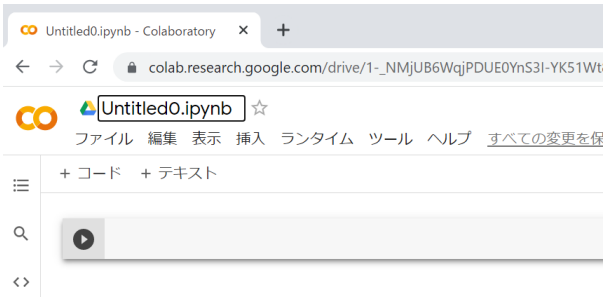
Google Colaboratory の使い方

プログラミング言語 Python は学習が容易な初心者向け言語の一つですが、機械学習を含めた科学計算にも使用される奥の深い言語です。また、Google が提供する Colaboratory（Colab）は Python の実行環境をクラウド上でブラウザを用いて操作するので、誰でも容易にプログラミングを始めることができます。Google アカウントを用意して Python プログラミングを Colab で始めてみましょう。

1）Google アカウントにログインして Colab の URL にアクセスしましょう。

https://colab.research.google.com/notebooks/welcome.ipynb

Google Colab のウエルカムページが表示されますので、最初は「ノートブックを新規作成」を選んで下さい。ちなみに、このページから過去に作成したノートブック（Colab におけるプログラムコードやメモをまとめたもの）を開くこともできます。

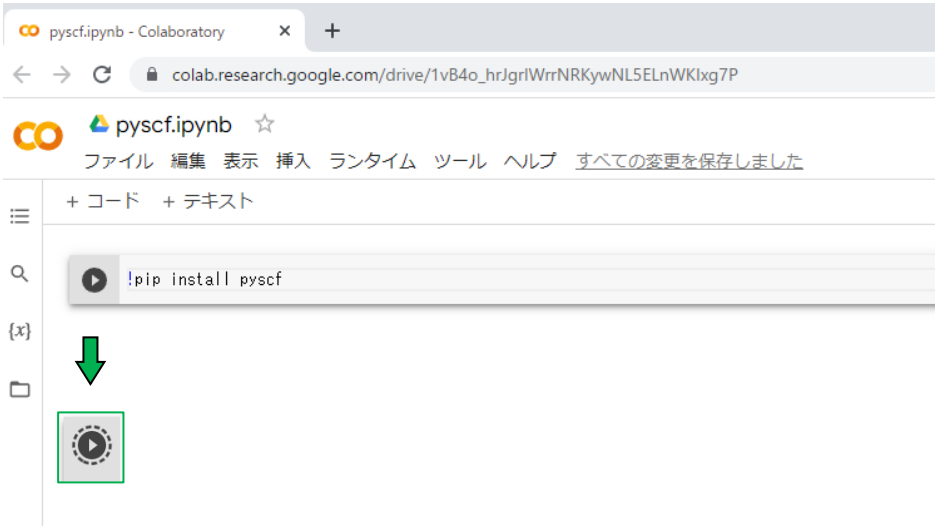


2）最初はノートブックの名前を変えましょう。

左上の○○.ipynb の部分です。ノートブックの拡張子は Jupyter などと同

じ.ipynb です。今回は pyscf.ipynb と変えました。

3）まず、外部ライブラリ PySCF を取り込みます。Colab では numpy や matplotlib など Python を実行する上で便利なライブラリが既にインストールされていますが、量子化学の計算環境は初期状態では入っていません。計算を始める前に毎回インストールする必要があります。



三角のマーク（▷）の右側がコードを打ち込む部分です。コピー＆ペーストで他のコードを写しても、キーボードから１文字ずつ入力しても構いません。ただし、全角スペースには注意して下さい。Python がコードとして認識するのは、半角の英数文字です。スペースが日本語モードの全角で入力されると、Python がスペースと理解できずにエラーを吐きます。コードを打ち込むときは、基本は英語入力モードにして下さい。

PySCF をインストールするために、コードを打ち込む部分に「!pip install pyscf」と入力します。pip とは Python でライブラリを管理するコマンドで Colab 上では「 ！ 」マークを付けて実行します。

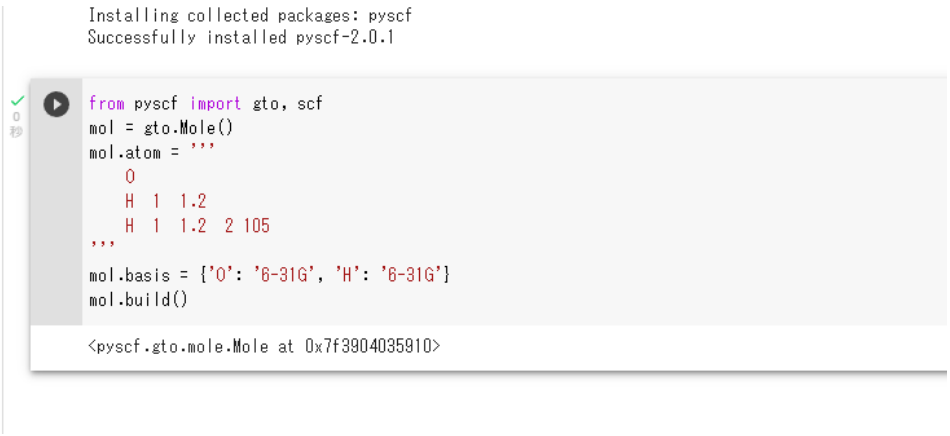
打ち込んだコードを実行するのは、三角のマーク（▷）をクリックします。三角の周りに破線が現れて実行待ちののち、コードが実行されます。しばらく待って、Successfully installed pyscf-**と表示されれば、ライブラリのインストールは完了です。

PySCF の詳しい情報はこちらのページを参照して下さい
https://pyscf.org/
量子化学計算の様々な手法が Python 環境で実行できます。

４）どんどんコードを打ち込んで実行していこう。

コードに実行が終了すると、三角のマーク（▷）に数字が表示されたり、左にチェックマークがつきます。さらに別のコードを入力するには、（＋コード）をクリックして、新たにコード入力エリア（セル）を作成して下さい。Colab にはプログラムのコードを保存するだけでなく、メモ書きも保存することができ（＋テキスト）をクリックすると、実行できるコードではなく、メモとして保存できるテキストセルが作成されます。

５）分子構造の構築

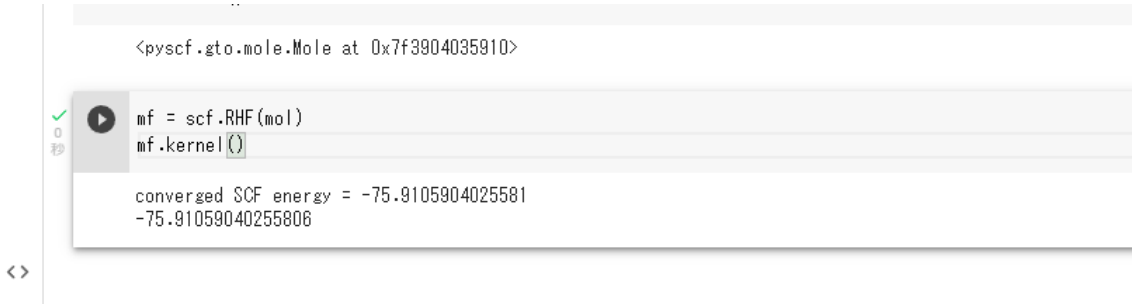


打ち込むコードを１行ごとに解説します。

「from pyscf import gto, scf」ここではインストールした PySCF より gto と scf という

名前のモジュールを読み込みます。gto は Gaussian Type Orbital で分子を計算するためで、scf は Self-field consistent に問題を解くために使います。
「mol = gto.Mole()」では mol という名前のインスタンスを作成しています。インスタンスとはオブジェクト指向型言語での言葉なので詳しく勉強して欲しいのですが、GTO 関数を使って分子を構築していく様々な情報を格納する変数の名前を mol と決めましたぐらいの理解で構わないと思います。
「mol.atom = '''
O
H 1 1.2
H 1 1.2 2 105
'''」 ここは5行まとめて説明します。Python ではシングルコーテーション「 ‘ 」3個で特別な意味があり、「'''」で囲まれた部分は改行も意味がある文字列になります。先ほど作成した mol という名前のインスタンスに対して、atom プロパティーとして原子位置の情報を設定します。2行目の「O」で最初に酸素原子を配置します。3行目の「H 1 1.2」では、水素原子を、1番目の原子、つまり、酸素原子から 1.2 Å の位置に配置します。さらに「H 1 1.2 2 105」では、もう1個の水素原子を1番目の酸素原子から 1.2 Å の距離で、2番目の水素原子と1番目の酸素原子の間でなす角が 105°となるように配置します。
「mol.basis = {'O': '6-31G', 'H': '6-31G'}」では GTO 基底関数として O 原子も H 原子も 6-31G 基底を使うことを定義しています。
最後の「mol.build()」で指定した原子位置の分子構造と基底関数の構築を行います。

5）量子化学計算の実行



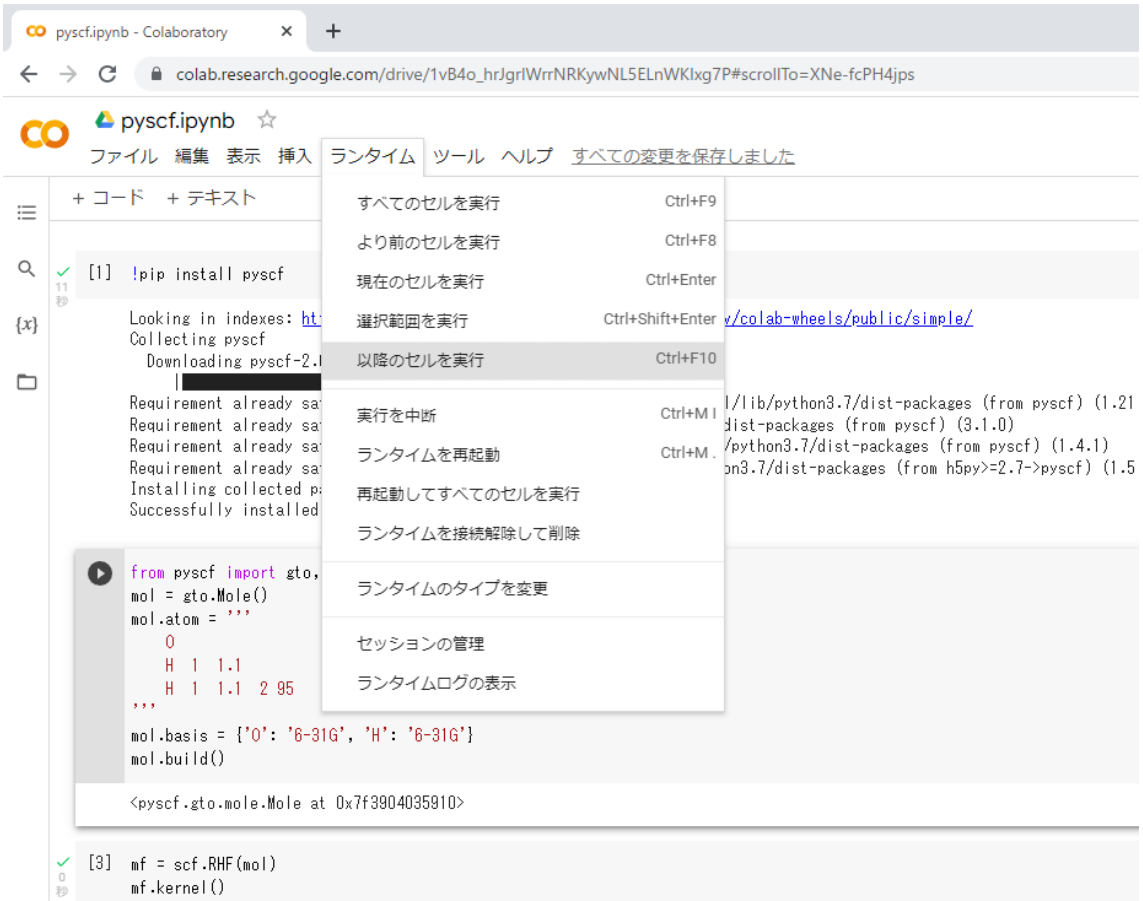
構築した分子の計算を行いましょう。「mf = scf.RHF(mol)」RHF 法で scf 計算を mol で定義した分子で行い、結果を mf に格納します。RHF とは restricted Hartree-Fock 法でスピンを制限した閉殻構造（スピン多重度1）での計算です。この程度のサイズの分子と基底関数であれば、すぐに結果は出ると思います。
「mf.kernel()」では計算した結果の核の部分を表示します。ここでは、求まったエネルギー値を Hartree 単位（原子単位 a.u.）で表示されます。他にも電荷情報が必要であれば

mf.mulliken_pop()を追加するなど、コードを修正すれば、様々な量子化学計算結果が表示されます。

６）続けて計算を行う

打ち込んだコードを実行するには、（ランタイム）メニューから（現在のセルを実行）を選ぶと、選択中のセルの三角マークをクリックした状態と同じことができます。また、キーボードの（SHIFT）キーを押しながら（ENTER）キーを押すと実行するだけでなく、新しいコード入力セルも生成されます。

一度実行したセルも、数値などを変えて再度実行することも可能です。具体的には２番目のセルの O 原子と H 原子の結合距離や結合角の数値を変更し、変更したセルを選択した状態で、（ランタイム）メニューから（以降のセルを実行）を選ぶと、２番目のセルと３番目のセルが実行され、新しく入力した結合距離と結合角の情報で、エネルギー計算した結果が表示されます。



７）Google Colab コード例

```
# ----- code 1/3
!pip install pyscf

# ----- code 2/3
from pyscf import gto, scf
mol = gto.Mole()
mol.atom = '''
   O
   H  1  1.2
   H  1  1.2  2 105
'''
mol.basis = {'O': '6-31G', 'H': '6-31G'}
mol.build()

# ----- code 3/3
mf = scf.RHF(mol)
mf.kernel()
```

（密度汎関数法 DFT 計算をする場合）　　B3LYP 汎関数

from pyscf import gto, scf ⇒ from pyscf import gto, dft

mf = scf.RHF(mol) ⇒ mf = dft.RKS(mol); mf.xc = ‘b3lyp’

（摂動法 MP2 計算をする場合）

from pyscf import gto, scf ⇒ from pyscf import gto, mp

mf = scf.RHF(mol) ⇒ mf = mp.MP2(mol)

（基底関数の例）

'6-31G' ⇒ 'STO-3G', '3-21G', '6-31+G', '6-31++G', '6-311G',
'6-311+G', '6-311++G', 'cc-pvdz', 'cc-pvtz', 'cc-pvqz', ……