# Anchorman Documentation

***Release 0.1.0***

**Matthias Rebel**

December 14, 2015

Contents


**1 anchorman package** **1**
- 1.1 Subpackages . . . 1
- 1.2 Submodules . . . 2
- 1.3 Module contents . . . 4

**2 anchorman.configure module** **5**

**3 anchorman.generator package** **7**
- 3.1 Submodules . . . 7
- 3.2 Module contents . . . 8

**4 anchorman.generator.candidate module** **9**

**5 anchorman.generator.element module** **11**

**6 anchorman.generator.highlight module** **13**

**7 anchorman.generator.tag module** **15**

**8 anchorman.generator.text module** **17**

**9 anchorman.main module** **19**

**10 anchorman.positioner package** **21**
- 10.1 Submodules . . . 21
- 10.2 Module contents . . . 21

**11 anchorman.positioner.interval module** **23**

**12 anchorman.positioner.slices module** **25**

**13 anchorman.utils module** **27**

**14 anchorman** **29**
- 14.1 anchorman package . . . 29

**15 Uml view** **33**

**16 Report** **35**
- 16.1 Statistics by type . . . 35

16.2 External dependencies . . . 35
16.3 Raw metrics . . . 35
16.4 Duplication . . . 36
16.5 Messages by category . . . 36
16.6 % errors / warnings by module . . . 36
16.7 Messages . . . 36
16.8 Global evaluation . . . 36

**17 Todo list** **37**

**18 Credits and contributions** **39**

**19 Indices and tables** **41**

**Python Module Index** **43**

CHAPTER 1

# anchorman package

## 1.1 Subpackages

### 1.1.1 anchorman.generator package

#### Submodules

##### anchorman.generator.candidate module

anchorman.generator.candidate.**data_val**(*item*, *replaces_per_attribute*)

anchorman.generator.candidate.**elements_of_unit**(*intervaltree*, *unit*, *setting*)
Get all items / elements of the actual unit to validate.

anchorman.generator.candidate.**retrieve_hits**(*intervaltree*, *units*, *config*, *own_validator*)
Loop the units and validate each item in unit.

anchorman.generator.candidate.**validate**(*item*, *candidates*, *this_unit*, *setting*, *own_validator*)
Apply the rules specified in setting to the item.

Take care of candidates already validated and the items already added to this_unit.

**Todo**

check context of replacement: do not add links in links, or inline of overlapping elements, ... replace only one item of an entity > e.g. A. Merkel, Mum Merkel, ...

##### anchorman.generator.element module

anchorman.generator.element.**create_element**(*element_pattern*, *item*, *mode*, *markup*)
Create the element that will be inserted in the text.

anchorman.generator.element.**create_element_pattern**(*mode*, *markup*)
Create the basic element pattern based on mode and markup.

##### anchorman.generator.highlight module

anchorman.generator.highlight.**augment_highlight**(*highlight*, *item*)
Fill the base highlight element with data of the item.

anchorman.generator.highlight.**create_highlight**(*highlight_markup*)
Use format to create a base highlight element.

#### anchorman.generator.tag module

anchorman.generator.tag.**augment_bs4tag**(*bs4tag*, *item*, *tag_markup*)
Fill the base bs4tag element with data of the item.

anchorman.generator.tag.**create_bs4tag**(*tag_markup*)
Use BeautifulSoup to create a base tag element.

#### anchorman.generator.text module

anchorman.generator.text.**augment**(*text*, *to_be_applied*)
Augment the text with the elements in to be applied.

### Module contents

## 1.1.2 anchorman.positioner package

### Submodules

#### anchorman.positioner.interval module

anchorman.positioner.interval.**intervals**(*text*, *elements*, *setting*)
From the slices of elements and units create an intervaltree.

anchorman.positioner.interval.**to_intervaltree**(*data*, *t=None*)
Create an intervaltree of all elements (elements, units, ...).

anchorman.positioner.interval.**unit_intervals**(*intervaltree*, *text_unit*)
Loop the intervaltree to get the text unit interval items.

#### anchorman.positioner.slices module

anchorman.positioner.slices.**element_slices**(*text*, *elements*, *element_identifier*)
Get slices of all elements in text.

anchorman.positioner.slices.**unit_slices**(*text*, *text_unit*)
Get slices of the text units specified in setting.

### Module contents

# 1.2 Submodules

## 1.2.1 anchorman.configure module

anchorman.configure.**get_config**(*project_conf=True*)
Load default configuration.

### 1.2.2 anchorman.main module

anchorman.main.**annotate**(*text, elements, own_validator=[], config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Find and annotate elements in text.

Create an invaltree with elements and units of text, validate the rules to apply elements and augment the text with this result.

**Parameters**

- **text** (*str*) – The first parameter.
- **elements** (*list*) – It is a list of element dicts like the following: {'fox': {'value': '/wiki/fox', 'data-type': 'animal'}}
- **own_validator** (*list*) – A list of functions that will be applied in the validation of an element, if it will be applied in the text.
- **config** (*dict*) – Load default config from etc/ or get_config the default config andd update to your own rules.

**Returns** **text** – The annotated text.

**Return type** str

#### Examples

Basic example with config overwrite:

```
>>> text = 'The quick brown fox jumps over the lazy dog.'
>>> elements = [
        {'fox': {
            'value': '/wiki/fox', 'data-type': 'animal'}},
        {'dog': {
            'value': '/wiki/dog', 'data-type': 'animal'}}]
>>> cfg = get_config()
>>> cfg['setting']['replaces_at_all'] = 1
>>> print annotate(text, elements, config=cfg)
'The quick brown <a href="/wiki/fox" data-type="animal">fox</a> jumps over the lazy dog .'
```

anchorman.main.**clean**(*text, config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Remove elements from text.

Use config to identify elements.

---

**Todo**

Implement me ...clean(text .

### 1.2.3 anchorman.utils module

## 1.3 Module contents

CHAPTER 2

# anchorman.configure module

anchorman.configure.**get_config**(*project_conf=True*)
Load default configuration.

CHAPTER 3

# anchorman.generator package

## 3.1 Submodules

### 3.1.1 anchorman.generator.candidate module

anchorman.generator.candidate.**data_val**(*item*, *replaces_per_attribute*)

anchorman.generator.candidate.**elements_of_unit**(*intervaltree*, *unit*, *setting*)
Get all items / elements of the actual unit to validate.

anchorman.generator.candidate.**retrieve_hits**(*intervaltree*, *units*, *config*, *own_validator*)
Loop the units and validate each item in unit.

anchorman.generator.candidate.**validate**(*item*, *candidates*, *this_unit*, *setting*, *own_validator*)
Apply the rules specified in setting to the item.

Take care of candidates already validated and the items already added to this_unit.

---

**Todo**

check context of replacement: do not add links in links, or inline of overlapping elements, ... replace only one item of an entity > e.g. A. Merkel, Mum Merkel, ...

---

### 3.1.2 anchorman.generator.element module

anchorman.generator.element.**create_element**(*element_pattern*, *item*, *mode*, *markup*)
Create the element that will be inserted in the text.

anchorman.generator.element.**create_element_pattern**(*mode*, *markup*)
Create the basic element pattern based on mode and markup.

### 3.1.3 anchorman.generator.highlight module

anchorman.generator.highlight.**augment_highlight**(*highlight*, *item*)
Fill the base highlight element with data of the item.

anchorman.generator.highlight.**create_highlight**(*highlight_markup*)
Use format to create a base highlight element.

### 3.1.4 anchorman.generator.tag module

anchorman.generator.tag.**augment_bs4tag**(*bs4tag*, *item*, *tag_markup*)
Fill the base bs4tag element with data of the item.

anchorman.generator.tag.**create_bs4tag**(*tag_markup*)
Use BeautifulSoup to create a base tag element.

### 3.1.5 anchorman.generator.text module

anchorman.generator.text.**augment**(*text*, *to_be_applied*)
Augment the text with the elements in to be applied.

## 3.2 Module contents

CHAPTER 4

# anchorman.generator.candidate module

anchorman.generator.candidate.**data_val**(*item*, *replaces_per_attribute*)

anchorman.generator.candidate.**elements_of_unit**(*intervaltree*, *unit*, *setting*)
Get all items / elements of the actual unit to validate.

anchorman.generator.candidate.**retrieve_hits**(*intervaltree*, *units*, *config*, *own_validator*)
Loop the units and validate each item in unit.

anchorman.generator.candidate.**validate**(*item*, *candidates*, *this_unit*, *setting*, *own_validator*)
Apply the rules specified in setting to the item.

Take care of candidates already validated and the items already added to this_unit.

**Todo**

check context of replacement: do not add links in links, or inline of overlapping elements, ... replace only one item of an entity > e.g. A. Merkel, Mum Merkel, ...

CHAPTER 5

# anchorman.generator.element module

anchorman.generator.element.**create_element**(*element_pattern*, *item*, *mode*, *markup*)
Create the element that will be inserted in the text.

anchorman.generator.element.**create_element_pattern**(*mode*, *markup*)
Create the basic element pattern based on mode and markup.

CHAPTER 6

# anchorman.generator.highlight module

anchorman.generator.highlight.**augment_highlight**(*highlight*, *item*)
Fill the base highlight element with data of the item.

anchorman.generator.highlight.**create_highlight**(*highlight_markup*)
Use format to create a base highlight element.

CHAPTER 7

# anchorman.generator.tag module

anchorman.generator.tag.**augment_bs4tag**(*bs4tag*, *item*, *tag_markup*)
Fill the base bs4tag element with data of the item.

anchorman.generator.tag.**create_bs4tag**(*tag_markup*)
Use BeautifulSoup to create a base tag element.

CHAPTER 8

# anchorman.generator.text module

anchorman.generator.text.**augment**(*text*, *to_be_applied*)
Augment the text with the elements in to be applied.

CHAPTER 9

# anchorman.main module

anchorman.main.**annotate**(*text, elements, own_validator=[], config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Find and annotate elements in text.

Create an invaltree with elements and units of text, validate the rules to apply elements and augment the text with this result.

**Parameters**

- **text** (*str*) – The first parameter.
- **elements** (*list*) – It is a list of element dicts like the following: {'fox': {'value': '/wiki/fox', 'data-type': 'animal'}}
- **own_validator** (*list*) – A list of functions that will be applied in the validation of an element, if it will be applied in the text.
- **config** (*dict*) – Load default config from etc/ or get_config the default config andd update to your own rules.

**Returns** **text** – The annotated text.

**Return type** str

### Examples

Basic example with config overwrite:

```
>>> text = 'The quick brown fox jumps over the lazy dog.'
>>> elements = [
        {'fox': {
            'value': '/wiki/fox', 'data-type': 'animal'}},
        {'dog': {
            'value': '/wiki/dog', 'data-type': 'animal'}}]
>>> cfg = get_config()
>>> cfg['setting']['replaces_at_all'] = 1
>>> print annotate(text, elements, config=cfg)
'The quick brown <a href="/wiki/fox" data-type="animal">fox</a> jumps over the lazy dog .'
```

anchorman.main.**clean**(*text, config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Remove elements from text.

Use config to identify elements.

---

**Todo**

Implement me ...clean(text .

---

CHAPTER 10

# anchorman.positioner package

## 10.1 Submodules

### 10.1.1 anchorman.positioner.interval module

anchorman.positioner.interval.**intervals**(*text*, *elements*, *setting*)
From the slices of elements and units create an intervaltree.

anchorman.positioner.interval.**to_intervaltree**(*data*, *t=None*)
Create an intervaltree of all elements (elements, units, ...).

anchorman.positioner.interval.**unit_intervals**(*intervaltree*, *text_unit*)
Loop the intervaltree to get the text unit interval items.

### 10.1.2 anchorman.positioner.slices module

anchorman.positioner.slices.**element_slices**(*text*, *elements*, *element_identifier*)
Get slices of all elements in text.

anchorman.positioner.slices.**unit_slices**(*text*, *text_unit*)
Get slices of the text units specified in setting.

## 10.2 Module contents

CHAPTER 11

# anchorman.positioner.interval module

anchorman.positioner.interval.**intervals**(*text*, *elements*, *setting*)
From the slices of elements and units create an intervaltree.

anchorman.positioner.interval.**to_intervaltree**(*data*, *t=None*)
Create an intervaltree of all elements (elements, units, ...).

anchorman.positioner.interval.**unit_intervals**(*intervaltree*, *text_unit*)
Loop the intervaltree to get the text unit interval items.

CHAPTER 12

# anchorman.positioner.slices module

anchorman.positioner.slices.**element_slices**(*text*, *elements*, *element_identifier*)
Get slices of all elements in text.

anchorman.positioner.slices.**unit_slices**(*text*, *text_unit*)
Get slices of the text units specified in setting.

CHAPTER 13

# anchorman.utils module

CHAPTER 14

# anchorman

## 14.1 anchorman package

### 14.1.1 Subpackages

#### anchorman.generator package

##### Submodules

**anchorman.generator.candidate module**

anchorman.generator.candidate.**data_val**(*item*, *replaces_per_attribute*)

anchorman.generator.candidate.**elements_of_unit**(*intervaltree*, *unit*, *setting*)

Get all items / elements of the actual unit to validate.

anchorman.generator.candidate.**retrieve_hits**(*intervaltree*, *units*, *config*, *own_validator*)

Loop the units and validate each item in unit.

anchorman.generator.candidate.**validate**(*item*, *candidates*, *this_unit*, *setting*, *own_validator*)

Apply the rules specified in setting to the item.

Take care of candidates already validated and the items already added to this_unit.

---

**Todo**

check context of replacement: do not add links in links, or inline of overlapping elements, ... replace only one item of an entity > e.g. A. Merkel, Mum Merkel, ...

---

**anchorman.generator.element module**

anchorman.generator.element.**create_element**(*element_pattern*, *item*, *mode*, *markup*)

Create the element that will be inserted in the text.

anchorman.generator.element.**create_element_pattern**(*mode*, *markup*)

Create the basic element pattern based on mode and markup.

**anchorman.generator.highlight module**

anchorman.generator.highlight.**augment_highlight**(*highlight*, *item*)

Fill the base highlight element with data of the item.

anchorman.generator.highlight.**create_highlight**(*highlight_markup*)

Use format to create a base highlight element.

**anchorman.generator.tag module**

anchorman.generator.tag.**augment_bs4tag**(*bs4tag*, *item*, *tag_markup*)
Fill the base bs4tag element with data of the item.

anchorman.generator.tag.**create_bs4tag**(*tag_markup*)
Use BeautifulSoup to create a base tag element.

**anchorman.generator.text module**

anchorman.generator.text.**augment**(*text*, *to_be_applied*)
Augment the text with the elements in to be applied.

#### Module contents

### anchorman.positioner package

#### Submodules

**anchorman.positioner.interval module**

anchorman.positioner.interval.**intervals**(*text*, *elements*, *setting*)
From the slices of elements and units create an intervaltree.

anchorman.positioner.interval.**to_intervaltree**(*data*, *t=None*)
Create an intervaltree of all elements (elements, units, ...).

anchorman.positioner.interval.**unit_intervals**(*intervaltree*, *text_unit*)
Loop the intervaltree to get the text unit interval items.

**anchorman.positioner.slices module**

anchorman.positioner.slices.**element_slices**(*text*, *elements*, *element_identifier*)
Get slices of all elements in text.

anchorman.positioner.slices.**unit_slices**(*text*, *text_unit*)
Get slices of the text units specified in setting.

#### Module contents

## 14.1.2 Submodules

### anchorman.configure module

anchorman.configure.**get_config**(*project_conf=True*)
Load default configuration.

### anchorman.main module

anchorman.main.**annotate**(*text, elements, own_validator=[], config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Find and annotate elements in text.

Create an invaltree with elements and units of text, validate the rules to apply elements and augment the text with this result.

**Parameters**

- **text** (*str*) – The first parameter.
- **elements** (*list*) – It is a list of element dicts like the following: {'fox': {'value': '/wiki/fox', 'data-type': 'animal'}}
- **own_validator** (*list*) – A list of functions that will be applied in the validation of an element, if it will be applied in the text.
- **config** (*dict*) – Load default config from etc/ or get_config the default config andd update to your own rules.

**Returns** **text** – The annotated text.

**Return type** str

#### Examples

Basic example with config overwrite:

```
>>> text = 'The quick brown fox jumps over the lazy dog.'
>>> elements = [
        {'fox': {
            'value': '/wiki/fox', 'data-type': 'animal'}},
        {'dog': {
            'value': '/wiki/dog', 'data-type': 'animal'}}]
>>> cfg = get_config()
>>> cfg['setting']['replaces_at_all'] = 1
>>> print annotate(text, elements, config=cfg)
'The quick brown <a href="/wiki/fox" data-type="animal">fox</a> jumps over the lazy dog .'
```

anchorman.main.**clean**(*text, config={'markup': {'highlight': {'pre': '${{', 'post': '}}'}, 'tag': {'attributes': ['style color:blue;cursor:pointer;', 'class anchorman'], 'value_key': 'href', 'tag': 'a', 'exclude_keys': ['score']}, 'coreferencer': {'attribute': 'token', 'value_key': 'text'}}, 'setting': {'mode': 'tag', 'element_identifier': 'entity', 'text_unit': {'name': 'html-paragraph', 'key': 'p', 'number_of_items': None}, 'longest_match_first': True, 'replaces_at_all': None, 'case_sensitive': True}}*)

Remove elements from text.

Use config to identify elements.

---

**Todo**

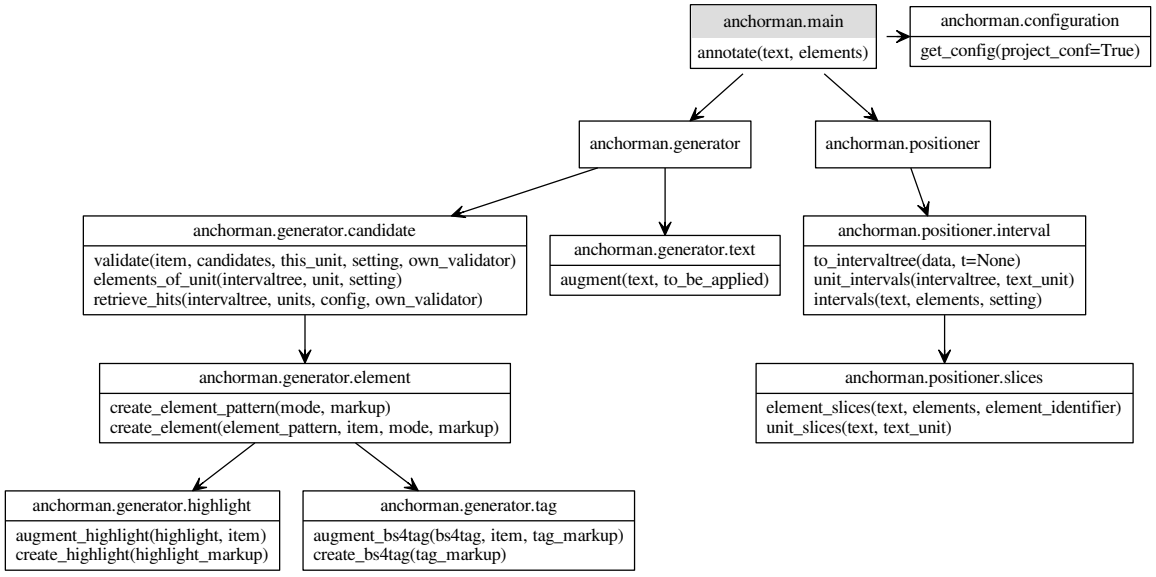
Implement me ...clean(text .

anchorman.utils module

### 14.1.3 Module contents

CHAPTER 15

# Uml view




**Danger:** Beware killer rabbits!

```
************* Module anchorman.configuration
C: 10, 0: Line too long (85/80) (line-too-long)
C: 13, 0: Line too long (86/80) (line-too-long)
************* Module anchorman.main
C: 38, 0: Line too long (112/80) (line-too-long)
W:  8, 0: Dangerous default value [] as argument (dangerous-default-value)
************* Module anchorman.utils
C: 14, 0: Line too long (121/80) (line-too-long)
C: 17, 0: Final newline missing (missing-final-newline)
************* Module anchorman.generator.candidate
C: 97, 0: Line too long (82/80) (line-too-long)
C:  5, 0: Empty function docstring (empty-docstring)
R: 12, 0: Too many local variables (18/15) (too-many-locals)
************* Module anchorman.generator.text
C:  7, 4: Invalid variable name "x" (invalid-name)
************* Module anchorman.positioner.interval
```

```
C:  6, 0: Invalid argument name "t" (invalid-name)
```

CHAPTER 16

# Report

184 statements analysed.

## 16.1 Statistics by type

| type | number | old number | difference | %documented | %badname |
|---|---|---|---|---|---|
| module | 13 | 13 | = | 30.77 | 0.00 |
| class | 0 | 0 | = | 0 | 0 |
| method | 0 | 0 | = | 0 | 0 |
| function | 18 | 17 | +1.00 | 94.44 | 0.00 |

## 16.2 External dependencies

```
anchorman
  \-configuration (anchorman.main)
  \-generator
  | \-candidate (anchorman.main)
  | \-element (anchorman.generator.candidate)
  | \-highlight (anchorman.generator.element)
  | \-tag (anchorman.generator.element)
  | \-text (anchorman.main)
  \-positioner
    \-interval (anchorman.main)
bs4 (anchorman.generator.tag,anchorman.positioner.slices)
intervaltree (anchorman.positioner.interval)
yaml (anchorman.configuration)
```

## 16.3 Raw metrics

| type | number | % | previous | difference |
|---|---|---|---|---|
| code | 200 | 50.38 | 188 | +12.00 |
| docstring | 54 | 13.60 | 57 | -3.00 |
| comment | 41 | 10.33 | 72 | -31.00 |
| empty | 102 | 25.69 | 108 | -6.00 |

## 16.4 Duplication

| | now | previous | difference |
|---|---|---|---|
| nb duplicated lines | 0 | 0 | = |
| percent duplicated lines | 0.000 | 0.000 | = |

## 16.5 Messages by category

| type | number | previous | difference |
|---|---|---|---|
| convention | 9 | 10 | -1.00 |
| refactor | 1 | 1 | = |
| warning | 1 | 1 | = |
| error | 0 | 0 | = |

## 16.6 % errors / warnings by module

| module | error | warning | refactor | convention |
|---|---|---|---|---|
| anchorman.main | 0.00 | 100.00 | 0.00 | 11.11 |
| anchorman.generator.candidate | 0.00 | 0.00 | 100.00 | 22.22 |
| anchorman.utils | 0.00 | 0.00 | 0.00 | 22.22 |
| anchorman.configuration | 0.00 | 0.00 | 0.00 | 22.22 |
| anchorman.positioner.interval | 0.00 | 0.00 | 0.00 | 11.11 |
| anchorman.generator.text | 0.00 | 0.00 | 0.00 | 11.11 |

## 16.7 Messages

| message id | occurrences |
|---|---|
| line-too-long | 5 |
| invalid-name | 2 |
| too-many-locals | 1 |
| missing-final-newline | 1 |
| empty-docstring | 1 |
| dangerous-default-value | 1 |

## 16.8 Global evaluation

Your code has been rated at 9.40/10 (previous run: 9.31/10, +0.09)

CHAPTER 17

# Todo list

**Todo**

check context of replacement: do not add links in links, or inline of overlapping elements, ... replace only one item of an entity > e.g. A. Merkel, Mum Merkel, ...

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/anchorman/checkouts/latest/anchorman/generator/ca
of anchorman.generator.candidate.validate, line 6.)

**Todo**

Implement me ...clean(text .

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/anchorman/checkouts/latest/anchorman/main.py:doc
of anchorman.main.clean, line 5.)

CHAPTER 18

# Credits and contributions

We published this at github and pypi to provide our solution to others, to get feedback and find contributers in the open source.

Thanks Tarn Barford for inspiration and first steps.

CHAPTER 19

# Indices and tables

- *genindex*
- *modindex*
- *search*

## a

anchorman, 32
anchorman.configure, 30
anchorman.generator, 30
anchorman.generator.candidate, 29
anchorman.generator.element, 29
anchorman.generator.highlight, 29
anchorman.generator.tag, 30
anchorman.generator.text, 30
anchorman.main, 31
anchorman.positioner, 30
anchorman.positioner.interval, 30
anchorman.positioner.slices, 30
anchorman.utils, 32

## A

anchorman (module), 4, 32
anchorman.configure (module), 2, 5, 30
anchorman.generator (module), 2, 8, 30
anchorman.generator.candidate (module), 1, 7, 9, 29
anchorman.generator.element (module), 1, 7, 11, 29
anchorman.generator.highlight (module), 1, 7, 13, 29
anchorman.generator.tag (module), 2, 8, 15, 30
anchorman.generator.text (module), 2, 8, 17, 30
anchorman.main (module), 3, 19, 31
anchorman.positioner (module), 2, 21, 30
anchorman.positioner.interval (module), 2, 21, 23, 30
anchorman.positioner.slices (module), 2, 21, 25, 30
anchorman.utils (module), 4, 27, 32
annotate() (in module anchorman.main), 3, 19, 31
augment() (in module anchorman.generator.text), 2, 8, 17, 30
augment_bs4tag() (in module anchorman.generator.tag), 2, 8, 15, 30
augment_highlight() (in module anchorman.generator.highlight), 1, 7, 13, 29

## C

clean() (in module anchorman.main), 3, 19, 31
create_bs4tag() (in module anchorman.generator.tag), 2, 8, 15, 30
create_element() (in module anchorman.generator.element), 1, 7, 11, 29
create_element_pattern() (in module anchorman.generator.element), 1, 7, 11, 29
create_highlight() (in module anchorman.generator.highlight), 1, 7, 13, 29

## D

data_val() (in module anchorman.generator.candidate), 1, 7, 9, 29

## E

element_slices() (in module anchorman.positioner.slices), 2, 21, 25, 30
elements_of_unit() (in module anchorman.generator.candidate), 1, 7, 9, 29

## G

get_config() (in module anchorman.configure), 2, 5, 30

## I

intervals() (in module anchorman.positioner.interval), 2, 21, 23, 30

## R

retrieve_hits() (in module anchorman.generator.candidate), 1, 7, 9, 29

## T

to_intervaltree() (in module anchorman.positioner.interval), 2, 21, 23, 30

## U

unit_intervals() (in module anchorman.positioner.interval), 2, 21, 23, 30
unit_slices() (in module anchorman.positioner.slices), 2, 21, 25, 30

## V

validate() (in module anchorman.generator.candidate), 1, 7, 9, 29