## 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。

CO Colaboratory へようごそ - Colabors × +							
← → C							
co	Colaboratory へようこそ ファイル 編集 表示 挿入 ランタイム ツール ヘルプ						
≡ ■	ノートブックを新規作成 ノートノックを開く	F Ctrl+0	・ + テキスト & ドライブにコピー				
Q (	ノートブックをアップロード	<b>_</b>	)Colaboratory とは				
<> t		b	oratory(略称: Colab)は、ブラウザから Python を記述、実行できるサービスです。次の特長を備				
{x} *	ドライブにコピーを保存 コピーを GitHub Gist として保存 GitHub にコピーを保存	3 ( 1	環境構築が不要 GPU への無料アクセス 簡単に共有 (ナー <b>学生</b> から <b>データ サイエンティスト、ムリリサーチャー</b> まで、皆さんの作業を効率化します、詳し				
	保存 変更屬歴	Ctrl+S Z	はい。下のリンクからすぐに使ってみることもできます。				
	ダウンロード	- "	めに				
	白扇」	Ctrl+P(こ ノィブ	こなっているこのドキュメントは静的なウェブページではなく、 <b>Colab ノートブック</b> という、コード げな環境です。				
		たとえ	tば次の <b>コードセル</b> には、値を計算して変数に保存し、結果を出力する短い Python スクリプトが記;				
		[] s s	seconds_in_a_day = 24 * 60 * 60 seconds_in_a_day				
		8	16400				

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

œ	Untit	tled0.ipynb	- Colabo	ratory	×	+				
$\leftarrow$	$\rightarrow$	C	colab	resear	ch.goo	gle.com/drive	e/1NMjl	JB6WqjP[	DUE0YnS3I-YK	(51Wt
C	C	<mark>∆</mark> Unt	itled0.	ipynk	<b>&gt;</b> ☆					
		ファイル	/ 編集	表示	挿入	ランタイム	ツール	ヘルプ	<u>すべての変更</u>	更を保
≣	+	コード	+ テキ	スト						
Q		0								
$\langle \rangle$										

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。

œ	pyscf.ipynb - Colaboratory X +
$\leftarrow$	→ C a colab.research.google.com/drive/1vB4o_hrJgrlWrrNRKywNL5ELnWKIxg7P
C	O ▲ pyscf.ipynb ☆ ファイル 編集 表示 挿入 ランタイム ツール ヘルプ <u>すべての変更を保存しました</u>
≣	+ コード + テキスト
Q	Ipip install pyscf
{ <i>x</i> }	Ļ

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 カします。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
 <a href="https://pyscf.org/">https://pyscf.org/</a>
 量子化学計算の様々な手法が Python 環境で実行できます。

CO pyscf.ipynb - Colaboratory × +							
$\leftarrow \   \rightarrow$	C a colab.research.google.com/drive/1vB4o_hrJgrlWrrNRKywNL5ELnWKIxg7P#scrollTo=XNe-fcPH4jps						
CO ▲ pyscf.ipynb ☆ ファイル 編集 表示 挿入 ランタイム ツール ヘルプ							
;							
٩ (	[]] pip install pyscf						
{x}	Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/ Collecting pyscf Downloading pyscf-2.0.1-cp37-cp37m-manylinux1_x86_64.whl (37.5 MB) 37.5 MB 1.5 MB/s Requirement already satisfied: numpy!=1.16,!=1.17,>=1.13 in /usr/local/lib/python3.7/dist-packages (from pyscf) (1.21.6) Requirement already satisfied: scipy!=1.16,!=1.5.1 in /usr/local/lib/python3.7/dist-packages (from pyscf) (3.1.0) Requirement already satisfied: scipy!=1.5.0,!=1.5.1 in /usr/local/lib/python3.7/dist-packages (from pyscf) (1.4.1) Requirement already satisfied: ached-property in /usr/local/lib/python3.7/dist-packages (from h5py>=2.7->pyscf) (1.5.2) Installing collected packages: pyscf Successfully installed pyscf-2.0.1						
	0						

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 = '''

0

- H 1 1.2
- H 1 1.2 2105

"」 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1.2Å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1.2Å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

			<pyscf.gto.mole.mole 0x7f3904035910="" at=""></pyscf.gto.mole.mole>					
	<b>~</b> 砂	D	mf = scf.RHF(mol) mf.kernel()					
			converged SCF energy = -75.9105904025581 -75.91059040255806					
$\rightarrow$	_							

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0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。

œ	co pyscf.ipynb - Colaboratory x +									
←	← → C  C colab.research.google.com/drive/1vB4o_hrJgrlWrrNRKywNL5ELnWKlxg7P#scrollTo=XNe-fcPH4jps									
C	0	<b>ム</b> ファ	pyscf.ipynb ☆ イル 編集 表示 挿入	ランタイム ツール ヘルプ す	べての変更を保存	しました				
≔	+		-ド + テキスト	すべてのセルを実行	Ctrl+F9					
				より前のセルを実行	Ctrl+F8					
Q	<b>×</b> 11	[1]	!pip install pyscf	現在のセルを実行	Ctrl+Enter					
{ <i>x</i> }	杪	Looking in indexes: <u>ht</u>					Looking in indexes: <u>ht</u>	選択範囲を実行	Ctrl+Shift+Enter	<u>v/colab-wheels/public/simple/</u>
_			Downloading pyscf-2.	以降のセルを実行	Ctrl+F10					
			Requirement already sa Requirement already sa Requirement already sa Requirement already sa Requirement already sa Installing collected p Successfully installed	実行を中断 実行を中断 ランタイムを再起動 再起動してすべてのセルを実行	//lib/python3.7/dist-packages (from pyscf) (1.21 dist-packages (from pyscf) (3.1.0)					
					/python3.//dist-packages (from pyscf) (1.4.1) on3.7/dist-packages (from h5py>=2.7->pyscf) (1.5					
		•	from pyscf import gto, mol = gto.Mole() mol.atom = ''' 0	ランタイムを接続解除して削除						
		o		ランタイムのタイプを変更						
				セッションの管理						
			,,, H 1 1.1 2 95	ランタイムログの表示						
			mol.basis = {'O': '6-3 mol.build()	1G', 'H': '8-31G'}						
			<pyscf.gto.mole.mole 0x7f3904035910="" at=""></pyscf.gto.mole.mole>							
	<b>~</b> 10 初	[3]	mf = scf.RHF(mol) mf.kernel()							

```
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 = ""
  0
  H 1 1.2
  H 1 1.2 2 105
m
mol.basis = {'0': '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 \Rightarrow from pyscf import gto, dft
mf = scf.RHF(mol) \Rightarrow mf = dft.RKS(mol); mf.xc = 'b3lyp'
 (摂動法 MP2 計算をする場合)
from pyscf import gto, scf \Rightarrow from pyscf import gto, mp
mf = scf.RHF(mol) \Rightarrow mf = mp.MP2(mol)
 (基底関数の例)
```