Natural Documentation

Release 0.2.0

Wijnand Modderman-Lenstra

June 07, 2016

Contents

Natural is a Python library to easily transform data into human-readable formats such as dates, time differences, numbers and sizes.

Features:

- Natural is not using any third-party libraries, so it runs from a bare Python installation
- Natural is fully internationalised and speaks your language
- Natural uses unicode strings where possible
- Natural is PEP8 compliant
- Natural has an extensive test suite

Demo:

```
>>> from natural import date, number, size
>>> import datetime
>>> date.compress(datetime.datetime(2012, 01, 01, 23, 42))
'23w2d17h52m59s'
>>> number.word(2300000)
'2.3 million'
>>> size.filesize(102410241024)
'95.377 GB'
```

We speak your language too!

```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'de_DE')
>>> from natural import date
>>> date.duration(1337000000)
'vor 4 Wochen'
```

CHAPTER 1

Download

Downloads:

- natural-0.2.0
- development tarball
- git repository at GitHub

CHAPTER 2

Contents

2.1 natural.bank

natural.bank.**bban**(*value*, *country=None*, *validate=False*)

Printable Basic Bank Account Number (BBAN) for the given country code. The `country` must be a valid ISO 3166-2 country code.

Parameters

- **value** – string or int
- **country** – string

```
>>> bban('068-9999995-01', 'BE')
'068999999501'
>>> bban('555', 'NL')
'555'
>>> bban('555', 'NL', validate=True)
Traceback (most recent call last):
    ...
ValueError: Invalid BBAN, number does not match specification
>>> bban('123', 'XY', validate=True)
Traceback (most recent call last):
    ...
ValueError: Invalid BBAN, country unknown
```

natural.bank.**bban_base10**(*number*)

Printable Basic Bank Account Number in base-10.

Parameters **number** – string

```
>>> bban_base10('01234567')
'45670123'
>>> bban_base10('ABCD')
'10111213'
```

natural.bank.**bban_compact**(*number*)

Printable compacted Basic Bank Account Number. Removes all the padding characters.

Parameters **number** – string

```
>>> bban_compact('1234.56.78.90')
'1234567890'
>>> bban_compact('068-9999995-01')
'068999999501'
```

natural.bank.**iban**(*number*, *validate=False*)

Printable International Bank Account Number (IBAN) as specified in ISO 13616.

Parameters **number** – string

```
>>> iban('BE43068999999501')
'BE43 0689 9999 9501'
>>> iban('XY32012341234123', validate=True)
Traceback (most recent call last):
    ...
ValueError: Invalid IBAN, country unknown
>>> iban('BE43068999999502', validate=True)
Traceback (most recent call last):
    ...
ValueError: Invalid IBAN, digits check failed
```

2.2 natural.data

natural.data.**hexdump**(*stream*)

Display stream contents in hexadecimal and ASCII format. The `stream` specified must either be a file-like object that supports the `read` method to receive bytes, or it can be a string.

To dump a file:

```
>>> hexdump(file(filename))
```

Or to dump stdin:

```
>>> import sys
>>> hexdump(sys.stdin)
```

Parameters **stream** – stream input

natural.data.**printable**(*sequence*)

Return a printable string from the input `sequence`

Parameters **sequence** – byte or string sequence

```
>>> printable('\x1b[1;34mtest\x1b[0m')
'.[1;34mtest.[0m'
>>> printable('\x00\x01\x02\x03\x04\x05\x06\x06')
'........'
>>> printable('12345678')
'12345678'
>>> printable('testing\n')
'testing.'
```

natural.data.**sparkline**(*data*)

Return a spark line for the given data set.

Value data sequence of numeric values

```
>>> print sparkline([1, 2, 3, 4, 5, 6, 5, 4, 3, 1, 5, 6])

```

natural.data.**throughput**(*sample*, *window=1*, *format='decimal'*)

Return the throughput in (intelli)bytes per second.

Parameters

- **sample** – number of samples sent
- **window** – default 1, sample window in seconds or `datetime.timedelta` object
- **format** – default 'decimal', see *natural.size.filesize()*

```
>>> throughput(123456, 42)
'2.87 kB/s'
```

2.3 natural.date

natural.date.**compress**(*t*, *sign=False*, *pad=''*)

Convert the input to compressed format, works with a `datetime.timedelta` object or a number that represents the number of seconds you want to compress. If you supply a timestamp or a `datetime.datetime` object, it will give the delta relative to the current time.

You can enable showing a sign in front of the compressed format with the `sign` parameter, the default is not to show signs.

Optionally, you can chose to pad the output. If you wish your values to be separated by spaces, set `pad` to `' '`.

Parameters

- **t** – seconds or `datetime.timedelta` object
- **sign** – default `False`
- **pad** – default `''`

```
>>> compress(1)
'1s'
>>> compress(12)
'12s'
>>> compress(123)
'2m3s'
>>> compress(1234)
'20m34s'
>>> compress(12345)
'3h25m45s'
>>> compress(123456)
'1d10h17m36s'
```

natural.date.**day**(*t*, *now=None*, *format='%B %d'*)

Date delta compared to `t`. You can override `now` to specify what date to compare to.

You can override the date format by supplying a `format` parameter.

Parameters

- **t** – timestamp, `datetime.date` or `datetime.datetime` object
- **now** – default `None`, optionally a `datetime.datetime` object
- **format** – default `'%B %d'`

```
>>> import time
>>> day(time.time())
'today'
>>> day(time.time() - 86400)
'yesterday'
>>> day(time.time() - 604800)
```

```
    'last week'
    >>> day(time.time() + 86400)
    'tomorrow'
    >>> day(time.time() + 604800)
    'next week'
```

natural.date.**delta**(*t1*, *t2*, *words=True*, *justnow=datetime.timedelta(0, 10)*)

Calculates the estimated delta between two time objects in human-readable format. Used internally by the *day()* and *duration()* functions.

Parameters

- **t1** – timestamp, `datetime.date` or `datetime.datetime` object
- **t2** – timestamp, `datetime.date` or `datetime.datetime` object
- **words** – default `True`, allow words like "yesterday", "tomorrow" and "just now"
- **justnow** – default `datetime.timedelta(seconds=10)`, `datetime.timedelta` object representing tolerance for considering a delta as meaning 'just now'

```
    >>> delta(_to_datetime('2012-06-13T15:24:17'), _to_datetime('2013-12-11T12:34:56'))
    ('77 weeks', -594639)
```

natural.date.**duration**(*t*, *now=None*, *precision=1*, *pad=' '*, *words=None*, *justnow=datetime.timedelta(0, 10)*)

Time delta compared to `t`. You can override `now` to specify what time to compare to.

Parameters

- **t** – timestamp, `datetime.date` or `datetime.datetime` object
- **now** – default `None`, optionally a `datetime.datetime` object
- **precision** – default `1`, number of fragments to return
- **words** – default `None`, allow words like "yesterday", if set to `None` this will be enabled if `precision` is set to `1`
- **justnow** – default `datetime.timedelta(seconds=10)`, `datetime.timedelta` object passed to *delta()* representing tolerance for considering argument `t` as meaning 'just now'

```
    >>> import time
    >>> from datetime import datetime
    >>> duration(time.time() + 1)
    'just now'
    >>> duration(time.time() + 11)
    '11 seconds from now'
    >>> duration(time.time() - 1)
    'just now'
    >>> duration(time.time() - 11)
    '11 seconds ago'
    >>> duration(time.time() - 3601)
    'an hour ago'
    >>> duration(time.time() - 7201)
    '2 hours ago'
    >>> duration(time.time() - 1234567)
    '2 weeks ago'
    >>> duration(time.time() + 7200, precision=1)
    '2 hours from now'
```

```
>>> duration(time.time() - 1234567, precision=3)
'2 weeks, 6 hours, 56 minutes ago'
>>> duration(datetime(2014, 9, 8), now=datetime(2014, 9, 9))
'yesterday'
>>> duration(datetime(2014, 9, 7, 23), now=datetime(2014, 9, 9))
'1 day ago'
>>> duration(datetime(2014, 9, 10), now=datetime(2014, 9, 9))
'tomorrow'
>>> duration(datetime(2014, 9, 11, 1), now=datetime(2014, 9, 9, 23))
'1 day from now'
```

2.4 natural.file

natural.file.**accessed**(*filename*)
Retrieve how long ago a file has been accessed.

Parameters **filename** – name of the file

```
>>> print accessed(__file__)
just now
```

natural.file.**created**(*filename*)
Retrieve how long ago a file has been created.

Parameters **filename** – name of the file

```
>>> print created('/')
8 weeks ago
```

natural.file.**modified**(*filename*)
Retrieve how long ago a file has been modified.

Parameters **filename** – name of the file

```
>>> print modified('/')
3 days ago
```

natural.file.**size**(*filename*, *format='decimal'*)
Retrieve the size of a file.

Parameters **filename** – name of the file

```
>>> size('/etc/mime.types')
23.70 kB
```

2.5 natural.number

natural.number.**double**(*value*, *digits=2*)
Converts a number to a formatted double based on the current locale.

Parameters

- **value** – number
- **digits** – default 2

```
>>> double(42)
'42.00'
>>> double(42, digits=1)
'42.0'
>>> double(12.34)
'12.34'
>>> double(1234.56)
'1,234.56'
```

natural.number.**number**(*value*)

Converts a number to a formatted number based on the current locale.

Parameters **value** – number

```
>>> number(42)
'42'
>>> number(12.34)
'12'
>>> number(1234)
'1,234'
>>> number(1234567)
'1,234,567'
```

natural.number.**ordinal**(*value*)

Converts a number to its ordinal representation.

Parameters **value** – number

```
>>> ordinal(1)
'1st'
>>> ordinal(11)
'11th'
>>> ordinal(101)
'101st'
>>> ordinal(104)
'104th'
>>> ordinal(113)
'113th'
>>> ordinal(123)
'123rd'
```

natural.number.**percentage**(*value*, *digits=2*)

Converts a fraction to a formatted percentage.

Parameters

- **value** – number
- **digits** – default 2

```
>>> percentage(1)
'100.00 %'
>>> percentage(0.23, digits=0)
'23 %'
>>> percentage(23.421)
'2,342.10 %'
```

natural.number.**word**(*value*, *digits=2*)

Converts a large number to a formatted number containing the textual suffix for that number.

Parameters **value** – number

```
>>> word(1)
'1'
>>> word(123456789)
'123.46 million'
```

2.6 natural.phone

natural.phone.**e123**(*number*, *areasize=3*, *groupsize=4*, *national=False*)

Printable E.123 (Notation for national and international telephone numbers from ITU) numbers.

Parameters

- **number** – string
- **areasize** – int
- **groupsize** – int
- **national** – bool

```
>>> e123(155542315678)
'+1 555 4231 5678'
>>> e123('+31654231567', areasize=1)
'+31 6 5423 1567'
>>> e123('+3114020', areasize=2)
'+31 14 020'
>>> e123('+312054231567', areasize=2, national=True)
'(020) 5423 1567'
```

natural.phone.**e161**(*number*, *alphabet={u'#': u'#', u'8tuv': u'8', u'3def': u'3', u'6mno': u'6', u'4ghi': u'4', u'*': u'*', u'1': u'1', u'0': u'0', u'2abc': u'2', u'5jkl': u'5', u'7prqs': u'7', u'9xyz': u'9'}*)

Printable a 26 Latin letters (A to Z) phone number to the 12-key telephone keypad number.

Parameters

- **number** – string
- **alphabet** – dict

```
>>> e161('0800-PIZZA123')
'080074992123'
>>> e161('0800^PIZZA123')
Traceback (most recent call last):
    ...
ValueError: Character "^" (0x5e) is not in the E.161 alphabet
```

natural.phone.**e164**(*number*)

Printable E.164 (The international public telecommunication numbering plan from ITU) numbers.

Parameters **number** – string

```
>>> e164(155542315678)
'+155542315678'
>>> e164('+31 20 5423 1567')
'+312054231567'
```

natural.phone.**enum**(*number*, *zone='e164.arpa'*)

Printable DNS ENUM (telephone number mapping) record.

Parameters

- **number** – string
- **zone** – string

```
>>> enum('+31 20 5423 1567')
'7.6.5.1.3.2.4.5.0.2.1.3.e164.arpa.'
>>> enum('+31 97 99 6642', zone='e164.spacephone.org')
'2.4.6.6.9.9.7.9.1.3.e164.spacephone.org.'
```

natural.phone.**imei**(*number*)

Printable International Mobile Station Equipment Identity (IMEI) numbers.

Parameters **number** – string or int

```
>>> imei(12345678901234)
'12-345678-901234-7'
>>> imei(1234567890123456)
'12-345678-901234-56'
```

natural.phone.**imsi**(*number*)

Printable International Mobile Subscriber Identity (IMSI) numbers. Mind that there is no validation done on the actual correctness of the MCC/MNC. If you wish to validate IMSI numbers, take a look at python-stdnum.

Parameters **number** – string or int

```
>>> imsi(2042312345)
'204-23-12345'
```

natural.phone.**meid**(*number*, *separator=u' '*)

Printable Mobile Equipment Identifier (MEID) number.

```
>>> meid(123456789012345678)
'1B 69B4BA 630F34 6'
>>> meid('1B69B4BA630F34')
'1B 69B4BA 630F34 6'
```

natural.phone.**pesn**(*number*, *separator=u' '*)

Printable Pseudo Electronic Serial Number.

Parameters **number** – hexadecimal string

```
>>> pesn('1B69B4BA630F34E')
'805F9EF7'
```

2.7 natural.size

natural.size.**binarysize**(*value*)

Wrapper for *filesize()*.

```
>>> binarysize(123)
'123.00 iB'
>>> binarysize(123456)
'123.46 KiB'
>>> binarysize(1234567890)
'1.23 GiB'
```

natural.size.**decimalsize**(*value*)

Wrapper for *filesize()*.

```
>>> decimalsize(123)
'123.00 B'
>>> decimalsize(123456)
'120.56 kB'
>>> decimalsize(1234567890)
'1.15 GB'
```

natural.size.**filesize**(*value*, *format='decimal'*, *digits=2*)
Convert a file size into natural readable format. Multiple formats are supported.

Parameters

- **value** – size
- **format** – default `decimal`, choices `binary`, `decimal` or `gnu`
- **digits** – default 2

```
>>> filesize(123)
'123.00 B'
>>> filesize(123456)
'120.56 kB'
>>> filesize(1234567890)
'1.15 GB'
```

natural.size.**gnusize**(*value*, *digits=1*)
Wrapper for *filesize()*.

```
>>> gnusize(123)
'123.0B'
>>> gnusize(123456)
'120.6K'
>>> gnusize(1234567890)
'1.1G'
```

2.8 natural.text

class natural.text.**Alphabet**(*mapping*)
Bases: *natural.text.Spelling*

Helper class for (spelling) alphabets.

static from_pair(*keys*, *values*)
Returns a new *Alphabet()* object for the translation items specified in `keys` and `values`.

transform(*letter*)

class natural.text.**Spelling**
Bases: object

transform(*letter*)

natural.text.**code**(*sentence*, *pad=u' '*, *format='army'*)
Transform a sentence using the code spelling alphabet, multiple international code alphabets are supported.

format	description
army	US (international) army code alphabet
faa	Federal Aviation Administration code alphabet, as described in "ICAO Phonetics in the FAA ATC Manual, §2-4-16"
icao	International Civil Aviation Organization, as described in "Annex 10 to the Convention on International Civil Aviation, Volume II (Fifth edition, 1995), Chapter 5, 38–40"
itu	International Telecommunication Union Roman alphabet, as described in "ITU Phonetic Alphabet and Figure Code"
morse	International morse code, as described in "International Morse code Recommendation ITU-R M.1677-1", http://itu.int/
word	International Civil Aviation Organization, as described in "Annex 10 to the Convention on International Civil Aviation, Volume II (Fifth edition, 1995), Chapter 5, 38–40"

Parameters

- **sentence** – input sentence
- **pad** – default `None` (reside to per-alphabet defaults)
- **format** – default `army`

```
>>> code('Python')
'PAH pah  YANG kee  TANG go  HO tell  OSS car  NOH vem ber'
>>> code('Python', format='faa')
'PAHPAH  YANGKEY  TANGGO  HOHTELL  OSSCAH  NOVEMBER'
>>> code('Python', format='icao')
'PAH PAH  YANG KEY  TANG GO  HOH TELL  OSS CAH  NO VEM BER'
>>> code('Python', format='itu')
'PAH PAH  YANG KEY  TANG GO  HOH TELL  OSS CAH  NO VEM BER'
>>> code('Python', format='morse')
'.--.  -.--  -  ....  ---  -.'
>>> code('Python', format='word')
'papa  yankee  tango  hotel  oscar  november'
```

`natural.text.`**`morse`**(*sentence*, *pad=u' '*)

Wrapper for *code()*.

```
>>> morse('Python')
'.--. -.-- - .... --- -.'
```

`natural.text.`**`nato`**(*sentence*, *pad=' '*, *format='telephony'*)

Transform a sentence using the NATO spelling alphabet.

Parameters

- **sentence** – input sentence
- **pad** – default `u' '`
- **format** – default `telephony`, options `telephony` or `phonetic`

```
>>> nato('Python')
'papa yankee tango hotel oscar november'
>>> nato('Python', format='phonetic')
'pah-pah yang-key tang-go hoh-tel oss-cah no-vem-ber'
```

`natural.text.`**`pronounce`**(*sentence*, *pad=u' '*)

Transform a sentence using the pronounciations of the international code spelling alphabet. This function is subject to change its behaviour to support internationalised pronounciations of letters.

Parameters

- **sentence** – input sentence
- **pad** – default `u' '`

```
>>> pronounce('abc')
'ælf brvo tli'
```

`natural.text.`**spell**(*sentence*, *pad=' '*)

Transform a sentence using the localised spelling alphabet.

Parameters

- **sentence** – input sentence
- **pad** – default `u' '`

```
>>> spell('Python')
'Paris  Yokohama  Tripoli  Havanna  Oslo  New York'
```

2.9 locales

Natural uses locales for translating most textual output using gettext. We need your help translating the project!

2.9.1 Help translating

We use the excellent online translation platform at **Transifex_**. If you want to help out translating the project please visit https://www.transifex.net/projects/p/natural/

2.9.2 Supported languages

We support the following languages:

locale	language
af_ZA	Afrikaans
de_DE	German
en_GB	English
en_US	English (American)
es_ES	Spanish
fr_FR	French
nl_NL	Dutch

2.10 Django integration

2.10.1 Usage

Simply install natural and put it in your `INSTALLED_APPS`:

```
INSTALLED_APPS += ('natural',)
```

Now you can use all features from the natural lib in your templates:

```
{% load naturalise %}
CPU load is: {{ percent|percentage:1 }}
Average load {{ load|sparkline }}
...
```

Alternatively, you may use `naturalize` if you feel it's more appropriate:

```
{% load naturalize %}
...
```

2.10.2 API

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

n

W