StringTree Documentation

Release 0.0.0

Luke Sloop

Dec 13, 2017

Tutorial

string_tree.py is a simple tree system for a non-linear parent-child string system such as a dialogue system.

Warning: This program is currently in pre-release, meaning that it should not be used in production.

CHAPTER 1

Tutorial

1.1 Guides

1.1.1 Getting Started

Installation

StringTree can be installed be running the setup.py file from terminal or command prompt

```
python setup.py install
```

Creating A Tree

To start simply create a *tree*

```
from string_tree import Tree

tree = Tree()
```

This creates a *root TreeNode* and adds it to the new Tree instance.

Adding to the Tree

So to add the first nodes use *add_node* on the path "root". Or use *add_child* on the root itself.

```
tree.add_node("root", "Foo", "Foo is a placeholder name")
tree.root.add_child("Bar", "Bar is a placeholder name")
```

From here children can be added to those nodes and so on. Added nodes can be accessed by using *get_node* more on accessing later.

```
tree.add_node("Foo", "Child", "Child of Foo")
tree.get_node("Bar").add_child("Child", "Child of Bar")
```

Accessing nodes

Now that there are two nodes that are named `Child` they can no longer be accessed by just their title. If "Child" was used as the path to that node it would always return Foo's Child and never Bar's because Foo was added to the Tree first. To access Bar's child the full or partial path must be used.

```
# This will return Foo's Child
tree.get_node("Child")

# Both of these will return Bar's Child
tree.get_node("root.Bar.Child")
tree.get_node("Bar.Child")

# It can also be accessed from the TreeNode itself
bar = tree.get_node("Bar")
bar.get_child("Child")
```

Accessing Contents

get_string can be used to access the contents of the TreeNode from a path. Or access the TreeNode's *string* property directly

```
tree.get_string("Foo")  # Will return "Foo is a placeholder name"
tree.get_string("Foo.Child")  # Will return "Child of Foo"

foo = tree.get_node("Foo")
foo.string # "Foo is a placeholder name"
foo_child = tree.get_node("Foo.Child")
foo_child.string  # "Child of Foo"
```

Adopted Children

Normally each child noe has one parent. However, if it is possible to connect two paths in the tree by adding any node as an adoptive parent to another node. The node's must have a unique title from all current *children* and can not be the *root* node. So Foo can not adopt "Bar.Child" because Foo already has a node named "Child". However if another child is added to Bar with a different name, then Foo can adopt that child as an adopted child.

See also:

get_adopted_children in *Tree* and *get_adopted_children* in *TreeNode*

```
bar.add_child("Unique", "This node has a unique title")

foo.add_adopted_child("Unique")

unique = foo.get_child("Unique")
unique.string  # Will return "This node has a unique title"

unique in foo.children  # Returns True
unique in foo.get_biological_children  # Returns False
unique in foo.get_adopted_children  # Returns True
```

```
unique.parent  # Still Bar
unique.adoptive_parents  # Returns a list including foo
```

Accessing Paths

To get a list of every node on a path use *get_lineage*. This will return the node at the path's children and their children's children and so on. *remove_lineage* can be used to remove all of a node's biological children as opposed to *remove_node* which will append the nodes children to its parent. *get_lineage* and *remove_lineage* are also accessible from individual *TreeNodes*

Note: The passed in node is not included in the returned list and the list only includes biological children.

```
foo in tree.get_lineage("root") # Returns True
tree.get_node("Unique") in tree.get_lineage("root") # Returns True

tree.root in tree.get_lineage("root")  # Returns False
tree.get_node("Bar.Child") in tree.get_lineage("Foo")  # Returns False
```

CHAPTER 2

StringTree API

2.1 Modules

2.1.1 string_tree.py

Tree

Inherits: object

Module: *string_tree*

Note: in this class "path" always refers to a string of the titles of each node joined by a period. Example: "Grandparent.Parent.Child" is a path to the "Child" TreeNode

Brief Description

Represents a simple tree system for a non-linear parent-child string system such as a dialogue system.

Instance Methods

Tree	*TreeNode* ( )
TreeNode	*add_node* (str path, str title, str string)
TreeNode	*get_node* (str path)
TreeNode	*remove_node* (str path)
str	*get_string* (str path)
list	*get_children* (str path)
list	*get_biological_children* (str path)
None	*add_adopted_child* (str path, str child)
list	*get_adopted_children* (str path)
TreeNode	*remove_adopted_child* (str path)
list	*get_lineage* (str path)
list	*remove_lineage* (str path)

Description

A tree system that stores in a parent-child relationship by title. Strings are added to the root and to the roots children and so on. Example: "root.foo.bar"

Instance Variables

- *list* **tree -** A list of TreeNodes that this tree contains.
- *TreeNode* **root -** The root node of this tree, with the *title* "root".

Instance Method Descriptions

- *Tree* **TreeNode ( )**

Creates a Tree instance

- *TreeNode* **add_node (str path, str title, str string)**

Creates a node using the `title` and `string`, and adds it to the parent node `path` then returns its instance.

- *TreeNode* **get_node (str path)**

Returns the TreeNode at the `path`

- *TreeNode* **remove_node (str path)**

Removes and returns the node at the `path`. This will add the children of the node as children of the nodes parent. Note that if one of the children of this node is the same as a child of the parents node it will raise a ValueError

- **str get_string (str path)**

Returns the contents of the TreeNode at path.

- *TreeNode* **get_children (str path)**

Returns the children of the TreeNode at `path`.

- **list get_biological_children (str path)**

Returns the node at `path's` children that are directly created from the node.

- **None add_adopted_child (str path, str child)**

Adds an adopted child to the node at path. `Child` is a path to a node that already exists as a child of another node but is also the indirect child of this node

- **list get_adopted_children (str path)**

Returns the node at path's children that are not created from the node, but are added to this node using *add_adopted_child ( )* from the *Tree* or *TreeNode*.

- *TreeNode* **remove_adopted_child (str path)**

Removes and returns the adopted child from the node at `path`. `Child` is the path to the existing child.

- **list get_lineage (str path)**

Returns the lineage of the node at `path`. This includes all biological children, their biological children, and so on.

- **list remove_lineage (str path)**

Removes and returns the lineage of the node at `path`. This includes all biological children, their biological children, and so on.

Supported Magic Methods

- *TreeNode* **__iter__ ( ):**

Iterates through this Tree's *tree* list

- **str __repr__ ( )**

Returns the representation of this Tree's *tree* list

- **str __str__ ( )**

Returns this Tree's *tree* list with each node on a new line and casting each node to a string

TreeNode

Inherits: object

Module: *string_tree*

Note: in this class "path" always refers to a string of the titles of each node joined by a period. Example: "Grandparent.Parent.Child" is a path to the "Child" TreeNode

---

Brief Description

A node in the *Tree* that holds references to this nodes *parent*, biological and adopted *children* and this nodes *string* contents.

Instance Methods

TreeNode	*TreeNode* (*Tree* tree, *TreeNode* parent, str title, str string)
str	*get_path* ( )
TreeNode	*add_child* (str title, str string)
TreeNode	*get_child* (str title)
TreeNode	*remove_child* (str title)
list	*get_biological_children* ( )
None	*add_adopted_child* (str path)
list	*get_adopted_children* ( )
None	*remove_adopted_child* (str title)
list	*get_lineage* ( )
list	*remove_lineage* ( )

Description

A node in the *Tree* that holds references to this nodes *parent*, biological and adopted *children* and this nodes *string* contents.

Instance Variables

- *Tree* **tree -** The tree that the node belongs to.
- *TreeNode* **parent -** The direct TreeNode this node derives from.
- **list adoptive_parents -** A list of this nodes adoptive parent TreeNodes, these are nodes that are not the direct parent of this node but still are connected as a parent.
- **list children -** A list of this nodes child TreeNodes.
- **str title -** The title that describes this node. This is used in the path to access the node. See *note*
- **str string -** The contents of this node.

Instance Method Descriptions

- *TreeNode* TreeNode (*Tree* tree, *TreeNode* parent, str title, str string)

Creates a TreeNode instance on the `tree` as a child of the `parent` with the `title` and `string` contents.

- **str get_path ( )**

Returns the path to this node.

- *TreeNode* **add_child (str title, str string)**

Creates a TreeNode with the `title` and `string` and adds it as a child to this node and appends it to this nodes *tree* .

- *TreeNode* **get_child (str title)**

Returns the child of this node with the `title` .

- *TreeNode* **remove_child (str title)**

Removes and returns the child of this node with the `title` . This will append all of the children of the removed node as children of this node.

- **list get_biological_children ( )**

Returns a list of this nodes *children* that are directly created from this node.

- **None add_adopted_child (str path)**

Adds the TreeNode from the `path` as an adopted child of this node. Adopted children are the same as biological children except they are not directly made from this parent.

- **list get_adopted_children ( )**

Returns a list of this nodes *children* that are adopted. Meaning they are not directly created from this node.

- None **remove_adopted_child (str title)**

Removes the child of this node with the `title` from this nodes *children*.

- **list get_lineage ( )**

Returns the lineage of this node. This includes all biological children, their biological children, and so on.

- **list remove_lineage ( )**

Removes and returns the lineage of this node. This includes all biological children, their biological children, and so on.

Supported Magic Methods

- **bool __eq__ ( )**

Returns true if this node and the other node have the same *title*, *parent*, and *children*