codegrapher Documentation

Release 0.2.1

Laura Rupprecht

Contents

1	Introduction to Codegrapher	3			
	1.1 codegrapher	3			
2	odegrapher package 5				
	2.1 Submodules				
	2.2 codegrapher.graph module	5			
	2.3 codegrapher.parser module	6			
	2.4 Module contents	10			
3	odegrapher 1				
	3.1 codegrapher package	11			
4	4 Indices and tables				
Pv	vthon Module Index	19			

Contents:

Contents 1

2 Contents

Introduction to Codegrapher

1.1 codegrapher

1.1.1 Code that graphs code

Uses the python AST to parse Python source code and build a call graph.

1.1.2 Output

An example of the current output of the parser parsing itself.

1.1.3 Installation

```
pip install codegrapher
```

To generate graphs, graphviz must be installed.

1.1.4 Usage

At the command line

To parse a file and output results to the console:

```
codegrapher path/to/file.py --printed
```

To parse a file and output results to a file:

```
codegrapher path/to/file.py --output output_file_name --output-type png
```

To analyze a directory of files, along with all files it contains:

```
\verb|codegrapher -r path/to/directory -- output multiple_file_analysis|\\
```

And if you have a list of functions that aren't useful in your graph, add it to a .cg_ignore file:

```
# cg_ignore file
# all lines beginning with '#' are ignored
# every function calls this, so it's not helpful in my graph:
```

```
log_error
# I don't want to see this in my graph:
parse
lower
```

Then add the *-ignore* flag to your command. Using the flag *-remove-builtins* provides the same functionality for ignoring items found in *_builtins__*.

As a Python module

To easily parse code in Python:

```
from codegrapher.parser import FileObject
file_object = FileObject('path/to/file.py')
file_object.visit()
```

And then to add that code to a graph and render it (using graphviz):

```
from codegrapher.graph import FunctionGrapher
graph = FunctionGrapher()
graph.add_file_to_graph(file_object)
graph.name = 'name.gv'
graph.format = 'png'
graph.render()
```

Which will produce your code as a png file, name.gv.png, along with a dot file name.gv

More documentation for the Python module can be found at Read the Docs.

codegrapher package

2.1 Submodules

2.2 codegrapher.graph module

```
exception codegrapher.graph.FilenameNotSpecifiedException
```

Bases: exceptions. Exception

An exception raised when a file name is not specified in a FunctionGrapher instance before calling FunctionGrapher.render() on it.

class codegrapher.graph.FunctionGrapher

Bases: object

FunctionGrapher is a class for producing graphviz graphs showing the call graph for sets of classes.

name

string

Name to be used when a graph is made.

nodes

set

Graphviz nodes to be graphed.

edges

set

Directional edges connecting one node to another.

format

string

File format for graph. Default is pdf.

add_classes_to_graph (classes, relative_namespace)

Adds classes with constructors to the set. This adds edges between a class constructor and the methods called on those items.

Parameters

- classes (list) list of codegrapher.parser.ClassObject items.
- relative_namespace (string) namespace of the current class.

add_dict_to_graph (class_names, dictionary, relative_namespace)

Creates a list of nodes and edges to be rendered. Deduplicates input.

Parameters

- **class_names** (*list*) List of class names to be recognized by the graph as *class_name*. __init__ nodes.
- **dictionary** (*dict*) *ClassObject.call_tree* dict to be added to graph nodes and edges.
- **relative_namespace** (*string*) Relative namespace for the current class, i.e. where the current class is located relative to the root, in dotted path notation.

```
add_file_to_graph (file_object)
```

When given a codegrapher.parser.FileObject object, this adds all classes to the current graph.

Parameters file_object (codegrapher.parser.FileObject) – Visitor objects to have all its classes added to the current graph.

```
render (name=None)
```

Renders the current graph. Graphviz must be installed for the graph to be rendered.

Parameters name (*string*) – filename to override *self.name*.

Raises FilenameNotSpecifiedException - If FunctionGrapher.name is not specified.

```
{\bf class} \; {\tt codegrapher.graph.Node} \; ({\it input\_node})
```

Bases: object

A class to more easily handle manipulations needed to properly display nodes in a graph. Optimized to handle nodes that represent functions in a program.

tuple

tuple

Contains the namespace, class, and function name for the current node. If namespace is an empty string, this contains just the class and function names. If a string is provided to the constructor this is a tuple containing just the function name.

represent

Provides a string representation of the current node

Returns (string): Dotted form of current node, as in *namespace.class.function_name*.

2.3 codegrapher.parser module

```
class codegrapher.parser.CallInspector
```

Bases: ast.NodeVisitor

Within a call, a Name or Attribute will provide the function name currently in use.

Identifies *Name* nodes, which are called as name(args), and *Attribute* nodes, which are called as object.attr(args)

module

string

Current module name on which the current call is made.

```
identifier
          string
          Name of the function called.
     visit Attribute(node)
     visit Name (node)
class codegrapher.parser.CallVisitor(**kwargs)
     Bases: codegrapher.parser.ImportVisitor
     Finds all calls present in the current scope and inspect them.
     call_names
          set
          set of CallInspector.identifier items within current AST node.
     calls
          list
          (module, identifier) items called within current AST node, with identifiers decoded form current alias, and
          modules expanded to their full import paths.
     continue_parsing(node)
     visit_Call (node)
class codegrapher.parser.ClassObject (node=None, aliases=None, modules=None)
     Bases: object
     Class for keeping track of classes in code.
     modules
          dict
          dict of current modules with alias: module_name, key:value pairs.
     aliases
          dict
          dict of current modules with alias: original_name, key:value pairs.
     node
          ast.AST
          AST node for entire class.
     name
          string
          Class name.
     functions
          list
          FunctionObject items defined in the current class.
     call_tree
          dict
          dict with key:value pairs (module, FunctionObject.name): (module, identifier).
     ignore_functions (ignore_set)
          Ignores all functions matching those specified in a pre-defined ignore set.
```

Parameters ignore_set (*set*) – Functions whose calls should be removed (ignored) in the class call tree.

namespace (relative_namespace)

Take the relative namespace for the class and prepend it to each item defined in the current class.

Parameters relative_namespace (*string*) – Namespace to be prepended to each item in the call tree.

pprint()

Pretty print formatter for class object.

Returns string

remove_builtins()

For many classes, we may not want to include builtin functions in the graph. Remove builtins from the call tree and from called functions list.

visit()

Visits all the nodes within the current class AST node.

Updates *self.functions* and *self.call_tree* for the current instance.

```
class codegrapher.parser.FileObject (file_name, modules=None, aliases=None)
```

Bases: object

Class for keeping track of files.

modules

dict

dict of current modules with alias: module_name, key:value pairs.

aliases

dict

dict of current modules with alias: original_name, key:value pairs.

node

ast.AST

AST node for entire file.

name

string

File name.

classes

list

ClassObject items defined in the current file.

relative_namespace

string

The namespace for the current file, taken from the relative path of the current file

ignore

set

Functions to be ignored, as defined in a .cg_ignore text file.

add_ignore_file()

Use a file .cg_ignore to ignore a list of functions from the call graph

```
ignore functions()
           Ignore all functions in the current class which are present in the instance's ignore attribute.
     namespace()
           Programmatically change the name of items in the call tree so they have relative path information
     remove builtins()
           Removes builtins from each class in a FileObject instance.
     visit()
           Visits all the nodes within the current file AST node.
           Updates self.classes for the current instance.
class codegrapher.parser.FileVisitor(**kwargs)
     Bases: codegrapher.parser.ImportVisitor
     First visitor that should be called on the file level.
     classes
          list
          list of ClassObject instances defined in the current file.
     continue_parsing(node)
     remove builtins()
           Removes builtins from each class in a FileVisitor instance.
     visit ClassDef(node)
     visit Module (node)
class codegrapher.parser.FunctionObject (node=None, aliases=None, modules=None)
     Bases: object
     Object that stores information within a single function definition
     modules
          dict of current modules with alias: module_name, key:value pairs.
     aliases
           dict of current modules with alias: original_name, key:value pairs.
     node
           ast.AST
           AST node for entire function.
     name
           string
           function name.
     calls
           list
           (module, identifier) tuples describing items called within current node, with identifiers decoded form cur-
           rent alias, and modules expanded to their full import paths.
     decorator_list
           list
           list of decorators, by name as a string, applied to the current function definition.
```

is classmethod

bool

True if the current function is designated as a classmethod by a decorator.

visit()

Visits all the nodes within the current function object's AST node.

Updates self.calls, self.modules, and self.aliases for the current instance.

```
class codegrapher.parser.FunctionVisitor(**kwargs)
```

```
Bases: codegrapher.parser.ImportVisitor
```

Function definitions are where the function is defined, and the call is where the ast for that function exists.

This only looks for items that are called within the scope of a function, and associates those items with the function.

defined_functions

set

names of functions found by function visitor instance.

functions

list

FunctionObject instances found by function visitor instance.

calls

dict

mapping from function names defined to calls within that function definition.

```
continue_parsing(node)
```

```
visit_FunctionDef (node)
```

class codegrapher.parser.ImportVisitor(aliases=None, modules=None)

```
Bases: ast.NodeVisitor
```

For import related calls, store the source modules and aliases used. Designed to be inherited by other classes that need to know about imports in their current scope.

modules

dict

dict of current modules with alias: module_name, key:value pairs.

aliases

dict

dict of current modules with alias: original_name, key:value pairs.

```
continue_parsing(node)
```

```
visit_Import (node)
```

 ${\tt visit_ImportFrom}\ (node)$

2.4 Module contents

codegrapher

3.1 codegrapher package

3.1.1 Submodules

3.1.2 codegrapher.graph module

```
exception codegrapher.graph.FilenameNotSpecifiedException
```

Bases: exceptions. Exception

An exception raised when a file name is not specified in a FunctionGrapher instance before calling FunctionGrapher.render() on it.

class codegrapher.graph.FunctionGrapher

Bases: object

FunctionGrapher is a class for producing graphviz graphs showing the call graph for sets of classes.

name

string

Name to be used when a graph is made.

nodes

set

Graphviz nodes to be graphed.

edges

set

Directional edges connecting one node to another.

format

string

File format for graph. Default is *pdf*.

add_classes_to_graph (classes, relative_namespace)

Adds classes with constructors to the set. This adds edges between a class constructor and the methods called on those items.

Parameters

• ${\bf classes}\ ({\it list}) - {\bf list}\ {\bf of}\ {\it codegrapher.parser.ClassObject}\ {\it items.}$

• **relative_namespace** (*string*) – namespace of the current class.

add_dict_to_graph (class_names, dictionary, relative_namespace)

Creates a list of nodes and edges to be rendered. Deduplicates input.

Parameters

- class_names (list) List of class names to be recognized by the graph as class name. init nodes.
- **dictionary** (*dict*) *ClassObject.call_tree* dict to be added to graph nodes and edges.
- **relative_namespace** (*string*) Relative namespace for the current class, i.e. where the current class is located relative to the root, in dotted path notation.

```
add_file_to_graph (file_object)
```

When given a codegrapher.parser.FileObject object, this adds all classes to the current graph.

Parameters file_object (codegrapher.parser.FileObject) – Visitor objects to have all its classes added to the current graph.

```
render (name=None)
```

Renders the current graph. Graphviz must be installed for the graph to be rendered.

Parameters name (*string*) – filename to override *self.name*.

Raises FilenameNotSpecifiedException - If FunctionGrapher.name is not specified.

```
class codegrapher.graph.Node(input_node)
```

Bases: object

A class to more easily handle manipulations needed to properly display nodes in a graph. Optimized to handle nodes that represent functions in a program.

tuple

tuple

Contains the namespace, class, and function name for the current node. If namespace is an empty string, this contains just the class and function names. If a string is provided to the constructor this is a tuple containing just the function name.

represent

Provides a string representation of the current node

Returns (string): Dotted form of current node, as in *namespace.class.function_name*.

3.1.3 codegrapher.parser module

class codegrapher.parser.CallInspector

Bases: ast.NodeVisitor

Within a call, a Name or Attribute will provide the function name currently in use.

Identifies *Name* nodes, which are called as name(args), and *Attribute* nodes, which are called as object.attr(args)

module

string

Current module name on which the current call is made.

```
identifier
          string
          Name of the function called.
     visit Attribute(node)
     visit Name (node)
class codegrapher.parser.CallVisitor(**kwargs)
     Bases: codegrapher.parser.ImportVisitor
     Finds all calls present in the current scope and inspect them.
     call_names
          set
          set of CallInspector.identifier items within current AST node.
     calls
          list
          (module, identifier) items called within current AST node, with identifiers decoded form current alias, and
          modules expanded to their full import paths.
     continue_parsing(node)
     visit_Call (node)
class codegrapher.parser.ClassObject (node=None, aliases=None, modules=None)
     Bases: object
     Class for keeping track of classes in code.
     modules
          dict
          dict of current modules with alias: module_name, key:value pairs.
     aliases
          dict
          dict of current modules with alias: original_name, key:value pairs.
     node
          ast.AST
          AST node for entire class.
     name
          string
          Class name.
     functions
          list
          FunctionObject items defined in the current class.
     call_tree
          dict
          dict with key:value pairs (module, FunctionObject.name): (module, identifier).
     ignore_functions (ignore_set)
          Ignores all functions matching those specified in a pre-defined ignore set.
```

Parameters ignore_set (*set*) – Functions whose calls should be removed (ignored) in the class call tree.

```
namespace (relative_namespace)
```

Take the relative namespace for the class and prepend it to each item defined in the current class.

Parameters relative_namespace (*string*) – Namespace to be prepended to each item in the call tree.

pprint()

Pretty print formatter for class object.

Returns string

remove_builtins()

For many classes, we may not want to include builtin functions in the graph. Remove builtins from the call tree and from called functions list.

visit()

Visits all the nodes within the current class AST node.

Updates *self.functions* and *self.call_tree* for the current instance.

```
class codegrapher.parser.FileObject (file_name, modules=None, aliases=None)
```

Bases: object

Class for keeping track of files.

modules

dict

dict of current modules with alias: module_name, key:value pairs.

aliases

dict

dict of current modules with alias: original_name, key:value pairs.

node

ast.AST

AST node for entire file.

name

string

File name.

classes

list

ClassObject items defined in the current file.

relative_namespace

string

The namespace for the current file, taken from the relative path of the current file

ignore

set

Functions to be ignored, as defined in a .cg_ignore text file.

add_ignore_file()

Use a file .cg_ignore to ignore a list of functions from the call graph

```
ignore functions()
           Ignore all functions in the current class which are present in the instance's ignore attribute.
     namespace()
           Programmatically change the name of items in the call tree so they have relative path information
     remove builtins()
           Removes builtins from each class in a FileObject instance.
     visit()
           Visits all the nodes within the current file AST node.
           Updates self.classes for the current instance.
class codegrapher.parser.FileVisitor(**kwargs)
     Bases: codegrapher.parser.ImportVisitor
     First visitor that should be called on the file level.
     classes
          list
          list of ClassObject instances defined in the current file.
     continue_parsing(node)
     remove builtins()
           Removes builtins from each class in a FileVisitor instance.
     visit ClassDef(node)
     visit Module (node)
class codegrapher.parser.FunctionObject (node=None, aliases=None, modules=None)
     Bases: object
     Object that stores information within a single function definition
     modules
          dict of current modules with alias: module_name, key:value pairs.
     aliases
           dict of current modules with alias: original_name, key:value pairs.
     node
           ast.AST
           AST node for entire function.
     name
           string
           function name.
     calls
           list
           (module, identifier) tuples describing items called within current node, with identifiers decoded form cur-
           rent alias, and modules expanded to their full import paths.
     decorator_list
           list
           list of decorators, by name as a string, applied to the current function definition.
```

is classmethod

bool

True if the current function is designated as a classmethod by a decorator.

visit()

Visits all the nodes within the current function object's AST node.

Updates self.calls, self.modules, and self.aliases for the current instance.

```
class codegrapher.parser.FunctionVisitor(**kwargs)
```

```
Bases: codegrapher.parser.ImportVisitor
```

Function definitions are where the function is defined, and the call is where the ast for that function exists.

This only looks for items that are called within the scope of a function, and associates those items with the function.

defined_functions

set

names of functions found by function visitor instance.

functions

list

FunctionObject instances found by function visitor instance.

calls

dict

mapping from function names defined to calls within that function definition.

```
continue_parsing(node)
```

```
visit_FunctionDef (node)
```

class codegrapher.parser.ImportVisitor(aliases=None, modules=None)

Bases: ast.NodeVisitor

For import related calls, store the source modules and aliases used. Designed to be inherited by other classes that need to know about imports in their current scope.

modules

dict

dict of current modules with alias: module_name, key:value pairs.

aliases

dict

dict of current modules with alias: original_name, key:value pairs.

```
continue_parsing(node)
```

```
visit_Import (node)
```

visit_ImportFrom (node)

3.1.4 Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

С

codegrapher, 16 codegrapher.graph, 11 codegrapher.parser, 12

20 Python Module Index

A	continue_parsing() (codegrapher.parser.ImportVisitor		
add_classes_to_graph() (codegra-	method), 10, 16		
pher.graph.FunctionGrapher method), 5,	D		
add_dict_to_graph() (codegra-	decorator_list (codegrapher.parser.FunctionObject		
pher.graph.FunctionGrapher method), 5,	attribute), 9, 15		
12	defined_functions (codegrapher.parser.FunctionVisitor at-		
add_file_to_graph() (codegrapher.graph.FunctionGrapher	tribute), 10, 16		
method), 6, 12	E		
add_ignore_file() (codegrapher.parser.FileObject	edges (codegrapher.graph.FunctionGrapher attribute), 5,		
method), 8, 14 aliases (codegrapher.parser.ClassObject attribute), 7, 13	11		
aliases (codegrapher.parser.FileObject attribute), 8, 14			
aliases (codegrapher.parser.FunctionObject attribute), 9,	F		
15	FilenameNotSpecifiedException, 5, 11		
aliases (codegrapher.parser.ImportVisitor attribute), 10,	FileObject (class in codegrapher.parser), 8, 14		
16	FileVisitor (class in codegrapher.parser), 9, 15		
0	format (codegrapher.graph.FunctionGrapher attribute), 5,		
C	II		
call_names (codegrapher.parser.CallVisitor attribute), 7,	FunctionGrapher (class in codegrapher graph), 5, 11		
13	FunctionObject (class in codegrapher.parser), 9, 15 functions (codegrapher.parser.ClassObject attribute), 7,		
call_tree (codegrapher.parser.ClassObject attribute), 7, 13	13		
CallInspector (class in codegrapher.parser), 6, 12	functions (codegrapher.parser.FunctionVisitor attribute),		
calls (codegrapher parser CallVisitor attribute), 7, 13	10, 16		
calls (codegrapher.parser.FunctionObject attribute), 9, 15 calls (codegrapher.parser.FunctionVisitor attribute), 10,	FunctionVisitor (class in codegrapher.parser), 10, 16		
16	1		
CallVisitor (class in codegrapher.parser), 7, 13	I		
classes (codegrapher.parser.FileObject attribute), 8, 14	identifier (codegrapher.parser.CallInspector attribute), 6,		
classes (codegrapher.parser.FileVisitor attribute), 9, 15	12		
ClassObject (class in codegrapher.parser), 7, 13	ignore (codegrapher.parser.FileObject attribute), 8, 14		
codegrapher (module), 10, 16	ignore_functions() (codegrapher.parser.ClassObject		
codegrapher.graph (module), 5, 11	method), 7, 13 ignore_functions() (codegrapher.parser.FileObject		
codegrapher.parser (module), 6, 12	method), 8, 14		
continue_parsing() (codegrapher.parser.CallVisitor	ImportVisitor (class in codegrapher.parser), 10, 16		
method), 7, 13 continue_parsing() (codegrapher.parser.FileVisitor	is_classmethod (codegrapher.parser.FunctionObject at-		
method), 9, 15	tribute), 9, 15		
continue_parsing() (codegrapher.parser.FunctionVisitor	NA.		
method), 10, 16	M		
	module (codegrapher parser CallInspector attribute) 6 12		

```
modules (codegrapher.parser.ClassObject attribute), 7, 13 visit FunctionDef() (codegrapher.parser.FunctionVisitor
modules (codegrapher.parser.FileObject attribute), 8, 14
modules (codegrapher.parser.FunctionObject attribute), 9,
                                                          visit Import()
modules (codegrapher.parser.ImportVisitor attribute), 10,
                                                          visit ImportFrom()
Ν
name (codegrapher.graph.FunctionGrapher attribute), 5, visit Name() (codegrapher.parser.CallInspector method),
                                                                    7, 13
name (codegrapher.parser.ClassObject attribute), 7, 13
name (codegrapher.parser.FileObject attribute), 8, 14
name (codegrapher.parser.FunctionObject attribute), 9, 15
namespace() (codegrapher.parser.ClassObject method), 8,
namespace() (codegrapher.parser.FileObject method), 9,
Node (class in codegrapher.graph), 6, 12
node (codegrapher.parser.ClassObject attribute), 7, 13
node (codegrapher.parser.FileObject attribute), 8, 14
node (codegrapher.parser.FunctionObject attribute), 9, 15
nodes (codegrapher.graph.FunctionGrapher attribute), 5,
         11
Р
pprint() (codegrapher.parser.ClassObject method), 8, 14
R
relative_namespace (codegrapher.parser.FileObject at-
         tribute), 8, 14
remove_builtins()
                        (codegrapher.parser.ClassObject
         method), 8, 14
remove_builtins()
                          (codegrapher.parser.FileObject
         method), 9, 15
remove_builtins()
                          (codegrapher.parser.FileVisitor
         method), 9, 15
render() (codegrapher.graph.FunctionGrapher method), 6,
represent (codegrapher.graph.Node attribute), 6, 12
Т
tuple (codegrapher.graph.Node attribute), 6, 12
V
visit() (codegrapher.parser.ClassObject method), 8, 14
visit() (codegrapher.parser.FileObject method), 9, 15
visit() (codegrapher.parser.FunctionObject method), 10,
         16
visit Attribute()
                       (codegrapher.parser.CallInspector
         method), 7, 13
visit_Call() (codegrapher.parser.CallVisitor method), 7,
visit_ClassDef() (codegrapher.parser.FileVisitor method),
         9.15
```

method), 10, 16 (codegrapher.parser.ImportVisitor method), 10, 16 (codegrapher.parser.ImportVisitor method), 10, 16 visit Module() (codegrapher.parser.FileVisitor method),

22 Index