PyArgWriter

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

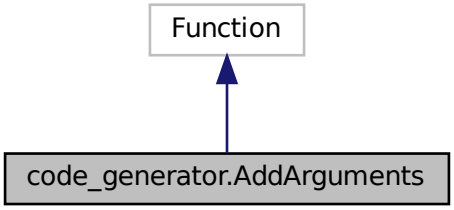
Chapter 3

Class Documentation

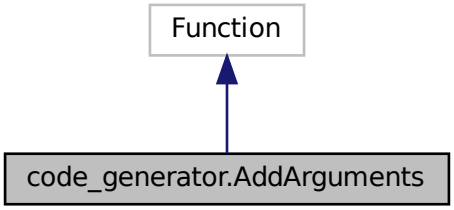
3.1 code_generator.AddArguments Class Reference

Represents a class for adding arguments to a function.

Inheritance diagram for code_generator.AddArguments:



Collaboration diagram for code_generator.AddArguments:



Public Member Functions

- None **__init__** (self, str infix, List[ArgumentStructure] arguments={})

3.1.1 Detailed Description

Represents a class for adding arguments to a function.

This class extends the Function class and is designed for generating functions that add arguments to an Argument↩ Parser instance. It automatically creates and appends the necessary code to add arguments to the parser function.

Parameters

infix	The infix string used to construct the function name and as a part of the argument names. arguments (List[ArgumentStructure], optional): A list of ArgumentStructure objects representing the arguments to be added.

```
(inherited attributes from Function...)
```

Methods

init(self, infix: str, arguments: List[ArgumentStructure] = {}) -> None: Initializes a new AddArguments instance with the specified infix and arguments.

_check_infix(self, infix: str) -> None: Checks the validity of the provided infix string and raises an error if it contains spaces, dashes, or is not in lowercase.

_add_function(self, arguments: List[ArgumentStructure]) -> None: Adds the code to add arguments to the ArgumentParser instance in the function.

Examples

```
>> arguments = [ArgumentStructure(dest='input_file'), ArgumentStructure(dest='output_file')]
>> add_args_function = AddArguments("input", arguments)
>> print(add_args_function)
```

def add_input_args(parser: ArgumentParser) -> ArgumentParser: parser.add_argument('–input-file', type = <class 'str'>, help = 'Path to input file') parser.add_argument('–output-file', type = <class 'str'>, help = 'Path to output file') return parser

The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

3.2 pyargwriter.process.ArgParseWriter Class Reference

Public Member Functions

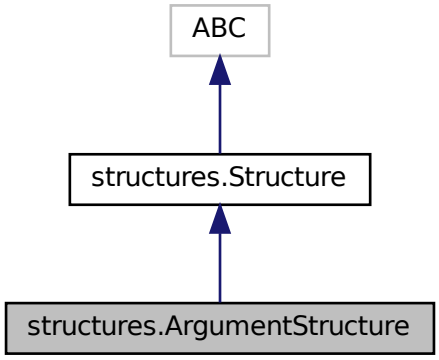
- None **__init__** (self, bool force=False, ∗∗kwargs)
- def **parse_code** (self, List[str] files, str output, ∗∗kwargs)
- def **write_code** (self, str file, str output, bool pretty=False, ∗∗kwargs)
- def **generate_parser** (self, List[str] files, str output, bool pretty=False, ∗∗kwargs)

The documentation for this class was generated from the following file:

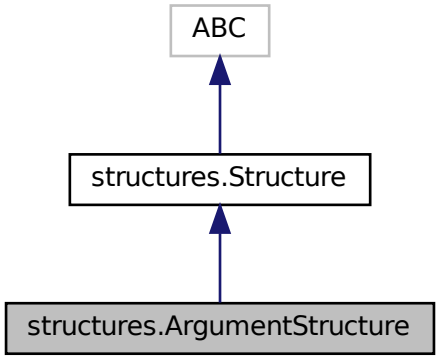
- documentation/tmp/pyargwriter/process.py

3.3 structures.ArgumentStructure Class Reference

Inheritance diagram for structures.ArgumentStructure:



Collaboration diagram for structures.ArgumentStructure:



Public Member Functions

- None **__init__** (self)
- ArgumentStructure from_dict (ArgumentStructure cls, Dict[str, str] data)

  Create an instance of the ArgumentStructure class from a dictionary.
- Dict[str, str] to_dict (self)

  Convert the argument structure to a dictionary representation.

3.3.1 Member Function Documentation

3.3.1.1 from_dict()

```
ArgumentStructure structures.ArgumentStructure.from_dict (
            ArgumentStructure cls,
            Dict[str, str] data )
```

Create an instance of the ArgumentStructure class from a dictionary.

Parameters

cls	The class itself. data (Dict[str, str]): The dictionary containing data to create the instance from.

Returns

ArgumentStructure An instance of the ArgumentStructure class created from the dictionary.

3.3.1.2 to_dict()

```
Dict[str, str] structures.ArgumentStructure.to_dict (
              self )
```

Convert the argument structure to a dictionary representation.

Returns

dict A dictionary representation of the argument structure.

Reimplemented from structures.Structure.

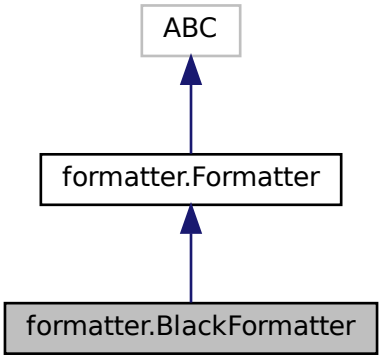
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/structures.py

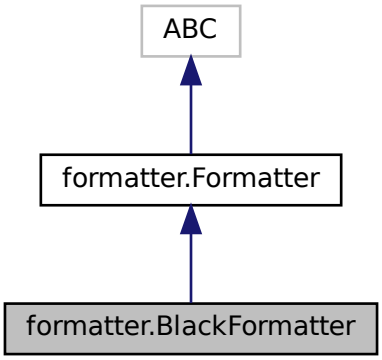
3.4 formatter.BlackFormatter Class Reference

Formatter for code using the 'black' code formatter.

Inheritance diagram for formatter.BlackFormatter:



Collaboration diagram for formatter.BlackFormatter:



Public Member Functions

- None **__init__** (self)
- def **format** (self, List[str] files)

Public Attributes

- name

  The name of the code formatter ('black').

3.4.1 Detailed Description

Formatter for code using the 'black' code formatter.

This class implements the 'format' method to format source code files using the 'black' code formatter.

Methods

format(self, files: List[str]) -> None: Format the given list of source code files using 'black'.

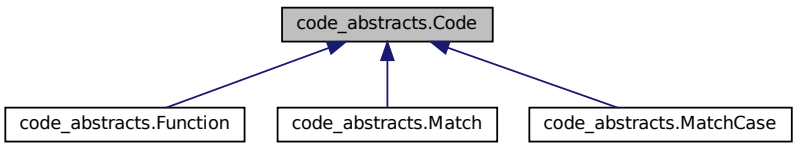
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/formatter.py

3.5 code_abstracts.Code Class Reference

Represents a collection of code lines.

Inheritance diagram for code_abstracts.Code:



Public Member Functions

- None **__init__** (self)
- str **__repr__** (self)
- int **__len__** (self)
- Code from_lines_of_code (Code cls, List[LineOfCode] code)

  Create a Code instance from a list of LineOfCode objects.
- Code from_str (Code cls, str code)

  Create a Code instance from a single string representing a line of code.
- None insert (self, List[LineOfCode]|LineOfCode|Code content, int index)

  insert given lines of code into self._file of class at given index
- None append (self, str|Code|LineOfCode|List[LineOfCode] content)

  Append content to the end of the code block.
- None write (self, str path)

  Write the code block to a file.
- def write_force (self, str path)

  Write the code block to a file without asking if you to overwrite existing files.
- List[LineOfCode] file (self)

  Get the list of code lines in the code block.
- None set_tab_level (self, int tab_level)

  Set the tab level for all lines of code in the code block.

3.5.1 Detailed Description

Represents a collection of code lines.

This class represents a collection of code lines with indentation. It is used to build and manipulate code blocks.

Methods

insert(self, content: List[LineOfCode] | LineOfCode | Code, index: int) -> None: Insert code lines at a specified index.

append(self, content: str | Code | LineOfCode | List[LineOfCode]) -> None: Append code lines to the end of the code block.

from_lines_of_code Code, code: List[LineOfCode]) -> Code: Create a Code instance from a list of LineOfCode objects.

from_str Code, code: str) -> Code: Create a Code instance from a single string representing a line of code.

_split_file(self, index: int) -> Tuple[List[LineOfCode], List[LineOfCode]]: Split the code block into two parts at the specified index.

_insert_line_of_code(first List[LineOfCode], line_of_code: LineOfCode, second: List[LineOfCode]) -> List[LineOfCode]: Insert a single line of code into the code block.

_insert_lines_of_code(first List[LineOfCode], lines_of_code List[LineOfCode], second List[LineOfCode],) -> List[LineOfCode]: Insert multiple lines of code into the code block.

_insert_code(self, first: List[LineOfCode], code: Code, second: List[LineOfCode]) -> List[LineOfCode]↩ : Insert another Code instance into the code block.

_write(self, path: str) -> None: Write the code block to a specified file path.

file(self) -> List[LineOfCode]: Get the list of code lines in the code block.

set_tab_level(self, tab_level: int) -> None: Set the tab level for all lines of code in the code block.

3.5.2 Member Function Documentation

3.5.2.1 append()

```
None code_abstracts.Code.append (
             self,
            str | Code | LineOfCode | List[LineOfCode] content )
```

Append content to the end of the code block.

content (str | Code | LineOfCode | List[LineOfCode]): Content to append.

3.5.2.2 file()

```
List[LineOfCode] code_abstracts.Code.file (
             self )
```

Get the list of code lines in the code block.

Returns

List The list of code lines.

3.5.2.3 from_lines_of_code()

```
Code code_abstracts.Code.from_lines_of_code (
            Code cls,
            List[LineOfCode] code )
```

Create a Code instance from a list of LineOfCode objects.

Parameters

code	List of LineOfCode objects.

Returns

Code A Code instance containing the specified code lines.

3.5.2.4 from_str()

```
Code code_abstracts.Code.from_str (
            Code cls,
            str code )
```

Create a Code instance from a single string representing a line of code.

Parameters

code	A single string representing a line of code.

Returns

Code A Code instance containing the specified code line.

3.5.2.5 insert()

```
None code_abstracts.Code.insert (
             self,
            List[LineOfCode] | LineOfCode | Code content,
            int index )
```

insert given lines of code into self._file of class at given index

```
        content (List[LineOfCode] | LineOfCode): Lines of code to insert
```

Parameters

index	Where to insert the given content

3.5.2.6 set_tab_level()

```
None code_abstracts.Code.set_tab_level (
             self,
            int tab_level )
```

Set the tab level for all lines of code in the code block.

```
   This method sets the tab level for all lines of code in the code block.
   It adjusts the indentation of each line based on the specified tab level.
```

Parameters

tab_level	The tab level to set. Must be a non-negative integer.

3.5.2.7 write()

```
None code_abstracts.Code.write (
              self,
             str path )
```

Write the code block to a file.

Parameters

path	The file path where the code block should be written.

3.5.2.8 write_force()

```
def code_abstracts.Code.write_force (
              self,
             str path )
```

Write the code block to a file without asking if you to overwrite existing files.

Parameters

path	The file path where the code block should be written.

The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_abstracts.py

3.6 code_generator.CodeGenerator Class Reference

Generates Python code for creating argparse-based command-line parsers.

Public Member Functions

- None **__init__** (self)
- None from_dict (self, List[Dict[str, Any]] modules, str parser_file)

  Generates code based on a list of module dictionaries and a parser file name.

- None from_yaml (self, str yaml_file, str parser_file)

  Generates code from a YAML file and a parser file name.
- None from_json (self, str json_file, str parser_file)

  Generates code from a JSON file and a parser file name.
- None write (self, str setup_parser_path, str main_path, bool force=False)

  Generates code from a JSON file and a parser file name.

3.6.1 Detailed Description

Generates Python code for creating argparse-based command-line parsers.

This class provides methods to generate Python code for creating argparse-based command-line parsers, including the setup parser, main function, and main caller.

Parameters

None	

Methods

from_dict List[Dict[str, Any]], parser_file: str) -> None: Generates code based on a list of module dictionaries and a parser file name. from_yaml str, parser_file: str): Generates code from a YAML file and a parser file name. from_json str, parser_file: str): Generates code from a JSON file and a parser file name. write str, main_path: str, force: bool = False): Writes the generated code to specified files.

Examples

```
>> generator = CodeGenerator()
>> modules_data = [{"name": "module1", "commands": [...]}, {"name": "module2", "commands": [...]}]
>> parser_file = "my_parser.py"
>> generator.from_dict(modules_data, parser_file)
>> generator.write("setup_parser.py", "main.py")
```

3.6.2 Member Function Documentation

3.6.2.1 from_dict()

```
None code_generator.CodeGenerator.from_dict (
             self,
            List[Dict[str, Any]] modules,
            str parser_file )
```

Generates code based on a list of module dictionaries and a parser file name.

```
        modules (List[Dict[str, Any]]): A list of dictionaries representing the module structure.
```

Parameters

parser_file	The path to the future parser file.

3.6.2.2 from_json()

```
None code_generator.CodeGenerator.from_json (
             self,
            str json_file,
            str parser_file )
```

Generates code from a JSON file and a parser file name.

Parameters

json_file	The path to the JSON file containing module structure data.
parser_file	The path to the future parser file.

3.6.2.3 from_yaml()

```
None code_generator.CodeGenerator.from_yaml (
             self,
            str yaml_file,
            str parser_file )
```

Generates code from a YAML file and a parser file name.

Parameters

yaml_file	The path to the YAML file containing module structure data.
parser_file	The path to the future parser file.

3.6.2.4 write()

```
None code_generator.CodeGenerator.write (
             self,
            str setup_parser_path,
            str main_path,
            bool  force = False )
```

Generates code from a JSON file and a parser file name.

Parameters

json_file	The path to the JSON file containing module structure data.
parser_file	The path to the future parser file.

The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

3.7 code_parser.CodeParser Class Reference

A parser for analyzing Python code and extracting structured information.

Public Member Functions

- None **__init__** (self)
- str __repr__ (self)

  Return a string representation of the parsed modules.
- def module_serialized (self)

  Serialize the module information as a list of dictionaries.
- def write (self, str path)

  Write the extracted structured information to a file in YAML or JSON format.
- def parse_tree (self, Module tree, str file)

  Parse the Abstract Syntax Tree (AST) of a Python module and extract structured information.

Public Attributes

- modules

  A collection of ModuleStructure objects.

3.7.1 Detailed Description

A parser for analyzing Python code and extracting structured information.

This class is designed to parse Python code and extract structured information about classes, methods, and their associated arguments and documentation.

Methods

parse_tree(self, tree: Module, file: str) -> None: Parse the Abstract Syntax Tree (AST) of a Python module and extract structured information.

write(self, path: str) -> None: Write the extracted structured information to a file in YAML or JSON format.

Properties

module_serialized A list of dictionaries representing serialized module information.

3.7.2 Member Function Documentation

3.7.2.1 __repr__()

```
str code_parser.CodeParser.__repr__ (
              self )
```

Return a string representation of the parsed modules.

Returns

str A string representation of the parsed modules.

3.7.2.2 module_serialized()

```
def code_parser.CodeParser.module_serialized (
              self )
```

Serialize the module information as a list of dictionaries.

Returns

List A list of dictionaries representing serialized module information.

3.7.2.3 parse_tree()

```
def code_parser.CodeParser.parse_tree (
              self,
             Module tree,
             str file )
```

Parse the Abstract Syntax Tree (AST) of a Python module and extract structured information.

Parameters

tree	The AST of the Python module.
file	The path to the Python module file.

3.7.2.4 write()

```
def code_parser.CodeParser.write (
              self,
             str path )
```

Write the extracted structured information to a file in YAML or JSON format.

Parameters

path	The path to the output file.

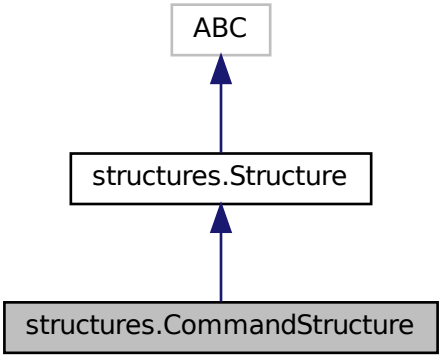
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_parser.py

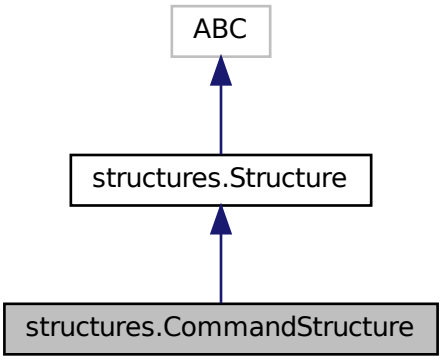
3.8 structures.CommandStructure Class Reference

Class representing a command structure.

Inheritance diagram for structures.CommandStructure:



Collaboration diagram for structures.CommandStructure:



Public Member Functions

- None **__init__** (self)
- int __len__ (self)

  Return the number of arguments for this command.
- CommandStructure from_dict (CommandStructure cls, dict data)

  Create an instance of the CommandStructure class from a dictionary.
- Dict[str, str] to_dict (self)

  Convert the command structure to a dictionary representation.

3.8.1 Detailed Description

Class representing a command structure.

This class defines the structure for commands.

Methods

from_dict(cls, data: dict) -> CommandStructure: Create an instance of the class from a dictionary.

to_dict(self) -> dict: Convert the command structure to a dictionary representation.

3.8.2 Member Function Documentation

3.8.2.1 __len__()

```
int structures.CommandStructure.__len__ (
            self )
```

Return the number of arguments for this command.

Returns

int The number of arguments for this command.

3.8.2.2 from_dict()

```
CommandStructure structures.CommandStructure.from_dict (
            CommandStructure cls,
            dict data )
```

Create an instance of the CommandStructure class from a dictionary.

Parameters

cls	The class itself.
data	The dictionary containing data to create the instance from.

Returns

CommandStructure An instance of the CommandStructure class created from the dictionary.

3.8.2.3 to_dict()

```
Dict[str, str] structures.CommandStructure.to_dict (
            self )
```

Convert the command structure to a dictionary representation.

Returns

dict A dictionary representation of the command structure.

Reimplemented from structures.Structure.

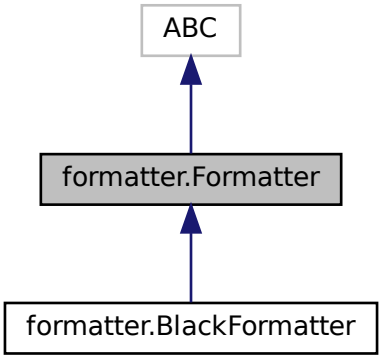
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/structures.py

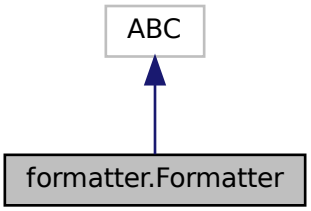
3.9 formatter.Formatter Class Reference

Abstract base class for code formatters.

Inheritance diagram for formatter.Formatter:



Collaboration diagram for formatter.Formatter:



Public Member Functions

- def format (self)

  Format the given list of source code files.

3.9.1 Detailed Description

Abstract base class for code formatters.

This class defines an abstract method 'format' that should be implemented by subclasses. Code formatters are used to automatically format source code files.

Methods

format(self, files: List[str]) -> None: Abstract method to format the given list of source code files.

3.9.2 Member Function Documentation

3.9.2.1 format()

```
def formatter.Formatter.format (
              self )
```

Format the given list of source code files.

Parameters

files	A list of file paths to be formatted.

Exceptions

NotImplementedError	This method should be implemented in subclasses.

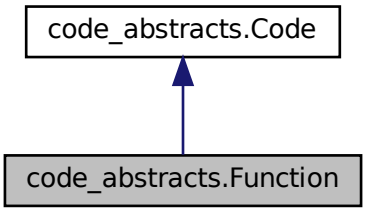
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/formatter.py

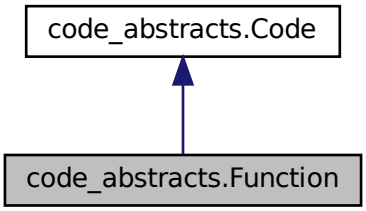
3.10 code_abstracts.Function Class Reference

Represents a Python function.

Inheritance diagram for code_abstracts.Function:



Collaboration diagram for code_abstracts.Function:



Public Member Functions

- None **__init__** (self, str name, Dict[str, Type] signature={}, Type return_type=None)
- def name (self)

  Return the name of the function.

3.10.1 Detailed Description

Represents a Python function.

This class represents a Python function and allows you to build and manipulate its structure.

Parameters

name	The name of the function. signature (Dict[str, Type], optional): A dictionary representing the function's signature. return_type (Type, optional): The return type of the function.

Methods

init(self, name: str, signature: Dict[str, Type] = {}, return_type: Type = None) -> None: Initializes a new Function instance.

_generate_header(self): Generates the function header based on the name, signature, and return type.

name(self): Returns the name of the function.

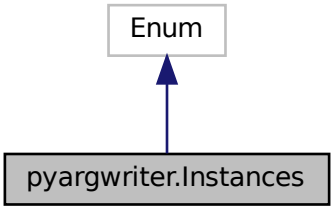
(other class methods...)

The documentation for this class was generated from the following file:

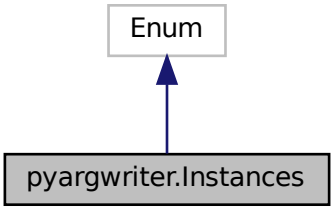
- documentation/tmp/pyargwriter/utils/code_abstracts.py

3.11 pyargwriter.Instances Class Reference

Inheritance diagram for pyargwriter.Instances:



Collaboration diagram for pyargwriter.Instances:



Static Public Attributes

- int **CLASSES** = 1

The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/__init__.py

3.12 code_abstracts.LineOfCode Class Reference

Represents a single line of code with indentation.

Public Member Functions

- None **__init__** (self, str content, int tab_level=0)
- str **__repr__** (self)
- int tab_level (self)

  The level of indentation for the line.
- str content (self)

  The content of the line of code.

3.12.1 Detailed Description

Represents a single line of code with indentation.

This class represents a single line of code with a specified level of indentation. It is used to build code blocks and maintain proper indentation.

Parameters

content	The content of the line of code. tab_level (int, optional): The level of indentation for the line. Defaults to 0.

3.12.2 Member Function Documentation

3.12.2.1 content()

```
code_abstracts.LineOfCode.content (
              self )
```

The content of the line of code.

Returns

str The content of the line of code.

3.12.2.2 tab_level()

```
code_abstracts.LineOfCode.tab_level (
              self )
```

The level of indentation for the line.

Returns

int The level of indentation for the line.

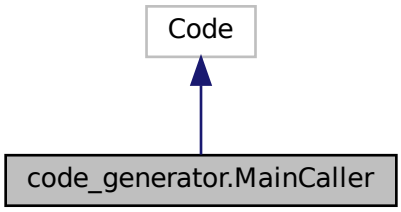
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_abstracts.py

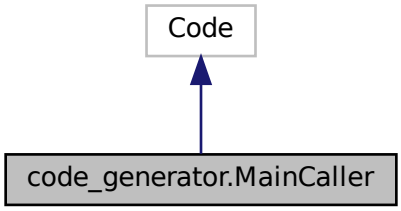
3.13 code_generator.MainCaller Class Reference

Represents a code block for calling the `main()` function if the script is executed as the main program.

Inheritance diagram for code_generator.MainCaller:



Collaboration diagram for code_generator.MainCaller:



Public Member Functions

- None **__init__** (self)

3.13.1 Detailed Description

Represents a code block for calling the `main()` function if the script is executed as the main program.

This class extends the Code class and is designed to generate code that calls the `main()` function if the script is executed as the main program.

Parameters

None	

Methods

None

Examples

```
>> main_caller = MainCaller()
>> print(main_caller)
```

if **name** == '**main**': main()

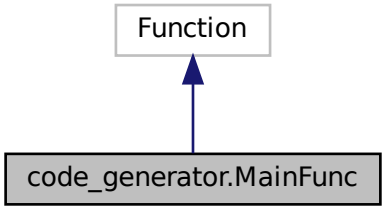
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

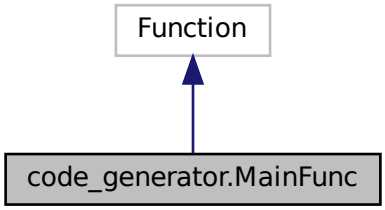
3.14 code_generator.MainFunc Class Reference

Represents the main function of a Python script that uses argparse for command-line arguments.

Inheritance diagram for code_generator.MainFunc:



Collaboration diagram for code_generator.MainFunc:



Public Member Functions

- None __init__ (self)

  summary
- None generate_code (self, ModuleStructures modules, str setup_parser_file="parser.py")

  Generates the code for the main function.

3.14.1 Detailed Description

Represents the main function of a Python script that uses argparse for command-line arguments.

This class extends the Function class and is designed to generate the code for the main function of a Python script that uses argparse for command-line argument parsing.

Parameters

None	

Methods

generate_code ModuleStructures, setup_parser_file: str = "parser.py") -> Any: Generates the code for the main function. _add_content Adds the main function's content. _generate_imports Dict[str, str]) -> Code: Generates import statements for modules. _add_module_logic ModuleStructures): Adds the logic to handle modules and commands. _generate_command_match_case List[CommandStructure]) -> MatchCase: Generates match cases for commands. _generate_module_match_case List[ModuleStructure]) -> MatchCase: Generates match cases for modules.

Examples

```
>> main_function = MainFunc()
>> print(main_function)
```

def main(): parser = ArgumentParser(description='TODO: make it a variable')

```
parser = setup_parser(parser)
args = parser.parse_args()
args_dict = vars(args)
# (module and command logic)
```

3.14.2 Constructor & Destructor Documentation

3.14.2.1 __init__()

```
None code_generator.MainFunc.__init__ (
            self )
```

summary

Parameters

setup_parser_file	relative path to file with setup parser functionalities

3.14.3 Member Function Documentation

3.14.3.1 generate_code()

```
None code_generator.MainFunc.generate_code (
              self,
             ModuleStructures modules,
             str  setup_parser_file = "parser.py" )
```

Generates the code for the main function.

Parameters

modules	A ModuleStructures object containing module and command information. setup_parser_file (str, optional): The relative path to the setup parser file. Defaults to "parser.py".

Returns

Any Generated code for the main function.

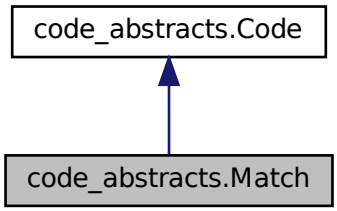
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

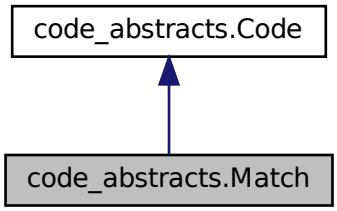
3.15 code_abstracts.Match Class Reference

Represents a Python 'match' expression.

Inheritance diagram for code_abstracts.Match:



Collaboration diagram for code_abstracts.Match:



Public Member Functions

- None **__init__** (self, Any match_value, Code body)
- str match_value (self)

  Return a string representation of the match value.

Public Attributes

- body

  The code block representing the body of the 'match' expression.

3.15.1 Detailed Description

Represents a Python 'match' expression.

This class represents a 'match' expression in Python and allows you to build and manipulate its structure.

Parameters

match_value	The value to match against.
body	The code block representing the body of the 'match' expression.

Methods

(other class methods...)

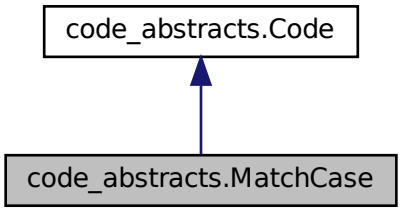
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_abstracts.py

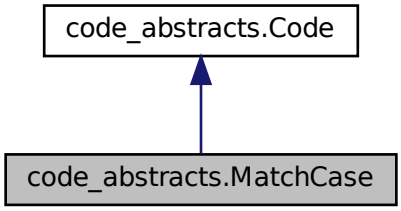
3.16 code_abstracts.MatchCase Class Reference

Represents a 'case' in a Python 'match' expression.

Inheritance diagram for code_abstracts.MatchCase:



Collaboration diagram for code_abstracts.MatchCase:



Public Member Functions

- None **__init__** (self, str match_name, List[Match] matches)

3.16.1 Detailed Description

Represents a 'case' in a Python 'match' expression.

This class represents a 'case' in a Python 'match' expression and allows you to build and manipulate its structure.

Parameters

match_name	The name of the matched value.
matches	A list of Match objects representing different cases.

Methods

(other class methods...)

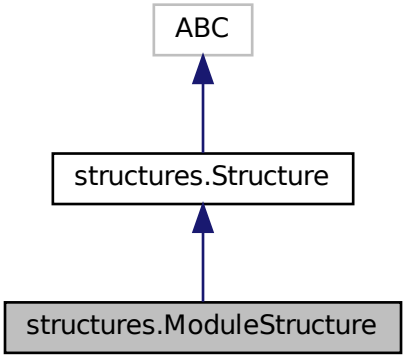
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_abstracts.py

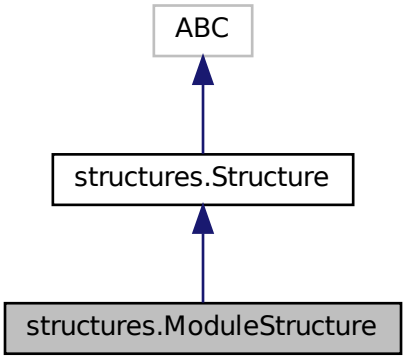
3.17 structures.ModuleStructure Class Reference

Class representing a module structure.

Inheritance diagram for structures.ModuleStructure:



Collaboration diagram for structures.ModuleStructure:



Public Member Functions

- None **__init__** (self)
- int __len__ (self)

  Return the number of commands in this module.
- ModuleStructure from_dict (ModuleStructure cls, Dict[str, str] data)

  Create an instance of the ModuleStructure class from a dictionary.
- Dict[str, str] to_dict (self)

  Convert the module structure to a dictionary representation.
- None add_args (self, List[ArgumentStructure] args)

  Add given arguments to all commands in the module.

3.17.1 Detailed Description

Class representing a module structure.

This class defines the structure for modules.

Methods

from_dict(cls, data: Dict[str, str]) -> ModuleStructure: Create an instance of the class from a dictionary.

to_dict(self) -> dict: Convert the module structure to a dictionary representation.

add_args(self, args: List[ArgumentStructure]) -> None: Add given arguments to all commands in the module.

3.17.2 Member Function Documentation

3.17.2.1 __len__()

```
int structures.ModuleStructure.__len__ (
            self )
```

Return the number of commands in this module.

Returns

int The number of commands in this module.

3.17.2.2 add_args()

```
None structures.ModuleStructure.add_args (
             self,
            List[ArgumentStructure] args )
```

Add given arguments to all commands in the module.

Parameters

args	A list of arguments to add to the module's commands.

3.17.2.3 from_dict()

```
ModuleStructure structures.ModuleStructure.from_dict (
            ModuleStructure cls,
            Dict[str, str] data )
```

Create an instance of the ModuleStructure class from a dictionary.

Parameters

cls	The class itself. data (Dict[str, str]): The dictionary containing data to create the instance from.

Returns

ModuleStructure An instance of the ModuleStructure class created from the dictionary.

3.17.2.4 to_dict()

```
Dict[str, str] structures.ModuleStructure.to_dict (
              self )
```

Convert the module structure to a dictionary representation.

Returns

dict A dictionary representation of the module structure.

Reimplemented from structures.Structure.

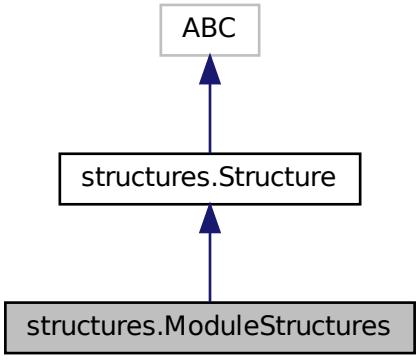
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/structures.py

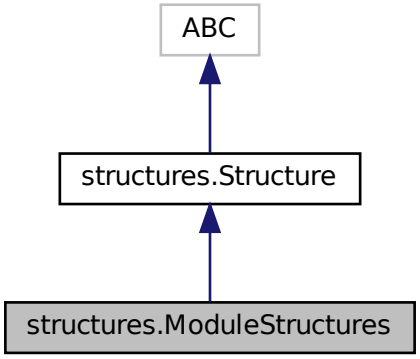
3.18 structures.ModuleStructures Class Reference

Class representing a collection of module structures.

Inheritance diagram for structures.ModuleStructures:



Collaboration diagram for structures.ModuleStructures:



Public Member Functions

- None **__init__** (self)
- int __len__ (self)

  Return the number of modules in the collection.
- def from_dict (ModuleStructures cls, Dict[str, str] data)

  Create an instance of the ModuleStructures class from a dictionary.

- Dict[str, str] to_dict (self)

  Convert the collection of module structures to a dictionary representation.
- Dict[str, str] locations (self)

  Return a dictionary of module names and their corresponding locations.

3.18.1 Detailed Description

Class representing a collection of module structures.

This class defines a collection of ModuleStructure objects.

Methods

from_dict(cls, data: Dict[str, str]) -> ModuleStructures: Create an instance of the class from a dictionary.

to_dict(self) -> dict: Convert the collection of module structures to a dictionary representation.

Properties

locations: Returns a dictionary of module names and their corresponding locations.

3.18.2 Member Function Documentation

3.18.2.1 __len__()

```
int structures.ModuleStructures.__len__ (
            self )
```

Return the number of modules in the collection.

Returns

int The number of modules in the collection.

3.18.2.2 from_dict()

```
def structures.ModuleStructures.from_dict (
            ModuleStructures cls,
            Dict[str, str] data )
```

Create an instance of the ModuleStructures class from a dictionary.

Parameters

cls	The class itself. data (Dict[str, str]): The dictionary containing data to create the instance from.

Returns

ModuleStructures An instance of the ModuleStructures class created from the dictionary.

3.18.2.3 locations()

```
Dict[str, str] structures.ModuleStructures.locations (
              self )
```

Return a dictionary of module names and their corresponding locations.

Returns

dict A dictionary mapping module names to their corresponding locations.

3.18.2.4 to_dict()

```
Dict[str, str] structures.ModuleStructures.to_dict (
              self )
```

Convert the collection of module structures to a dictionary representation.

Returns

dict A dictionary representation of the collection of module structures.

Reimplemented from structures.Structure.

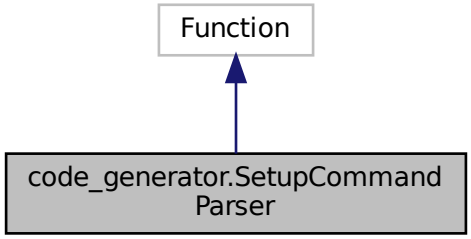
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/structures.py

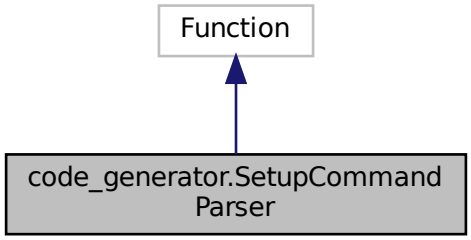
3.19 code_generator.SetupCommandParser Class Reference

Represents a function generator for setting up an ArgumentParser with subcommands.

Inheritance diagram for code_generator.SetupCommandParser:



Collaboration diagram for code_generator.SetupCommandParser:



Public Member Functions

- None **__init__** (self, str module_name, bool no_imports=False)
- None generate_code (self, List[CommandStructure] commands)

  Generates the code to set up the ArgumentParser with subcommands, including imports (if enabled).

3.19.1 Detailed Description

Represents a function generator for setting up an ArgumentParser with subcommands.

This class extends the Function class and is designed for generating functions that set up an ArgumentParser with subcommands based on a list of CommandStructure objects. It can optionally include imports for ArgumentParser.

Parameters

module_name	The name of the module or command group. no_imports (bool, optional): If True, omit importing ArgumentParser; otherwise, include the import.

```
(inherited attributes from Function...)
```

Methods

generate_code(self, commands: List[CommandStructure]) -> Any: Generates the code to set up the ArgumentParser with subcommands, including imports (if enabled).

Examples

```
>> commands = [CommandStructure(name='create', help='Create a new item'), CommandStructure(name='delete',
       help='Delete an existing item')]
>> setup_parser_function = SetupCommandParser('my_module', no_imports=True)
>> setup_parser_function.generate_code(commands)
>> print(setup_parser_function)
```

def setup_my_module_parser(parser: ArgumentParser) -> ArgumentParser: command_subparser = parser.add↩
_subparsers(dest='command', title='command')

```
create = command_subparser.add_parser('create', help='Create a new item')
delete = command_subparser.add_parser('delete', help='Delete an existing item')
```

return parser

3.19.2 Member Function Documentation

3.19.2.1 generate_code()

```
None code_generator.SetupCommandParser.generate_code (
              self,
             List[CommandStructure] commands )
```

Generates the code to set up the ArgumentParser with subcommands, including imports (if enabled).

Parameters

commands	A list of CommandStructure objects representing the subcommands to be added.

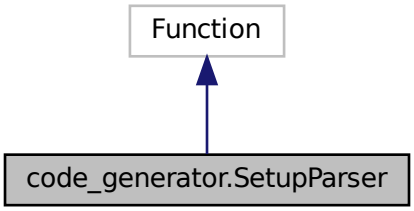
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

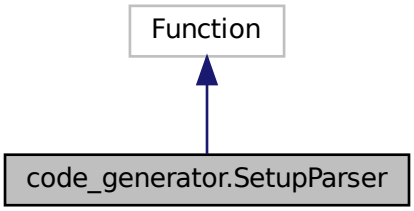
3.20 code_generator.SetupParser Class Reference

Represents a function generator for setting up an ArgumentParser with subcommands for multiple modules.

Inheritance diagram for code_generator.SetupParser:



Collaboration diagram for code_generator.SetupParser:



Public Member Functions

- None **__init__** (self)
- None generate_code (self, ModuleStructures modules)

  Generates the code to set up the ArgumentParser with subcommands for multiple modules.
- def from_yaml (self, str yaml_file)

  Generates the code based on a YAML configuration file.
- def from_json (self, str json_file)

  Generates the code based on a JSON configuration file.

3.20.1 Detailed Description

Represents a function generator for setting up an ArgumentParser with subcommands for multiple modules.

This class extends the Function class and is designed for generating functions that set up an ArgumentParser with subcommands for multiple modules based on ModuleStructures. It can handle the case of a single module or multiple modules with individual subparsers. Imports for ArgumentParser are included based on the number of modules.

Parameters

None	(inherited attributes from Function...)

Methods

generate_code(self, modules: ModuleStructures) -> None: Generates the code to set up the ArgumentParser with subcommands for multiple modules. from_yaml(self, yaml_file: str) -> None: Generates the code based on a YAML configuration file. from_json(self, json_file: str) -> None: Generates the code based on a JSON configuration file.

Examples

```
>> modules = ModuleStructures(modules=[ModuleStructure(name='module1'), ModuleStructure(name='module2')])
>> setup_parser_function = SetupParser()
>> setup_parser_function.generate_code(modules)
>> print(setup_parser_function)
```

def setup_parser(parser: ArgumentParser) -> ArgumentParser: module_subparser = parser.add_subparsers(dest='module', title='module')

```
module1_parser = module_subparser.add_parser(name='module1', help='TODO')
setup_module1_parser = SetupCommandParser('module1', no_imports=False)
setup_module1_parser.generate_code([])
module1_parser = setup_module1_parser(module1_parser)
module2_parser = module_subparser.add_parser(name='module2', help='TODO')
setup_module2_parser = SetupCommandParser('module2', no_imports=True)
setup_module2_parser.generate_code([])
module2_parser = setup_module2_parser(module2_parser)
```

return parser

3.20.2 Member Function Documentation

3.20.2.1 from_json()

```
def code_generator.SetupParser.from_json (
              self,
             str json_file )
```

Generates the code based on a JSON configuration file.

Parameters

json_file	The path to the JSON configuration file.

3.20.2.2 from_yaml()

```
def code_generator.SetupParser.from_yaml (
              self,
             str yaml_file )
```

Generates the code based on a YAML configuration file.

Parameters

yaml_file	The path to the YAML configuration file.

3.20.2.3 generate_code()

```
None code_generator.SetupParser.generate_code (
              self,
             ModuleStructures modules )
```

Generates the code to set up the ArgumentParser with subcommands for multiple modules.

Parameters

modules	A ModuleStructures object containing information about the modules and their subcommands.

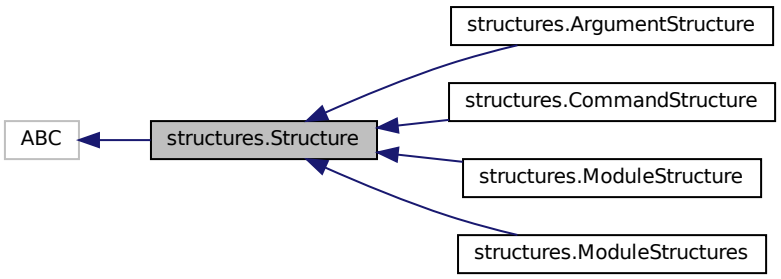
The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/code_generator.py

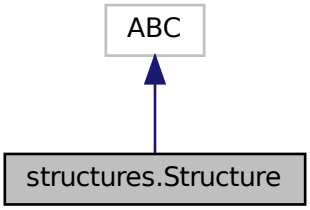
3.21 structures.Structure Class Reference

Abstract base class for defining structured objects.

Inheritance diagram for structures.Structure:



Collaboration diagram for structures.Structure:



Public Member Functions

- def __repr__ (self)

  Return a JSON representation of the structured object.
- def from_dict (Structure cls, dict data)

  Create an instance of the class from a dictionary.
- dict to_dict (self)

  Convert the object to a dictionary representation.

3.21.1 Detailed Description

Abstract base class for defining structured objects.

This class defines methods for converting structured objects to and from dictionaries, as well as providing a JSON representation of the object.

Methods

from_dict(cls, data: dict) -> Structure: Abstract method for creating an instance of the class from a dictionary.

to_dict(self) -> dict: Abstract method for converting the object to a dictionary representation.

3.21.2 Member Function Documentation

3.21.2.1 from_dict()

```
def structures.Structure.from_dict (
            Structure cls,
            dict data )
```

Create an instance of the class from a dictionary.

Parameters

cls	The class itself.
data	The dictionary containing data to create the instance from.

Exceptions

NotImplementedError	This method should be implemented in subclasses.

Returns

Structure An instance of the class created from the dictionary.

3.21.2.2 to_dict()

```
dict structures.Structure.to_dict (
              self )
```

Convert the object to a dictionary representation.

Exceptions

NotImplementedError	This method should be implemented in subclasses.

Returns

dict A dictionary representation of the object.

Reimplemented in structures.ModuleStructures, structures.ModuleStructure, structures.CommandStructure, and structures.ArgumentStructure.

The documentation for this class was generated from the following file:

- documentation/tmp/pyargwriter/utils/structures.py

Index