structoscope

Release 0.0.1

Matteo Sandrin

Dec 22, 2020

REFERENCE

Welcome to the documentation for the `structoscope` Python library!

Structoscope is a Python library for visualizing and inspecting any data structure.

CHAPTER

ONE

STRUCTOSCOPE PACKAGE

1.1 structoscope.lib

class structoscope.lib.**Scope**(*dataMemberName=None*, *childrenMemberName=None*)
Bases: object

The Scope class is a wrapper around a single visualization window

Parameters

- **childMemberName** – The name of the member containing the data of the node object
- **childrenMemberName** (*str*) – The name of the member containing the children of the node object

_displayGraph(*graph*)
Converts the graph into a PNG image and displays it as a plot

Parameters **graph** (*graphviz.Digraph*) – The graph object to display

print(*data*, *raw=False*)
Display a visualization of an arbitrary Python object. Supports lists, dictionaries and trees.

Parameters

- **data** (*Object*) – The object to visualize
- **raw** (*bool*) – If true returns a string representing the dot-notation graph

static wait(*secs*)
Block the main thread for any number of seconds

Parameters **secs** (*float*) – Amount of time to wait for, in seconds

1.2 structoscope.slist

class structoscope.slist.**List**
Bases: object

_findNestedLists(*data*, *result=None*)
Finds every nested array in the supplied data and returns is as a flat, one-dimensional list.

Parameters

- **data** (*list*) – The multi-dimensional list
- **result** – The one-dimensional list holding the nested lists

_getLabelForList(*data*)
Creates the label for a single graph node representing a list. This label is formatted as an HTML-like markup language specific to the Graphviz library.

Parameters **data** (*list*) – The list populating the label

makeGraph(*data*)
Creates a graph to visualize a Python list

Parameters **data** (*list*) – The list to visualize

1.3 structoscope.sdict

class structoscope.sdict.**Dict**
Bases: object

_getLabelForDict(*data*)
Creates the label for a single graph node representing a dictionary. This label is formatted as an HTML-like markup language specific to the Graphviz library.

Parameters **data** (*dict*) – The dictionary populating the label

makeGraph(*data*)
Creates a graph to visualize of a Python dictionary

Parameters **data** (*dict*) – The dictionary to visualize

1.4 structoscope.stree

class structoscope.stree.**Tree**(*members*)
Bases: object

_findChildren(*data*, *result=None*)
Finds every node in the supplied tree and returns is as a flat, one-dimensional list.

Parameters

- **data** (*Object*) – The root of the tree
- **result** – The one-dimensional list holding the nodes

_getLabelForNode(*data*)
Creates the label for a single graph node representing the node of a tree. This label is formatted as an HTML-like markup language specific to the Graphviz library.

Parameters **data** (*Object*) – The node populating the label

makeGraph(*data*)
Creates a graph to visualize a tree

Parameters **data** (*Object*) – The root object of the tree to visualize

CHAPTER

TWO

INSTALL

The only external dependency is the `graphviz` binary, which you can install buy running the following command in the terminal.

```
brew install graphviz
```

Now you can install structoscope by running the following command in the terminal.

```
pip3 install structoscope
```

CHAPTER

THREE

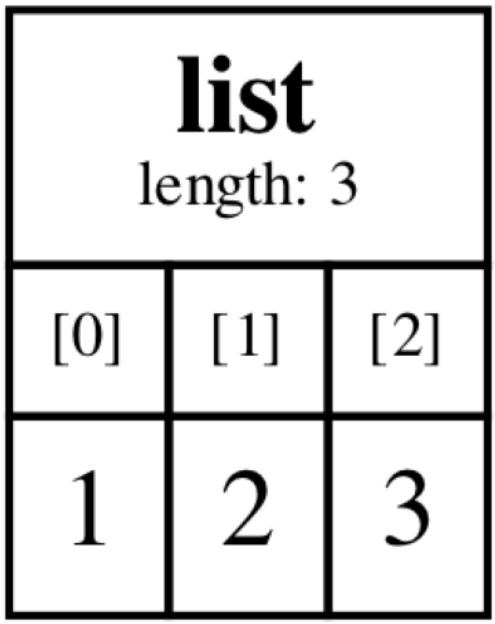
EXAMPLES

3.1 Lists

Structoscope can easily display Python lists:

```
from structoscope import Scope

s = Scope()
testList = [1,2,3]
s.print(testList)
input() # block the main thread
```



Example

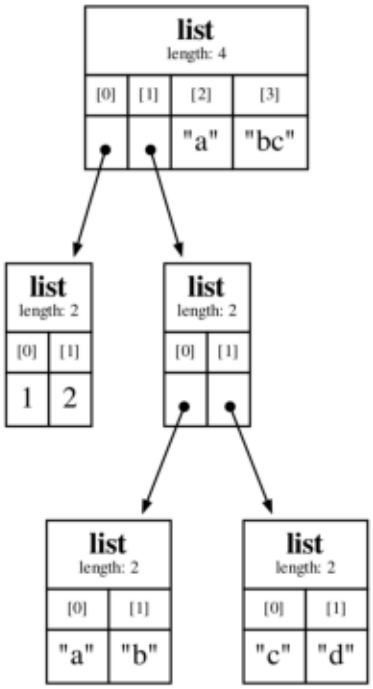
1

3.2 Multi-dimensional Lists

It can even display multi-dimensional lists:

```
from structoscope import Scope

s = Scope()
testList = [
    [1,2],
    [
        ['a', 'b'],
        ['c', 'd']
    ],
    'abc'
]
s.print(testList)
input() # block the main thread
```



Example

2

3.3 Dictionaries

Or it can display dictionaries:

```
from structoscope import Scope

s = Scope()
testDict = {
    'first' : 101,
    'second' : 102,
    'third' : 103,
}
s.print(testDict)
input() # block the main thread
```

dict length: 3	
key	**value**
"first"	101
"second"	102
"third"	103

Example 3

3.4 Trees

It can even display trees:

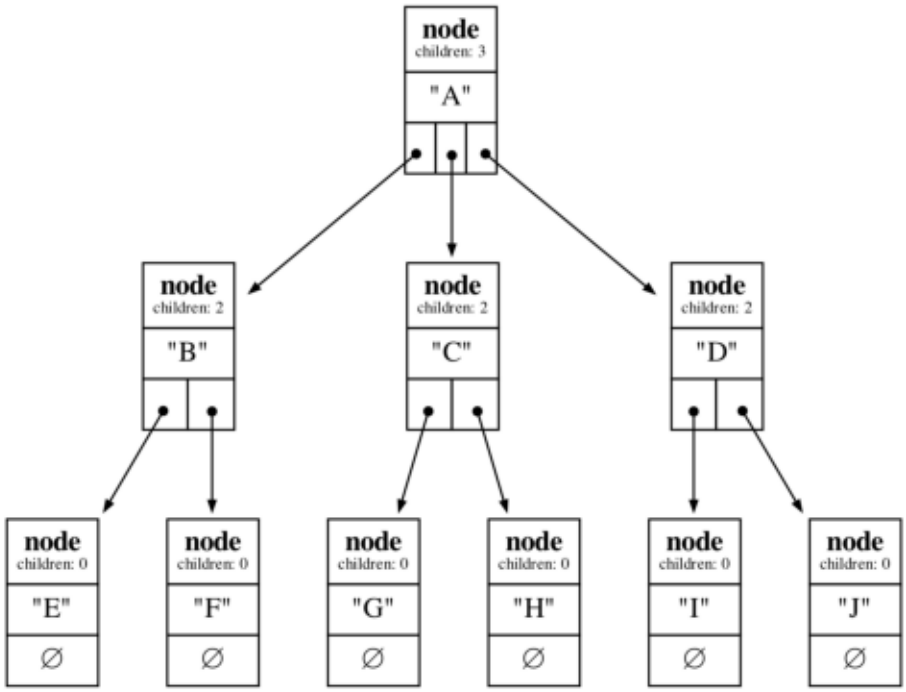
```
from structoscope import Scope

class Node:
    def __init__(self, val=None, children=[]):
        self.val = val
        self.children = children

s = Scope(
    dataMemberName='val',
    childrenMemberName='children'
)

node9 = Node(val='J')
node8 = Node(val='I')
node7 = Node(val='H')
node6 = Node(val='G')
node5 = Node(val='F')
node4 = Node(val='E')
node3 = Node(val='D', children=[node8, node9])
node2 = Node(val='C', children=[node6, node7])
node1 = Node(val='B', children=[node4, node5])
root = Node(val='A', children=[node1, node2, node3])

s.print(root)
input() # block the main thread
```


node
children: 3
"A"
node
children: 2
"B"
node
children: 2
"C"
node
children: 2
"D"
node
children: 0
"E"
∅
node
children: 0
"F"
∅
node
children: 0
"G"
∅
node
children: 0
"H"
∅
node
children: 0
"I"
∅
node
children: 0
"J"
∅

Example 4

PYTHON MODULE INDEX

S

INDEX

Symbols

D

L

M

P

S

T

W