arsenal

Release 0.0.1

Will Price

Jun 16, 2020

CONTENTS:

CHAPTER

ONE

API REFERENCE

This page contains auto-generated API reference documentation[1].

1.1 arsenal

1.1.1 Subpackages

arsenal.nn

Submodules

arsenal.nn.activation

Module Contents

arsenal.nn.activation.**softmax**(*xs*, *axis=- 1*)

Apply softmax function over a dimension of `xs`.

Parameters

- **xs** – Array of floats to softmax.
- **axis** – Dimension to softmax

Returns Softmaxed `xs`

1.1.2 Submodules

arsenal.__version__

Module Contents

arsenal.__version__.**__title__ = arsenal**

arsenal.__version__.**__author__ = Will Price**

arsenal.__version__.**__author_email__ = will.price94@gmail.com**

arsenal.__version__.**__url__**

[1] Created with sphinx-autoapi

arsenal.__version__.__description__ = A package of miscellaneous utils that I use day to da

arsenal.__version__.__version__ = 0.0.1

arsenal.collections

Module Contents

arsenal.collections.**intersperse**(*ls*, *elem*, *first=False*, *last=False*)

Parameters

- **ls** – A list of elements
- **elem** – The element to insert in between each element
- **first** – Whether to add the element at the beginning of the sequence
- **last** – Whether to add the element at the end of the sequence

Returns ls interspersed with elem`

Examples

```
>>> intersperse([1, 2, 3], 0)
[1, 0, 2, 0, 3]
>>> intersperse([1, 2, 3], 0, first=True)
[0, 1, 0, 2, 0, 3]
>>> intersperse([1, 2, 3], 0, first=True, last=True)
[0, 1, 0, 2, 0, 3, 0]
```

arsenal.collections.**index_collated_dict**(*d: Dict[str, Any]*, *idxs: Sequence[int]*) → Dict[str, Any]

Parameters

- **d** – dictionary of sequences with arbitrary levels of sub-dictionary nesting
- **idxs** – list/array of indices

Returns Dictionary where all sequences are indexed by idxs.

arsenal.debug

Module Contents

arsenal.debug.**extract**(*source=None*)

Copies the variables of the caller up to iPython. Useful for debugging.

In a Jupyter notebook, create a cell after an exception has occurred with

```
%%debug
from arsenal.debug import extract; extract()
```

Parameters **source** – A method or module from which to extract local variables. If not specified the current scope's locals will be used.

Notes

Taken from Andy Jones' personal library https://github.com/andyljones/aljpy/blob/master/aljpy/debugging.py All rights go to him.

See also:

Andy wrote a blog post explaining how he uses this code: https://andyljones.com/posts/post-mortem-plotting.html

```
def f():
    x = 'hello world'
    extract()

f() # raises an error

print(x) # prints 'hello world'
```

arsenal.image

Module Contents

arsenal.image.**_default_sep_color = [100, 100, 100]**

arsenal.image.**Color**

arsenal.image.**resize_image**(*image: np.ndarray*, ***, *height: Optional[int] = None*, *width: Optional[int] = None*, *resample=Image.NEAREST*) → np.ndarray

Resize image.

arsenal.image.**vstack_with_sep**(*rows: List[np.ndarray]*, *sep_width: int = 3*, *sep_color: Color = _default_sep_color*, ***kwargs*) → np.ndarray

Stack images on-top of one another with separator

arsenal.image.**hstack_with_sep**(*cols: List[np.ndarray]*, *sep_width: int = 3*, *sep_color: Color = _default_sep_color*, ***kwargs*) → np.ndarray

Stack images side-by-side with separator

arsenal.image.**img_to_base64**(*img: Image.Image*) → str

Encode image to base64 encoded JPEG

arsenal.image.**base64_to_img**(*b64_img: str*) → Image.Image

Decode base64 encoded image.

arsenal.numpy

Module Contents

arsenal.numpy.**select**(*xs: np.ndarray*, *xs_ids: np.ndarray*, *selection_ids: np.ndarray*) → np.ndarray

Parameters

- **xs** – Array to select elements from
- **xs_ids** – Array of ids for each element in xs
- **selection_ids** – Array of ids to select

Returns A selection of elements from `xs`

Examples

```
>>> select(            np.array([1, 2, 3]),            np.array(['a', 'b', 'c']),
↪            np.array(['a'])         )
array([1])
>>> select(            np.array([1, 2, 3]),            np.array(['a', 'b', 'c']),
↪            np.array(['a', 'c'])         )
array([1, 3])
>>> select(            np.array([1, 2, 3]),            np.array(['a', 'b', 'c']),
↪            np.array(['a', 'c', 'a'])         )
array([1, 3, 1])
```

arsenal.pandas

Module Contents

arsenal.pandas.**swap_index_values**(*series: pd.Series*) → pd.Series
Swap index and values in series.

Examples

```
>>> s = pd.Series({'a': 1, 'b': 2})
>>> s.name = 'val'
>>> s.index.name = 'char'
>>> swap_index_values(s)
val
1    a
2    b
Name: char, dtype: object
```

arsenal.pickle

Module Contents

arsenal.pickle.**load_pickle**(*filepath: Union[str, Path]*) → Any
Load pickled data from disk

Parameters **filepath** – Path to pickle file

Returns Contents of pickle.

arsenal.pickle.**save_pickle**(*obj: Any, filepath: Union[str, Path], protocol=pickle.DEFAULT_PROTOCOL*) → None

Parameters

- **obj** – The object to persist to disk
- **filepath** – The path to save
- **protocol** – The pickle protocol to use

arsenal.video

Module Contents

arsenal.video.**clip_to_html**(*clip: Union[VideoClip, np.ndarray]*, *verbose=False*, *fps=24*, ***kwargs*) → str

Convert a MoviePy clip to an HTML string.

```
from IPython.display import display
clip = ImageSequenceClip(list(np_video))
display(clip_to_html(clip))
```

Parameters

- **clip** – MoviePy clip.
- **verbose** – Whether to print out FFmpeg information during encoding
- **fps** – FPS of clip
- ****kwargs** – Any kwargs to pass down to `html_embed()`

Returns String of HTML with a `<video>` tag and base64 encoded media. Useful for use with `IPython.display.display` to show videos.

CHAPTER

TWO

INSTALLATION

```
$ pip install git+https://github.com/willprice/arsenal.git
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

Symbols

A

B

C

E

H

I

L

M

R

S

V