargs*)

A form field for Norwegian bank account numbers.

Performs MOD11 with the custom weights for the Norwegian bank account numbers, including a check for a remainder of 0, in which event the checksum is also 0.

Usually their string representation is along the lines of ZZZZ.YY.XXXXX, where the last X is the check digit. They're always a total of 11 digits long, with 10 out of these 11 being the actual account number itself.

- Accepts, and strips, account numbers with extra spaces.
- Accepts, and strips, account numbers provided in form of XXXX.YY.XXXXX.

Note: No consideration is taking for banking clearing numbers as of yet, seeing as these are only used between banks themselves.

New in version 1.5.

to_python(*value*)

Return a string.

class localflavor.no.forms.**NOMunicipalitySelect**(*attrs=None*)
A Select widget that uses a list of Norwegian municipalities (fylker) as its choices.

class localflavor.no.forms.**NOSocialSecurityNumber**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)
Algorithm is documented at http://no.wikipedia.org/wiki/Personnummer.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.no.forms.**NOZipCodeField**(***kwargs*)
A form field that validates input as a Norwegian zip code.

Valid codes have four digits.

6.45.2 Data

localflavor.no.no_municipalities.**MUNICIPALITY_CHOICES = (('akershus', 'Akershus'), ('austag**
An alphabetical list of Norwegian municipalities (fylker) for use as *choices* in a formfield.

6.46 Nepal (np)

6.46.1 Forms

Nepal specific form helpers.

class localflavor.np.forms.**NPDistrictSelect**(*attrs=None*)
A Select widget with option to select a district from list of all districts of Nepal.

class localflavor.np.forms.**NPPostalCodeFormField**(***kwargs*)
A form field that accepts Nepali postal code. Format : XXXXX

Postal codes: https://en.wikipedia.org/wiki/List_of_postal_codes_in_Nepal

class localflavor.np.forms.**NPProvinceSelect**(*attrs=None*)
A Select widget with option to select a province from list of all provinces of Nepal.

class localflavor.np.forms.**NPZoneSelect**(*attrs=None*)
A Select widget with option to select a zone from list of all zones of Nepal.

6.46.2 Data

localflavor.np.np_provinces.**PROVINCES = [('bagmati', 'Bagmati'), ('gandaki', 'Gandaki'), (**
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

localflavor.np.np_zones.**ZONES = [('bagmati', 'Bagmati'), ('bheri', 'Bheri'), ('dhawalagiri**
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

localflavor.np.np_districts.**DISTRICTS = [('achham', 'Achham'), ('arghakhanchi', 'Arghakhanc**
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

6.47 New Zealand (nz)

New in version 1.1.

6.47.1 Forms

New Zealand specific form helpers.

class localflavor.nz.forms.**NZBankAccountNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field that validates its input as New Zealand bank account number.

Formats:

XX-XXXX-XXXXXXX-XX

XX-XXXX-XXXXXXX-XXX

Where:

- the first two digits is the bank ID
- the next four digits are the branch number where the account was opened
- the next 7 digits are the account numbers
- the last two or three digits define type of the account.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.nz.forms.**NZNorthIslandCouncilSelect**(*attrs=None*)

A select widget with list of New Zealand North Island city and district councils as its choices.

class localflavor.nz.forms.**NZPostCodeField**(***kwargs*)

A form field that validates its input as New Zealand postal code.

class localflavor.nz.forms.**NZProvinceSelect**(*attrs=None*)

A select widget with list of New Zealand provinces as its choices.

class localflavor.nz.forms.**NZRegionSelect**(*attrs=None*)

A select widget with list of New Zealand regions as its choices.

class localflavor.nz.forms.**NZSouthIslandCouncilSelect**(*attrs=None*)

A select widget with list of New Zealand South Island city and district councils as its choices.

6.47.2 Data

localflavor.nz.nz_regions.**REGION_CHOICES = (('NZ-NTL', 'Northland'), ('NZ-AUK', 'Auckland'**

A list of regions

localflavor.nz.nz_provinces.**PROVINCE_CHOICES = (('Auckland', 'Auckland'), ('Taranaki', 'Ta**

A list of provinces (abolished in 1876, use regions instead)

localflavor.nz.nz_councils.**NORTH_ISLAND_COUNCIL_CHOICES = (('Far North', 'Far North Distric**

A list of North Island city and district councils

localflavor.nz.nz_councils.**SOUTH_ISLAND_COUNCIL_CHOICES = (('Tasman', 'Tasman District'),**

A list of South Island city and district councils

6.48 Peru (pe)

6.48.1 Forms

PE-specific Form helpers.

class localflavor.pe.forms.**PEDNIField**(*max_length=8*, *min_length=8*, ***kwargs*)
A field that validates Documento Nacional de Identidad (DNI) numbers.

clean(*value*)
Value must be a string in the XXXXXXXX formats.

class localflavor.pe.forms.**PERUCField**(*max_length=11*, *min_length=11*, ***kwargs*)
This field validates a RUC (Registro Unico de Contribuyentes).

A RUC is of the form XXXXXXXXXXX.

clean(*value*)
Value must be an 11-digit number.

class localflavor.pe.forms.**PERegionSelect**(*attrs=None*)
A Select widget that uses a list of Peruvian Regions as its choices.

6.48.2 Data

localflavor.pe.pe_region.**REGION_CHOICES = (('AMA', 'Amazonas'), ('ANC', 'Ancash'), ('APU',**
A list of Peru regions as *choices* in a formfield.

6.49 Pakistan (pk)

6.49.1 Forms

Pakistani-specific Form helpers.

class localflavor.pk.forms.**PKPostCodeField**(***kwargs*)
Pakistani post code field.

Assumed to be 5 digits.

class localflavor.pk.forms.**PKStateSelect**(*attrs=None*)
A Select widget that uses a list of Pakistani states/territories as its choices.

6.49.2 Models

class localflavor.pk.models.**PKPostCodeField**(**args*, ***kwargs*)
A model field that stores the five-digit Pakistani postcode in the database

Forms represent it as a *PKPostCodeField* field.

description = 'Pakistani Postcode'

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.pk.models.**PKStateField**(**args*, ***kwargs*)
A model field that stores the five-letter Pakistani state abbreviation in the database.

It is represented with `STATE_CHOICES` choices.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

description = 'Pakistani State'

6.49.3 Data

localflavor.pk.pk_states.**STATE_CHOICES = (('PK-JK', 'Azad Jammu & Kashmir'), ('PK-BA', 'Ba**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

6.50 Poland (pl)

6.50.1 Forms

Polish-specific form helpers.

class localflavor.pl.forms.**PLCountySelect**(*attrs=None*)
A select widget with list of Polish administrative units as choices.

class localflavor.pl.forms.**PLNIPField**(***kwargs*)
A form field that validates as Polish Tax Number (NIP).

Valid forms are: XXX-YYY-YY-YY, XXX-YY-YY-YYY or XXXYYYYYYY. Checksum algorithm based on documentation at http://wipos.p.lodz.pl/zylla/ut/nip-rego.html

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

has_valid_checksum(*number*)
Calculates a checksum with the provided algorithm.

class localflavor.pl.forms.**PLNationalIDCardNumberField**(***kwargs*)
A form field that validates as Polish National ID Card Number.

Checks the following rules:

- the length consist of 3 letter and 6 digits
- has a valid checksum

The algorithm is documented at http://en.wikipedia.org/wiki/Polish_identity_card.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

has_valid_checksum(*number*)
Calculates a checksum with the provided algorithm.

class localflavor.pl.forms.**PLPESELField**(***kwargs*)
A form field that validates as Polish Identification Number (PESEL).

Checks the following rules:

- the length consist of 11 digits
- has a valid checksum
- contains a valid birth date

The algorithm is documented at http://en.wikipedia.org/wiki/PESEL.

Changed in version 1.4.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

has_valid_birth_date(*number*)
Checks whether the birth date encoded in PESEL is valid.

has_valid_checksum(*number*)
Calculates a checksum with the provided algorithm.

class localflavor.pl.forms.**PLPostalCodeField**(***kwargs*)
A form field that validates as Polish postal code.

Valid code is XX-XXX where X is digit.

class localflavor.pl.forms.**PLProvinceSelect**(*attrs=None*)
A select widget with list of Polish administrative provinces as choices.

class localflavor.pl.forms.**PLREGONField**(***kwargs*)
A form field that validates its input is a REGON number.

Valid regon number consists of 9 or 14 digits. See http://www.stat.gov.pl/bip/regon_ENG_HTML.htm for more information.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

has_valid_checksum(*number*)
Calculates a checksum with the provided algorithm.

6.50.2 Data

localflavor.pl.pl_administrativeunits.**ADMINISTRATIVE_UNIT_CHOICES = (('wroclaw', 'Wrocław')**

(locally NISS - 'Número de Identificação na Segurança Social'). - Social Security numbers must be in the format XYYYYYYYYYY (where X is either 1 or 2 and Y is any other digit).

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.pt.forms.**PTZipCodeField**(***kwargs*)
A field which validates Portuguese zip codes.

NOTE - Zip codes have the format XYYY-YYY (where X is a digit between 1 and 9 and Y is any other digit).

6.51.2 Data

localflavor.pt.pt_regions.**REGION_CHOICES = (('01', 'Aveiro'), ('02', 'Beja'), ('03', 'Braga**
A tuple representing Portuguese regions (as per ISO3166:2-PT).

6.52 Paraguay (py)

6.52.1 Forms

PY-specific Form helpers.

class localflavor.py_.forms.**PyDepartmentSelect**(*attrs=None*)
A Select widget with a list of Paraguayan departments as choices.

class localflavor.py_.forms.**PyNumberedDepartmentSelect**(*attrs=None*)
A Select widget with a roman numbered list of Paraguayan departments as choices.

6.52.2 Data

localflavor.py_.py_department.**DEPARTMENT_CHOICES = (('AG', 'Alto Paraguay'), ('AA', 'Alto**
http://www.statoids.com/upy.html

localflavor.py_.py_department.**DEPARTMENT_ROMAN_CHOICES = (('CN', 'I Concepción'), ('SP', '**
list of departments sorted by its roman number

6.53 Romania (ro)

6.53.1 Forms

Romanian specific form helpers.

class localflavor.ro.forms.**ROCIFField**(*max_length=10*, *min_length=2*, ***kwargs*)
A Romanian fiscal identity code (CIF) field.

For CIF validation algorithm see: https://ro.wikipedia.org/wiki/Cod_de_Identificare_Fiscal%C4%83

clean(*value*)
CIF validation.

Args: value: the CIF code

class localflavor.ro.forms.**ROCNPField**(*max_length=13*, *min_length=13*, ***kwargs*)
A Romanian personal identity code (CNP) field.

For CNP validation algorithm see: https://ro.wikipedia.org/wiki/Cod_numeric_personal

clean(*value*)
CNP validations.

Args: value: the CNP code

class localflavor.ro.forms.**ROCountyField**(**, max_length=None, min_length=None, strip=True, empty_value=”, **kwargs*)
A form field that validates its input is a Romanian county name or abbreviation.

It normalizes the input to the standard vehicle registration abbreviation for the given county.

WARNING: This field will only accept names written with diacritics (using comma bellow for ș and ț); consider using ROCountySelect if this behavior is unacceptable for you

For more information regarding diacritics see *Comma-below (ș and ț) versus cedilla (ş and ţ)* and *Unicode and HTML* sections from: Romanian alphabet.

Example:

Argeș => valid (comma bellow)

Argeş => invalid (cedilla)

Arges => invalid (no diacritic)

clean(*value*)
Validate the given value and return its “cleaned” value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.ro.forms.**ROCountySelect**(*attrs=None*)
A Select widget that uses a list of Romanian counties (județe) as its choices.

class localflavor.ro.forms.**ROPostalCodeField**(*max_length=6*, *min_length=6*, ***kwargs*)
Romanian postal code field.

6.53.2 Data

localflavor.ro.ro_counties.**COUNTIES_CHOICES = (('AB', 'Alba'), ('AR', 'Arad'), ('AG', 'Arge**
A list of Romanian counties as *choices* in a formfield.

6.54 Russia (ru)

6.54.1 Forms

Russian-specific forms helpers.

class localflavor.ru.forms.**RUAlienPassportNumberField**(***kwargs*)
Russian alien’s passport number format.

XX XXXXXXX where X - any digit.

class localflavor.ru.forms.**RUCountySelect**(*attrs=None*)
A Select widget that uses a list of Russian Counties as its choices.

class localflavor.ru.forms.**RUPassportNumberField**(***kwargs*)
Russian internal passport number format.

XXXX XXXXXX where X - any digit.

class localflavor.ru.forms.**RUPostalCodeField**(***kwargs*)
Russian Postal code field.

Format: XXXXXX, where X is any digit, and first digit is not zero.

class localflavor.ru.forms.**RURegionSelect**(*attrs=None*)
A Select widget that uses a list of Russian Regions as its choices.

6.54.2 Data

localflavor.ru.ru_regions.**RU_COUNTY_CHOICES = (('Central Federal County', 'Central Federal**
http://ru.wikipedia.org/wiki/___

localflavor.ru.ru_regions.**RU_REGIONS_CHOICES = (('77', 'Moskva'), ('78', 'Saint-Peterburg'**
http://ru.wikipedia.org/wiki/___

6.55 Sweden (se)

6.55.1 Forms

Swedish specific Form helpers.

class localflavor.se.forms.**SECountySelect**(*attrs=None*)
A Select form widget that uses a list of the Swedish counties (län) as its choices.

The cleaned value is the official county code – see http://en.wikipedia.org/wiki/Counties_of_Sweden for a list.

class localflavor.se.forms.**SEOrganisationNumberField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field that validates input as a Swedish organisation number (organisationsnummer).

It accepts the same input as SEPersonalIdentityField (for sole proprietorships (enskild firma). However, coordination numbers are not accepted.

It also accepts ordinary Swedish organisation numbers with the format NNNNNNNNNN.

The return value will be YYYYMMDDXXXX for sole proprietors, and NNNNNNNNNN for other organisations.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.se.forms.**SEPersonalIdentityNumberField**(*coordination_number=True, interim_number=False, **kwargs*)
A form field that validates input as a Swedish personal identity number (personnummer).

The correct formats are YYYYMMDD-XXXX, YYYYMMDDXXXX, YYMMDD-XXXX, YYMMDDXXXX and YYMMDD+XXXX.

A + indicates that the person is older than 100 years, which will be taken into consideration when the date is validated.

The checksum will be calculated and checked. The birth date is checked to be a valid date.

By default, co-ordination numbers (samordningsnummer) will be accepted. To only allow real personal identity numbers, pass the keyword argument coordination_number=False to the constructor.

Interim numbers (interimspersonnummer), used by educational institutions within the Ladok system, are supported but not accepted by default, since they are not considered valid outside Ladok. They have the same format and semantics as real personal identity numbers, except that the first control digit is replaced by a letter (A-Z). To allow the use of interim numbers, pass the keyword argument interim_numbers=True to the constructor.

The cleaned value will always have the format YYYYMMDDXXXX.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.se.forms.**SEPostalCodeField**(***kwargs*)
A form field that validates input as a Swedish postal code (postnummer).

Valid codes consist of five digits (XXXXX). The number can optionally be formatted with a space after the third digit (XXX XX).

The cleaned value will never contain the space.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.55.2 Utilities

localflavor.se.utils.**id_number_checksum**(*gd*)
Calculates a Swedish ID number checksum, using the "Luhn"-algoritm.

localflavor.se.utils.**validate_id_birthday**(*gd*, *fix_coordination_number_day=True*)
Validates the birth_day and returns the datetime.date object for the birth_day.

If the date is an invalid birth day, a ValueError will be raised.

6.55.3 Data

localflavor.se.se_counties.**COUNTY_CHOICES = (('AB', 'Stockholm'), ('AC', 'Västerbotten'),**
An alphabetical list of Swedish counties, sorted by codes. http://en.wikipedia.org/wiki/Counties_of_Sweden

6.56 Singapore (sg)

New in version 1.1.

6.56.1 Forms

Singapore-specific Form helpers.

6.57 Slovenia (si)

6.57.1 Forms

Slovenian specific form helpers.

class localflavor.si.forms.**SIEMSOField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form for validating Slovenian personal identification number.

Additionally stores gender, nationality and birthday to self.info dictionary.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.si.forms.**SIPostalCodeField**(***kwargs*)

Slovenian post codes field.

class localflavor.si.forms.**SIPostalCodeSelect**(*attrs=None*)

A Select widget that uses Slovenian postal codes as its choices.

class localflavor.si.forms.**SITaxNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

Slovenian tax number field.

Valid input is SIXXXXXXXX or XXXXXXXX where X is a number.

http://zylla.wipos.p.lodz.pl/ut/translation.html#PZSI

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

6.57.2 Data

localflavor.si.si_postalcodes.**SI_POSTALCODES = [(1000, 'Ljubljana'), (1215, 'Medvode'), (12**

class localflavor.sk.forms.**SKRegionSelect**(*attrs=None*)
A select widget widget with list of Slovak regions as choices.

6.58.2 Data

localflavor.sk.sk_districts.**DISTRICT_CHOICES = (('BB', 'Banska Bystrica'), ('BS', 'Banska S**
http://sk.wikipedia.org/wiki/Administrat%C3%ADvne_%C4%8Dlenenie_Slovenska

localflavor.sk.sk_regions.**REGION_CHOICES = (('BB', 'Banska Bystrica region'), ('BA', 'Brati**
http://sk.wikipedia.org/wiki/Administrat%C3%ADvne_%C4%8Dlenenie_Slovenska

6.59 Tunisia (tn)

Tunisia-specific Form helpers.

class localflavor.tn.forms.**TNGovernorateSelect**(*attrs=None*)
A Select widget that uses a list of the Tunisian governorates as its choices.

localflavor.tn.tn_governorates.**GOVERNORATE_CHOICES = (('ariana', 'Ariana'), ('beja', 'Beja**
All the 24 Tunisian Governorates http://en.wikipedia.org/wiki/Governorates_of_Tunisia

6.60 Turkey (tr)

6.60.1 Forms

class localflavor.tr.forms.**TRIdentificationNumberField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A Turkey Identification Number number.

See: http://tr.wikipedia.org/wiki/T%C3%BCrkiye_Cumhuriyeti_Kimlik_Numaras%C4%B1

Checks the following rules to determine whether the number is valid:

- The number is 11-digits.
- First digit is not 0.
- Conforms to the following two formula: (sum(1st, 3rd, 5th, 7th, 9th)*7 - sum(2nd,4th,6th,8th)) % 10 = 10th digit sum(1st to 10th) % 10 = 11th digit

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.tr.forms.**TRPostalCodeField**(*max_length=5, min_length=5, **kwargs*)
A form field that validates input as a Turkish zip code.

Valid codes consist of five digits.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.tr.forms.**TRProvinceSelect**(*attrs=None*)
A Select widget that uses a list of provinces in Turkey as its choices.

6.60.2 Data

localflavor.tr.tr_provinces.**PROVINCE_CHOICES = (('01', 'Adana'), ('02', 'Adıyaman'), ('03',**
A list of Turkish provinces

6.61 Ukraine (ua)

6.61.1 Forms

class localflavor.ua.forms.**UAPostalCodeField**(***kwargs*)
A form field that validates input as a Ukrainian postal code.

Valid format is XXXXX. Note: first two numbers cannot be '00'.

Whitespace around a postal code is accepted and automatically trimmed.

New in version 1.5.

to_python(*value*)
Return a string.

class localflavor.ua.forms.**UARegionSelect**(**args*, ***kwargs*)
A Select widget that uses a list of Ukrainian regions as its choices.

New in version 1.5.

class localflavor.ua.forms.**UAVatNumberField**(***kwargs*)
A form field that validates input as a Ukrainian analog of a VAT number.

Valid format is XXXXXXXXXX.

Whitespace around a VAT number is accepted and automatically trimmed.

New in version 1.5.

to_python(*value*)
Return a string.

6.61.2 Models

class localflavor.ua.models.**UAPostalCodeField**(**args*, ***kwargs*)
A model field which stores a Ukrainian postal code.

This field is represented by forms as a *UAPostalCodeField* field.

New in version 1.5.

class localflavor.ua.models.**UARegionField**(**args*, ***kwargs*)
A model field which stores a Ukrainian region.

This field is represented by forms as a *UARegionSelect* field.

New in version 1.5.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.ua.models.**UAVatNumberField**(**args*, ***kwargs*)
A model field which stores a Ukrainian analog of a VAT number.

This field is represented by forms as a *UAVatNumberField* field.

New in version 1.5.

6.61.3 Data

localflavor.ua.ua_regions.**UA_REGION_CHOICES = (('UA-71', 'Cherkasy Oblast'), ('UA-74', 'Che**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

6.62 United States of America (us)

6.62.1 Forms

USA-specific Form helpers.

class localflavor.us.forms.**USPSSelect**(*attrs=None*)
A Select widget that uses a list of US Postal Service codes as its choices.

Note: If you are looking for a form field that validates U.S. ZIP codes please use *USZipCodeField*.

class localflavor.us.forms.**USSocialSecurityNumberField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A United States Social Security number.

Checks the following rules to determine whether the number is valid:

- Conforms to the XXX-XX-XXXX format.
- No group consists entirely of zeroes.
- The leading group is not "666" (block "666" will never be allocated).
- The number is not in the promotional block 987-65-4320 through 987-65-4329, which are permanently invalid.
- The number is not one known to be invalid due to otherwise widespread promotional use or distribution (e.g., the Woolworth's number or the 1962 promotional number).

New in version 1.1.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.us.forms.**USStateField**(*, *max_length=None*, *min_length=None*, *strip=True*, *empty_value=''*, ***kwargs*)

A form field that validates its input is a U.S. state, territory, or COFA territory. The input is validated against a dictionary which includes names and abbreviations.

It normalizes the input to the standard two-letter postal service abbreviation for the given state.

clean(*value*)

Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.us.forms.**USStateSelect**(*attrs=None*)

A Select widget that uses a list of U.S. states/territories as its choices.

class localflavor.us.forms.**USZipCodeField**(***kwargs*)

A form field that validates input as a U.S. ZIP code.

Valid formats are XXXXX or XXXXX-XXXX.

Note: If you are looking for a form field with a list of U.S. Postal Service locations please use *USPSSelect*.

New in version 1.1.

Whitespace around the ZIP code is accepted and automatically trimmed.

to_python(*value*)

Return a string.

6.62.2 Models

class localflavor.us.models.**USPostalCodeField**(**args*, ***kwargs*)

A model field that stores the two-letter U.S. Postal Service abbreviation in the database.

Forms represent it as a `USPSSelect` field.

Note: If you are looking for a model field that validates U.S. ZIP codes please use *USZipCodeField*.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.us.models.**USSocialSecurityNumberField**(**args*, ***kwargs*)
A model field that stores the security number in the format `XXX-XX-XXXX`.

Forms represent it as `forms.USSocialSecurityNumberField` field.

New in version 1.1.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

class localflavor.us.models.**USStateField**(**args*, ***kwargs*)
A model field that stores the two-letter U.S. state abbreviation in the database.

Forms represent it as a `forms.USStateField` field.

deconstruct()
Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict

- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

class localflavor.us.models.**USZipCodeField**(**args*, ***kwargs*)

A model field that stores the U.S. ZIP code in the database.

Forms represent it as a *USZipCodeField* field.

Note: If you are looking for a model field with a list of U.S. Postal Service locations please use *USPostalCodeField*.

New in version 1.1.

formfield(***kwargs*)

Return a django.forms.Field instance for this field.

6.62.3 Data

localflavor.us.us_states.**CONTIGUOUS_STATES = (('AL', 'Alabama'), ('AZ', 'Arizona'), ('AR',**

The 48 contiguous states, plus the District of Columbia.

localflavor.us.us_states.**NON_CONTIGUOUS_STATES = (('AK', 'Alaska'), ('HI', 'Hawaii'))**

Non contiguous states (Not connected to mainland USA)

localflavor.us.us_states.**US_TERRITORIES = (('AS', 'American Samoa'), ('GU', 'Guam'), ('MP',**

Non-state territories.

localflavor.us.us_states.**ARMED_FORCES_STATES = (('AA', 'Armed Forces Americas'), ('AE', 'A**

Military postal "states". Note that 'AE' actually encompasses Europe, Canada, Africa and the Middle East.

localflavor.us.us_states.**COFA_STATES = (('FM', 'Federated States of Micronesia'), ('MH', 'M**

Non-US locations serviced by USPS (under Compact of Free Association).

localflavor.us.us_states.**OBSOLETE_STATES = (('CM', 'Commonwealth of the Northern Mariana Is**

Obsolete abbreviations (no longer US territories/USPS service, or code changed).

localflavor.us.us_states.**US_STATES = CONTIGUOUS_STATES + NON_CONTIGUOUS_STATES**

All US states.

This tuple is lazily generated and may not work as expected in all cases due to tuple optimizations in the Python interpreter which do not account for lazily generated tuples. For example:

```
US_STATES + ('XX', _('Select a State'))
```

should work as expected, but:

```
('XX', _('Select a State')) + US_STATES
```

may throw:

```
TypeError: can only concatenate tuple (not "proxy") to tuple
```

due to a Python optimization that causes the concatenation to occur before US_STATES has been lazily generated. To work around these issues, you can use a slice index (`[:]`) to force the generation of US_STATES before any other operations are processed by the Python interpreter:

```
('XX', _('Select a State')) + US_STATES[:]
```

localflavor.us.us_states.**STATE_CHOICES = CONTIGUOUS_STATES + NON_CONTIGUOUS_STATES + US_TE**

All US states and territories plus DC and military mail.

This tuple is lazily generated and may not work as expected in all cases due to tuple optimizations in the Python interpreter which do not account for lazily generated tuples. For example:

```
STATE_CHOICES + ('XX', _('Select a State'))
```

should work as expected, but:

```
('XX', _('Select a State')) + STATE_CHOICES
```

may throw:

```
TypeError: can only concatenate tuple (not "proxy") to tuple
```

due to a Python optimization that causes the concatenation to occur before STATE_CHOICES has been lazily generated. To work around these issues, you can use a slice index (`[:]`) to force the generation of STATE_CHOICES before any other operations are processed by the Python interpreter:

```
('XX', _('Select a State')) + STATE_CHOICES[:]
```

localflavor.us.us_states.**USPS_CHOICES = CONTIGUOUS_STATES + NON_CONTIGUOUS_STATES + US_TERR**

All US Postal Service locations.

This tuple is lazily generated and may not work as expected in all cases due to tuple optimizations in the Python interpreter which do not account for lazily generated tuples. For example:

```
USPS_CHOICES + ('XX', _('Select a State'))
```

should work as expected, but:

```
('XX', _('Select a State')) + USPS_CHOICES
```

may throw:

```
TypeError: can only concatenate tuple (not "proxy") to tuple
```

due to a Python optimization that causes the concatenation to occur before USPS_CHOICES has been lazily generated. To work around these issues, you can use a slice index (`[:]`) to force the generation of USPS_CHOICES before any other operations are processed by the Python interpreter:

```
('XX', _('Select a State')) + USPS_CHOICES[:]
```

localflavor.us.us_states.**STATES_NORMALIZED = {'aa': 'AA', 'ae': 'AE', 'ak': 'AK', 'al':**

Normalized versions of state names

6.63 Uruguay (uy)

6.63.1 Forms

UY-specific form helpers.

class localflavor.uy.forms.**UYCIField**(***kwargs*)
A field that validates Uruguayan 'Cedula de identidad' (CI) numbers.

clean(*value*)
Validates format and validation digit.

The official format is [X.]XXX.XXX-X but usually dots and/or slash are omitted so, when validating, those characters are ignored if found in the correct place. The three typically used formats are supported: [X]XXXXXXX, [X]XXXXXX-X and [X.]XXX.XXX-X.

class localflavor.uy.forms.**UYDepartmentSelect**(*attrs=None*)
A Select widget that uses a list of Uruguayan departments as its choices.

6.63.2 Utilities

localflavor.uy.util.**get_validation_digit**(*number*)
Calculates the validation digit for the given number.

6.63.3 Data

localflavor.uy.uy_departments.**DEPARTMENT_CHOICES = (('G', 'Artigas'), ('A', 'Canelones'),**
A list of Uruguayan departments as *choices* in a formfield.

6.64 South Africa (za)

6.64.1 Forms

South Africa-specific Form helpers.

class localflavor.za.forms.**ZAIDField**(**, max_length=None, min_length=None, strip=True, empty_value='', **kwargs*)
A form field for South African ID numbers.

The checksum is validated using the Luhn checksum, and uses a simplistic (read: not entirely accurate) check for the birth date.

clean(*value*)
Validate the given value and return its "cleaned" value as an appropriate Python object. Raise ValidationError for any errors.

class localflavor.za.forms.**ZAPostCodeField**(***kwargs*)
A form field that validates input as a South African postcode.

Valid postcodes must have four digits.

class localflavor.za.forms.**ZAProvinceSelect**(*attrs=None*)
A Select widget that uses a list of South African Provinces as its choices.

6.64.2 Data

localflavor.za.za_provinces.**PROVINCE_CHOICES = (('EC', 'Eastern Cape'), ('FS', 'Free State**
A list of South African provinces as *choices* in a formfield.

6.65 Generic helpers

6.65.1 Forms

class localflavor.generic.forms.**BICFormField**(***kwargs*)
A BIC consists of 8 (BIC8) or 11 (BIC11) alphanumeric characters.

BICs are also known as SWIFT-BIC, BIC code, SWIFT ID, SWIFT code or ISO 9362.

https://en.wikipedia.org/wiki/ISO_9362

New in version 1.1.

to_python(*value*)
Return a string.

class localflavor.generic.forms.**DateField**(*input_formats=None*, ***kwargs*)
A date input field which uses non-US date input formats by default.

class localflavor.generic.forms.**DateTimeField**(*input_formats=None*, ***kwargs*)
A date and time input field which uses non-US date and time input formats by default.

class localflavor.generic.forms.**IBANFormField**(*use_nordea_extensions=False*, *include_countries=None*, ***kwargs*)
An IBAN consists of up to 34 alphanumeric characters.

To limit validation to specific countries, set the 'include_countries' argument with a tuple or list of ISO 3166-1 alpha-2 codes. For example, *include_countries=('NL', 'BE, 'LU')*.

A list of countries that use IBANs as part of SEPA is included for convenience. To use this feature, set *include_countries=IBAN_SEPA_COUNTRIES* as an argument to the field.

Example:

```
from django import forms
from localflavor.generic.forms import IBANFormField
from localflavor.generic.countries.sepa import IBAN_SEPA_COUNTRIES

class MyForm(forms.Form):
    iban = IBANFormField(include_countries=IBAN_SEPA_COUNTRIES)
```

In addition to validating official IBANs, this field can optionally validate unofficial IBANs that have been catalogued by Nordea by setting the *use_nordea_extensions* argument to True.

https://en.wikipedia.org/wiki/International_Bank_Account_Number

New in version 1.1.

prepare_value(*value*)
The display format for IBAN has a space every 4 characters.

to_python(*value*)
Return a string.

class localflavor.generic.forms.**SplitDateTimeField**(*input_date_formats=None*, *input_time_formats=None*, ***kwargs*)

Split date and time input fields which use non-US date and time input formats by default.

6.65.2 Models

class localflavor.generic.models.**BICField**(**args*, ***kwargs*)

A BIC consists of 8 (BIC8) or 11 (BIC11) alphanumeric characters.

BICs are also known as SWIFT-BIC, BIC code, SWIFT ID, SWIFT code or ISO 9362.

https://en.wikipedia.org/wiki/ISO_9362

New in version 1.1.

formfield(***kwargs*)

Return a django.forms.Field instance for this field.

to_python(*value*)

Convert the input value into the expected Python data type, raising django.core.exceptions.ValidationError if the data can't be converted. Return the converted value. Subclasses should override this.

class localflavor.generic.models.**IBANField**(**args*, ***kwargs*)

An IBAN consists of up to 34 alphanumeric characters.

To limit validation to specific countries, set the 'include_countries' argument with a tuple or list of ISO 3166-1 alpha-2 codes. For example, *include_countries=('NL', 'BE, 'LU')*.

A list of countries that use IBANs as part of SEPA is included for convenience. To use this feature, set *include_countries=IBAN_SEPA_COUNTRIES* as an argument to the field.

Example:

```
from django.db import models
from localflavor.generic.models import IBANField
from localflavor.generic.countries.sepa import IBAN_SEPA_COUNTRIES

class MyModel(models.Model):
    iban = IBANField(include_countries=IBAN_SEPA_COUNTRIES)
```

In addition to validating official IBANs, this field can optionally validate unofficial IBANs that have been catalogued by Nordea by setting the *use_nordea_extensions* argument to True.

https://en.wikipedia.org/wiki/International_Bank_Account_Number

New in version 1.1.

deconstruct()

Return enough information to recreate the field as a 4-tuple:

- The name of the field on the model, if contribute_to_class() has been run.
- The import path of the field, including the class:e.g. django.db.models.IntegerField This should be the most portable version, so less specific may be better.
- A list of positional arguments.
- A dict of keyword arguments.

Note that the positional or keyword arguments must contain values of the following types (including inner values of collection types):

- None, bool, str, int, float, complex, set, frozenset, list, tuple, dict
- UUID
- datetime.datetime (naive), datetime.date
- top-level classes, top-level functions - will be referenced by their full import path
- Storage instances - these have their own deconstruct() method

This is because the values here must be serialized into a text format (possibly new Python code, possibly JSON) and these are the only types with encoding handlers defined.

There's no need to return the exact way the field was instantiated this time, just ensure that the resulting field is the same - prefer keyword arguments over positional ones, and omit parameters with their default values.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

to_python(*value*)
Convert the input value into the expected Python data type, raising django.core.exceptions.ValidationError if the data can't be converted. Return the converted value. Subclasses should override this.

6.65.3 Validators

class localflavor.generic.validators.**BICValidator**
A validator for SWIFT Business Identifier Codes (ISO 9362:2009).

Validation is based on the BIC structure found on wikipedia.

https://en.wikipedia.org/wiki/ISO_9362#Structure

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

class localflavor.generic.validators.**EANValidator**(*strip_nondigits=False*, *message=None*)
A generic validator for EAN like codes with the last digit being the checksum.

http://en.wikipedia.org/wiki/International_Article_Number_(EAN)

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

class localflavor.generic.validators.**IBANValidator**(*use_nordea_extensions=False*, *include_countries=None*)
A validator for International Bank Account Numbers (IBAN - ISO 13616-1:2007).

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

static iban_checksum(*value*)
Returns check digits for an input IBAN number.

Original checksum in input value is ignored.

class localflavor.generic.validators.**VATINValidator**
A validator for VAT identification numbers.

Currently only supports European VIES VAT identification numbers.

See See https://en.wikipedia.org/wiki/VAT_identification_number

clean(*value*)
Return tuple of country code and number.

deconstruct()
Return a 3-tuple of class import path, positional arguments, and keyword arguments.

localflavor.generic.validators.**VATIN_COUNTRY_CODE_LENGTH = 2**
Length of the country code prefix of a VAT identification number.

Codes are two letter ISO 3166-1 alpha-2 codes except for Greece that uses ISO 639-1.

localflavor.generic.validators.**VATIN_PATTERN_MAP = {'AT': '^ATU\\d{8}$', 'BE': '^BE0?\\d{9**
Map of country codes and regular expressions.

See https://en.wikipedia.org/wiki/VAT_identification_number

New in version 1.1.

6.65.4 Data

localflavor.generic.countries.iso_3166.**ISO_3166_1_ALPHA2_COUNTRY_CODES = ('AD', 'AE', 'AF',**
ISO 3166-1 country list. Sourced from https://www.iso.org/obp/ui on 2014-11-08

localflavor.generic.countries.sepa.**IBAN_SEPA_COUNTRIES = ('AD', 'AT', 'BE', 'BG', 'CH', 'CY**
European Payments Council list of SEPA scheme countries as of 2 Sep 2015. http://www.europeanpaymentscouncil.eu/index.cfm/knowledge-bank/epc-documents/epc-list-of-sepa-scheme-countries/

Python Module Index

l

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z