bpyutils

Release 0.5.8

Achilles Rasquinha

Nov 10, 2022

CONTENTS

Release v0.5.8. (Installation)

CHAPTER

ONE

THE API GUIDE

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

1.1 Utilities

1.1.1 Dict

bpyutils.util._dict.**autodict**(**args*, ***kwargs*)

Automatically adds a key to a dictionary.

Example:

```
>>> d = bpy.autodict()
>>> d['foo']['bar']['baz'] = 'boo'
{'foo': {'bar': {'baz': 'boo'}}}
```

bpyutils.util._dict.**dict_from_list**(*keys*, *values=None*)

Generate a dictionary from a list of keys and values.

Parameters

- **keys** – A list of keys.
- **values** – A list of values.

Returns dict

Example:

```
>>> bpy.dict_from_list(['a', 'b', 'c'], [1, 2, 3])
{'a': 1, 'b': 2, 'c': 3}
```

bpyutils.util._dict.**lkeys**(*d*)

Get the keys of a dictionary as a list.

Parameters **d** – A dictionary.

Returns list

Example:

```
>>> bpy.lkeys({ 'foo': 'bar', 'baz': 'boo' })
['foo', 'baz']
```

bpyutils.util._dict.**lvalues**(*d*)

Get the values of a dictionary as a list.

Parameters **d** – A dictionary.

Returns list

Example:

```
>>> bpy.lvalues({ 'foo': 'bar', 'baz': 'boo' })
['bar', 'boo']
```

bpyutils.util._dict.**merge_dict**(**args*, ***kwargs*)

Merge Dictionaries.

Parameters **args** – arguments of dictionaries to be merged. *merge_dict* will override keys from right to left.

Returns dict

Example:

```
>>> bpy.merge_dict({ 'foo': 'bar' }, { 'bar': 'baz' }, { 'baz': 'boo' })
{'foo': 'bar', 'bar': 'baz', 'baz': 'boo'}
>>> bpy.merge_dict({ 'foo': 'bar' }, { 'foo': 'baz', 'bar': 'boo' })
{'foo': 'baz', 'bar': 'boo'}
```

1.1.2 Array

bpyutils.util.array.**chunkify**(*arr*, *n*)

Divide an array into chunks wherein each chunk contains "n" elements.

Parameters

- **arr** – The array to be chunked.
- **n** – The number of elements in each chunk.

Returns A generator consisting of arrays containing "n" elements each.

Example:

```
>>> bpy.sequencify([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 3)
```

bpyutils.util.array.**compact**(*arr*, *type_=<class 'list'>*)

Creates an array with all falsey values removed. The values False, None, 0, "" are falsey.

Parameters

- **arr** (*list, tuple*) – The array to be compacted.
- **type** – The type of sequence to be returned, defaults to list.

Returns Compacted array.

Example:

```
>>> bpy.compact([1, None, 2, False, 3, 4, "", 5])
[1, 2, 3, 4, 5]
```

bpyutils.util.array.**flatten**(*arr*)

Flatten an array in case it is multi-dimensional.

Parameters **arr** – The array to be flattened.

Returns The flattened array.

Example:

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

bpyutils.util.array.**sequencify**(*value*, *type_=<class 'list'>*)

Convert a value into array-like.

Parameters **arr** – The object to be converted to array-like.

Returns A sequence.

Example:

```
>>> bpy.sequencify([1])
[1]
>>> bpy.sequencify(3)
[3]
```

bpyutils.util.array.**squash**(*seq*)

Return the object in an array in case there is just a single element.

Parameters **arr** – The array to be squashed.

Returns The squashed array.

Example:

```
>>> bpy.squash([1, 2, 3, 4, 5])
[1, 2, 3, 4, 5]
>>> bpy.squash([1])
1
```

1.1.3 Datetime

bpyutils.util.datetime.**check_datetime_format**(*datetime*, *format_*, *raise_err=False*)

Check if a given "date-string" is of the format given.

Parameters

- **datetime** – Datetime string.
- **format** – Python-compatible datetime format.
- **raise_err** – Raise a *ValueError* in case the format is not compliant.

Returns bool

Raises ValueError

Example:

```
>>> bpy.check_datetime_format('2011-11-11', '%Y-%m-%d')
True
>>> bpy.check_datetime_format('2011-11-11 11:12:13', '%Y-%m-%d')
False
>>> bpy.check_datetime_format('2011-11-11 11:12:13', '%Y-%m-%d', raise_err = True)
ValueError: Incorrect datetime format, expected %Y-%m-%d
```

bpyutils.util.datetime.**get_timestamp_str**(*format_='%Y-%m-%d %H:%M:%S'*)

Get current timestamp string.

Parameters **format** – Python-compatible datetime format. (optional)

Example:

```
>>> bpy.get_timestamp_str()
'2021-09-15 14:24:11'
>>> bpy.get_timestamp_str(format_ = '%d/%m/%Y')
'15/09/2021'
```

bpyutils.util.datetime.**now**(*tz=None*)

Returns new datetime object representing current time local to tz.

tz Timezone object.

If no tz is specified, uses local timezone.

1.1.4 Types

bpyutils.util.types.**auto_typecast**(*value*)

Automatically convert a string into its desired data type.

Parameters **value** – The value to be converted.

Example:

```
>>> bpy.auto_typecast("True")
True
>>> bpy.auto_typecast("1.2345")
1.2345
```

bpyutils.util.types.**build_fn**(*fn*, ***kwargs*)

Build a function caller with default arguments.

Args: fn (function): The function to be called.

Returns: function: A function wrapper with default arguments passed.

Example:

```
>>> def add(a, b):
        return a + b
>>> fn = bpy.build_fn(add, a = 1, b = 2)
>>> fn()
3
```

bpyutils.util.types.**check_array**(*o*, *raise_err=True*)

Check if an object is an array.

Parameters

- **o** – The object to be checked.
- **raise_err** – If True, raises an error if the object is not an array.

Example:

```
>>> bpy.check_array([1, 2, 3])
True
>>> bpy.check_array(1)
False
```

bpyutils.util.types.**get_function_arguments**(*fn*)

Get arguments of a function

Args: fn (function): The function to retrieve arguments from.

Returns: dict: A dictionary of arguments. If there is no default argument, the value associated to that argument would be inpsect._empty.

Example

```
>>> def add(a = 0, b = 1):
        return a + b
>>> params = bpy.get_function_arguments(add)
>>> params
{'a': 0, 'b': 1}
```

1.1.5 Request

1.2 Tree

class bpyutils.tree.**Node**(*obj*, *children=[]*, *parent=None*)

Construct a Tree.

Parameters

- **obj** – Object within the node.
- **children** – Children of the node.

Usage:

```
>>> from bpyutils.tree import Node
>>> family = Node('grandparent', [Node('parent-1', [Node('child-1'), Node('child-2
↪')]), Node('parent-2', [Node('child-3'), Node('child-4')])])
>>> print(family.render())
grandparent
    parent-1
        child-1
        child-2
    parent-2
```

(continues on next page)

(continued from previous page)

```
        child-3
        child-4
```

find(*query*)

Performs a Depth-First Search to find a Node based on a query provieded.

PYTHON MODULE INDEX

b

INDEX