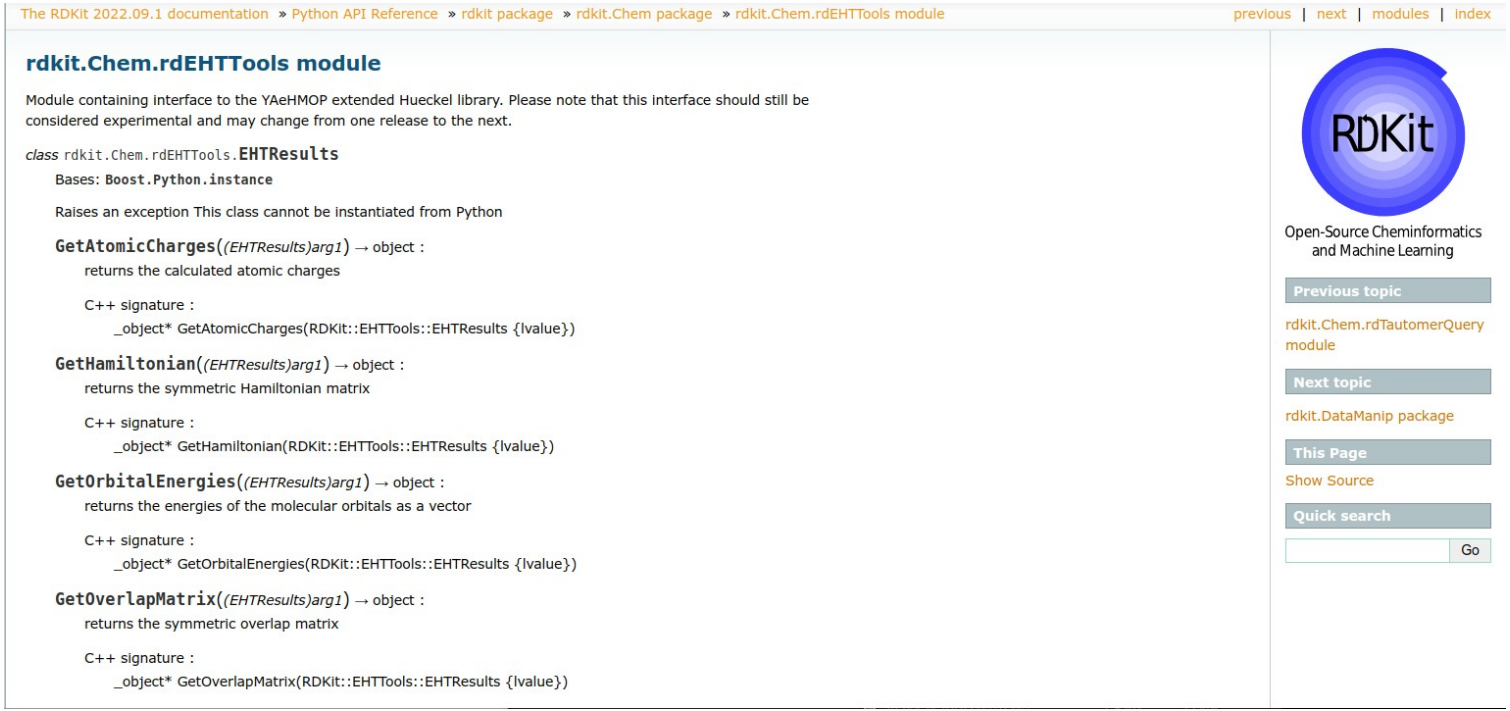
# Let’s QM with OSS

20230218 Mishima.syk
pen(Taka)@iwatobipen

# Python で QM をしたいときのパッケージ

- RDKit (!)
- Psi4
- Pyscf
- Fanpy （使ったことない）
- pygamess ( 別途 GAMESS インストール必要 )
- psikit ( 別途 psi4 インストール必要 )

# RDKit は拡張 Hückel 法を実装している（今回は話さない）



The RDKit 2022.09.1 documentation » Python API Reference » rdkit package » rdkit.Chem package » rdkit.Chem.rdEHTTools module

previous | next | modules | index

## rdkit.Chem.rdEHTTools module

Module containing interface to the YAeHMOP extended Hueckel library. Please note that this interface should still be considered experimental and may change from one release to the next.

*class* rdkit.Chem.rdEHTTools.**EHTResults**

Bases: **Boost.Python.instance**

Raises an exception This class cannot be instantiated from Python

**GetAtomicCharges**(*(EHTResults)arg1*) → object :
returns the calculated atomic charges

C++ signature :
_object* GetAtomicCharges(RDKit::EHTTools::EHTResults {lvalue})

**GetHamiltonian**(*(EHTResults)arg1*) → object :
returns the symmetric Hamiltonian matrix

C++ signature :
_object* GetHamiltonian(RDKit::EHTTools::EHTResults {lvalue})

**GetOrbitalEnergies**(*(EHTResults)arg1*) → object :
returns the energies of the molecular orbitals as a vector

C++ signature :
_object* GetOrbitalEnergies(RDKit::EHTTools::EHTResults {lvalue})

**GetOverlapMatrix**(*(EHTResults)arg1*) → object :
returns the symmetric overlap matrix

C++ signature :
_object* GetOverlapMatrix(RDKit::EHTTools::EHTResults {lvalue})

RDKit
Open-Source Cheminformatics and Machine Learning

Previous topic
rdkit.Chem.rdTautomerQuery module

Next topic
rdkit.DataManip package

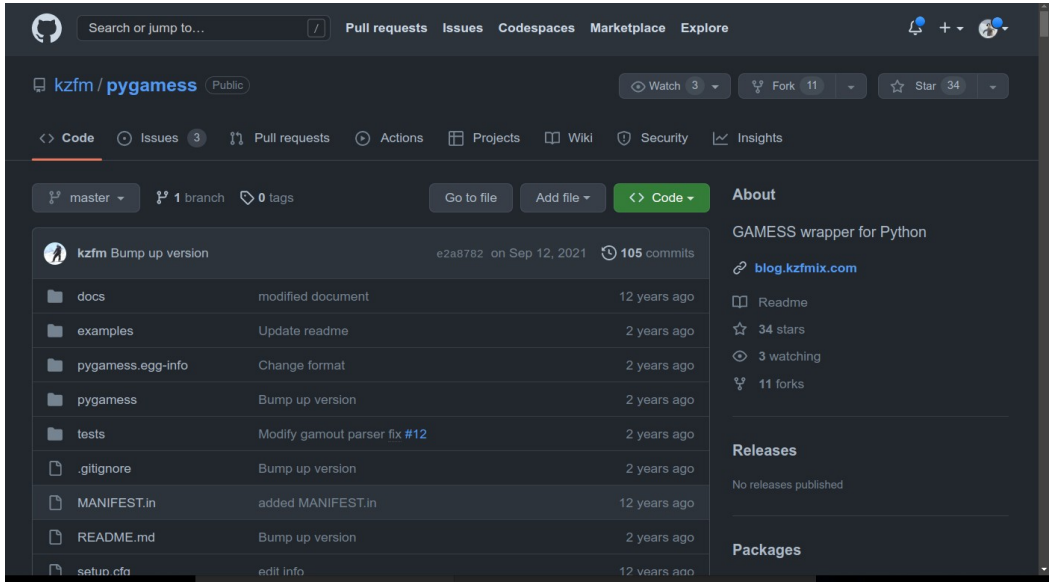
This Page
Show Source

Quick search
Go

https://www.rdkit.org/docs/source/rdkit.Chem.rdEHTTools.html
http://rdkit.blogspot.com/2019/06/doing-extended-hueckel-calculations.html

# Masterpiece of QM with RDKit -1

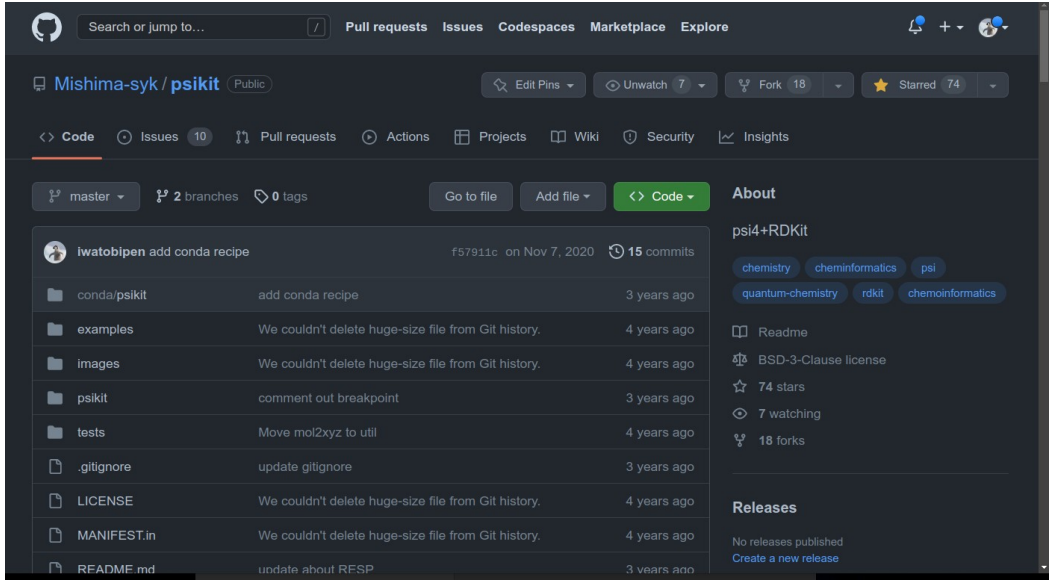
https://github.com/kzfm/pygamess




GAMESS と RDKit でいい感じに QM できます。

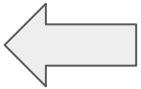
# Masterpiece of QM with RDKit -2

https://github.com/Mishima-syk/psikit

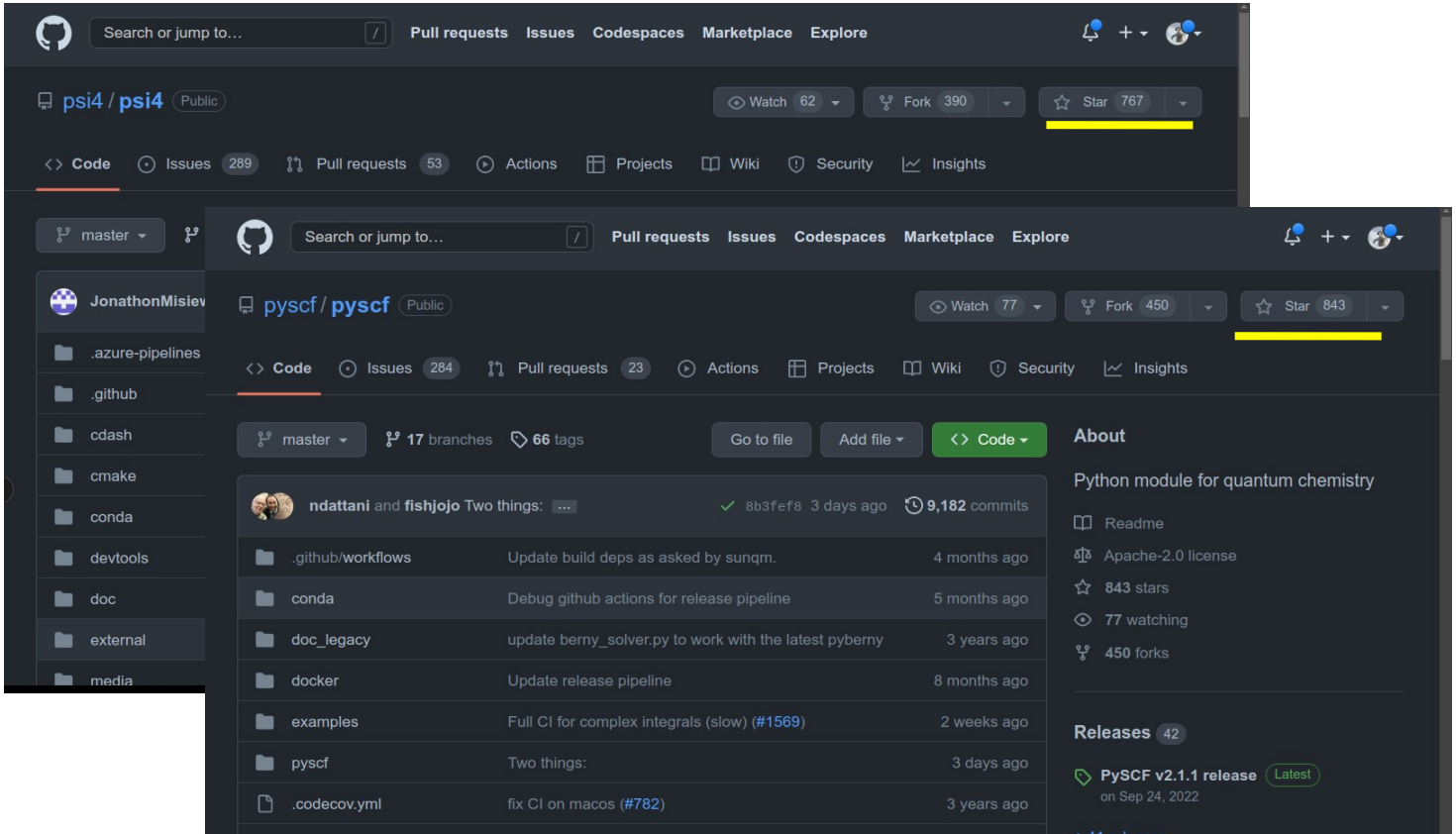



Psi4 と RDKit でいい感じに QM できます。

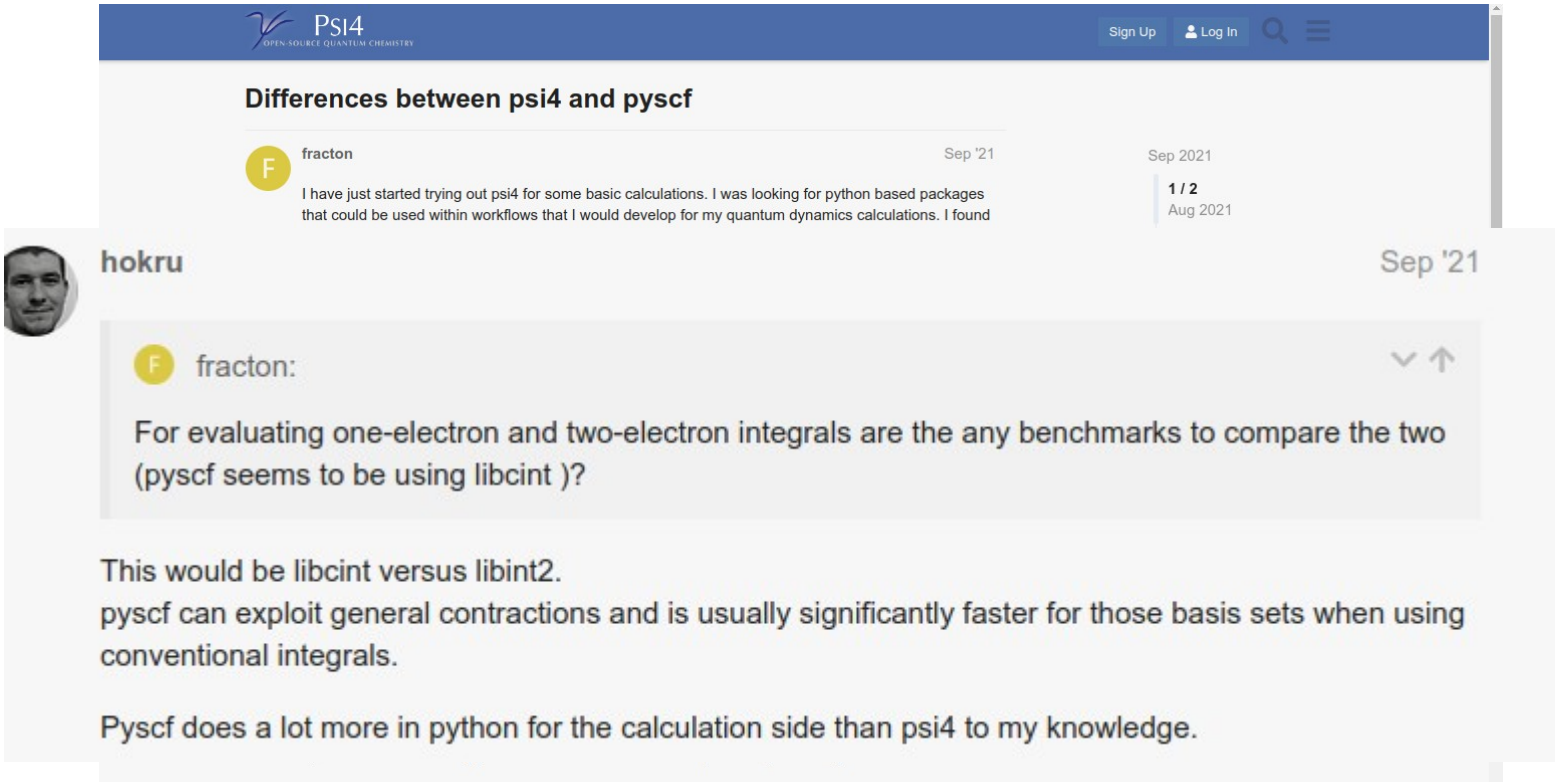
# Python で QM をしたいときのパッケージ

- Psi4
- Pyscf 
- Fanpy （使ったことない）
- pygamess ( 別途 GAMESS インストール必要 )
- psikit ( 別途 psi4 インストール必要 )

# psi4 と pyscf　☆の数は同じくらい



# From psi4 user forum



Psi4
OPEN-SOURCE QUANTUM CHEMISTRY

Sign Up  Log In

**Differences between psi4 and pyscf**

**fracton** Sep '21

I have just started trying out psi4 for some basic calculations. I was looking for python based packages that could be used within workflows that I would develop for my quantum dynamics calculations. I found

Sep 2021

1 / 2

Aug 2021

**hokru** Sep '21

> fracton:
>
> For evaluating one-electron and two-electron integrals are the any benchmarks to compare the two (pyscf seems to be using libcint )?

This would be libcint versus libint2.
pyscf can exploit general contractions and is usually significantly faster for those basis sets when using conventional integrals.

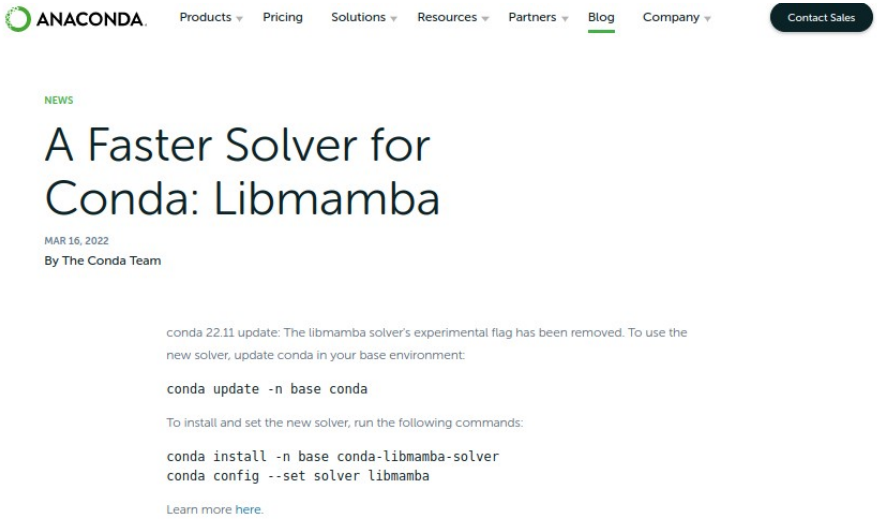
Pyscf does a lot more in python for the calculation side than psi4 to my knowledge.

# Install

$ pip install pyscf

$ conda install -c pyscf pyscf

横道にそれますが
<condaの依存関係解決を高速化したい場合>




The conda team is pleased to announce the availability of 'libmamba' as a new, much faster dependency solver for conda! Three different companies worked to make this release possible:

https://www.anaconda.com/blog/a-faster-conda-for-a-growing-community

# では実際にコード書いてみましょ〜

# 必要なパッケージインポート

In [1]:
```
from rdkit import Chem
from rdkit.Chem import Draw
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import AllChem, rdCoordGen
import py3Dmol
```

In [2]:
```
from pyscf import gto, scf, lo, tools
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
%matplotlib inline
sns.set_theme(style="ticks", context="talk", palette="muted")
```

In [4]:
```
import numpy as np
import pandas as pd
pd.options.display.float_format = "{:,.3f}".format
```

# gto.Mole, gto.M の違い

## Initializing a molecule

There are three ways to define and initialize a molecule. The first is to use the keyword arguments of the `Mole.build()` method to initialize a molecule:

```
>>> from pyscf import gto
>>> mol = gto.Mole()
>>> mol.build(
...     atom = '''O 0 0 0; H  0 1 0; H 0 0 1''',
...     basis = 'sto-3g')
```

The second way is to assign the geometry, basis etc., to the `Mole` object, followed by calling the `build()` method:

```
>>> from pyscf import gto
>>> mol = gto.Mole()
>>> mol.atom = '''O 0 0 0; H  0 1 0; H 0 0 1'''
>>> mol.basis = 'sto-3g'
>>> mol.build()
```

The third way is to use the shortcut functions `pyscf.M()` or `Mole.M()`. These functions pass all the arguments to the `build()` method:

```
>>> import pyscf
>>> mol = pyscf.M(
...     atom = '''O 0 0 0; H  0 1 0; H 0 0 1''',
...     basis = 'sto-3g')

>>> from pyscf import gto
>>> mol = gto.M(
...     atom = '''O 0 0 0; H  0 1 0; H 0 0 1''',
...     basis = 'sto-3g')
```

In any of these, you may have noticed two keywords `atom` and `basis`. They are used to hold the molecular geometry and basis sets, which can be defined along with other input options as follows.

https://pyscf.org/user/gto.html

gto.Mole()　より gtoM() のほうがコードが少なくなる。

## pyscf.gto.mole module

Mole class and helper functions to handle paramters and attributes for GTO integrals. This module serves the interface to the integral library libcint.

pyscf.gto.mole.M(***kwargs*)[source]

This is a shortcut to build up Mole object.

Args: Same to `Mole.build()`

Examples:

```
>>> from pyscf import gto
>>> mol = gto.M(atom='H 0 0 0; F 0 0 1', basis='6-31g')
```

*class* pyscf.gto.mole.Mole(***kwargs*)[source]

Bases: `StreamObject`

Basic class to hold molecular structure and global options

Attributes:

verbose : int

Print level

output : str or None

# MolToXYZBlock で座標情報取得

```
def smi2xyz(smi, optimize=False):
    mol = Chem.MolFromSmiles(smi)
    mol = Chem.AddHs(mol)
    AllChem.EmbedMolecule(mol)
    if optimize:
        AllChem.MMFFOptimizeMolecule(mol)
    xyz = Chem.MolToXYZBlock(mol)[3:]
    return mol, xyz
```

# https://www.rdkit.org/docs/source/rdkit.Chem.rdmolfiles.html

rdkit.Chem.rdmolfiles.**MolToXYZBlock**(*(Mol)mol*[, *(int)confId=-1*]) → str :

Returns a XYZ block for a molecule

ARGUMENTS:

- mol: the molecule
- confId: (optional) selects which conformation to output (-1 = default)

RETURNS:

a string

C++ signature :

std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
MolToXYZBlock(RDKit::ROMol [,int=-1])

rdkit.Chem.rdmolfiles.**MolToXYZFile**(*(Mol)mol*, *(str)filename*[, *(int)confId=-1*]) → None :

Writes a XYZ file for a molecule

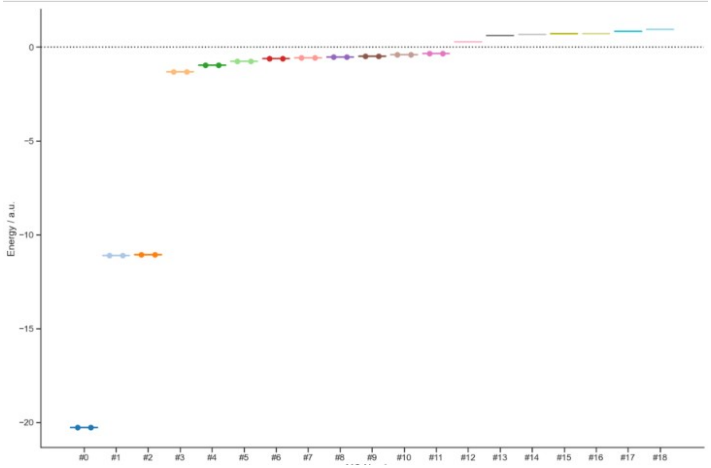
ARGUMENTS:

- mol: the molecule
- filename: the file to write to
- confId: (optional) selects which conformation to output (-1 = default)

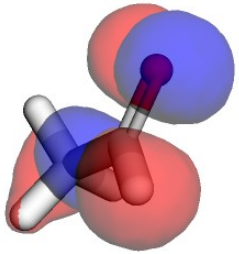
# pyscf での計算実行パート

```
def run_qm(xyz, basis='sto-3g'):
    scfmol = gto.Mole()
    scfmol.atom = xyz
    scfmol.basis =basis
    scfmol.unit = 'ANG'
    scfmol.build()
    mf = scf.RHF(scfmol).run()
```

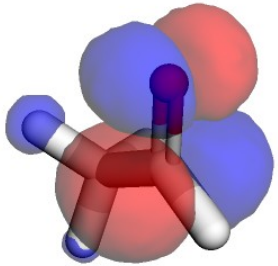
# Example




```
v = py3Dmol.view()
v.addVolumetricData(homo_voldata, "cube", {'isoval': -0.03, 'color': "red", 'opacity': 0.75})
v.addVolumetricData(homo_voldata, "cube", {'isoval': 0.03, 'color': "blue", 'opacity': 0.75})
v.addModel(Chem.MolToMolBlock(mol), 'mol')
v.setStyle({'stick':{}})
v.zoomTo()
v.show()
```



```
v = py3Dmol.view()
v.addVolumetricData(lumo_voldata, "cube", {'isoval': -0.03, 'color': "red", 'opacity': 0.75})
v.addVolumetricData(lumo_voldata, "cube", {'isoval': 0.03, 'color': "blue", 'opacity': 0.75})
v.addModel(Chem.MolToMolBlock(mol), 'mol')
v.setStyle({'stick':{}})
v.zoomTo()
v.show()
```



# Tips

rdkit.DataStructs.cDataStructs.**AsymmetricSimilarity**(*(SparseBitVect)bv1*, *(SparseBitVect)bv2*[, *(bool)returnDistance=0*]) → float :

C++ signature :
double AsymmetricSimilarity(SparseBitVect,SparseBitVect [,bool=0])

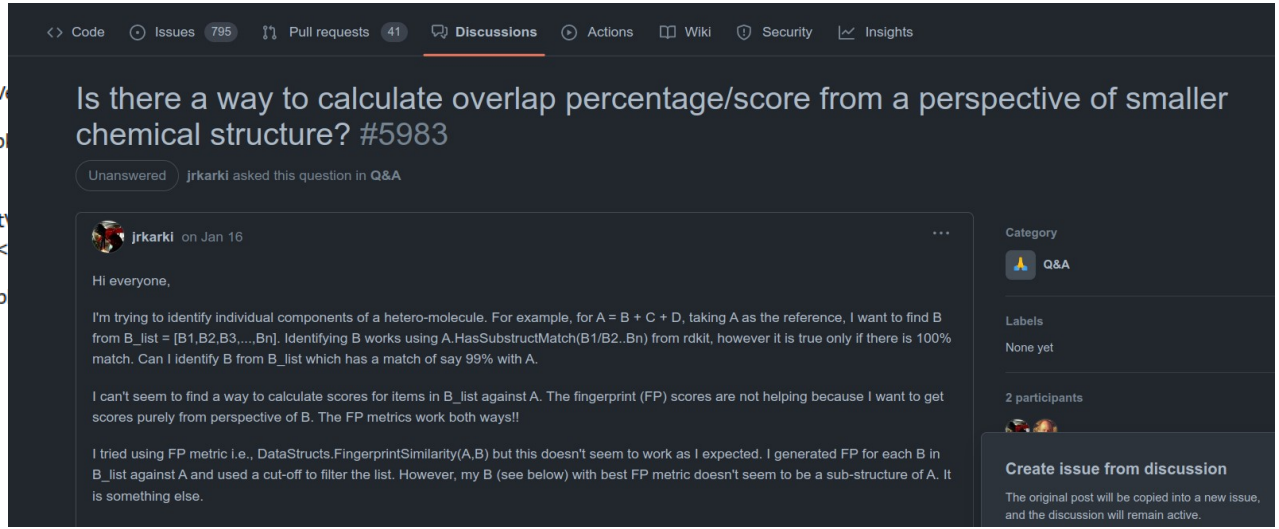
AsymmetricSimilarity( (ExplicitBitVect)bv1, (ExplicitBitVect)bv2 [, (bool)returnDistance=0]) -> float :
B(bv1&bv2) / min(B(bv1),B(bv2))

C++ signature :
double AsymmetricSimilarity(ExplicitBitV

AsymmetricSimilarity( (SparseBitVect)bv1, (str)p

C++ signature :
double AsymmetricSimilarity(SparseBit
std::char_traits<char>, std::allocator<

AsymmetricSimilarity( (ExplicitBitVect)bv1, (str)p
B(bv1&bv2) / min(B(bv1),B(bv2))




https://github.com/rdkit/rdkit/discussions/5983

# まとめ

- Pyscf と RDKit で QM<>Chemoinfo の橋渡しができる。
- py3Dmol を利用して軌道が眺められる！
- psikit も便利（最新の Psi4 をフォローできていない、、）
- RDKit  の Github レポの Discussion は要チェック！