OPath Documentation

Release 0.5.1

Justin D. Vrana

Jan 13, 2018

Contents

OPath is a library for encapsulating directory and file trees. It makes creating, moving, and deleting directories and files a breeze.

GitHub: https://github.com/jvrana/opath

CHAPTER 1

Build/Coverage Status

OPath is currently in development (Version 0.5.1)

Branch	Build	Coverage
master		
development		

CHAPTER 2

Installation

Its suggested you use pipenv to install OPath.

OPath can be downloaded from PyPI here

Option 1 (recommended): pipenv

To install, just cd into your project and run. Make sure to review how to use pipenv.

To install your project to the virtual environment

```
pipenv install opath
```

Option 2: pip

To install on your machine using pip/pip3

```
pip install opath
```

CHAPTER 3

Usage

Use `add` to create folders.

```
from opath import *

env = ODir('bin')
env.add('subfolder1')
env.add('subfolder2')
env.print()

>>>
*bin
|   *subfolder1
|   *subfolder2
```

Functions return ODir objects and so can be chained together.

```
env = ODir('bin')
env.add('subfolder1').add('subsubfolder')
env.print()

>>>
*bin
|   *subfolder1
|   |   *subsubfolder
```

Files can be written quickly

```
env = ODir('bin')
env.add('subfolder1').add('subsubfolder')
env.subsubfolder.write('My Data', 'w')
```

Or a OFile can be added:

```
env = ODir('bin')
env.add_file('myfile.txt', attr='myfile')
env.myfile.write('this is my data', 'w')
```

Folders create accesible ODir attributes automatically. Alternative attribute names can be set using 'alias='

```
env = ODir('bin')
env.add('subfolder1')
env.subfolder1.add('misc')
env.subfolder1.misc.add('.hidden', alias='hidden')
env.subfolder1.misc.hidden.add('hiddenbin')
env.print()

*bin
|   *subfolder1
|   |   *misc
|   |   |   *.hidden ("hidden")
|   |   |   |   *hiddenbin
```

By default, attributes are *pushed* back the the root directory. The following is equivalent to above.

```
env = ODir('bin')
env.add('subfolder1')
env.subfolder1.add('misc')
env.misc.add('.hidden', alias='hidden')
env.hidden.add('hiddenbin')
env.print()

*bin
|   *subfolder1
|   |   *misc
|   |   |   *.hidden ("hidden")
|   |   |   |   *hiddenbin
```

3.1 Making, moving, copying, and deleting directories

The location of the root folder can be set by `set_bin`

```
env.set_bin('../bin')
```

Directories can be created, deleted, copied or moved using `mkdirs`, `cpdirs`, `mvdirs`, `rmdirs`

```
env.mkdirs()
env_copy = env.cpdirs()
# you can do stuff with env_copy independently
env.mvdirs('~/Document')
env_copy.rmdirs()
```

3.2 Advanced usage

All iterables return special list-like objects that can be chained in one-liners.

```
env.descendents() # returns a ChainList object

# find all txt files
env.descendents(include_self=True).glob("*.txt")

# recursively change permissions for directories
env.abspaths.chmod(0o444)
```

CHAPTER 4

Running Tests

To run tests, cd into the OPath directory and run

```
pipenv run pytest
```

CHAPTER 5

API Reference

5.1 Submodules

opath	Object encapsulation of directory/file trees
objchain	Chaining methods for chaining nested objects together

5.1.1 opath.opath

Object encapsulation of directory/file trees

Functions

deepcopy(x[, memo, _nil])	Deep copy operation on arbitrary Python objects.

Classes

ChainList	List-like class that collects attributes and applies functions but functions like a list in every other regard.
ODir(name[, push_up, check_attr])	A directory object
OFile(name[, push_up, check_attr])	A file object
OPath(name[, push_up, check_attr])	A generic path
ObjChain([push_up, check_attr])	A tree-like class for chaining commands and attributes together with special root/head handling.
Path	

class opath.opath.**ODir**(*name*, *push_up=True*, *check_attr=True*)

Bases: *opath.opath.OPath*

A directory object

Initializer for OPath

Parameters

- **name** (*str*) – basename of the path
- **push_up** (*boolean*) – default of whether to 'push' access of this path to the root path node
- **check_attr** (*boolean*) – default to validate attributes. For example 'this is not valid' is not a valid attribute since it contains spaces but 'this_is_a_valid_attribute' is a valid attribute.

_add(*attr*, *child*, *push_up=None*, *check_attr=None*)

Adds child node to this node.

Parameters

- **attr** (*str*) – name to use to reference the child
- **child** (ObjChain) – child node to add
- **push_up** (*boolean*) – whether to add the child node to the root node. If True, the child will be able to be accessed from the root node.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the child node

Return type *ObjChain*

_add_grandchild(*child*)

Adds a node to the roots grandchildren

_create_and_add_child(*attr*, *with_attributes=None*, *push_up=None*, *check_attr=None*)

Copy this node and adds the node as a child

Parameters

- **attr** (*str*) – name of the new node
- **with_attributes** (*dict*) – attribute to apply to the new node
- **push_up** (*boolean*) – whether to push the new node to root.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the newly added child node

Return type *ObjChain*

_create_child(*with_attributes=None*)

Create a new copy node with with a set of attributes

Parameters **with_attributes** (*dict*) – list of attributes to apply to child

Returns child node

Return type *ObjChain*

_opts(***opts*)

Returns the options dictionary. If passed opts has a None value, default option is used.

_remove_child(*attr*)

Removes a child from this node

Parameters **attr** (*str*) – the attribute name of the node

Returns the removed child, else return None

Return type *ObjChain* or None

_sanitize_identifier(*iden*)

Validates the identifier to ensure it is not a reserve keyword used in python ('which', 'in', 'class', 'else', etc.). Other strings such as 'something.else' that cannot be translated into an attribute are also disallowed.

_update_grandchildren(*push_up*)

Updates accessible children

_validate_attr(*attr*, *push_up=None*)

Validates the attribute name for a node and checks if that attribute (i) already exists or (ii) already exists in the root attributes (if push_up=True)

param attr: attribute name for potential node :type attr: str :param push_up: whether to validate if the attribute exists in the list of root attributes :type push_up: boolean :return: None :rtype: None

abspath

Absolute path of this location

abspaths

Returns all absolute paths to all directories (includes parent_dir).

Returns directory absolute paths in this directory (inclusive)

Return type list of PosixPath

add(*name*, *attr=None*, *push_up=None*, *check_attr=None*)

Adds a new directory to the directory tree.

Parameters

- **name** (*basestring*) – name of the new directory
- **attr** (*basestring*) – attribute to use to access this directory. Defaults to name.
- **push_up** (*boolean*) – whether to 'push' attribute to the root, where it can be accessed
- **check_attr** (*boolean|None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns new directory

Return type *ODir*

add_file(*name*, *attr=None*, *push_up=None*, *check_attr=False*)

Adds a new file to the directory tree.

Parameters

- **name** (*basestring*) – name of the new file
- **attr** (*basestring*) – attribute to use to access this directory. Defaults to name.
- **push_up** (*boolean*) – whether to 'push' attribute to the root, where it can be accessed
- **check_attr** (*boolean|None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns new directory

Return type *ODir*

all_exists()
Whether all directories in the tree exist.

Returns directory tree exists

Return type boolean

ancestors(*include_self=False*)
All ancestral nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of ancestor nodes (list of ObjChain instances)

Return type list

attr
The name for this node from the parent's reference.

children
This nodes descendent notes

cpdirs(*new_parent*)
Copies the directory tree to a new location. Returns the root of the newly copied tree.

Parameters **new_parent** (*basestring or PosixPath or Path object*) – path of new parent directory

Returns copied directory

Return type *ODir*

descendents(*include_self=False*)
All descendent nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of descendent nodes (list of ObjChain instances)

Return type list

dir
The parent directory of this location

dirs
Recursively returns all directories *ODir* of this directory. Does not include parent directories.

Returns list of ODir

Return type list

dump_json(*filename*, *mode*, *data*, **args*, ***json_kwargs*)
Dump data to json

exists()
Whether the directory exists

files
Recursively returns all files *OFile* of this directory. Does not include parent directories.

Returns list of OFiles

Return type list

get(*attr*)
Short for getattr(self, attr)

has (*attr*)
Short for hasattr(self, attr)

is_root ()
Whether this node is the root/head node

list_dirs ()
List immediate directories in this directory

list_files ()
List immediate files in this directory

load_json (*filename*, *mode*, ***kwargs*)
Load data from a json

ls ()
Lists the files that exist in directory

mkdirs ()

Creates all directories in the directory tree. Existing directories are ignored.

Returns self

Return type *ODir*

mvdirs (*new_parent*)
Moves the directory. If this directory has a parent connection, the connection will be broken and this directory will become a root directory. The original root will be left in place but will no longer have access to the moved directories.

open_file (*filename*, *mode*, ***kwargs*)
Open a file at this location

parent
This nodes parent

path
Path of this location

paths
Returns all paths to all directories (includes parent_dir).

Returns directory paths in this directory (inclusive)

Return type list of PosixPath

read_file (*filename*, *mode*, ***kwargs*)
Read a file at this location

relpath
Relative path of this location

relpaths
Returns all paths to all directories (includes parent_dir).

Returns directory paths in this directory (inclusive)

Return type list of PosixPath

remove_parent ()
Remove parent from this path

rmdirs ()

Recursively removes all files and directories of this directory (inclusive)

Returns self

Return type *ODir*

root
The root/head node

set_dir (*path*)
Set the parent directory

show (*print_files=False*, *indent=4*, *max_level=None*, *level=0*, *list_missing=True*)
Recursively print the file structure

Parameters

- **print_files** (*boolean*) – whether to print files
- **indent** (*int*) – number of spaces per indent
- **max_level** (*int*) – max level depth to print
- **level** (*int*) – start at level
- **list_missing** (*boolean*) – whether to annotate files that do not exist

Returns None

Return type None

write_file (*filename*, *mode*, *data*, ***kwargs*)
Write a file at this location

class opath.opath.**OFile** (*name*, *push_up=True*, *check_attr=True*)
Bases: *opath.opath.OPath*

A file object

Initializer for OPath

Parameters

- **name** (*str*) – basename of the path
- **push_up** (*boolean*) – default of whether to 'push' access of this path to the root path node
- **check_attr** (*boolean*) – default to validate attributes. For example 'this is not valid' is not a valid attribute since it contains spaces but 'this_is_a_valid_attribute' is a valid attribute.

_add (*attr*, *child*, *push_up=None*, *check_attr=None*)
Adds child node to this node.

Parameters

- **attr** (*str*) – name to use to reference the child
- **child** (ObjChain) – child node to add
- **push_up** (*boolean*) – whether to add the child node to the root node. If True, the child will be able to be accessed from the root node.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the child node

Return type *ObjChain*

_add_grandchild(*child*)
Adds a node to the roots grandchildren

_create_and_add_child(*attr*, *with_attributes=None*, *push_up=None*, *check_attr=None*)
Copy this node and adds the node as a child

Parameters

- **attr** (*str*) – name of the new node
- **with_attributes** (*dict*) – attribute to apply to the new node
- **push_up** (*boolean*) – whether to push the new node to root.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the newly added child node

Return type *ObjChain*

_create_child(*with_attributes=None*)
Create a new copy node with with a set of attributes

Parameters **with_attributes** (*dict*) – list of attributes to apply to child

Returns child node

Return type *ObjChain*

_opts(***opts*)
Returns the options dictionary. If passed opts has a None value, default option is used.

_remove_child(*attr*)
Removes a child from this node

Parameters **attr** (*str*) – the attribute name of the node

Returns the removed child, else return None

Return type *ObjChain* or None

_sanitize_identifier(*iden*)
Validates the identifier to ensure it is not a reserve keyword used in python ('which', 'in', 'class', 'else', etc.). Other strings such as 'something.else' that cannot be translated into an attribute are also disallowed.

_update_grandchildren(*push_up*)
Updates accessible children

_validate_attr(*attr*, *push_up=None*)
Validates the attribute name for a node and checks if that attribute (i) already exists or (ii) already exists in the root attributes (if push_up=True)

param attr: attribute name for potential node :type attr: str :param push_up: whether to validate if the attribute exists in the list of root attributes :type push_up: boolean :return: None :rtype: None

abspath
Absolute path of this location

ancestors(*include_self=False*)
All ancestral nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of ancestor nodes (list of ObjChain instances)

Return type list

attr
The name for this node from the parent's reference.

children
This nodes descendent notes

descendents(*include_self=False*)
All descendent nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of descendent nodes (list of ObjChain instances)

Return type list

dir
The parent directory of this location

dump_json(*data*, *mode='w'*, ***json_kwargs*)
Dump data as a json

exists()
Whether the file exists

get(*attr*)
Short for getattr(self, attr)

has(*attr*)
Short for hasattr(self, attr)

is_root()
Whether this node is the root/head node

load_json(*mode='r'*, ***json_kwargs*)
Load data from json

open(*mode='r'*, ***kwargs*)
Opens a file for reading or writing

parent
This nodes parent

path
Path of this location

read(*mode='r'*, ***kwargs*)
Read data from a file

relpath
Relative path of this location

remove_parent()
Remove parent from this path

rm()
Removes file if it exists.

root
The root/head node

set_dir(*path*)
Set the parent directory

show (*print_files=False*, *indent=4*, *max_level=None*, *level=0*, *list_missing=True*)
Recursively print the file structure

Parameters

- **print_files** (*boolean*) – whether to print files
- **indent** (*int*) – number of spaces per indent
- **max_level** (*int*) – max level depth to print
- **level** (*int*) – start at level
- **list_missing** (*boolean*) – whether to annotate files that do not exist

Returns None

Return type None

write (*data*, *mode='w'*, ***kwargs*)
Write data to a file

class opath.opath.**OPath** (*name*, *push_up=True*, *check_attr=True*)
Bases: *opath.objchain.ObjChain*

A generic path

Initializer for OPath

Parameters

- **name** (*str*) – basename of the path
- **push_up** (*boolean*) – default of whether to 'push' access of this path to the root path node
- **check_attr** (*boolean*) – default to validate attributes. For example 'this is not valid' is not a valid attribute since it contains spaces but 'this_is_a_valid_attribute' is a valid attribute.

_add (*attr*, *child*, *push_up=None*, *check_attr=None*)
Adds child node to this node.

Parameters

- **attr** (*str*) – name to use to reference the child
- **child** (ObjChain) – child node to add
- **push_up** (*boolean*) – whether to add the child node to the root node. If True, the child will be able to be accessed from the root node.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the child node

Return type *ObjChain*

_add_grandchild (*child*)
Adds a node to the roots grandchildren

_create_and_add_child (*attr*, *with_attributes=None*, *push_up=None*, *check_attr=None*)
Copy this node and adds the node as a child

Parameters

- **attr** (*str*) – name of the new node

- **with_attributes** (*dict*) – attribute to apply to the new node
- **push_up** (*boolean*) – whether to push the new node to root.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the newly added child node

Return type *ObjChain*

_create_child(*with_attributes=None*)

Create a new copy node with with a set of attributes

Parameters **with_attributes** (*dict*) – list of attributes to apply to child

Returns child node

Return type *ObjChain*

_opts(***opts*)

Returns the options dictionary. If passed opts has a None value, default option is used.

_remove_child(*attr*)

Removes a child from this node

Parameters **attr** (*str*) – the attribute name of the node

Returns the removed child, else return None

Return type *ObjChain* or None

_sanitize_identifier(*iden*)

Validates the identifier to ensure it is not a reserve keyword used in python ('which', 'in', 'class', 'else', etc.). Other strings such as 'something.else' that cannot be translated into an attribute are also disallowed.

_update_grandchildren(*push_up*)

Updates accessible children

_validate_attr(*attr*, *push_up=None*)

Validates the attribute name for a node and checks if that attribute (i) already exists or (ii) already exists in the root attributes (if push_up=True)

param attr: attribute name for potential node :type attr: str :param push_up: whether to validate if the attribute exists in the list of root attributes :type push_up: boolean :return: None :rtype: None

abspath

Absolute path of this location

ancestors(*include_self=False*)

All ancestral nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of ancestor nodes (list of ObjChain instances)

Return type list

attr

The name for this node from the parent's reference.

children

This nodes descendent notes

descendents(*include_self=False*)

All descendent nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of descendent nodes (list of ObjChain instances)

Return type list

dir
The parent directory of this location

get (*attr*)
Short for getattr(self, attr)

has (*attr*)
Short for hasattr(self, attr)

is_root ()
Whether this node is the root/head node

parent
This nodes parent

path
Path of this location

relpath
Relative path of this location

remove_parent ()
Remove parent from this path

root
The root/head node

set_dir (*path*)
Set the parent directory

show (*print_files=False*, *indent=4*, *max_level=None*, *level=0*, *list_missing=True*)
Recursively print the file structure

Parameters

- **print_files** (*boolean*) – whether to print files
- **indent** (*int*) – number of spaces per indent
- **max_level** (*int*) – max level depth to print
- **level** (*int*) – start at level
- **list_missing** (*boolean*) – whether to annotate files that do not exist

Returns None

Return type None

5.1.2 opath.objchain

Chaining methods for chaining nested objects together

Functions

chainable_list(fxn)	Decorator that turns a returned value from a list to a ChainList (if possible)
copy(x)	Shallow copy operation on arbitrary Python objects.
wraps(wrapped[, assigned, updated])	Decorator factory to apply update_wrapper() to a wrapper function

Classes

ChainList	List-like class that collects attributes and applies functions but functions like a list in every other regard.
ObjChain([push_up, check_attr])	A tree-like class for chaining commands and attributes together with special root/head handling.

class opath.objchain.**ChainList**

Bases: list

List-like class that collects attributes and applies functions but functions like a list in every other regard.

```
m1 = ChainList(["string1   ", "   string2"])
m1.strip().upper() # ["STRING1", "STRING2"]
```

append(*object*) → None – append object to end

clear() → None – remove all items from L

copy() → list – a shallow copy of L

count(*value*) → integer – return number of occurrences of value

extend(*iterable*) → None – extend list by appending elements from the iterable

index(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert()
L.insert(index, object) – insert object before index

pop([*index*]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove(*value*) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse()
L.reverse() – reverse *IN PLACE*

sort(*key=None*, *reverse=False*) → None – stable sort *IN PLACE*

class opath.objchain.**ObjChain**(*push_up=False*, *check_attr=True*)

Bases: object

A tree-like class for chaining commands and attributes together with special root/head handling.

Each attribute creates a new ObjChain instance (a 'node') which acts like a node in a linked list.

```
root.child1.child2.child3 # etc.
```

Chainer constructor

Parameters

- **parent** (ObjChain) – parent node that called this object
- **push_up** (*boolean*) – whether to push up node to the root node by default

_add(*attr*, *child*, *push_up=None*, *check_attr=None*)
Adds child node to this node.

Parameters

- **attr** (*str*) – name to use to reference the child
- **child** (ObjChain) – child node to add
- **push_up** (*boolean*) – whether to add the child node to the root node. If True, the child will be able to be accessed from the root node.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the child node

Return type *ObjChain*

_add_grandchild(*child*)
Adds a node to the roots grandchildren

_create_and_add_child(*attr*, *with_attributes=None*, *push_up=None*, *check_attr=None*)
Copy this node and adds the node as a child

Parameters

- **attr** (*str*) – name of the new node
- **with_attributes** (*dict*) – attribute to apply to the new node
- **push_up** (*boolean*) – whether to push the new node to root.
- **check_attr** (*boolean or None*) – if True, will raise exception if attr is not a valid attribute. If None, value will default to defaults defined on initialization

Returns the newly added child node

Return type *ObjChain*

_create_child(*with_attributes=None*)
Create a new copy node with with a set of attributes

Parameters **with_attributes** (*dict*) – list of attributes to apply to child

Returns child node

Return type *ObjChain*

_opts(***opts*)
Returns the options dictionary. If passed opts has a None value, default option is used.

_remove_child(*attr*)
Removes a child from this node

Parameters **attr** (*str*) – the attribute name of the node

Returns the removed child, else return None

Return type *ObjChain* or None

_sanitize_identifier(*iden*)
Validates the identifier to ensure it is not a reserve keyword used in python ('which', 'in', 'class', 'else', etc.). Other strings such as 'something.else' that cannot be translated into an attribute are also disallowed.

_update_grandchildren(*push_up*)
Updates accessible children

_validate_attr(*attr*, *push_up=None*)
Validates the attribute name for a node and checks if that attribute (i) already exists or (ii) already exists in the root attributes (if push_up=True)

param attr: attribute name for potential node :type attr: str :param push_up: whether to validate if the attribute exists in the list of root attributes :type push_up: boolean :return: None :rtype: None

ancestors(*include_self=False*)
All ancestral nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of ancestor nodes (list of ObjChain instances)

Return type list

attr
The name for this node from the parent's reference.

children
This nodes descendent notes

descendents(*include_self=False*)
All descendent nodes

Parameters **include_self** (*boolean*) – Whether to include this node in the return list

Returns list of descendent nodes (list of ObjChain instances)

Return type list

get(*attr*)
Short for getattr(self, attr)

has(*attr*)
Short for hasattr(self, attr)

is_root()
Whether this node is the root/head node

parent
This nodes parent

remove_parent(*push_up=None*)
Remove this node's parent, effectively breaking the chain

root
The root/head node

opath.objchain.**chainable_list**(*fxn*)
Decorator that turns a returned value from a list to a ChainList (if possible)

Python Module Index

o

Index

Symbols

A

C

D