
mpu Documentation

Release 0.23.0

Martin Thoma

Aug 11, 2020

CONTENTS

1	mpu	3
2	mpu.aws	7
3	mpu.datastructures	9
4	mpu.datetime	15
5	mpu.decorators	17
6	mpu.geometry	19
7	mpu.image	21
8	mpu.io	23
9	mpu.math	27
10	mpu.ml	31
11	mpu.path	33
12	mpu.pd	35
13	mpu.shell	37
14	mpu.string	39
15	mpu.units	45
	15.1 Module contents	45
	15.2 Allowed operations with Money	46
16	Indices and tables	47
	Python Module Index	49
	Index	51

This package contains various small functions and classes. All of the functionality is not offered by any mayor package.

Core design principles are:

- **Lightweight:**
 - No dependencies
 - Small file size
- **Documentation:** Every parameter is properly documented. For each opened issue or question I will think about adding the information to the docs
- **Testing:** >90% test coverage. For each issue found I will think about creating a test which could have shown the issue.

Please note that this is not in version 1.0 yet. So there will likely be breaking changes.

Contents:

mpu: Martins Python Utilities.

class `mpu.Location` (*latitude: float, longitude: float*)

Bases: `object`

Define a single point.

Parameters

- **latitude** (*float*) – in [-90, 90] - from North to South
- **longitude** (*float*) – in [-180, 180] - from West to East

distance (*there: mpu.Location*) → `float`

Calculate the distance from this location to there.

Parameters *there* (`Location`) –

Returns `distance_in_m`

Return type `float`

get_google_maps_link ()

Get a Google Maps link to this location.

property latitude

Getter for latitude.

property longitude

Getter for longitude.

`mpu.clip` (*number: Union[int, float], lowest: Union[None, int, float] = None, highest: Union[None, int, float] = None*) → `Union[int, float]`

Clip a number to a given lowest / highest value.

Parameters

- **number** (*number*) –
- **lowest** (*number, optional*) –
- **highest** (*number, optional*) –

Returns `clipped_number`

Return type `number`

Examples

```
>>> clip(42, lowest=0, highest=10)
10
```

`mpu.consistent_shuffle` (**lists: List[List[Any]]*) → Tuple[List[Any], ...]
Shuffle lists consistently.

Parameters **lists* – Variable length number of lists

Returns *shuffled_lists* – All of the lists are shuffled consistently

Return type tuple of lists

Examples

```
>>> import mpu, random; random.seed(8)
>>> mpu.consistent_shuffle([1,2,3], ['a', 'b', 'c'], ['A', 'B', 'C'])
([3, 2, 1], ['c', 'b', 'a'], ['C', 'B', 'A'])
```

`mpu.exception_logging` (*exctype, value, tb*)
Log exception by using the root logger.

Use it as `sys.excepthook = exception_logging`.

Parameters

- **exctype** (*type*) –
- **value** (*NameError*) –
- **tb** (*traceback*) –

`mpu.haversine_distance` (*origin: Tuple[float, float], destination: Tuple[float, float]*) → float
Calculate the Haversine distance.

Parameters

- **origin** (*Tuple[float, float]*) – (lat, long)
- **destination** (*Tuple[float, float]*) – (lat, long)

Returns *distance_in_km*

Return type float

Examples

```
>>> munich = (48.1372, 11.5756)
>>> berlin = (52.5186, 13.4083)
>>> round(haversine_distance(munich, berlin), 1)
504.2
```

```
>>> new_york_city = (40.712777777778, -74.005833333333) # NYC
>>> round(haversine_distance(berlin, new_york_city), 1)
6385.3
```

`mpu.is_in_intervall` (*value, min_value, max_value, name='variable'*)
Raise an exception if value is not in an interval.

Parameters

- **value** (*orderable*) –
- **min_value** (*orderable*) –
- **max_value** (*orderable*) –
- **name** (*str*) – Name of the variable to print in exception.

`mpu.parallel_for` (*loop_function*: *Callable*[*Any*, *Any*], *parameters*: *List*[*Tuple*[*Any*, ...]], *nb_threads*:
int = 100) → *List*[*Any*]

Execute the loop body in parallel.

Note: Race-Conditions Executing code in parallel can cause an error class called “race-condition”.

Parameters

- **loop_function** (*Callable*) – Python function which takes a tuple as input
- **parameters** (*List*[*Tuple*]) – Each element here should be executed in parallel.

Returns *return_values*

Return type list of return values

MPU.AWS

Convenience functions for AWS interactions.

```
class mpu.aws.ExistsStrategy
```

```
    Bases: enum.Enum
```

```
    Strategies what to do when a file already exists.
```

```
    ABORT = 'abort'
```

```
    RAISE = 'raise'
```

```
    REPLACE = 'replace'
```

```
class mpu.aws.S3Path(bucket_name, key)
```

```
    Bases: tuple
```

```
    property bucket_name
```

```
        Alias for field number 0
```

```
    property key
```

```
        Alias for field number 1
```

```
mpu.aws.list_files(bucket, prefix="", profile_name=None)
```

```
    List up to 1000 files in a bucket.
```

Parameters

- **bucket** (*str*) –
- **profile_name** (*str*, *optional*) – AWS profile

Returns s3_paths

Return type List[str]

```
mpu.aws.s3_download(source, destination=None, exists_strategy=<ExistsStrategy.RAISE: 'raise'>, profile_name=None)
```

```
    Copy a file from an S3 source to a local destination.
```

Parameters

- **source** (*str*) – Path starting with s3://, e.g. 's3://bucket-name/key/foo.bar'
- **destination** (*str*, *optional*) – If none is given, a temporary file is created
- **exists_strategy** (*{'raise', 'replace', 'abort'}*) – What is done when the destination already exists? * *ExistsStrategy.RAISE* means a RuntimeError is raised, * *ExistsStrategy.REPLACE* means the local file is replaced, * *ExistsStrategy.ABORT* means the download is not done.
- **profile_name** (*str*, *optional*) – AWS profile

Returns `download_path` – Path of the downloaded file.

Return type `str`

Raises `botocore.exceptions.NoCredentialsError` – Botocore is not able to find your credentials. Either specify `profile_name` or add the environment variables `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN`. See <https://boto3.readthedocs.io/en/latest/guide/configuration.html>

`mpu.aws.s3_read` (*source*, *profile_name=None*)

Read a file from an S3 source.

Parameters

- **source** (*str*) – Path starting with `s3://`, e.g. `'s3://bucket-name/key/foo.bar'`
- **profile_name** (*str*, *optional*) – AWS profile

Returns `content`

Return type `bytes`

Raises `botocore.exceptions.NoCredentialsError` – Botocore is not able to find your credentials. Either specify `profile_name` or add the environment variables `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN`. See <https://boto3.readthedocs.io/en/latest/guide/configuration.html>

`mpu.aws.s3_upload` (*source*, *destination*, *profile_name=None*)

Copy a file from a local source to an S3 destination.

Parameters

- **source** (*str*) –
- **destination** (*str*) – Path starting with `s3://`, e.g. `'s3://bucket-name/key/foo.bar'`
- **profile_name** (*str*, *optional*) – AWS profile

MPU.DATASTRUCTURES

Utility datastructures.

class mpu.datastructures.**EList** (*args)

Bases: list

Enhanced List.

This class supports every operation a normal list supports. Additionally, you can call it with a list as an argument.

Examples

```
>>> l = EList([2, 1, 0])
>>> l[2]
0
>>> l[[2, 0]]
[0, 2]
>>> l[1]
[0, 1, 2]
```

remove_indices (*indices*)

Remove rows by which have the given indices.

Parameters *indices* (*list*) –

Returns *filtered_list*

Return type *EList*

class mpu.datastructures.**Interval** (*left=None, right=None*)

Bases: *mpu.datastructures.IntervalLike*

Representation of an interval.

The empty interval is represented as *left=None, right=None*. Left and right have to be comparable. Typically, it would be numbers or dates.

Parameters

- **left** (*object*) –
- **right** (*object*) –

intersection (*other*)

Intersect two *IntervalLike* objects.

Parameters *other* (*IntervalLike*) –

Returns *intersected*

Return type *IntervalLike*

is_empty ()

Return if the interval is empty.

issubset (*other*)

Check if the interval “self” is completely inside of other.

Parameters **other** (*IntervalLike*) –

Returns **is_inside**

Return type bool

union (*other*)

Combine two Intervals.

Parameters **other** (*IntervalLike*) –

Returns **interval_union**

Return type *IntervalLike*

class mpu.datastructures.**IntervalLike**

Bases: object

Anything like an interval or a union of an interval.

As mpu supports Python 2.7 until 2020 and does not want to include extra dependencies, ABC cannot be used.

intersection (*other*)

Intersect two IntervalLike objects.

Parameters **other** (*IntervalLike*) –

Returns **intersected**

Return type *IntervalLike*

is_empty ()

Return if the IntervalLike is empty.

issubset (*other*)

Check if the interval “self” is completely inside of other.

Parameters **other** (*IntervalLike*) –

Returns **is_inside**

Return type bool

union (*other*)

Combine two Intervals.

Parameters **other** (*IntervalLike*) –

Returns **interval_union**

Return type *IntervalLike*

class mpu.datastructures.**IntervalUnion** (*intervals*)

Bases: *mpu.datastructures.IntervalLike*

A union of Intervals.

intersection (*other*)

Return the intersection between this IntervalUnion and another object.

This changes the object itself!

Parameters `other` (`Interval` or `IntervalUnion`)–

Returns `intersection`

Return type `Interval` or `IntervalUnion`

`is_empty()`

Return if the `IntervalUnion` is empty.

`issubset(other)`

Check if this `IntervalUnion` is completely inside of `other`.

Parameters `other` (`Interval` or `IntervalUnion`)–

Returns `is_inside`

Return type `bool`

`union(other)`

Return the union between this `IntervalUnion` and another object.

Parameters `other` (`Interval` or `IntervalUnion`)–

Returns `union`

Return type `Interval` or `IntervalUnion`

`mpu.datastructures.dict_merge(dict_left, dict_right, merge_method='take_left_shallow')`

Merge two dictionaries.

This method does NOT modify `dict_left` or `dict_right`!

Apply this method multiple times if the dictionary is nested.

Parameters

- `dict_left` (`dict`)–
- `dict_right` (`dict`)–
- `merge_method` (`{'take_left_shallow', 'take_left_deep', 'take_right_shallow', 'take_right_deep', 'sum'}`)–

- `take_left_shallow`: Use both dictionaries. If both have the same key, take the value of `dict_left`
- `take_left_deep`: If both dictionaries have the same key and the value is a dict for both again, then merge those sub-dictionaries
- `take_right_shallow`: See `take_left_shallow`
- `take_right_deep`: See `take_left_deep`
- `sum`: sum up both dictionaries. If one does not have a value for a key of the other, assume the missing value to be zero.

Returns `merged_dict`

Return type `dict`

Examples

```
>>> dict_merge({'a': 1, 'b': 2}, {'c': 3}) == {'a': 1, 'b': 2, 'c': 3}
True
```

```
>>> out = dict_merge({'a': {'A': 1}},
...                 {'a': {'A': 2, 'B': 3}}, 'take_left_deep')
>>> expected = {'a': {'A': 1, 'B': 3}}
>>> out == expected
True
```

```
>>> out = dict_merge({'a': {'A': 1}},
...                 {'a': {'A': 2, 'B': 3}}, 'take_left_shallow')
>>> expected = {'a': {'A': 1}}
>>> out == expected
True
```

```
>>> out = dict_merge({'a': 1, 'b': {'c': 2}},
...                 {'b': {'c': 3, 'd': 4}},
...                 'sum')
>>> expected = {'a': 1, 'b': {'c': 5, 'd': 4}}
>>> out == expected
True
```

`mpu.datastructures.does_keychain_exist` (*dict_*, *list_*)

Check if a sequence of keys exist in a nested dictionary.

Parameters

- **dict_** (*Dict[str/int/tuple, Any]*) –
- **list_** (*List[str/int/tuple]*) –

Returns `keychain_exists`

Return type `bool`

Examples

```
>>> d = {'a': {'b': {'c': 'd'}}}
>>> l_exists = ['a', 'b']
>>> does_keychain_exist(d, l_exists)
True
```

```
>>> l_no_existant = ['a', 'c']
>>> does_keychain_exist(d, l_no_existant)
False
```

`mpu.datastructures.flatten` (*iterable*, *string_flattening=False*)

Flatten an given iterable of iterables into one list.

Parameters

- **iterable** (*iterable*) –
- **string_flattening** (*bool*) – If this is `False`, then strings are NOT flattened

Returns `flat_list`

Return type List

Examples

```
>>> flatten([1, [2, [3]]])
[1, 2, 3]
```

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

```
>>> flatten(EList([EList([1, 2]), (3, [4, [[5]]])]))
[1, 2, 3, 4, 5]
```

`mpu.datastructures.set_dict_value` (*dictionary*, *keys*, *value*)
Set a value in a (nested) dictionary by defining a list of keys.

Note: Side-effects This function does not make a copy of dictionary, but directly edits it.

Parameters

- **dictionary** (*dict*) –
- **keys** (*List[Any]*) –
- **value** (*object*) –

Returns dictionary

Return type dict

Examples

```
>>> d = {'a': {'b': {'c': 'x', 'f': 'g'}, 'd': 'e'}}
>>> expected = {'a': {'b': {'c': 'foobar', 'f': 'g'}, 'd': 'e'}}
>>> set_dict_value(d, ['a', 'b', 'c'], 'foobar') == expected
True
```


MPU.DATETIME

Datetime related utility functions.

`mpu.datetime.add_time` (*datetime_obj*, *days=0*, *hours=0*, *minutes=0*, *seconds=0*)
Add time to a timezone-aware datetime object.

This keeps the timezone correct, even if it changes due to daylight saving time (DST).

Parameters

- **datetime_obj** (*datetime.datetime*) –
- **days** (*int*) –
- **hours** (*int*) –
- **minutes** (*int*) –
- **seconds** (*int*) –

Returns datetime

Return type `datetime.datetime`

`mpu.datetime.generate` (*minimum*, *maximum*, *local_random=<random.Random object>*)
Generate a random date.

The generated dates are uniformly distributed.

Parameters

- **minimum** (*datetime object*) –
- **maximum** (*datetime object*) –
- **local_random** (*random.Random*) –

Returns generated_date

Return type `datetime object`

Examples

```
>>> import random; r = random.Random(); r.seed(0)
>>> from datetime import datetime
```

```
>>> generate(datetime(2018, 1, 1), datetime(2018, 1, 2), local_random=r)
datetime.datetime(2018, 1, 1, 20, 15, 58, 47972)
```

```
>>> generate(datetime(2018, 1, 1), datetime(2018, 1, 2), local_random=r)
datetime.datetime(2018, 1, 1, 18, 11, 27, 260414)
```

MPU.DECORATORS

Decorators which are not in *functools*.

`mpu.decorators.deprecated` (*func*)

Mark functions as deprecated.

It will result in a warning being emitted when the function is used.

`mpu.decorators.timing` (*func*)

Measure the execution time of a function call and print the result.

MPU.GEOMETRY

Create and manipulate two-dimensional geometrical entities such as lines.

For more advanced use cases, see:

- [sympy.geometry](#)
- [Shapely](#)

```
class mpu.geometry.LineSegment (p1: mpu.geometry.Point, p2: mpu.geometry.Point, name: str =  
                                'LineSegment')
```

Bases: object

A line segment a a 2-dimensional Euclidean space.

Parameters

- **p1** (Point) –
- **p2** (Point) –

angle () → float

Get the angle of this line.

bounding_box () → Tuple[mpu.geometry.Point, mpu.geometry.Point]

Get the bounding box of this line represented by two points.

The p1 point is in the lower left corner, the p2 one at the upper right corner.

intersect (other) → Union[None, mpu.geometry.LineSegment, mpu.geometry.Point]

Get the intersection between this LineSegment and another LineSegment.

Parameters other (LineSegment) –

Returns intersection

Return type Union[None, LineSegment, Point]

is_point ()

Check if this LineSegment is a point.

length () → float

Get the length of this line segment.

simplify ()

Simplify this line segment to a point, if possible.

```
class mpu.geometry.Point (x: float, y: float)
```

Bases: object

A point in a 2-dimensional Euclidean space.

Parameters

- **x** (*float*) –
- **y** (*float*) –

`mpu.geometry.crossproduct` (*a*: `mpu.geometry.Point`, *b*: `mpu.geometry.Point`) → `float`
 Get the cross product of two points.

`mpu.geometry.do_bounding_boxes_intersect` (*a*: `Tuple[mpu.geometry.Point, mpu.geometry.Point]`, *b*: `Tuple[mpu.geometry.Point, mpu.geometry.Point]`) → `bool`

Check if bounding boxes do intersect.

If one bounding box touches the other, they do intersect.

`mpu.geometry.do_lines_intersect` (*a*: `mpu.geometry.LineSegment`, *b*: `mpu.geometry.LineSegment`) → `bool`

Check if LineSegments a and b intersect.

`mpu.geometry.get_all_intersecting_lines_by_brute_force` (*lines*: `List[mpu.geometry.LineSegment]`) → `Set[FrozenSet[mpu.geometry.LineSegment]]`

Get all intersectionLines by applying a brute force algorithm.

Parameters `lines` (*all lines you want to check, in no order*) –

Returns

Return type a list that contains all pairs of intersecting lines

`mpu.geometry.is_point_on_line` (*a*: `mpu.geometry.LineSegment`, *b*: `mpu.geometry.Point`) → `bool`
 Check if point b is on LineSegment a.

`mpu.geometry.is_point_right_of_line` (*a*: `mpu.geometry.LineSegment`, *b*: `mpu.geometry.Point`) → `bool`
 Check if point b is right of line a.

`mpu.geometry.line_segment_touches_or_crosses_line` (*a*: `mpu.geometry.LineSegment`, *b*: `mpu.geometry.LineSegment`) → `bool`
 Check if line segment a touches or crosses line segment b.

MPU.IMAGE

Image manipulation.

`mpu.image.get_meta(filepath)`

Get meta-information of an image.

Parameters `filepath` (*str*) –

Returns `meta`

Return type `dict`

Reading and writing common file formats.

`mpu.io.download` (*source: str, sink=None*)
Download a file.

Parameters

- **source** (*str*) – Where the file comes from. Some URL.
- **sink** (*str or None (default: same filename in current directory)*) – Where the file gets stored. Some filepath in the local file system.

`mpu.io.get_access_datetime` (*filepath: str*)
Get the last time filepath was accessed.

Parameters `filepath` (*str*) –

Returns `access_datetime`

Return type `datetime.datetime`

`mpu.io.get_creation_datetime` (*filepath: str*)
Get the date that a file was created.

Parameters `filepath` (*str*) –

Returns `creation_datetime`

Return type `datetime.datetime` or `None`

`mpu.io.get_file_meta` (*filepath: str*)
Get meta-information about a file.

Parameters `filepath` (*str*) –

Returns `meta`

Return type `dict`

`mpu.io.get_modification_datetime` (*filepath: str*)
Get the datetime that a file was last modified.

Parameters `filepath` (*str*) –

Returns `modification_datetime`

Return type `datetime.datetime`

`mpu.io.gzip_file` (*source: str, sink: str*)
Create a GZIP file from a source file.

Parameters

- **source** (*str*) – Filepath
- **sink** (*str*) – Filepath

`mpu.io.hash` (*filepath: str, method='sha1', buffer_size=65536*)

Calculate a hash of a local file.

Parameters

- **filepath** (*str*) –
- **method** (*{'sha1', 'md5'}*) –
- **buffer_size** (*int, optional (default: 65536 byte = 64 KiB)*) – in byte

Returns hash

Return type `str`

`mpu.io.read` (*filepath: str, **kwargs*)

Read a file.

Supported formats:

- CSV
- JSON, JSONL
- pickle

Parameters

- **filepath** (*str*) – Path to the file that should be read. This methods action depends mainly on the file extension.
- **kwargs** (*dict*) – Any keywords for the specific file format. For CSV, this is 'delimiter', 'quotechar', 'skiprows', 'format'

Returns data

Return type `Union[str, bytes]` or other (e.g. `format=dicts`)

`mpu.io.urlread` (*url: str, encoding='utf8'*)

Read the content of an URL.

Parameters `url` (*str*) –

Returns content

Return type `str`

`mpu.io.write` (*filepath: str, data, **kwargs*)

Write a file.

Supported formats:

- CSV
- JSON, JSONL
- pickle

Parameters

- **filepath** (*str*) – Path to the file that should be read. This methods action depends mainly on the file extension.

- **data** (*dict or list*) – Content that should be written
- **kwargs** (*dict*) – Any keywords for the specific file format.

Returns data

Return type str or bytes

Mathematical functions which are not adequately covered by standard libraries.

Standard libraries are:

- `math`
- `scipy`
- `sympy`

`mpu.math.argmax` (*iterable*)

Find the first index of the biggest value in the iterable.

Parameters `iterable` (*iterable*) –

Returns `argmax`

Return type `int`

Examples

```
>>> argmax([0, 0, 0])
0
>>> argmax([1, 0, 0])
0
>>> argmax([0, 1, 0])
1
>>> argmax([])
```

`mpu.math.factorize` (*number*)

Get the prime factors of an integer except for 1.

Parameters `number` (*int*) –

Returns `primes`

Return type `iterable`

Examples

```
>>> factorize(-17)
[-1, 17]
>>> factorize(8)
[2, 2, 2]
>>> factorize(3**25)
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
>>> factorize(1)
[1]
```

`mpu.math.gcd(a: int, b: int) → int`

Calculate the greatest common divisor.

Currently, this uses the Euclidean algorithm.

Parameters

- **a** (*int*) – Non-zero
- **b** (*int*) –

Returns `greatest_common_divisor`

Return type `int`

Examples

```
>>> gcd(1, 7)
1
>>> gcd(-1, -1)
1
>>> gcd(1337, 42)
7
>>> gcd(-1337, -42)
7
>>> gcd(120, 364)
4
>>> gcd(273, 1870)
1
```

`mpu.math.generate_primes()`

Generate an infinite sequence of prime numbers.

The algorithm was originally written by David Eppstein, UC Irvine. See: <http://code.activestate.com/recipes/117119/>

Examples

```
>>> g = generate_primes()
>>> next(g)
2
>>> next(g)
3
>>> next(g)
5
```


`mpu.math.is_prime(number)`

Check if a number is prime.

Parameters `number` (*int*) –

Returns `is_prime_number`

Return type `bool`

Examples

```
>>> is_prime(-17)
False
>>> is_prime(17)
True
>>> is_prime(47055833459)
True
```

`mpu.math.product(iterable, start=1)`

Calculate the product of the iterables.

Parameters

- **iterable** (*iterable*) – List, tuple or similar which contains numbers
- **start** (*number, optional (default: 1)*) –

Returns `product`

Return type `number`

Examples

```
>>> product([1, 2, 3, 4, 5])
120
>>> product([])
1
```

`mpu.math.round_down(x, decimal_places)`

Round a float down to `decimal_places`.

Parameters

- **x** (*float*) –
- **decimal_places** (*int*) –

Returns `rounded_float`

Return type `float`

Examples

```
>>> round_down(1.23456, 3)
1.234
>>> round_down(1.23456, 2)
1.23
```

`mpu.math.round_up(x, decimal_places)`

Round a float up to `decimal_places`.

Parameters

- **x** (*float*) –
- **decimal_places** (*int*) –

Returns `rounded_float`

Return type float

Examples

```
>>> round_up(1.2344, 3)
1.235
>>> round_up(1.234, 3)
1.234
>>> round_up(1.23456, 3)
1.235
>>> round_up(1.23456, 2)
1.24
```

Machine Learning functions.

`mpu.ml.indices2one_hot` (*indices*, *nb_classes*)

Convert an iterable of indices to one-hot encoded list.

You might also be interested in `sklearn.preprocessing.OneHotEncoder`

Parameters

- **indices** (*iterable*) – iterable of indices
- **nb_classes** (*int*) – Number of classes
- **dtype** (*type*) –

Returns `one_hot`

Return type list

Examples

```
>>> indices2one_hot([0, 1, 1], 3)
[[1, 0, 0], [0, 1, 0], [0, 1, 0]]
>>> indices2one_hot([0, 1, 1], 2)
[[1, 0], [0, 1], [0, 1]]
```

`mpu.ml.one_hot2indices` (*one_hots*)

Convert an iterable of one-hot encoded targets to a list of indices.

Parameters `one_hot` (*list*) –

Returns `indices`

Return type list

Examples

```
>>> one_hot2indices([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
[0, 1, 2]
```

```
>>> one_hot2indices([[1, 0], [1, 0], [0, 1]])
[0, 0, 1]
```


MPU.PATH

Functions for path manipulation and retrieval of files.

`mpu.path.get_all_files` (*root*, *followlinks=False*)

Get all files within the given root directory.

Note that this list is not ordered.

Parameters

- **root** (*str*) – Path to a directory
- **followlinks** (*bool*, *optional* (default: *False*)) –

Returns `filepath` – List of absolute paths to files

Return type `list`

`mpu.path.get_from_package` (*package_name*, *path*)

Get the absolute path to a file in a package.

Parameters

- **package_name** (*str*) – e.g. 'mpu'
- **path** (*str*) – Path within a package

Returns `filepath`

Return type `str`

Pandas utility functions.

`mpu.pd.describe(df, dtype=None)`

Print a description of a Pandas dataframe.

Parameters

- **df** (*Pandas.DataFrame*) –
- **dtype** (*dict*) – Maps column names to types

`mpu.pd.example_df()`

Create an example dataframe.

MPU.SHELL

Enhancing printed terminal output.

```
class mpu.shell.Codes
    Bases: object

    Escape sequences for enhanced shell output.

    BACKGROUND_BLACK = '\x1b[40m'
    BACKGROUND_BLUE = '\x1b[44m'
    BACKGROUND_CYAN = '\x1b[46m'
    BACKGROUND_DARK_GRAY = '\x1b[100m'
    BACKGROUND_DEFAULT = '\x1b[49m'
    BACKGROUND_GREEN = '\x1b[42m'
    BACKGROUND_LIGHT_BLUE = '\x1b[104m'
    BACKGROUND_LIGHT_CYAN = '\x1b[106m'
    BACKGROUND_LIGHT_GRAY = '\x1b[47m'
    BACKGROUND_LIGHT_GREEN = '\x1b[102m'
    BACKGROUND_LIGHT_MAGENTA = '\x1b[105m'
    BACKGROUND_LIGHT_RED = '\x1b[101m'
    BACKGROUND_LIGHT_YELLOW = '\x1b[103m'
    BACKGROUND_MAGENTA = '\x1b[45m'
    BACKGROUND_RED = '\x1b[41m'
    BACKGROUND_WHITE = '\x1b[107m'
    BACKGROUND_YELLOW = '\x1b[43m'

    BLACK = '\x1b[30m'
    BLINK = '\x1b[5m'
    BLUE = '\x1b[34m'
    BOLD = '\x1b[1m'
    CYAN = '\x1b[36m'
    DARK_GRAY = '\x1b[90m'
    DEFAULT = '\x1b[39m'
```

```
DIM = '\x1b[2m'  
GREEN = '\x1b[32m'  
HIDDEN = '\x1b[8m'  
LIGHT_BLUE = '\x1b[94m'  
LIGHT_CYAN = '\x1b[96m'  
LIGHT_GRAY = '\x1b[37m'  
LIGHT_GREEN = '\x1b[92m'  
LIGHT_MAGENTA = '\x1b[95m'  
LIGHT_RED = '\x1b[91m'  
LIGHT_YELLOW = '\x1b[93m'  
MAGENTA = '\x1b[35m'  
RED = '\x1b[31m'  
RESET_ALL = '\x1b[0m'  
RESET_BLINK = '\x1b[25m'  
RESET_BOLD = '\x1b[21m'  
RESET_DIM = '\x1b[22m'  
RESET_HIDDEN = '\x1b[28m'  
RESET_REVERSE = '\x1b[27m'  
RESET_UNDERLINED = '\x1b[24m'  
REVERSE = '\x1b[7m'  
UNDERLINED = '\x1b[4m'  
WHITE = '\x1b[97m'  
YELLOW = '\x1b[33m'
```

`mpu.shell.print_table` (*table*)
Print as a table.

I recommend looking at [*tabulate*](<https://pypi.org/project/tabulate/>).

Parameters `table` (*list*) –

Examples

```
>>> print_table([[1, 2, 3], [41, 0, 1]])  
1 2 3  
41 0 1
```

`mpu.shell.text_input` (*text*)
Ask the user for textual input.

Parameters `text` (*str*) – What the user sees.

Returns `inputted_text` – What the user wrote.

Return type `str`

MPU.STRING

String manipulation, verification and formatting.

For more complex checks, you might want to use the [validators](<http://validators.readthedocs.io>) package.

`mpu.string.human_readable_bytes` (*nb_bytes*: *Union[int, float]*, *suffix*: *str = 'B'*) → *str*
Convert a byte number into a human readable format.

Parameters

- **nb_bytes** (*Union[int, float]*) –
- **suffix** (*str*, *optional* (default: "B")) –

Returns *size_str*

Return type *str*

Examples

```
>>> human_readable_bytes(123)
'123.0 B'
```

```
>>> human_readable_bytes(1025)
'1.0 KiB'
```

```
>>> human_readable_bytes(9671406556917033397649423)
'8.0 YiB'
```

`mpu.string.is_email` (*potential_email_address*: *str*) → *bool*
Check if *potential_email_address* is a valid e-mail address.

Please note that this function has no false-negatives but many false-positives. So if it returns that the input is not a valid e-mail address, it certainly isn't. If it returns True, it might still be invalid. For example, the domain could not be registered.

Parameters *potential_email_address* (*str*) –

Returns *is_email*

Return type *bool*

Examples

```
>>> is_email('')
False
>>> is_email('info@martin-thoma.de')
True
>>> is_email('info@math.martin-thoma.de')
True
>>> is_email('Martin Thoma <info@martin-thoma.de>')
False
>>> is_email('info@martin-thoma')
False
>>> is_email('Martin <>')
False
```

`mpu.string.is_float` (*potential_float: str*) → bool
Check if `potential_float` is a valid float.

Returns `is_float`

Return type bool

Examples

```
>>> is_float('123')
True
>>> is_float('1234567890123456789')
True
>>> is_float('0')
True
>>> is_float('-123')
True
>>> is_float('123.45')
True
>>> is_float('a')
False
>>> is_float('0x8')
False
```

`mpu.string.is_iban` (*potential_iban: str*) → bool
Check if a string is a valid IBAN number.

IBAN is described in ISO 13616-1:2007 Part 1.

Spaces are ignored.

CODE 0 = always zero b = BIC or National Bank code c = Account number i = holder's kennitala (national identification number) k = IBAN check digits n = Branch number t = Account type x = National check digit or character

Examples

```
>>> is_iban('DE89 3704 0044 0532 0130 00')
True
>>> is_iban('DE89 3704 0044 0532 0130 01')
False
```

`mpu.string.is_int` (*potential_int: str*) → bool
Check if *potential_int* is a valid integer.

Parameters *potential_int* (*str*) –

Returns `is_int`

Return type bool

Examples

```
>>> is_int('123')
True
>>> is_int('1234567890123456789')
True
>>> is_int('0')
True
>>> is_int('-123')
True
>>> is_int('123.45')
False
>>> is_int('a')
False
>>> is_int('0x8')
False
```

`mpu.string.is_ipv4` (*potential_ipv4: str*, *allow_leading_zeros: bool = False*, *allow_shortened_addresses=False*) → bool
Check if a string is a valid IPv4 address.

Parameters *potential_ipv4* (*str*) –

Returns `is_valid`

Return type bool

Examples

```
>>> is_ipv4("192.168.0.4")
True
>>> is_ipv4("192.168..4")
False
>>> is_ipv4("192.168.01.4", allow_leading_zeros=True)
True
>>> is_ipv4("192.168.01.4", allow_leading_zeros=False)
False
>>> is_ipv4("256.168.01.4")
False
>>> is_ipv4("4", allow_shortened_addresses=True)
```

(continues on next page)

(continued from previous page)

```
True
>>> is_ipv4("4", allow_shortened_addresses=False)
False
```

`mpu.string.is_none` (*string_*: *str*, *default*='raise') → bool

Check if a string is equivalent to None.

Parameters

- **string_** (*str*) –
- **default** (*{'raise', False}*) – Default behaviour if none of the “None” strings is detected.

Returns `is_none`

Return type bool

Examples

```
>>> is_none('2', default=False)
False
>>> is_none('undefined', default=False)
True
```

`mpu.string.str2bool` (*string_*: *str*, *default*='raise') → bool

Convert a string to a bool.

Parameters

- **string_** (*str*) –
- **default** (*{'raise', False}*) – Default behaviour if none of the “true” strings is detected.

Returns boolean

Return type bool

Examples

```
>>> str2bool('True')
True
>>> str2bool('1')
True
>>> str2bool('0')
False
```

`mpu.string.str2bool_or_none` (*string_*: *str*, *default*='raise') → Optional[bool]

Convert a string to a bool or to None.

Parameters

- **string_** (*str*) –
- **default** (*{'raise', False}*) – Default behaviour if none of the “true” or “none” strings is detected.

Returns `bool_or_none`

Return type bool or None

Examples

```
>>> str2bool_or_none('True')
True
>>> str2bool_or_none('1')
True
>>> str2bool_or_none('0')
False
>>> str2bool_or_none('undefined')
```

`mpu.string.str2float_or_none(string_: str) → Optional[float]`
Convert a string to a float or to None.

Parameters `string_ (str)` –

Returns `float_or_none`

Return type float or None

Examples

```
>>> str2float_or_none('1')
1.0
>>> str2float_or_none('1.2')
1.2
>>> str2float_or_none('undefined')
```

`mpu.string.str2int_or_none(string_: str) → Optional[int]`
Convert a string to a int or to None.

Parameters `string_ (str)` –

Returns `int_or_none`

Return type int or None

Examples

```
>>> str2int_or_none('2')
2
>>> str2int_or_none('undefined')
```

`mpu.string.str2str_or_none(string_: str) → Optional[str]`
Convert a string to a str or to None.

Parameters `string_ (str)` –

Returns `str_or_none`

Return type bool or None

Examples

```
>>> str2str_or_none('True')
'True'
>>> str2str_or_none('1')
'1'
>>> str2str_or_none('0')
'0'
>>> str2str_or_none('undefined')
```


15.1 Module contents

Handle units - currently only currencies.

class `mpu.units.Currency` (*name, code, numeric_code, symbol, exponent, entities, withdrawal_date, subunits*)

Bases: `object`

Currency base class which contains information similar to ISO 4217.

for_json ()

Return a JSON-serializable object.

classmethod from_json (*json*)

Create a Currency object from a JSON dump.

class `mpu.units.Money` (*value: Union[str, fractions.Fraction, int, Tuple], currency: Union[str, mpu.units.Currency]*)

Bases: `object`

Unit of account.

Parameters

- **value** (*Union[str, fractions.Fraction, int, Tuple]*)–
- **currency** (*Currency or str*)–

Examples

```
>>> rent = Money(500, 'USD')
>>> '{:.2f,shortcode}'.format(rent)
'USD 500.00'
>>> '{:.2f,postshortcode}'.format(rent)
'500.00 USD'
>>> '{:.2f,symbol}'.format(rent)
'$500.00'
>>> '{:.2f,postsymbol}'.format(rent)
'500.00$'
>>> '{:.2f}'.format(rent)
'500.00 USD'
```

for_json ()

Return a JSON-serializable object.

classmethod `from_json` (*json*)

Create a Money object from a JSON dump.

`mpu.units.get_currency` (*currency_str*)

Convert an identifier for a currency into a currency object.

Parameters `currency_str` (*str*) –

Returns `currency`

Return type *Currency*

15.2 Allowed operations with Money

Here you can see which operations are allowed by two Money objects of currencies (A and B):

Money A	Operator	Money A	Money B	int, Fraction
	+ , -	Money A	N/A	N/A
	*	N/A	N/A	Money A
	/	N/A	N/A	N/A
	//	Fraction	N/A	Money A
	>, >=, <, <=	Bool	N/A	N/A
	==	Bool	False	False

INDICES AND TABLES

- modindex
- search

PYTHON MODULE INDEX

m

- mpu, 3
- mpu.aws, 7
- mpu.datastructures, 9
- mpu.datetime, 15
- mpu.decorators, 17
- mpu.geometry, 19
- mpu.image, 21
- mpu.io, 23
- mpu.math, 27
- mpu.ml, 31
- mpu.path, 33
- mpu.pd, 35
- mpu.shell, 37
- mpu.string, 39
- mpu.units, 45

A

ABORT (*mpu.aws.ExistsStrategy* attribute), 7
 add_time() (*in module mpu.datetime*), 15
 angle() (*mpu.geometry.LineSegment* method), 19
 argmax() (*in module mpu.math*), 27

B

BACKGROUND_BLACK (*mpu.shell.Codes* attribute), 37
 BACKGROUND_BLUE (*mpu.shell.Codes* attribute), 37
 BACKGROUND_CYAN (*mpu.shell.Codes* attribute), 37
 BACKGROUND_DARK_GRAY (*mpu.shell.Codes* attribute), 37
 BACKGROUND_DEFAULT (*mpu.shell.Codes* attribute), 37
 BACKGROUND_GREEN (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_BLUE (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_CYAN (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_GRAY (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_GREEN (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_MAGENTA (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_RED (*mpu.shell.Codes* attribute), 37
 BACKGROUND_LIGHT_YELLOW (*mpu.shell.Codes* attribute), 37
 BACKGROUND_MAGENTA (*mpu.shell.Codes* attribute), 37
 BACKGROUND_RED (*mpu.shell.Codes* attribute), 37
 BACKGROUND_WHITE (*mpu.shell.Codes* attribute), 37
 BACKGROUND_YELLOW (*mpu.shell.Codes* attribute), 37
 BLACK (*mpu.shell.Codes* attribute), 37
 BLINK (*mpu.shell.Codes* attribute), 37
 BLUE (*mpu.shell.Codes* attribute), 37
 BOLD (*mpu.shell.Codes* attribute), 37
 bounding_box() (*mpu.geometry.LineSegment* method), 19
 bucket_name() (*mpu.aws.S3Path* property), 7

C

clip() (*in module mpu*), 3
 Codes (*class in mpu.shell*), 37
 consistent_shuffle() (*in module mpu*), 4
 crossproduct() (*in module mpu.geometry*), 20
 Currency (*class in mpu.units*), 45
 CYAN (*mpu.shell.Codes* attribute), 37

D

DARK_GRAY (*mpu.shell.Codes* attribute), 37
 DEFAULT (*mpu.shell.Codes* attribute), 37
 deprecated() (*in module mpu.decorators*), 17
 describe() (*in module mpu.pd*), 35
 dict_merge() (*in module mpu.datastructures*), 11
 DIM (*mpu.shell.Codes* attribute), 37
 distance() (*mpu.Location* method), 3
 do_bounding_boxes_intersect() (*in module mpu.geometry*), 20
 do_lines_intersect() (*in module mpu.geometry*), 20
 does_keychain_exist() (*in module mpu.datastructures*), 12
 download() (*in module mpu.io*), 23

E

EList (*class in mpu.datastructures*), 9
 example_df() (*in module mpu.pd*), 35
 exception_logging() (*in module mpu*), 4
 ExistsStrategy (*class in mpu.aws*), 7

F

factorize() (*in module mpu.math*), 27
 flatten() (*in module mpu.datastructures*), 12
 for_json() (*mpu.units.Currency* method), 45
 for_json() (*mpu.units.Money* method), 45
 from_json() (*mpu.units.Currency* class method), 45
 from_json() (*mpu.units.Money* class method), 45

G

gcd() (*in module mpu.math*), 28
 generate() (*in module mpu.datetime*), 15

generate_primes() (in module *mpu.math*), 28
 get_access_datetime() (in module *mpu.io*), 23
 get_all_files() (in module *mpu.path*), 33
 get_all_intersecting_lines_by_brute_force() (in module *mpu.geometry*), 20
 get_creation_datetime() (in module *mpu.io*), 23
 get_currency() (in module *mpu.units*), 46
 get_file_meta() (in module *mpu.io*), 23
 get_from_package() (in module *mpu.path*), 33
 get_google_maps_link() (in module *mpu.Location* method), 3
 get_meta() (in module *mpu.image*), 21
 get_modification_datetime() (in module *mpu.io*), 23
 GREEN (*mpu.shell.Codes* attribute), 38
 gzip_file() (in module *mpu.io*), 23

H

hash() (in module *mpu.io*), 24
 haversine_distance() (in module *mpu*), 4
 HIDDEN (*mpu.shell.Codes* attribute), 38
 human_readable_bytes() (in module *mpu.string*), 39

I

indices2one_hot() (in module *mpu.ml*), 31
 intersect() (*mpu.geometry.LineSegment* method), 19
 intersection() (*mpu.datastructures.Interval* method), 9
 intersection() (*mpu.datastructures.IntervalLike* method), 10
 intersection() (*mpu.datastructures.IntervalUnion* method), 10
 Interval (class in *mpu.datastructures*), 9
 IntervalLike (class in *mpu.datastructures*), 10
 IntervalUnion (class in *mpu.datastructures*), 10
 is_email() (in module *mpu.string*), 39
 is_empty() (*mpu.datastructures.Interval* method), 10
 is_empty() (*mpu.datastructures.IntervalLike* method), 10
 is_empty() (*mpu.datastructures.IntervalUnion* method), 11
 is_float() (in module *mpu.string*), 40
 is_iban() (in module *mpu.string*), 40
 is_in_intervall() (in module *mpu*), 4
 is_int() (in module *mpu.string*), 41
 is_ipv4() (in module *mpu.string*), 41
 is_none() (in module *mpu.string*), 42
 is_point() (*mpu.geometry.LineSegment* method), 19
 is_point_on_line() (in module *mpu.geometry*), 20
 is_point_right_of_line() (in module *mpu.geometry*), 20
 is_prime() (in module *mpu.math*), 28

issubset() (*mpu.datastructures.Interval* method), 10
 issubset() (*mpu.datastructures.IntervalLike* method), 10
 issubset() (*mpu.datastructures.IntervalUnion* method), 11

K

key() (*mpu.aws.S3Path* property), 7

L

latitude() (*mpu.Location* property), 3
 length() (*mpu.geometry.LineSegment* method), 19
 LIGHT_BLUE (*mpu.shell.Codes* attribute), 38
 LIGHT_CYAN (*mpu.shell.Codes* attribute), 38
 LIGHT_GRAY (*mpu.shell.Codes* attribute), 38
 LIGHT_GREEN (*mpu.shell.Codes* attribute), 38
 LIGHT_MAGENTA (*mpu.shell.Codes* attribute), 38
 LIGHT_RED (*mpu.shell.Codes* attribute), 38
 LIGHT_YELLOW (*mpu.shell.Codes* attribute), 38
 line_segment_touches_or_crosses_line() (in module *mpu.geometry*), 20
 LineSegment (class in *mpu.geometry*), 19
 list_files() (in module *mpu.aws*), 7
 Location (class in *mpu*), 3
 longitude() (*mpu.Location* property), 3

M

MAGENTA (*mpu.shell.Codes* attribute), 38
 module
 mpu, 3
 mpu.aws, 7
 mpu.datastructures, 9
 mpu.datetime, 15
 mpu.decorators, 17
 mpu.geometry, 19
 mpu.image, 21
 mpu.io, 23
 mpu.math, 27
 mpu.ml, 31
 mpu.path, 33
 mpu.pd, 35
 mpu.shell, 37
 mpu.string, 39
 mpu.units, 45
 Money (class in *mpu.units*), 45
mpu
 module, 3
mpu.aws
 module, 7
mpu.datastructures
 module, 9
mpu.datetime
 module, 15
mpu.decorators

module, 17
 mpu.geometry
 module, 19
 mpu.image
 module, 21
 mpu.io
 module, 23
 mpu.math
 module, 27
 mpu.ml
 module, 31
 mpu.path
 module, 33
 mpu.pd
 module, 35
 mpu.shell
 module, 37
 mpu.string
 module, 39
 mpu.units
 module, 45

O

one_hot2indices() (in module mpu.ml), 31

P

parallel_for() (in module mpu), 5
 Point (class in mpu.geometry), 19
 print_table() (in module mpu.shell), 38
 product() (in module mpu.math), 29

R

RAISE (mpu.aws.ExistsStrategy attribute), 7
 read() (in module mpu.io), 24
 RED (mpu.shell.Codes attribute), 38
 remove_indices() (mpu.datastructures.EList
 method), 9
 REPLACE (mpu.aws.ExistsStrategy attribute), 7
 RESET_ALL (mpu.shell.Codes attribute), 38
 RESET_BLINK (mpu.shell.Codes attribute), 38
 RESET_BOLD (mpu.shell.Codes attribute), 38
 RESET_DIM (mpu.shell.Codes attribute), 38
 RESET_HIDDEN (mpu.shell.Codes attribute), 38
 RESET_REVERSE (mpu.shell.Codes attribute), 38
 RESET_UNDERLINED (mpu.shell.Codes attribute), 38
 REVERSE (mpu.shell.Codes attribute), 38
 round_down() (in module mpu.math), 29
 round_up() (in module mpu.math), 30

S

s3_download() (in module mpu.aws), 7
 s3_read() (in module mpu.aws), 8
 s3_upload() (in module mpu.aws), 8

S3Path (class in mpu.aws), 7
 set_dict_value() (in module mpu.datastructures),
 13
 simplify() (mpu.geometry.LineSegment method), 19
 str2bool() (in module mpu.string), 42
 str2bool_or_none() (in module mpu.string), 42
 str2float_or_none() (in module mpu.string), 43
 str2int_or_none() (in module mpu.string), 43
 str2str_or_none() (in module mpu.string), 43

T

text_input() (in module mpu.shell), 38
 timing() (in module mpu.decorators), 17

U

UNDERLINED (mpu.shell.Codes attribute), 38
 union() (mpu.datastructures.Interval method), 10
 union() (mpu.datastructures.IntervalLike method), 10
 union() (mpu.datastructures.IntervalUnion method),
 11
 urlread() (in module mpu.io), 24

W

WHITE (mpu.shell.Codes attribute), 38
 write() (in module mpu.io), 24

Y

YELLOW (mpu.shell.Codes attribute), 38