

The RDKit Cookbook

rdkit.org/docs/Cookbook.html

2020 RDKit UGM Virtual Poster

Vincent F. Scalfani | contact: vfscalfani@ua.edu

Rodgers Library for Science and Engineering, The University of Alabama, Tuscaloosa, AL 35487, USA.

What is the RDKit Cookbook?

The [Cookbook](#) is an official section of the RDKit Documentation and provides example code snippets, or “recipes”, for how to carry out particular tasks using the RDKit functionality from Python. Entries from the Cookbook are a compilation of user contributed RDKit code from various sources including:

- [Sourceforge mailing list](#)
- Blogs
- GitHub Gists and Issues
- Direct contributions

The scope of the RDKit Cookbook is to capture useful short recipes (about 5-50 lines of code), test the code, then format the code into the Cookbook.

Currently, you can expect to find recipes such as drawing molecules with specific properties, identifying bond types, rings, substructure matching, descriptor calculations, and more.

We assign appropriate credit and links to the original source.

Isomeric SMILES without isotopes
Author: Andrew Dalke
Source: <https://sourceforge.net/projects/rdkit/mailman/message/36877847/>
Index ID: RDKitCB_5
Summary: Write Isomeric SMILES without isotope information (i.e. only stereochemistry)

```
from rdkit.Chem import Chem
def MolWithoutIsotopesToSmiles(mol):
    atom_data = [(atom, atom.GetIsotope()) for atom in mol.GetAtoms()]
    for atom, isotope in atom_data:
        # restore original isotope values
        if isotope:
            atom.SetIsotope(0)
    smiles = Chem.MolToSmiles(mol)
    for atom, isotope in atom_data:
        if isotope:
            atom.SetIsotope(isotope)
    return smiles

mol = Chem.MolFromSmiles('[19F][13C@H]([16O])[135Cl]')
print(MolWithoutIsotopesToSmiles(mol))
```

O[C@@H](F)Cl

N.B. There are two limitations noted with this Isomeric SMILES without isotopes method including with isotopic hydrogens, and a requirement to recalculate stereochemistry. See the source discussion linked above for further explanation and examples.

RDKit Cookbook Version 1.0 and Challenges

The Cookbook started in 2012 and prior to 2020, the [Cookbook markdown file](#) code snippets were not tested. As a result, some of the recipes in the Cookbook were no longer functional or relevant as RDKit evolved since 2012.

One nice feature of using a basic markdown file is that it is easy to contribute, as minimal markup formatting is needed. For example, most contributions would need to use only the `#` symbols for a heading and `'''` for code fence blocks:

```
# Aromaticity Heading
'''
from rdkit.Chem import Chem
m = Chem.MolFromSmiles('N1C=Cc2ccccc12')
m.GetSubstructMatches(Chem.MolFromSmarts('c'))
'''
```

Details of the Current RDKit Cookbook Version 2.0

Earlier this year, the Cookbook was rewritten in Python reStructuredText (.rst) and uses [Sphinx](#) to build the documentation and test the code, similarly to other RDKit documentation. Specifically, the Sphinx doctest extension is used with the `testcode::` and `testoutput::` directives to execute the code snippets and check the output. Here is the main workflow:

1

We first have to format the code in reStructuredText and the Sphinx doctest directives.

It is a bit of work, but worth the effort to maintain working code as RDKit evolves.

2

Next, run the Sphinx doctest builder.

```
(my-rdkit-env) user:~/rdkit/Docs/Book$ make doctest

Results of doctest builder
=====
Document: Cookbook
-----
1 items passed all tests:
149 tests in default
149 tests in 1 items.
149 passed and 0 failed.
Test passed.
1 items passed all tests:
1 tests in default (cleanup code)
1 tests in 1 items.
1 passed and 0 failed.
Test passed.
```

If a test fails, the doctest builder has helpful output:

```
File "Cookbook.rst", line 633, in default
Failed example:
'''
Expected:
[[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12],
[14, 15, 16, 17, 18, 19]]
Got:
None
```

```
--/--RDKitCB_3 example (abridged)

.. testcode::

def GetRingSystems(mol, includeSpiro=False):
    ri = mol.GetRingInfo()
    systems = []
    for ring in ri.AtomRings():
        ringAts = set(ring)
        nSystems = []
        for system in systems:
            nInCommon = len(ringAts.intersection(system))
            if nInCommon and (includeSpiro or nInCommon>1):
                ringAts = ringAts.union(system)
            else:
                nSystems.append(system)
        nSystems.append(ringAts)
    systems = nSystems
    return systems
mol = Chem.MolFromSmiles('CN1C(=O)CN=C(C2=C1C=CC(=C2)C1)C3=CC=CC=C3')
print(GetRingSystems(mol))

.. testoutput::

[[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12], [14, 15, 16, 17, 18, 19]]

.. testcode::

# Draw molecule with atom index (see RDKitCB_0)
def mol_with_atom_index(mol):
    for atom in mol.GetAtoms():
        atom.SetAtomMapNum(atom.GetIdx())
    return mol
mol_with_atom_index(mol)

.. image:: images/RDKitCB_3_im0.png

--/--
```

3

Build the documents.

```
(my-rdkit-env) user:~/rdkit/Docs/Book$ make html
```

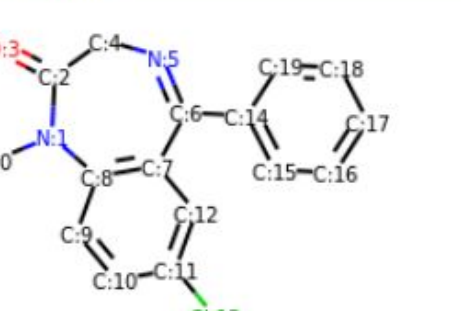
Rings, Aromaticity, and Kekulization
Count Ring Systems
Author: Greg Landrum
Source: <https://gist.github.com/greglandrum/1751a42b3cae54011041d867ae7415>
Index ID: RDKitCB_3
Summary: Count ring systems in a molecule

```
from rdkit.Chem import Chem
from rdkit.Chem.Draw import IPythonConsole

def GetRingSystems(mol, includeSpiro=False):
    ri = mol.GetRingInfo()
    systems = []
    for ring in ri.AtomRings():
        ringAts = set(ring)
        nSystems = []
        for system in systems:
            nInCommon = len(ringAts.intersection(system))
            if nInCommon and (includeSpiro or nInCommon>1):
                ringAts = ringAts.union(system)
            else:
                nSystems.append(system)
        nSystems.append(ringAts)
    systems = nSystems
    return systems
mol = Chem.MolFromSmiles('CN1C(=O)CN=C(C2=C1C=CC(=C2)C1)C3=CC=CC=C3')
print(GetRingSystems(mol))

[[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12], [14, 15, 16, 17, 18, 19]]

# Draw molecule with atom index (see RDKitCB_0)
def mol_with_atom_index(mol):
    for atom in mol.GetAtoms():
        atom.SetAtomMapNum(atom.GetIdx())
    return mol
mol_with_atom_index(mol)
```



Jupyter Book...RDKit Cookbook Version 3.0?

In August, a major update to the [Jupyter Book project](#) was released. The Jupyter Book project allows for compilation of Jupyter Notebooks and Markdown files into professionally formatted online and PDF books. My early experiments with the new Jupyter Book suggest that it may be an ideal platform for a future version of the RDKit Cookbook.

1

Contributions can be submitted as Jupyter Notebooks without any special formatting or separate image files. Markup is needed only for title, author, and source link, so contributing has a low learning curve.

2

We should be able to test the notebooks with [nbval](#) and [pytest](#). The nbval plugin compares saved outputs to the newly executed output.

3

Each Jupyter Notebook recipe is then added to the Cookbook as a separate file in the book toc.yml file.

4

Build the documents.

```
(my-rdkit-env) user:~/rdkit/Docs/Book$ jupyter-book build cookbook/
```

```
- file: RDKitCookbook_intro
- file: drawing_molecules_overview
sections:
- file: draw_atom_index
- file: draw_calc
- file: draw_bw
- file: draw_highlight
- file: draw_noHs
- file: Rings_Aromaticity_Kekulization
sections:
- file: count_ring_systems
- file: identify_aromatic_rings
- file: identify_aromatic_atoms
...
```

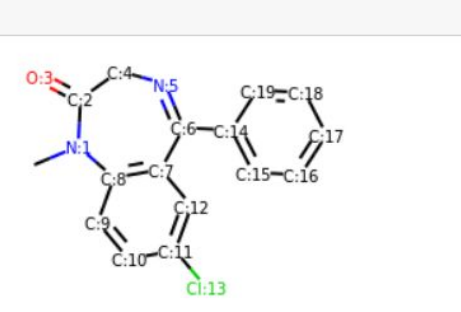
Count Ring Systems
Author: Greg Landrum
Source: <https://gist.github.com/greglandrum/1751a42b3cae54011041d867ae7415>
Index ID: RDKitCB_3
Summary: Count ring systems in a molecule

```
from rdkit.Chem import Chem
from rdkit.Chem.Draw import IPythonConsole

def GetRingSystems(mol, includeSpiro=False):
    ri = mol.GetRingInfo()
    systems = []
    for ring in ri.AtomRings():
        ringAts = set(ring)
        nSystems = []
        for system in systems:
            nInCommon = len(ringAts.intersection(system))
            if nInCommon and (includeSpiro or nInCommon>1):
                ringAts = ringAts.union(system)
            else:
                nSystems.append(system)
        nSystems.append(ringAts)
    systems = nSystems
    return systems
mol = Chem.MolFromSmiles('CN1C(=O)CN=C(C2=C1C=CC(=C2)C1)C3=CC=CC=C3')
print(GetRingSystems(mol))

[[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12], [14, 15, 16, 17, 18, 19]]

# Draw molecule with atom index (see RDKitCB_0)
def mol_with_atom_index(mol):
    for atom in mol.GetAtoms():
        atom.SetAtomMapNum(atom.GetIdx())
    return mol
mol_with_atom_index(mol)
```



Jupyter Book pages can be setup interactively (e.g., with Google Colab).

Many more possibilities:

1. Add a DOI to contributions?

2. RDKit Cookbook Journal? Short, tested, and peer-reviewed recipes.

Contributing and Feedback

Any feedback is appreciated. If you are interested in contributing, you can edit the cookbook.rst file. I am happy to help you. **Please also feel free to send code to me for addition into the Cookbook.** Depending on feedback received, I am interested in transitioning the Cookbook to Jupyter Book.

Acknowledgments

Thanks to Greg Landrum for helping me with the new Cookbook workflow and the contributors that helped with reviewing code from the Cookbook. My sincere thanks to the RDKit Community for your open sharing of code and discussions; this makes it easy to compile great code recipes for the Cookbook. Lastly, I thank the University of Alabama Libraries Administration for their support of open source and my time with contributing to RDKit.