
Natch

Release 1.0.4

Jul 04, 2020

Contents

| | | |
|----------|---|-----------|
| 1 | Guide | 3 |
| 1.1 | Installation | 3 |
| 1.2 | Examples | 3 |
| 1.2.1 | Example: Authorization | 3 |
| 1.2.2 | Example: Factorial | 4 |
| 1.2.3 | Example: JSON Encoder | 4 |
| 1.2.4 | Example: Perfect Square (Guard) | 5 |
| 1.2.5 | Example: Traversal | 6 |
| 1.2.6 | Example: Traversal (Functional) | 7 |
| 2 | Abstract | 9 |
| 3 | Core | 11 |
| 4 | Decorators | 13 |
| 5 | Exceptions | 15 |
| 6 | Hashers | 17 |
| 7 | Rules | 19 |
| | Python Module Index | 25 |
| | Index | 27 |

Pattern matching library.

1.1 Installation

```
pip install natch
```

1.2 Examples

1.2.1 Example: Authorization

```
import natch

@natch.pattern(
    natch.Any(),
    natch.Partial(
        user=natch.Partial(
            is_authenticated=True,
        ),
    ),
)
def get_user_profile(user_id, request):
    profile = dict()
    return profile

@natch.any()
def get_user_profile(user_id, request):
    raise Exception('Not authenticated.')
```

1.2.2 Example: Factorial

```
import natch

@natch.lt(0)
def factorial(x):
    x = abs(x)
    x = factorial(x)
    x = -1 * x
    return x

@natch.eq(0)
def factorial(x):
    return 1

@natch.eq(1)
def factorial(x):
    return 1

@natch.gt(1)
def factorial(x):
    x = x * factorial(x - 1)
    return x

for i in range(-5, 6):
    result = factorial(i)
    print(i, result)
```

1.2.3 Example: JSON Encoder

```
import natch

@natch.condition(lambda x: isinstance(x, bool))
def encode(data):
    return 'true' if data is True else 'false'

@natch.condition(lambda x: isinstance(x, int))
def encode(data):
    return str(data)

@natch.condition(lambda x: isinstance(x, str))
def encode(data):
    return f'"{data}"'

@natch.condition(lambda x: isinstance(x, list))
def encode(data):
```

(continues on next page)

(continued from previous page)

```

    encoded_segments = list()
    for segment in data:
        encoded_segment = encode(segment)
        encoded_segments.append(encoded_segment)
    inner = ', '.join(encoded_segments)
    return f'[{inner}]'

@natch.condition(lambda x: isinstance(x, dict))
def encode(data):
    inner_split = []
    for key, value in data.items():
        encoded_segment = encode(value)
        inner_split.append(f'"{key}": {encoded_segment}')
    inner = ', '.join(inner_split)
    return f'{{{inner}}}'

data = {'id': -1, 'username': 'ertgl', 'follows': ['neuro-sys'], 'is_active': True}

encoded_data = encode(data)
print(encoded_data)

```

1.2.4 Example: Perfect Square (Guard)

Natch is extendable. This is a how-to guide that shows extending Natch by writing custom rules.

```

import math
import natch

class IsPerfectSquare(natch.Rule):

    def __init__(self, *args, **kwargs):
        super(IsPerfectSquare, self).__init__()

    def does_match(self, *args, **kwargs):
        x = args[0]
        root = math.sqrt(x)
        does_match = int(root + 0.5) ** 2 == x
        return does_match

is_perfect_square = natch.make_rule_decorator(IsPerfectSquare)

@is_perfect_square()
def is_perfect(x):
    """

    Or:
    @natch.all_of(
        natch.Condition(

```

(continues on next page)

(continued from previous page)

```

        lambda x: isinstance(x, int),
    ),
    IsPerfectSquare(),
)

Or:
@natch.pattern(IsPerfectSquare())

"""
return True

@natch.any()
def is_perfect(x):
    return False

for i in range(10):
    result = is_perfect(i)
    print(i, result)

```

1.2.5 Example: Traversal

```

import natch

class Node(object):

    def __init__(self, **kwargs):
        self.value = kwargs.get('value')
        self.next = kwargs.get('next')

    @natch.partial(
        next=natch.Neq(None),
    )
    def get_next_node(self):
        return self.next

    @natch.partial(
        next=natch.Eq(None),
    )
    def get_next_node(self):
        return None

    def __str__(self):
        return str(self.value)

node = Node(
    value=0,
    next=Node(
        value=1,
        next=Node(
            value=2,
        ),
    ),
)

```

(continues on next page)

(continued from previous page)

```

    ),
)

current_node = node

for i in range(3):
    print(current_node)
    current_node = current_node.get_next_node()

```

1.2.6 Example: Traversal (Functional)

```

import natch

class Node(object):

    def __init__(self, **kwargs):
        self.value = kwargs.get('value')
        self.next = kwargs.get('next')

    def __str__(self):
        return str(self.value)

    @classmethod
    def new(cls, **kwargs):
        node = Node(**kwargs)
        return node

    @classmethod
    @natch.pattern(
        natch.Any(),
        natch.Partial(
            next=natch.Neq(None),
        ),
    )
    def get_next_node(cls, node):
        next_node = node.next
        return next_node

    @classmethod
    @natch.pattern(
        natch.Any(),
        natch.Partial(
            next=natch.Eq(None),
        ),
    )
    def get_next_node(cls, node):
        return None

    @classmethod
    @natch.pattern(
        natch.Any(),
        natch.Eq(None),
    )
    def get_next_node(cls, node):

```

(continues on next page)

(continued from previous page)

```
        return None

node = Node.new(
    value=0,
    next=Node.new(
        value=1,
        next=Node.new(
            value=2,
        ),
    ),
)

current_node = node

for i in range(5):
    print(current_node)
    current_node = Node.get_next_node(current_node)
```

```
class natch.abstract.Hasher(*args, **kwargs)

    hash(obj)

class natch.abstract.Registry(*args, **kwargs)

    del_hasher()
    del_index()
    get_hasher()
    get_index()
    hasher
    index
    lookup(func, *args, **kwargs)
    register(func, rule)
        Register a virtual subclass of an ABC.

        Returns the subclass, to allow usage as a class decorator.
    set_hasher(hasher)
    set_index(index)
    unregister(func, rule)
class natch.abstract.Rule(*args, **kwargs)

    args
    del_args()
    del_kwargs()
```

```
does_match (*args, **kwargs)
get_args ()
get_kwargs ()
kwargs
set_args (args)
set_kwargs (kwargs)
```

```
class natch.core.Decoration

    classmethod make_rule_decorator(rule_cls)
class natch.core.Registry(*args, **kwargs)

    del_hasher()
    del_index()
    get_hasher()
    get_index()
    hasher
    index
    lookup(func, *args, **kwargs)
    register(func, rule)
        Register a virtual subclass of an ABC.

        Returns the subclass, to allow usage as a class decorator.
    set_hasher(hasher)
    set_index(index)
    unregister(func, rule)
```


CHAPTER 4

Decorators

```
natch.decorators.any(*rule_args, **rule_kwargs)
natch.decorators.any_of(*rule_args, **rule_kwargs)
natch.decorators.all_of(*rule_args, **rule_kwargs)
natch.decorators.eq(*rule_args, **rule_kwargs)
natch.decorators.neq(*rule_args, **rule_kwargs)
natch.decorators.gt(*rule_args, **rule_kwargs)
natch.decorators.gte(*rule_args, **rule_kwargs)
natch.decorators.lt(*rule_args, **rule_kwargs)
natch.decorators.lte(*rule_args, **rule_kwargs)
natch.decorators.contains(*rule_args, **rule_kwargs)
natch.decorators.not_contains(*rule_args, **rule_kwargs)
natch.decorators.condition(*rule_args, **rule_kwargs)
natch.decorators.partial(*rule_args, **rule_kwargs)
natch.decorators.pattern(*rule_args, **rule_kwargs)
```


CHAPTER 5

Exceptions

exception `natch.exceptions.NeverMatchesError(*args, **kwargs)`

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

CHAPTER 6

Hashers

```
class natch.hashers.QualnameHasher(*args, **kwargs)
```

```
    hash(obj)
```



```
class match.rules.Any(*args, **kwargs)

    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)

class match.rules.AnyOf(*args, **kwargs)

    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)
```

```
class natch.rules.AllOf(*args, **kwargs)
```

```
    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)
```

```
class natch.rules.Eq(value, **kwargs)
```

```
    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)
```

```
class natch.rules.Neq(value, **kwargs)
```

```
    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)
```

```
class natch.rules.Gt(value, **kwargs)
```

```
    args
    del_args()
```



```
del_kwargs ()
does_match (*args, **kwargs)
get_args ()
get_kwargs ()
kwargs
set_args (args)
set_kwargs (kwargs)
class natch.rules.Gte (value, **kwargs)
```

```
args
del_args ()
del_kwargs ()
does_match (*args, **kwargs)
get_args ()
get_kwargs ()
kwargs
set_args (args)
set_kwargs (kwargs)
class natch.rules.It (value, **kwargs)
```

```
args
del_args ()
del_kwargs ()
does_match (*args, **kwargs)
get_args ()
get_kwargs ()
kwargs
set_args (args)
set_kwargs (kwargs)
class natch.rules.Lte (value, **kwargs)
```

```
args
del_args ()
del_kwargs ()
does_match (*args, **kwargs)
get_args ()
get_kwargs ()
```

```
    kwargs
    set_args (args)
    set_kwargs (kwargs)
class natch.rules.Contains (value, **kwargs)

    args
    del_args ()
    del_kwargs ()
    does_match (*args, **kwargs)
    get_args ()
    get_kwargs ()
    kwargs
    set_args (args)
    set_kwargs (kwargs)
class natch.rules.NotContains (value, **kwargs)

    args
    del_args ()
    del_kwargs ()
    does_match (*args, **kwargs)
    get_args ()
    get_kwargs ()
    kwargs
    set_args (args)
    set_kwargs (kwargs)
class natch.rules.Condition (func, **kwargs)

    args
    del_args ()
    del_kwargs ()
    does_match (*args, **kwargs)
    get_args ()
    get_kwargs ()
    kwargs
    set_args (args)
    set_kwargs (kwargs)
```

```
class natch.rules.Partial(*args, **kwargs)

    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)

class natch.rules.Pattern(*rules, **kwargs)

    args
    del_args()
    del_kwargs()
    does_match(*args, **kwargs)
    get_args()
    get_kwargs()
    kwargs
    set_args(args)
    set_kwargs(kwargs)
```

- [modindex](#)

n

- `natch.abstract`, [9](#)
- `natch.core`, [11](#)
- `natch.decorators`, [13](#)
- `natch.exceptions`, [15](#)
- `natch.hashers`, [17](#)
- `natch.rules`, [19](#)

A

`all_of()` (in module *natch.decorators*), 13
`AllOf` (class in *natch.rules*), 19
`Any` (class in *natch.rules*), 19
`any()` (in module *natch.decorators*), 13
`any_of()` (in module *natch.decorators*), 13
`AnyOf` (class in *natch.rules*), 19
`args` (*natch.abstract.Rule* attribute), 9
`args` (*natch.exceptions.NeverMatchesError* attribute), 15
`args` (*natch.rules.AllOf* attribute), 20
`args` (*natch.rules.Any* attribute), 19
`args` (*natch.rules.AnyOf* attribute), 19
`args` (*natch.rules.Condition* attribute), 22
`args` (*natch.rules.Contains* attribute), 22
`args` (*natch.rules.Eq* attribute), 20
`args` (*natch.rules.Gt* attribute), 20
`args` (*natch.rules.Gte* attribute), 21
`args` (*natch.rules.Lt* attribute), 21
`args` (*natch.rules.Lte* attribute), 21
`args` (*natch.rules.Neq* attribute), 20
`args` (*natch.rules.NotContains* attribute), 22
`args` (*natch.rules.Partial* attribute), 23
`args` (*natch.rules.Pattern* attribute), 23

C

`Condition` (class in *natch.rules*), 22
`condition()` (in module *natch.decorators*), 13
`Contains` (class in *natch.rules*), 22
`contains()` (in module *natch.decorators*), 13

D

`Decoration` (class in *natch.core*), 11
`del_args()` (*natch.abstract.Rule* method), 9
`del_args()` (*natch.rules.AllOf* method), 20
`del_args()` (*natch.rules.Any* method), 19
`del_args()` (*natch.rules.AnyOf* method), 19
`del_args()` (*natch.rules.Condition* method), 22
`del_args()` (*natch.rules.Contains* method), 22

`del_args()` (*natch.rules.Eq* method), 20
`del_args()` (*natch.rules.Gt* method), 20
`del_args()` (*natch.rules.Gte* method), 21
`del_args()` (*natch.rules.Lt* method), 21
`del_args()` (*natch.rules.Lte* method), 21
`del_args()` (*natch.rules.Neq* method), 20
`del_args()` (*natch.rules.NotContains* method), 22
`del_args()` (*natch.rules.Partial* method), 23
`del_args()` (*natch.rules.Pattern* method), 23
`del_hasher()` (*natch.abstract.Registry* method), 9
`del_hasher()` (*natch.core.Registry* method), 11
`del_index()` (*natch.abstract.Registry* method), 9
`del_index()` (*natch.core.Registry* method), 11
`del_kwargs()` (*natch.abstract.Rule* method), 9
`del_kwargs()` (*natch.rules.AllOf* method), 20
`del_kwargs()` (*natch.rules.Any* method), 19
`del_kwargs()` (*natch.rules.AnyOf* method), 19
`del_kwargs()` (*natch.rules.Condition* method), 22
`del_kwargs()` (*natch.rules.Contains* method), 22
`del_kwargs()` (*natch.rules.Eq* method), 20
`del_kwargs()` (*natch.rules.Gt* method), 20
`del_kwargs()` (*natch.rules.Gte* method), 21
`del_kwargs()` (*natch.rules.Lt* method), 21
`del_kwargs()` (*natch.rules.Lte* method), 21
`del_kwargs()` (*natch.rules.Neq* method), 20
`del_kwargs()` (*natch.rules.NotContains* method), 22
`del_kwargs()` (*natch.rules.Partial* method), 23
`del_kwargs()` (*natch.rules.Pattern* method), 23
`does_match()` (*natch.abstract.Rule* method), 9
`does_match()` (*natch.rules.AllOf* method), 20
`does_match()` (*natch.rules.Any* method), 19
`does_match()` (*natch.rules.AnyOf* method), 19
`does_match()` (*natch.rules.Condition* method), 22
`does_match()` (*natch.rules.Contains* method), 22
`does_match()` (*natch.rules.Eq* method), 20
`does_match()` (*natch.rules.Gt* method), 21
`does_match()` (*natch.rules.Gte* method), 21
`does_match()` (*natch.rules.Lt* method), 21
`does_match()` (*natch.rules.Lte* method), 21
`does_match()` (*natch.rules.Neq* method), 20

`does_match()` (*natch.rules.NotContains method*), 22
`does_match()` (*natch.rules.Partial method*), 23
`does_match()` (*natch.rules.Pattern method*), 23

E

`Eq` (*class in natch.rules*), 20
`eq()` (*in module natch.decorators*), 13

G

`get_args()` (*natch.abstract.Rule method*), 10
`get_args()` (*natch.rules.AllOf method*), 20
`get_args()` (*natch.rules.Any method*), 19
`get_args()` (*natch.rules.AnyOf method*), 19
`get_args()` (*natch.rules.Condition method*), 22
`get_args()` (*natch.rules.Contains method*), 22
`get_args()` (*natch.rules.Eq method*), 20
`get_args()` (*natch.rules.Gt method*), 21
`get_args()` (*natch.rules.Gte method*), 21
`get_args()` (*natch.rules.Lt method*), 21
`get_args()` (*natch.rules.Lte method*), 21
`get_args()` (*natch.rules.Neq method*), 20
`get_args()` (*natch.rules.NotContains method*), 22
`get_args()` (*natch.rules.Partial method*), 23
`get_args()` (*natch.rules.Pattern method*), 23
`get_hasher()` (*natch.abstract.Registry method*), 9
`get_hasher()` (*natch.core.Registry method*), 11
`get_index()` (*natch.abstract.Registry method*), 9
`get_index()` (*natch.core.Registry method*), 11
`get_kwargs()` (*natch.abstract.Rule method*), 10
`get_kwargs()` (*natch.rules.AllOf method*), 20
`get_kwargs()` (*natch.rules.Any method*), 19
`get_kwargs()` (*natch.rules.AnyOf method*), 19
`get_kwargs()` (*natch.rules.Condition method*), 22
`get_kwargs()` (*natch.rules.Contains method*), 22
`get_kwargs()` (*natch.rules.Eq method*), 20
`get_kwargs()` (*natch.rules.Gt method*), 21
`get_kwargs()` (*natch.rules.Gte method*), 21
`get_kwargs()` (*natch.rules.Lt method*), 21
`get_kwargs()` (*natch.rules.Lte method*), 21
`get_kwargs()` (*natch.rules.Neq method*), 20
`get_kwargs()` (*natch.rules.NotContains method*), 22
`get_kwargs()` (*natch.rules.Partial method*), 23
`get_kwargs()` (*natch.rules.Pattern method*), 23
`Gt` (*class in natch.rules*), 20
`gt()` (*in module natch.decorators*), 13
`Gte` (*class in natch.rules*), 21
`gte()` (*in module natch.decorators*), 13

H

`hash()` (*natch.abstract.Hasher method*), 9
`hash()` (*natch.hashers.QualnameHasher method*), 17
`Hasher` (*class in natch.abstract*), 9
`hasher` (*natch.abstract.Registry attribute*), 9
`hasher` (*natch.core.Registry attribute*), 11

I

`index` (*natch.abstract.Registry attribute*), 9
`index` (*natch.core.Registry attribute*), 11

K

`kwargs` (*natch.abstract.Rule attribute*), 10
`kwargs` (*natch.rules.AllOf attribute*), 20
`kwargs` (*natch.rules.Any attribute*), 19
`kwargs` (*natch.rules.AnyOf attribute*), 19
`kwargs` (*natch.rules.Condition attribute*), 22
`kwargs` (*natch.rules.Contains attribute*), 22
`kwargs` (*natch.rules.Eq attribute*), 20
`kwargs` (*natch.rules.Gt attribute*), 21
`kwargs` (*natch.rules.Gte attribute*), 21
`kwargs` (*natch.rules.Lt attribute*), 21
`kwargs` (*natch.rules.Lte attribute*), 21
`kwargs` (*natch.rules.Neq attribute*), 20
`kwargs` (*natch.rules.NotContains attribute*), 22
`kwargs` (*natch.rules.Partial attribute*), 23
`kwargs` (*natch.rules.Pattern attribute*), 23

L

`lookup()` (*natch.abstract.Registry method*), 9
`lookup()` (*natch.core.Registry method*), 11
`Lt` (*class in natch.rules*), 21
`lt()` (*in module natch.decorators*), 13
`Lte` (*class in natch.rules*), 21
`lte()` (*in module natch.decorators*), 13

M

`make_rule_decorator()` (*natch.core.Decoration class method*), 11

N

`natch.abstract` (*module*), 9
`natch.core` (*module*), 11
`natch.decorators` (*module*), 13
`natch.exceptions` (*module*), 15
`natch.hashers` (*module*), 17
`natch.rules` (*module*), 19
`Neq` (*class in natch.rules*), 20
`neq()` (*in module natch.decorators*), 13
`NeverMatchesError`, 15
`not_contains()` (*in module natch.decorators*), 13
`NotContains` (*class in natch.rules*), 22

P

`Partial` (*class in natch.rules*), 22
`partial()` (*in module natch.decorators*), 13
`Pattern` (*class in natch.rules*), 23
`pattern()` (*in module natch.decorators*), 13

Q

`QualnameHasher` (*class in natch.hashers*), 17

R

register() (*natch.abstract.Registry method*), 9
 register() (*natch.core.Registry method*), 11
 Registry (*class in natch.abstract*), 9
 Registry (*class in natch.core*), 11
 Rule (*class in natch.abstract*), 9

S

set_args() (*natch.abstract.Rule method*), 10
 set_args() (*natch.rules.AllOf method*), 20
 set_args() (*natch.rules.Any method*), 19
 set_args() (*natch.rules.AnyOf method*), 19
 set_args() (*natch.rules.Condition method*), 22
 set_args() (*natch.rules.Contains method*), 22
 set_args() (*natch.rules.Eq method*), 20
 set_args() (*natch.rules.Gt method*), 21
 set_args() (*natch.rules.Gte method*), 21
 set_args() (*natch.rules.Lt method*), 21
 set_args() (*natch.rules.Lte method*), 22
 set_args() (*natch.rules.Neq method*), 20
 set_args() (*natch.rules.NotContains method*), 22
 set_args() (*natch.rules.Partial method*), 23
 set_args() (*natch.rules.Pattern method*), 23
 set_hasher() (*natch.abstract.Registry method*), 9
 set_hasher() (*natch.core.Registry method*), 11
 set_index() (*natch.abstract.Registry method*), 9
 set_index() (*natch.core.Registry method*), 11
 set_kwargs() (*natch.abstract.Rule method*), 10
 set_kwargs() (*natch.rules.AllOf method*), 20
 set_kwargs() (*natch.rules.Any method*), 19
 set_kwargs() (*natch.rules.AnyOf method*), 19
 set_kwargs() (*natch.rules.Condition method*), 22
 set_kwargs() (*natch.rules.Contains method*), 22
 set_kwargs() (*natch.rules.Eq method*), 20
 set_kwargs() (*natch.rules.Gt method*), 21
 set_kwargs() (*natch.rules.Gte method*), 21
 set_kwargs() (*natch.rules.Lt method*), 21
 set_kwargs() (*natch.rules.Lte method*), 22
 set_kwargs() (*natch.rules.Neq method*), 20
 set_kwargs() (*natch.rules.NotContains method*), 22
 set_kwargs() (*natch.rules.Partial method*), 23
 set_kwargs() (*natch.rules.Pattern method*), 23

U

unregister() (*natch.abstract.Registry method*), 9
 unregister() (*natch.core.Registry method*), 11

W

with_traceback() (*natch.exceptions.NeverMatchesError method*), 15