Report generated on 27-May-2017 at 19:16:16 by pytest-html v1.14.2

Environment

JAVA_HOME	/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home
Packages	{'pytest': '3.0.7', 'py': '1.4.33', 'pluggy': '0.4.0'}
Platform	Darwin-16.5.0-x86_64-i386-64bit
Plugins	{'metadata': '1.4.0', 'html': '1.14.2'}
Python	3.6.0

Summary

223 tests ran in 0.95 seconds.

(Un)check the boxes to filter the results.

☑ 207 passed, ☑ 0 skipped, ☑ 16 failed, ☑ 0 errors, ☑ 0 expected failures, ☑ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Failed (hide details)	tests/unit_test_configuration.py::TestWriteConfig::test_load_from_dict	0.00	

```
self = <tests.unit_test_configuration.TestWriteConfig testMethod=test_load_from_dict>

    def setUp(self):
        self.dest_path = os.path.join(make_temp_dir(), "test_config.yaml")

        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:61:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x106334c88>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestWriteConfig::test_setup	0.00	

```
self = <tests.unit_test_configuration.TestWriteConfig testMethod=test_setup>

    def setUp(self):
        self.dest_path = os.path.join(make_temp_dir(), "test_config.yaml")

        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:61:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x1071415c0>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestWriteConfig::test_write_and_verify	0.00	

```
self = <tests.unit_test_configuration.TestWriteConfig testMethod=test_write_and_verify>

    def setUp(self):
        self.dest_path = os.path.join(make_temp_dir(), "test_config.yaml")

        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:61:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10632a3c8>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestWriteConfig::test_write_config	0.00	

```
self = <tests.unit_test_configuration.TestWriteConfig testMethod=test_write_config>

    def setUp(self):
        self.dest_path = os.path.join(make_temp_dir(), "test_config.yaml")

        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:61:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10741dd68>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestDefaultConfig::test_default_configuration_contain_at_least_two_rules	0.00	

```
self = <tests.unit_test_configuration.TestDefaultConfig testMethod=test_default_configuration_contain_at_least_two_rules>

    def setUp(self):
        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:87:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x106808a20>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestDefaultConfig::test_default_configuration_contain_rules	0.00	

```
self = <tests.unit_test_configuration.TestDefaultConfig testMethod=test_default_configuration_contain_rules>

    def setUp(self):
        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:87:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamekw/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x1071464a8>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamekw/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details)	tests/unit_test_configuration.py::TestDefaultConfig::test_default_configuration_exists	0.00	

Result | Test | Duration | Links

```
self = <tests.unit_test_configuration.TestDefaultConfig testMethod=test_default_configuration_exists>

    def setUp(self):
        self.configuration = Configuration()
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:87:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x106808ef0>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestWriteDefaultConfig::test_load_default_config_from_dict_before_write | 0.00

```
self = <tests.unit_test_configuration.TestWriteDefaultConfig testMethod=test_load_default_config_from_dict_before_write>

    def test_load_default_config_from_dict_before_write(self):
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:116:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10681a550>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestWriteDefaultConfig::test_write_default_config_to_disk | 0.00

```
self = <tests.unit_test_configuration.TestWriteDefaultConfig testMethod=test_write_default_config_to_disk>

    def test_write_default_config_to_disk(self):
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:121:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10681e898>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestWriteDefaultConfig::test_write_default_config_to_disk_and_verify | 0.00

```
self = <tests.unit_test_configuration.TestWriteDefaultConfig testMethod=test_write_default_config_to_disk_and_verify>

    def test_write_default_config_to_disk_and_verify(self):
>       self.configuration.load_from_dict(DEFAULT_CONFIG)

tests/unit_test_configuration.py:127:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:163: in load_from_dict
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10635d400>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationInit::test_configuration_parsers_in_not_none | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationInit testMethod=test_configuration_parsers_in_not_none>

    def setUp(self):
        self.maxDiff = None
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:138:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x107146470>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationInit::test_configuration_parsers_instance_of_parser | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationInit testMethod=test_configuration_parsers_instance_of_parser>

    def setUp(self):
        self.maxDiff = None
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:138:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10652d530>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationInit::test_configuration_parsers_subclass_of_parser | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationInit testMethod=test_configuration_parsers_subclass_of_parser>

    def setUp(self):
        self.maxDiff = None
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:138:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10635bf60>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with *validated* data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationDataAccess::test_get_data_does_not_return_none | 0.00

Result | Test | Duration | Links

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationDataAccess::test_get_data_does_not_return_none | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationDataAccess testMethod=test_get_data_does_not_return_none>

    def setUp(self):
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:157:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10637a048>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with "validated" data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationDataAccess::test_get_file_rules_does_not_return_none | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationDataAccess testMethod=test_get_file_rules_does_not_return_none>

    def setUp(self):
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:157:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10636bf98>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with "validated" data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Failed (hide details) | tests/unit_test_configuration.py::TestConfigurationDataAccess::test_get_file_rules_returns_expected | 0.00

```
self = <tests.unit_test_configuration.TestConfigurationDataAccess testMethod=test_get_file_rules_returns_expected>

    def setUp(self):
>       self.configuration = Configuration(DEFAULT_CONFIG)

tests/unit_test_configuration.py:157:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
autonamenow/core/config/configuration.py:93: in __init__
    self._load_file_rules()
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

self = <core.config.configuration.Configuration object at 0x10635da40>

    def _load_file_rules(self):
        # Check raw dictionary data.
        # Create and populate "FileRule" objects with "validated" data.
        for fr in self._data['file_rules']:

            # Prioritize 'name_format', the "raw" name format string.
            # If it is not defined in the rule, check that 'name_template'
            # refers to a valid entry in 'name_templates'.
            _valid_template = None
            if 'name_format' in fr:
                _valid_template = self.validate(fr, 'name_format')
            elif 'name_template' in fr:
                _template_name = fr.get('name_template')
                if _template_name in self.name_templates:
                    _valid_template = self.name_templates.get(_template_name)
            else:
                # TODO: Handle case where all name format fields are missing.
                pass

            if not _valid_template:
>               raise ConfigurationSyntaxError('Invalid name format')
E               core.exceptions.ConfigurationSyntaxError: Invalid name format

autonamenow/core/config/configuration.py:128: ConfigurationSyntaxError
```

Skipped (hide details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_analyzer_classes_returns_expected_contents | 0.00

```
('tests/unit_test_analysis.py', 90, 'Skipped: TODO: Implement ..')
```

Skipped (hide details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_instantiated_analyzers | 0.00

```
('tests/unit_test_analysis.py', 94, 'Skipped: Fix/skip this test ..')
```

Skipped (hide details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test__get_datetime_from_filesystem | 0.00

```
('tests/unit_test_analyze_filesystem.py', 108, 'Skipped: Create and access time might have changed. Should be set to a known state as part of the tests setup.')
```

Skipped (hide details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_filesystem_accessed_is_valid | 0.00

```
('tests/unit_test_analyze_filesystem.py', 90, 'Skipped: Access time might have changed between test runs.')
```

Skipped (hide details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_filesystem_created_is_valid | 0.00

```
('tests/unit_test_analyze_filesystem.py', 75, 'Skipped: Create time might have changed. Should be set to a known time as part of the tests setup.')
```

Passed (show details) | tests/unit_test_analysis.py::TestTests::test_constant_expect_analyzer_classes | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestTests::test_constant_expect_analyzer_classes_basename | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_analysis_mime_mappings | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_analyzer_classes_basename_returns_expected_contents | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_analyzer_classes_basename_returns_expected_count | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestAnalysisUtilityFunctions::test_get_analyzer_classes_returns_expected_count | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_add | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_add_with_invalid_analyzer_raises_exception | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_add_with_invalid_field_raises_exception | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_results_init_contains_all_valid_results_fields | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_results_init_contains_valid_results_fields | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_results_init_results_fields_are_empty | 0.00
Passed (show details) | tests/unit_test_analysis.py::TestResults::test_results_init_results_fields_are_type_dict | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_datetime_contains_special_case | 0.02
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_datetime_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_datetime_special_case_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_tags_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_tags_returns_empty_list | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_get_title_returns_empty_list | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithImageFile::test_setup | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_datetime_does_not_return_none | 0.01
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_datetime_returns_empty_list | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_tags_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_tags_returns_empty_list | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_title_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_get_title_return_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithEmptyFile::test_setup | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_datetime_contains_special_case | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_datetime_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_datetime_special_case_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_tags_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_tags_return_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_title_contains_filename | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_title_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_get_title_return_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filename.py::TestFilenameAnalyzerWithTaggedFile::test_setup | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_author_should_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_contains_filesystem_accessed | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_contains_filesystem_created | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_contains_filesystem_modified | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_does_not_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_datetime_filesystem_modified_is_valid | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_tags_returns_none | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_get_title_should_return_none | 0.00
Passed (show details) | tests/unit_test_analyze_filesystem.py::TestFilesystemAnalyzerWithEmptyFile::test_setup | 0.00
Passed (show details) | tests/unit_test_autonamenow.py::TestAutonamenow::test_setup | 0.00
Passed (show details) | tests/unit_test_configuration.py::TestWriteDefaultConfig::test_setup | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestDateAndTime::test_hyphenate_date | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchUnixTimestamp::test_match_unix_timestamp | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchUnixTimestamp::test_match_unix_timestamp_with_invalid_argument | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchSpecialCase::test_match_special_case_1st_variation | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchSpecialCase::test_match_special_case_2nd_variation | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchSpecialCase::test_match_special_case_3rd_variation | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchSpecialCase::test_match_special_case_4th_variation | 0.00
Passed (show details) | tests/unit_test_dateandtime.py::TestMatchSpecialCase::test_match_special_case_with_invalid_argument | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestMimeTypes::test_magic_type_lookup_arbitrarily_assert_image_is_jpeg | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestMimeTypes::test_magic_type_lookup_table_exists | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestMimeTypes::test_magic_type_lookup_table_is_instance_of_dict | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestMimeTypes::test_magic_type_lookup_table_values_are_type_list | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestSplitFileName::test_split_filename_has_multiple_extensions | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestSplitFileName::test_split_filename_has_no_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestSplitFileName::test_split_filename_has_one_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestSplitFileName::test_split_filename_no_name | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_file_has_many_suffixes | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_file_has_no_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_file_has_one_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_from_hidden_file | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_from_hidden_file_many_suffixes | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileSuffix::test_file_suffix_no_name | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_file_has_many_suffixes | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_file_has_no_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_file_has_one_extension | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_from_hidden_file | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_from_hidden_file_many_suffixes | 0.00
Passed (show details) | tests/unit_test_diskutils.py::TestFileBase::test_file_base_no_name | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameNoTags::test__filenamepart_base | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameNoTags::test__filenamepart_ext | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameNoTags::test__filenamepart_tags | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameNoTags::test__filenamepart_ts | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameNoTags::test_setUp | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTags::test__filenamepart_base | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTags::test__filenamepart_ext | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTags::test__filenamepart_tags | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTags::test__filenamepart_ts | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTags::test_setUp | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTagsDashesInName::test__filenamepart_base | 0.00
Passed (show details) | tests/unit_test_filedobject.py::TestFileObjectFilenamePartitioningLongNameWithTagsDashesInName::test__filenamepart_ext | 0.00

Result	Test	Duration	Links
Passed *(show details)*	tests.unit_test_filedbject.py::TestFileObjectFilenamePartitioningLongNameWithTagsDashesInName::test__filenamepart_tags	0.00	
Passed *(show details)*	tests.unit_test_filedbject.py::TestFileObjectFilenamePartitioningLongNameWithTagsDashesInName::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningLongNameWithTagsDashesInName::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoExtensionNoTags::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoExtensionNoTags::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoExtensionNoTags::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoExtensionNoTags::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoExtensionNoTags::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoTags::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoTags::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoTags::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoTags::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileNoTags::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffix::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffix::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffix::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffix::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffix::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileWithTags::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileWithTags::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileWithTags::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileWithTags::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileWithTags::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffixTags::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffixTags::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffixTags::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffixTags::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningHiddenFileCompoundSuffixTags::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningDifficultCombination::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningDifficultCombination::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningDifficultCombination::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningDifficultCombination::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningDifficultCombination::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningAnotherDifficultCombination::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningAnotherDifficultCombination::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningAnotherDifficultCombination::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningAnotherDifficultCombination::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningAnotherDifficultCombination::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningEvenMoreDifficultCombination::test__filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningEvenMoreDifficultCombination::test__filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningEvenMoreDifficultCombination::test__filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningEvenMoreDifficultCombination::test__filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningEvenMoreDifficultCombination::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueType::test__filenamepart_base_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueType::test__filenamepart_ext_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueType::test__filenamepart_tags_returns_list_of_string_or_empty_list	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueType::test__filenamepart_ts_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueType::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTags::test__filenamepart_base_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTags::test__filenamepart_ext_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTags::test__filenamepart_tags_returns_list_of_string_or_empty_list	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTags::test__filenamepart_ts_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTags::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTagsNoExt::test__filenamepart_base_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTagsNoExt::test__filenamepart_ext_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTagsNoExt::test__filenamepart_tags_returns_list_of_string_or_empty_list	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTagsNoExt::test__filenamepart_ts_returns_string_or_none	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningReturnValueTypeNoTagsNoExt::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename::test_filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename::test_filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename::test_filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename::test_filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename2::test_filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename2::test_filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename2::test_filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename2::test_filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename2::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename3::test_filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename3::test_filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename3::test_filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename3::test_filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename3::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename4::test_filenamepart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename4::test_filenamepart_ext	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename4::test_filenamepart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename4::test_filenamepart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenamePartitioningWithActualFilename4::test_setUp	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameIsInFiletagsFormat::test_has_filetags_format	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameIsInFiletagsFormat::test_has_filetags_format_date_only	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameIsInFiletagsFormat::test_has_filetags_format_no_extension	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameNotInFiletagsFormat::test_doesnt_have_filetags_format_missing_fnpart_base	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameNotInFiletagsFormat::test_doesnt_have_filetags_format_missing_fnpart_tags	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileObjectFilenameNotInFiletagsFormat::test_doesnt_have_filetags_format_missing_fnpart_ts	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileTypeMagic::test_filetype_magic	0.05	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileTypeMagic::test_filetype_magic_with_invalid_args	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileTypeMagic::test_test_files_are_readable	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileTypeMagic::test_test_files_defined	0.00	
Passed *(show details)*	tests/unit_test_filedbject.py::TestFileTypeMagic::test_test_files_exist	0.00	
Passed *(show details)*	tests/unit_test_filerule.py::TestFileRuleInstantiation::test_init_description_is_str_or_none	0.00	
Passed *(show details)*	tests/unit_test_filerule.py::TestFileRuleInstantiation::test_init_exact_match_is_boolean	0.00	
Passed *(show details)*	tests/unit_test_misc_utils.py::TestUniqueIdentifier::test_unique_identifier_has_prefix_uuid	0.00	
Passed *(show details)*	tests/unit_test_misc_utils.py::TestUniqueIdentifier::test_unique_identifier_returns_1000_unique_values	0.00	
Passed *(show details)*	tests/unit_test_misc_utils.py::TestUniqueIdentifier::test_unique_identifier_returns_100_unique_values	0.00	
Passed *(show details)*	tests/unit_test_misc_utils.py::TestUniqueIdentifier::test_unique_identifier_returns_not_none	0.00	
Passed *(show details)*	tests/unit_test_misc_utils.py::TestUniqueIdentifier::test_unique_identifier_returns_string	0.00	
Passed *(show details)*	tests/unit_test_namebuilder.py::TestNameBuilder::test_assemble_basename_using_template_1_given_all_fields	0.00	
Passed *(show details)*	tests/unit_test_namebuilder.py::TestNameBuilder::test_assemble_basename_using_template_1_some_fields_missing	0.00	
Passed *(show details)*	tests/unit_test_namebuilder.py::TestNameBuilder::test_assemble_basename_using_template_2_all_fields_missing	0.00	
Passed *(show details)*	tests/unit_test_namebuilder.py::TestNameBuilder::test_assemble_basename_using_template_2_given_all_fields	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_readable_file_raises_exception_arg_is_dev_null	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_readable_file_raises_exception_arg_is_directory	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_readable_file_raises_exception_empty_string_arg	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_readable_file_raises_exception_missing_file	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_readable_file_raises_exception_none_argument	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_year_passes_valid_years	0.00	
Passed *(show details)*	tests/unit_test_options.py::TestArgumentValidators::test_arg_is_year_raises_exception	0.00	
Passed *(show details)*	tests/unit_test_rulematcher.py::TestRuleMatcher::test_setup	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserFunctions::test_get_available_parsers	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserFunctions::test_get_available_parsers_returns_list_of_strings	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserFunctions::test_get_instantiated_parsers_returns_arbitrary_number	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserFunctions::test_get_instantiated_parsers_returns_class_objects	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserFunctions::test_get_instantiated_parsers_returns_list	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParser::test_get_validation_function_should_raise_error_if_unimplemented	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserSubclasses::test_get_validation_function_should_return_function	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserSubclasses::test_setup	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRuleParserSubclasses::test_validation_function_should_return_booleans	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRegexRuleParser::test_get_validation_function_should_not_return_none	0.00	
Passed *(show details)*	tests/unit_test_ruleparsers.py::TestRegexRuleParser::test_get_validation_function_should_return_function	0.00	
Passed *(show details)*	tests/unit_test_unit_utils.py::TestUtilities::test_make_temp_dir	0.00	