

組成式から記述子を生成してみよう！ Xenonpyインストールマニュアル

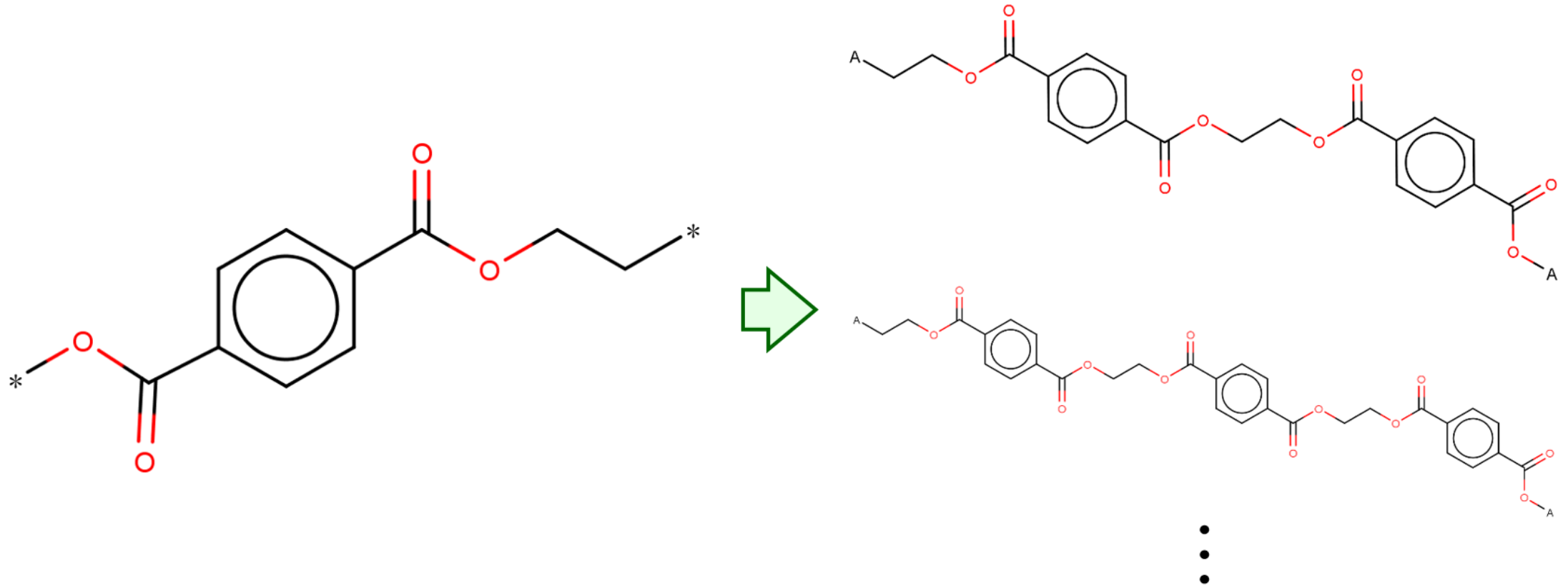
Masaki Open Lab



お断り

- 当資料の利用は自由ですが、内容に誤りを含む場合もございます
(何分素人です)
- 怪しい点を発見された場合は、コメントいただけますと幸甚です
(環境構築は個々人のPCによる部分が大きいので、対応には限界がございます点ご容赦ください)
- セキュアな社内ネットワークにおける環境構築手法については「知らん！」

背景：マテリアルズインフォマティクスとは



- ・マテリアルズインフォマティクス（MI）：機械学習により材料開発を支援
- ・**材料の情報を説明変数**とし、目的性能を予測するモデルを構築
- ・モデルを逆解析する事で、所望の目的性能を有する材料を提案

背景：重要なのは「如何に記述子を生成するか」

分子を描いてSmilesアイコンをクリックすると、Smilesの構造式が得られるので、それをコピーする。

SMILES

C=Cc1ccccc1

Close

S

F

Cl

Br

I

X

ブック9

新規作成 開く 保存 プリント インポート コピー ペースト 書式 元に戻す やり直し オート

MS Pゴシック 11 B I U

	A	B	C	D	E
1	No	Smiles			
2	1	O=C(C)C			
3	2	O=C(C)CCC			

ここに入力して検索

13:38 2020/05/14

- ・有機分子について、材料の情報を説明変数に変換する方法は複数ある（例：SMILESキーからRDKitにて～1800個の説明変数に自動変換）
- ・一方で、無機固体に関する記述子の開発は比較的遅れていた

目的

無機物の「組成式」から、目的性能の予測に使える
簡便な記述子を自動生成する方法を紹介する

既往の文献例

Journal of Computer Aided Chemistry , “無機材料の組成式を元にした物性予測のための記述子開発”（佐方、船津ら）

Table 5. Feature importance of density prediction model

記述子名	寄与率
典型元素の割合	0.3184
構成元素の原子量の平均	0.1220
全原子の原子量の平均	0.1203
遷移元素の割合	0.1091
原子量の最大値	0.0584
原子番号の最大値	0.0571
全原子の原子番号の標準偏差	0.0218
全原子の原子番号の分散	0.0168
全原子のイオン半径の平均	0.0121
第6周期元素の割合	0.0113
構成元素のイオン半径の和	0.0105
全原子の電気陰性度の平均	0.0098
第3周期元素の割合	0.0090
全原子の原子番号の平均	0.0089
イオン半径の最大値	0.0047

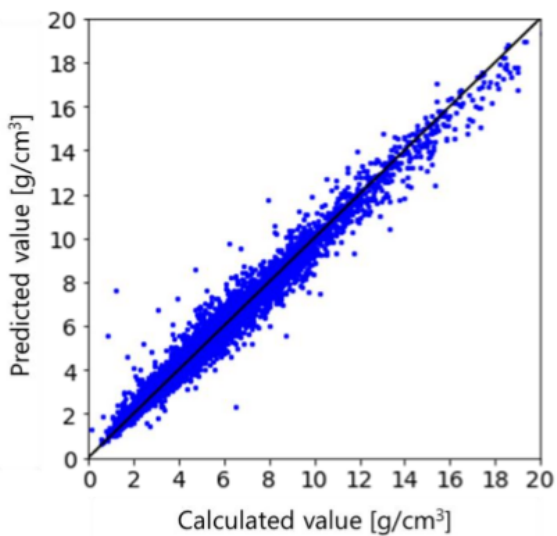


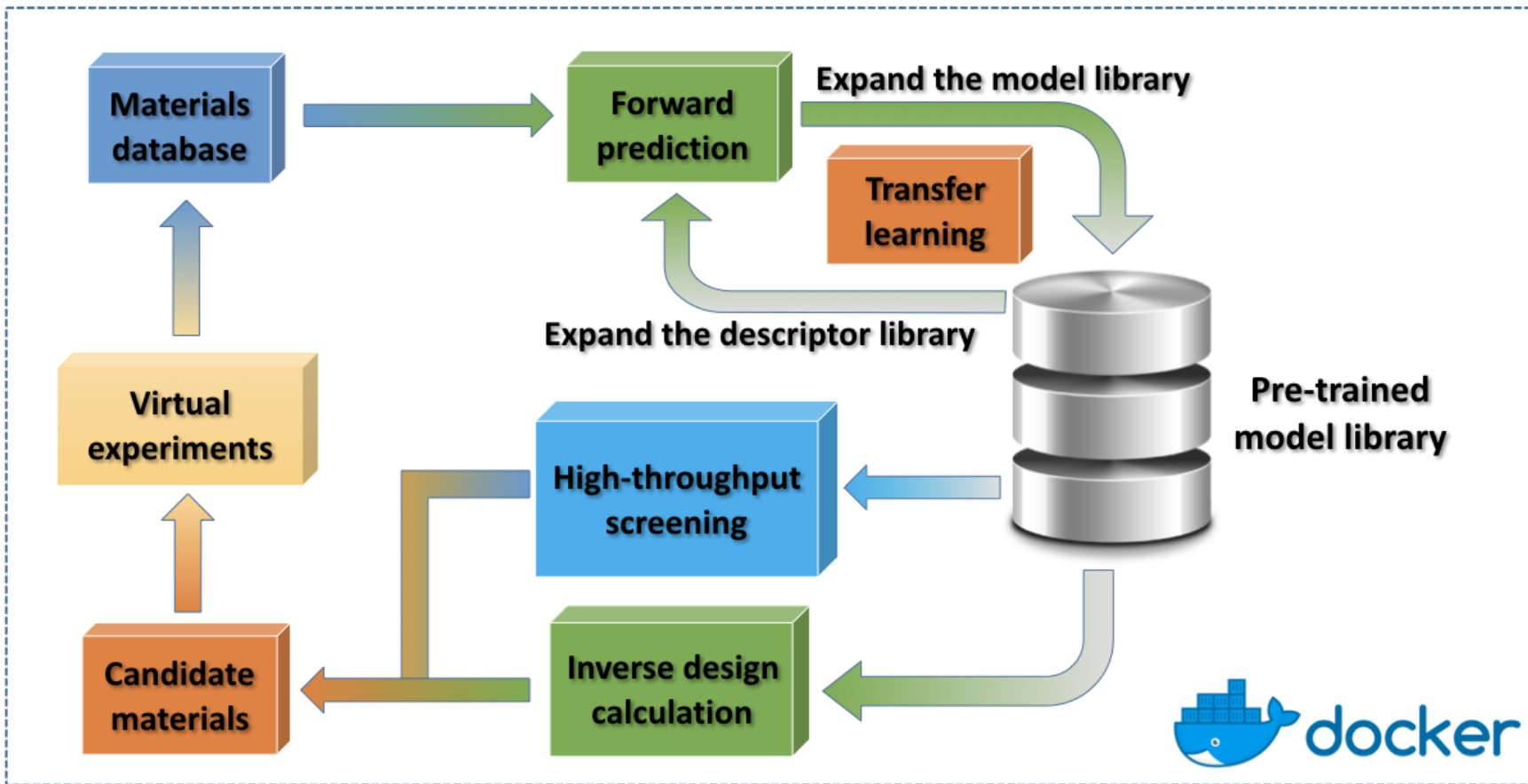
Figure 3. Predicted values for calculated values of density

Table 6. Evaluation value of accuracy of density prediction

R^2_{train}	R^2_{test}	RMSE	MAE
0.997	0.977	0.426	0.269

- ・組成式から読み取れる「当たり前前の説明変数（～350個）」でモデル構築
- ・割と現実的な精度で各種性能を予測できる事もあるらしい
- ・ソースコードが公開されてない（おこ）

今回紹介するパッケージ "XenonPy"について



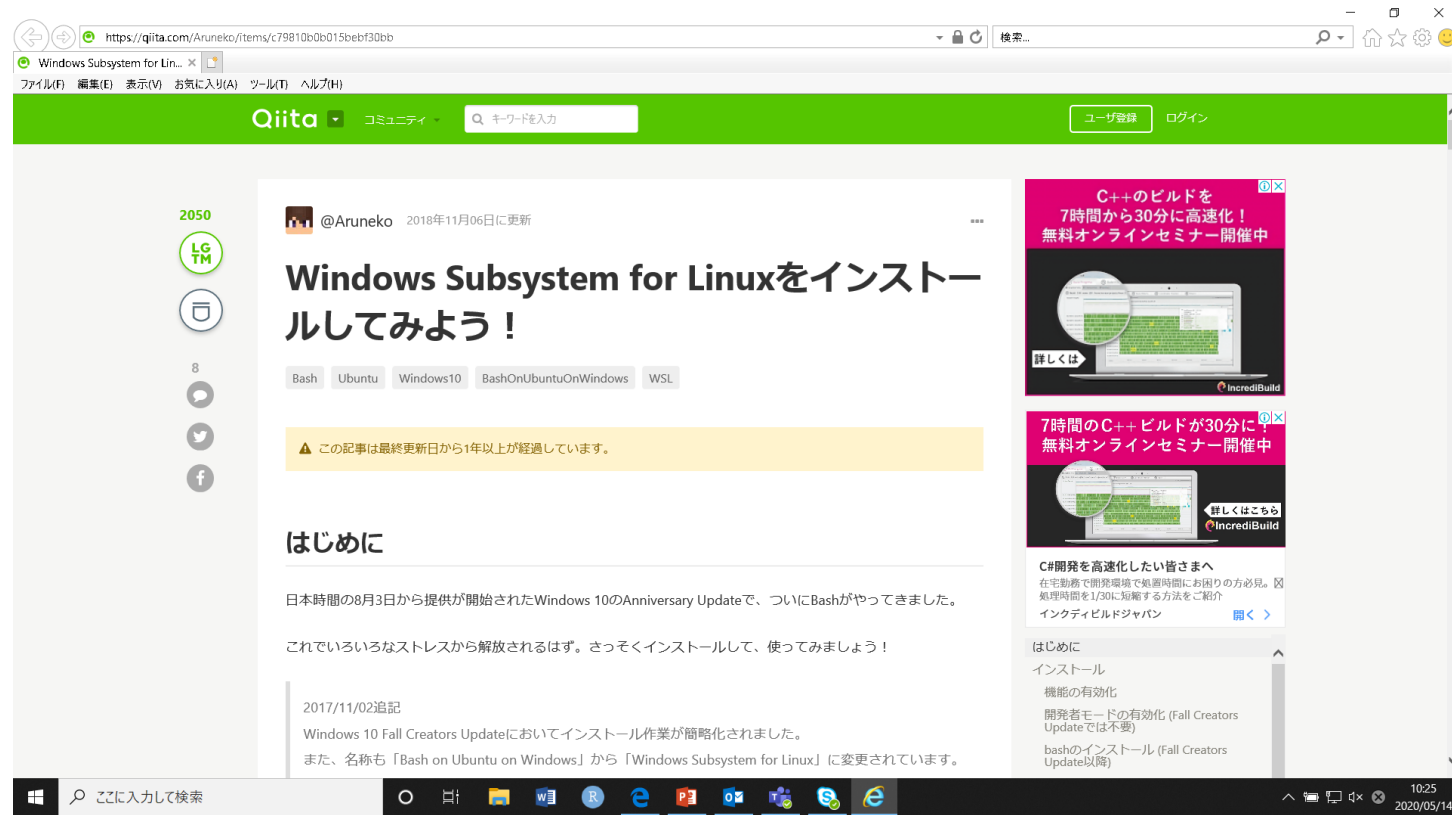
- ・組成式から読み取れる「当たり前前の説明変数（290個）」を自動生成
- ・Pythonパッケージ、但しLinux環境じゃないと動きません
- ・WindowsユーザはWSL経由する必要がある、ややこしい←このマニュアル

マニュアル目次

- WSL
 - WSLのインストール
 - Ubuntuのインストール
 - gccのインストール
- Anaconda
- XenonPy
 - Pytorchのインストール
 - Pymatgenのインストール
 - RDKitのインストール
- Tutorial

WSL（環境構築とUbuntuのインストール）

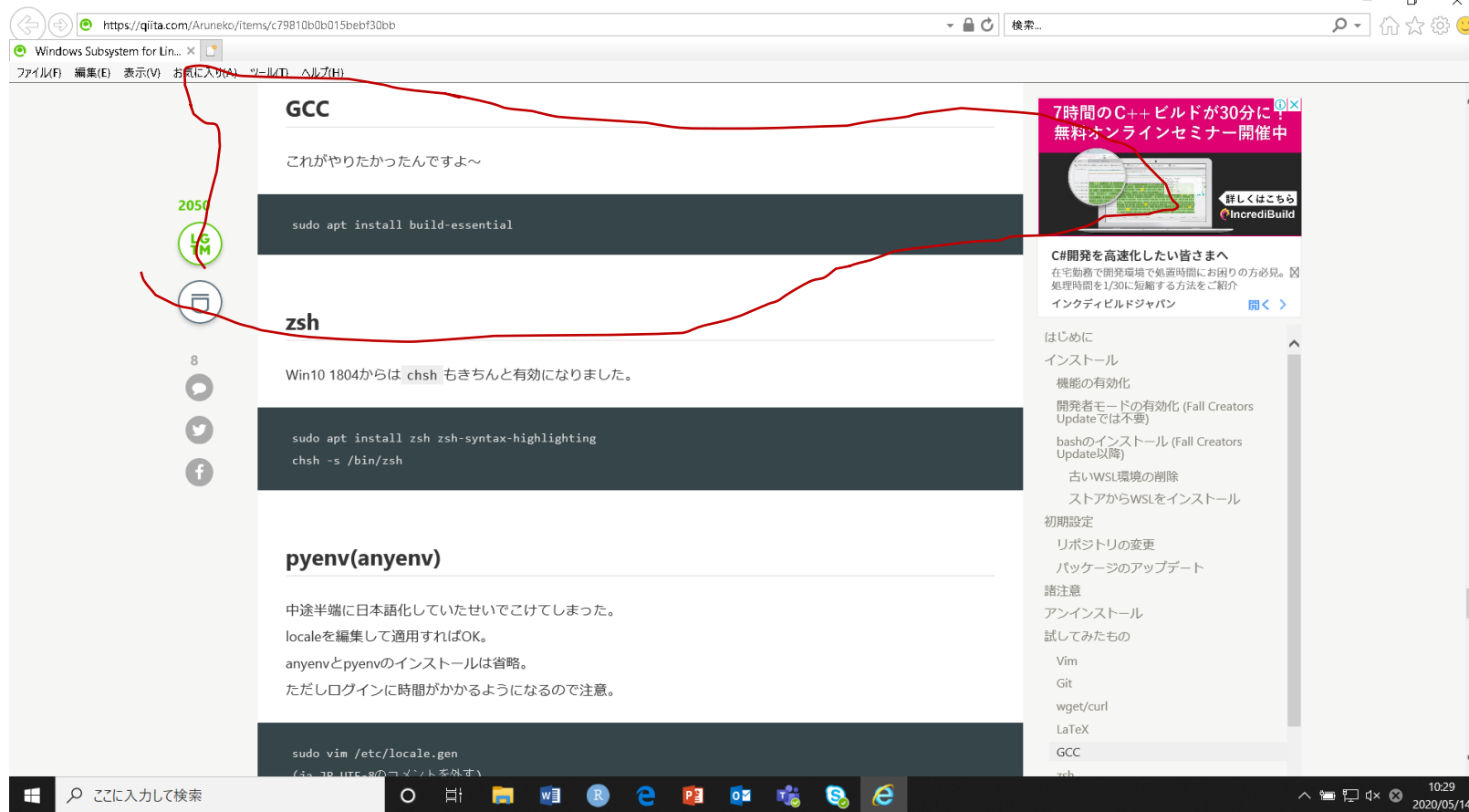
<https://qiita.com/Arunekeo/items/c79810b0b015bebf30bb>



- このページに書いてある内容を（パッケージのアップデートまで）やる
- 古いWSLの削除は（元々入れてない限り）必要ありません
- リポジトリの変更も私はやってないです

WSL (gccのインストール)

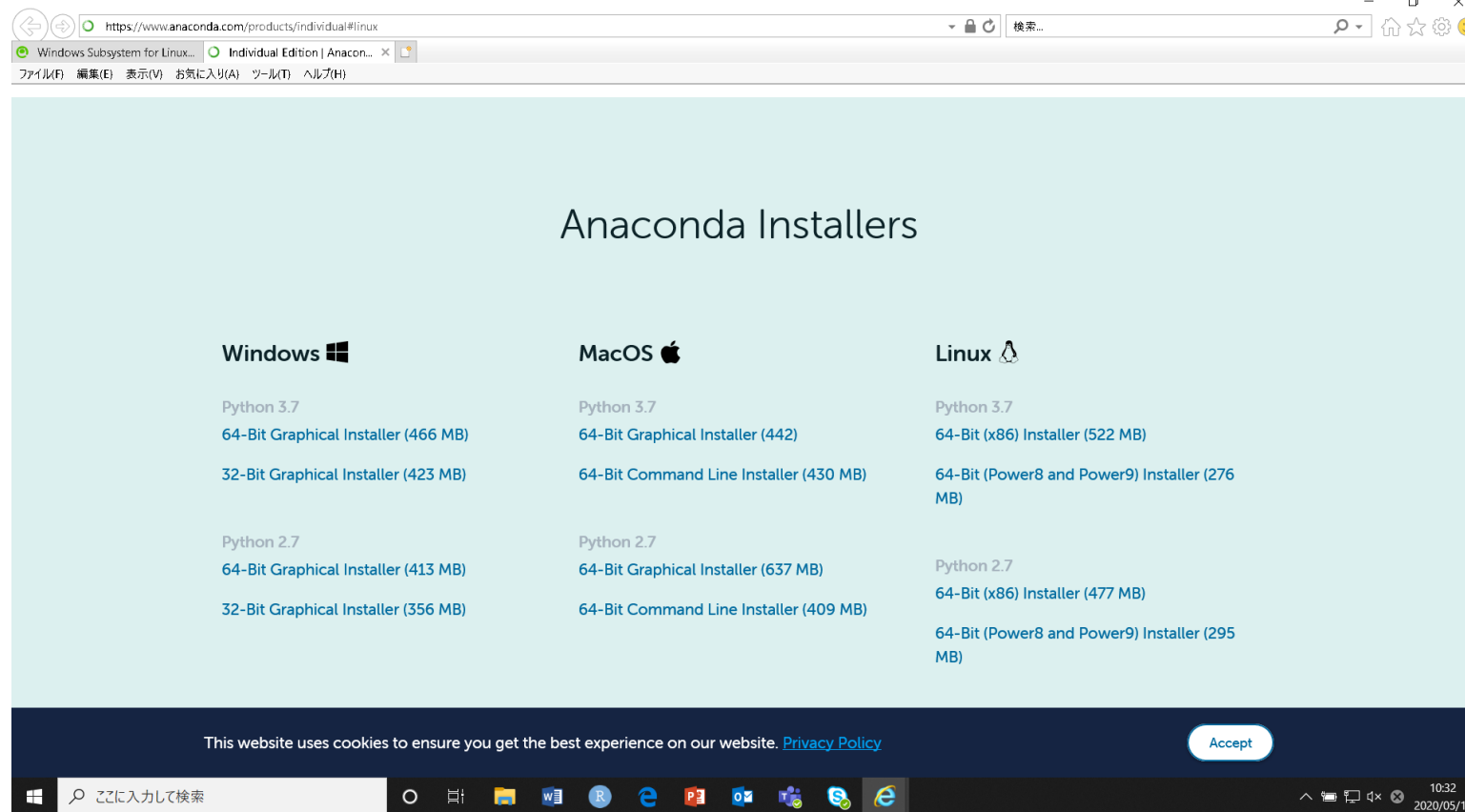
<https://qiita.com/Aruneke/items/c79810b0b015bebf30bb>



- 同じページの下の方に書いてある
- これもインストールしておく

Anaconda (ダウンロード)

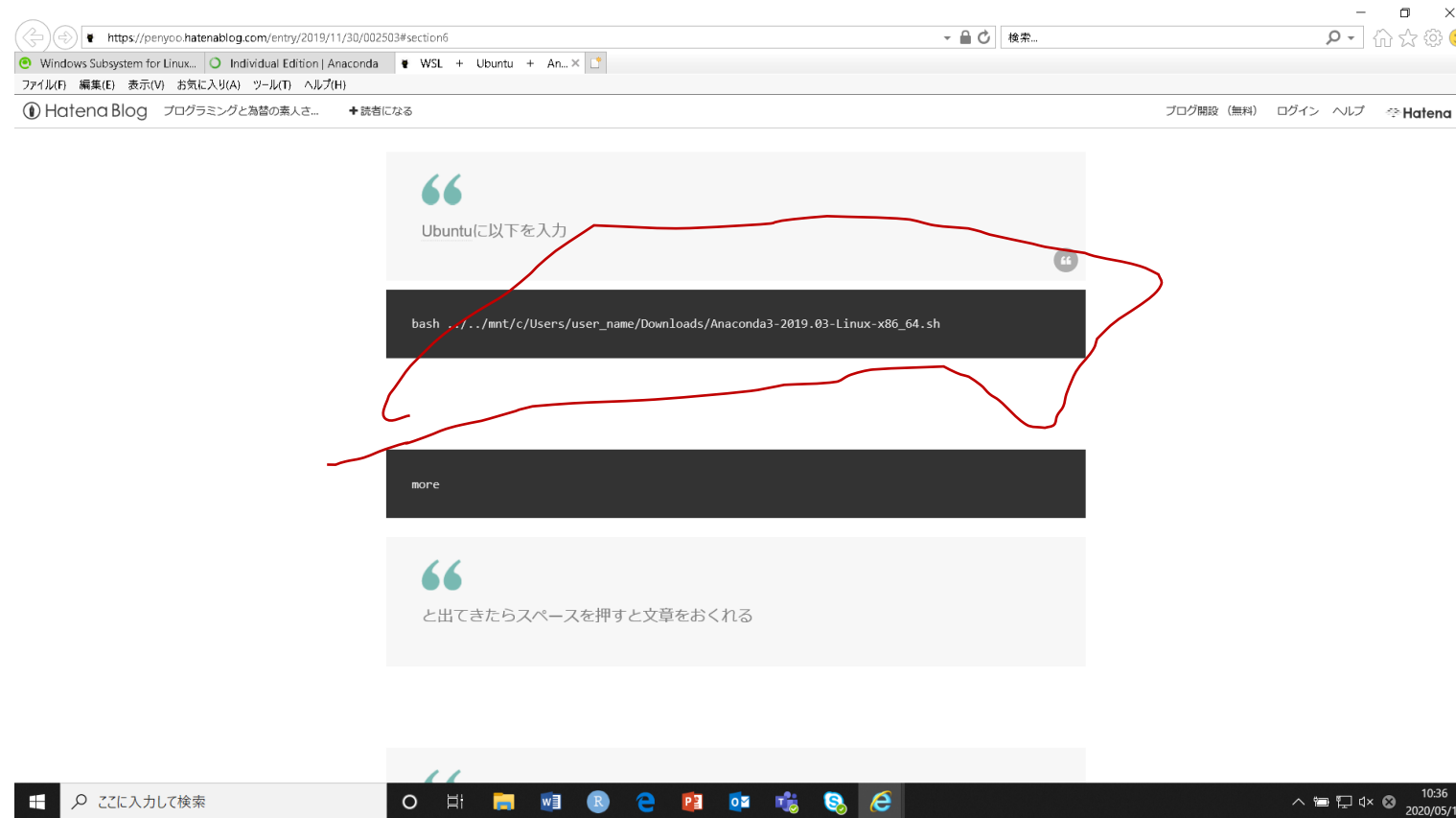
<https://www.anaconda.com/products/individual#linux>



- Anacondaのホームページから、Linux版を落とす

Anaconda (Ubuntuからインストール)

<https://penyoo.hatenablog.com/entry/2019/11/30/002503#section6>

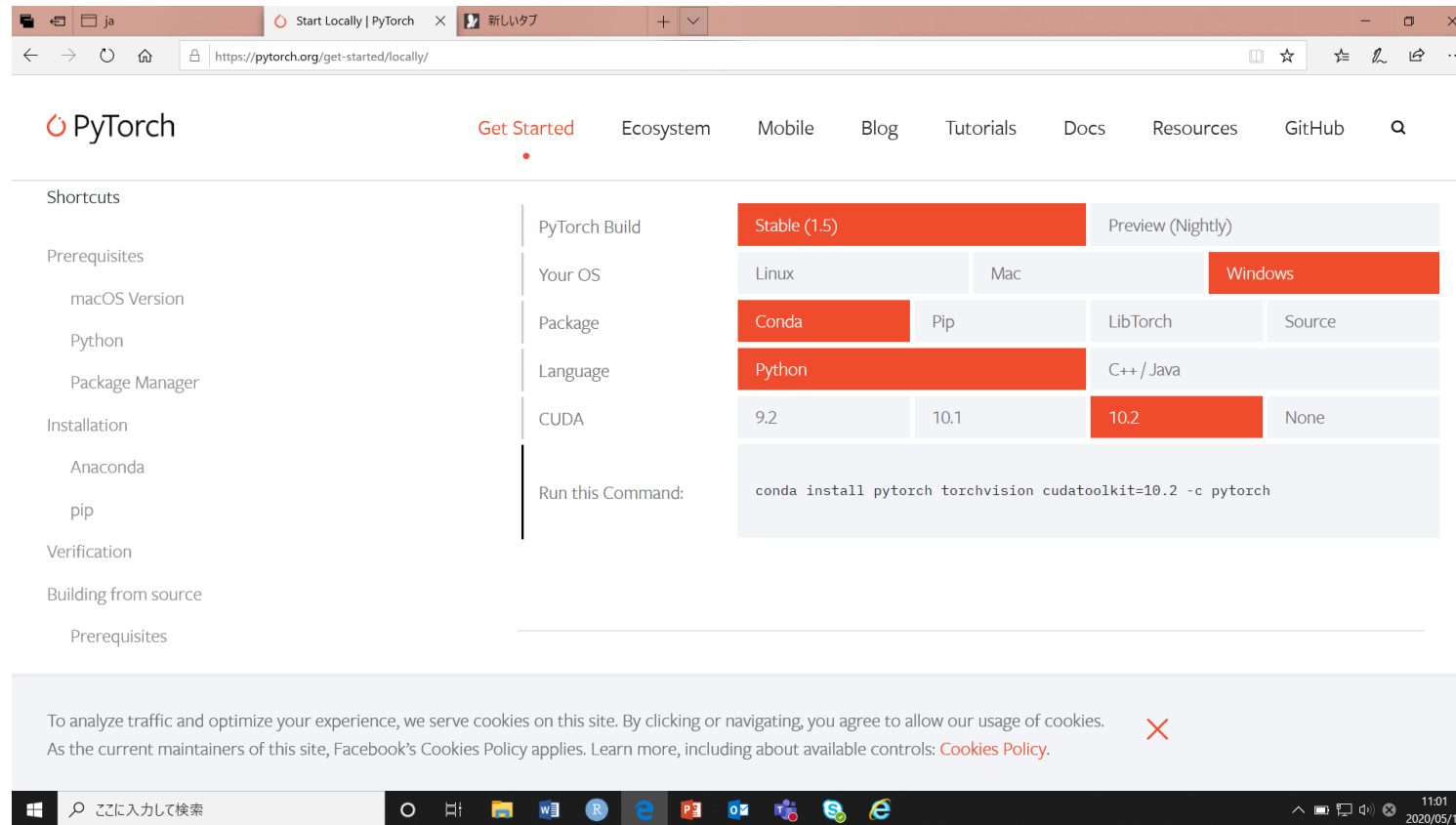


- リンク先にUbuntu上での作業内容が書いてある（囲み部分）
※ 勿論ユーザ名は自分のです。ダウンロードに落としたAnacondaのプロパティを見れば番地名が分かります。
バージョンは上記と違うので落としたAnacondaの名前を正確に入力ください。
- 後は基本的にyesと答えてEnterを押すだけの簡単なお仕事です

XenonPy (Pytorch)

<https://xenonpy.readthedocs.io/en/latest/installation.html>

<https://pytorch.org/get-started/locally/>

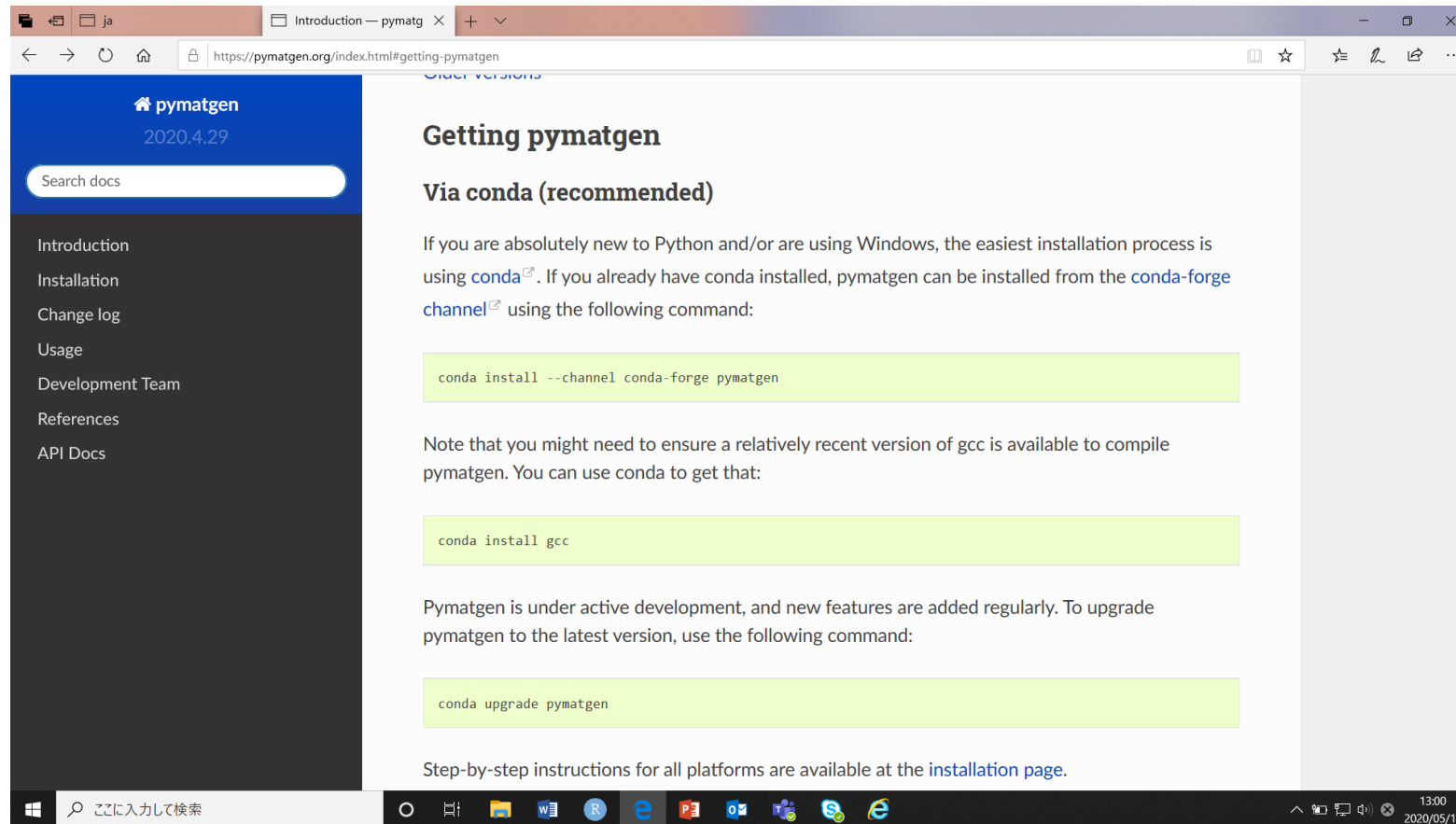


・環境を選べと、適したコマンドが現れるのでこれを実行

XenonPy (pymatgen)

<https://xenonpy.readthedocs.io/en/latest/installation.html>

<https://pymatgen.org/index.html#getting-pymatgen>

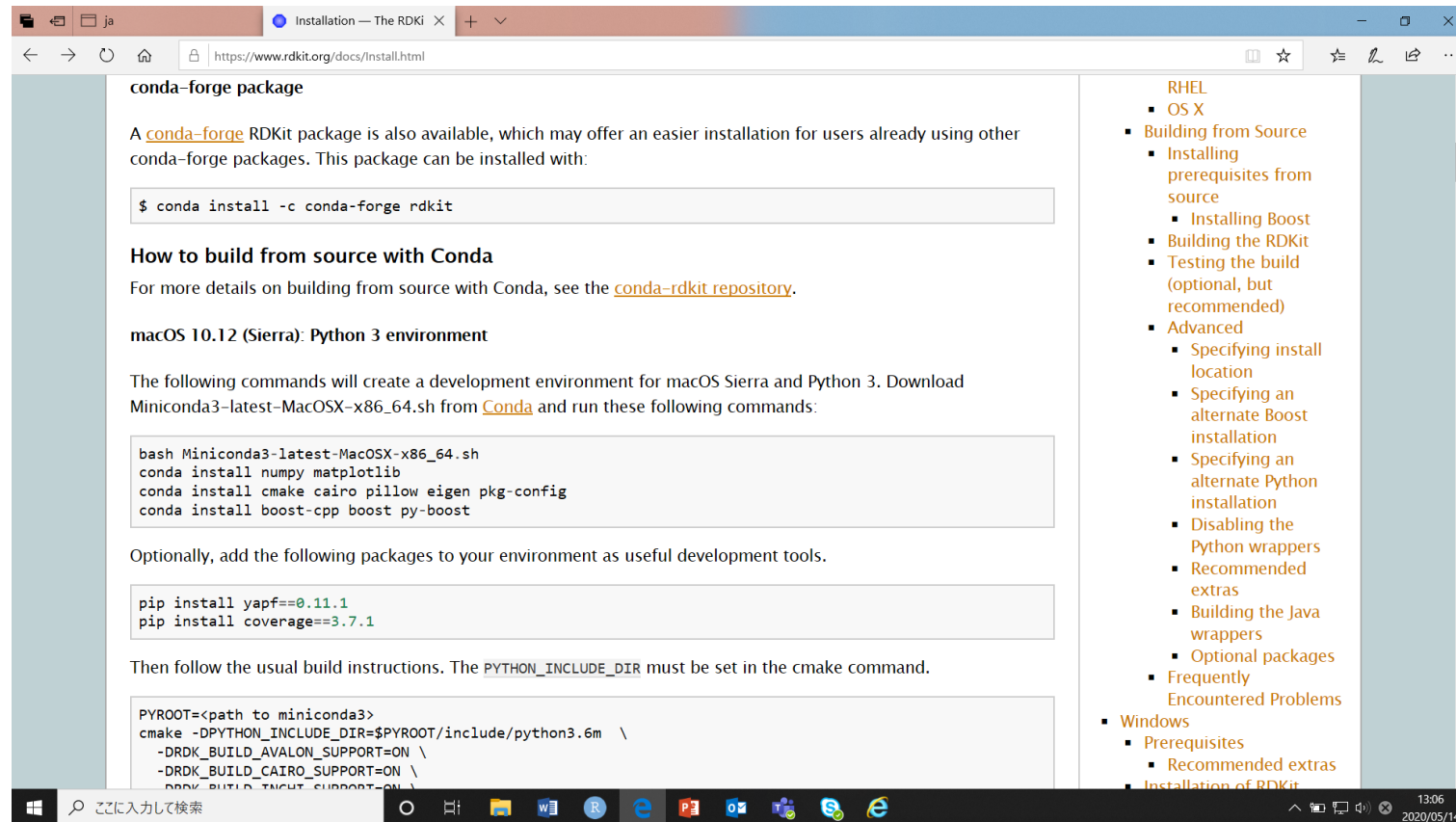


• Via condaの2行（gccは先にインストール済み）を実行

XenonPy (RDKit)

<https://xenonpy.readthedocs.io/en/latest/installation.html>

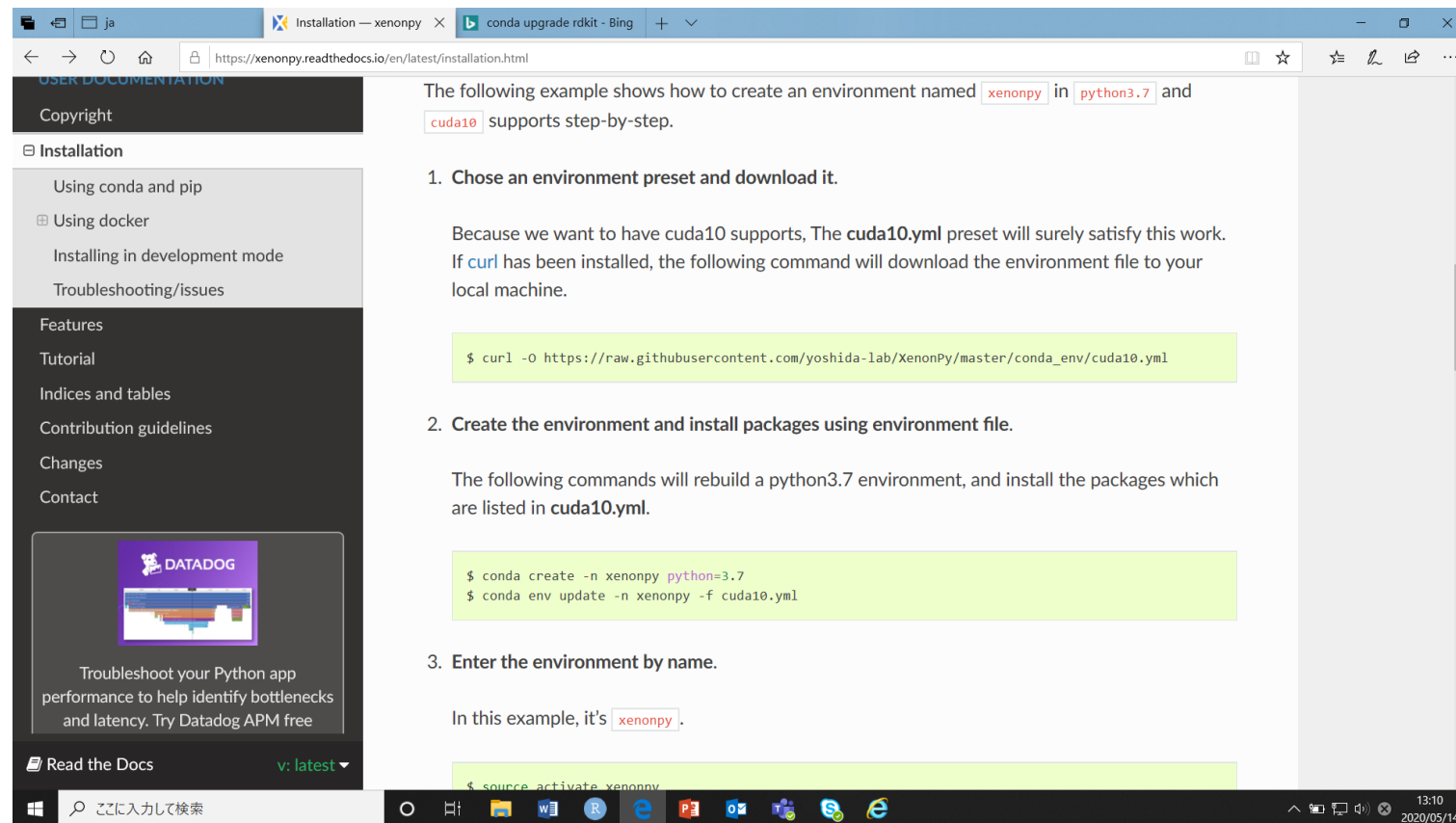
<https://www.rdkit.org/docs/Install.html>



• conda forge packageの行を実行

XenonPy (本体)

<https://xenonpy.readthedocs.io/en/latest/installation.html>

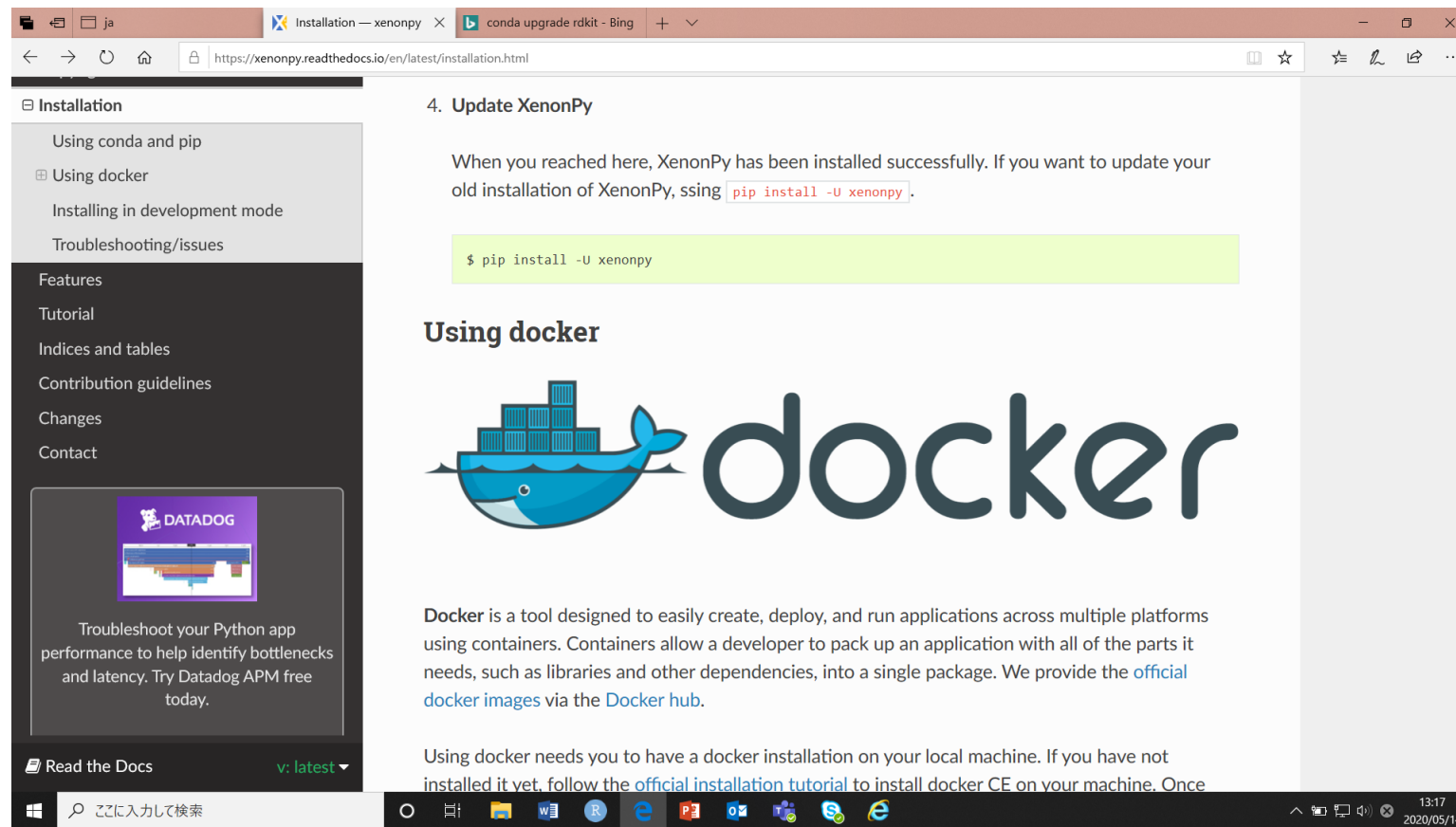


- 1. Choose an environment preset and download it
- 2. Create the environment and install packages...を実行

※\$ curl -O https://raw.githubusercontent.com/yoshida-lab/XenonPy/master/conda_env/cpu.yml
(GPU使わない想定)

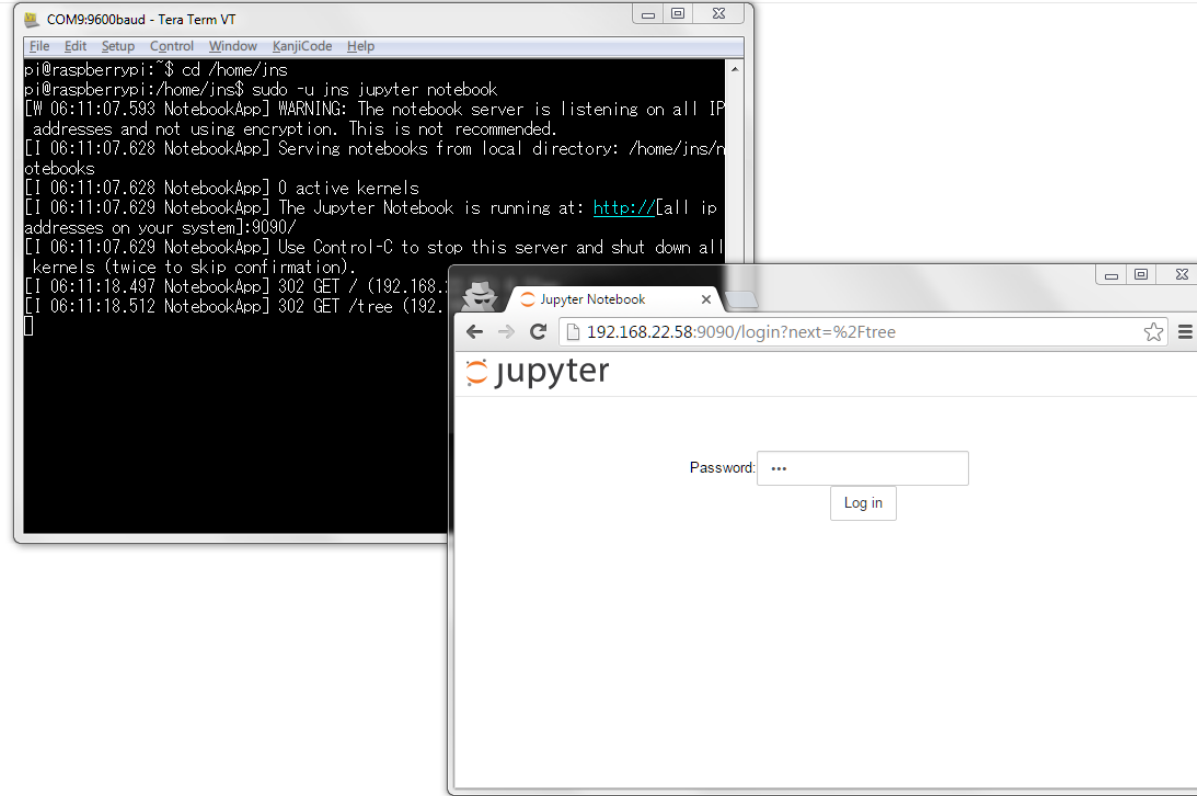
XenonPy (update)

<https://xenonpy.readthedocs.io/en/latest/installation.html>



- 4. Update XenonPyを実行
- 最後にactivate xenonpyと入力すれば、XenonPyが使える状態に

Tutorial : Jupyter notebookの起動

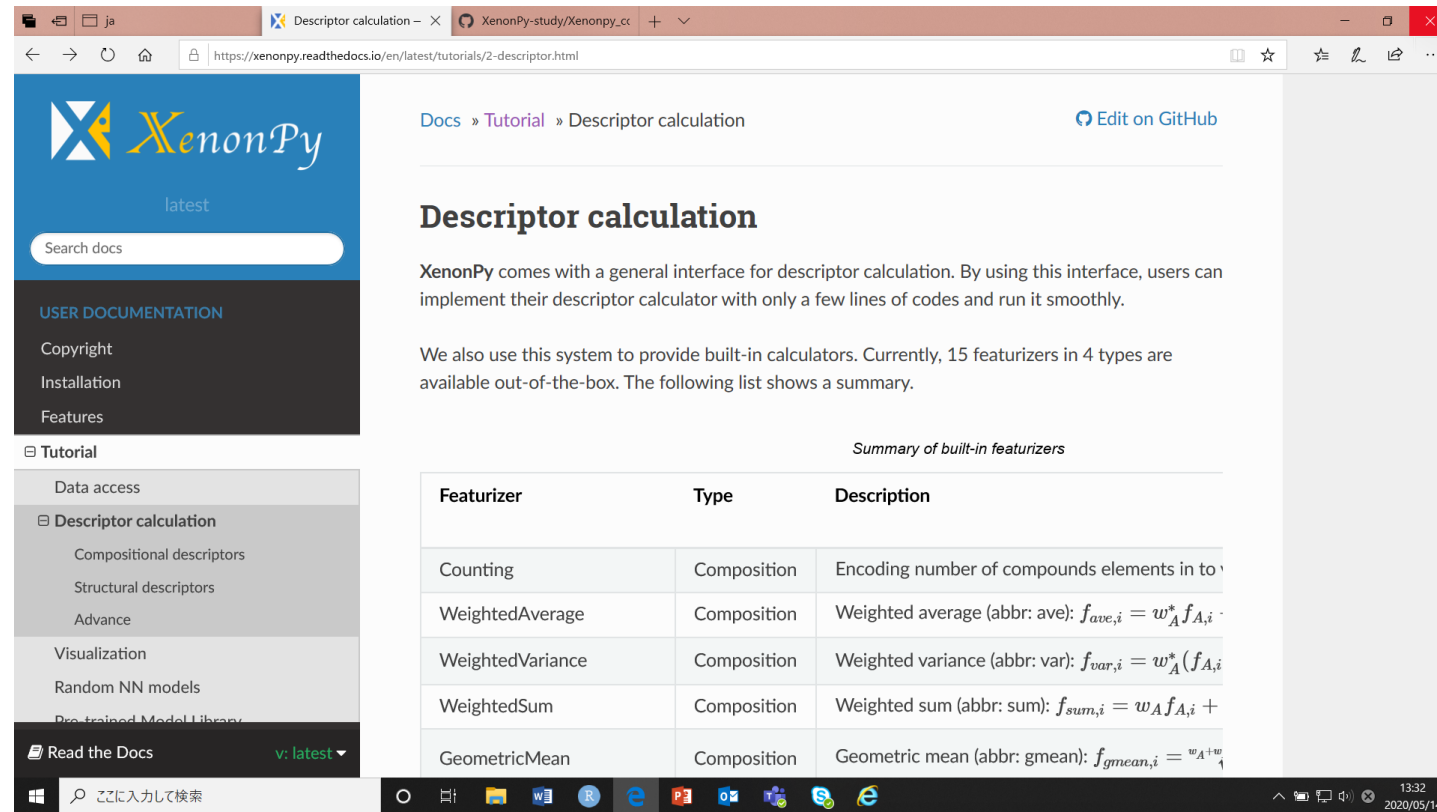


- jupyter notebook --no-browser で起動
- <http://localhost.8888/>（正確にはターミナルに表示）にアクセスすれば使用可能に
- tokenの入力はターミナル画面表示をコピーすればOK

Tutorial : XenonPyによる記述子の計算

<https://xenonpy.readthedocs.io/en/latest/tutorials/2-descriptor.html>

[https://github.com/M-asaki-K/XenonPy-study/blob/master/Xenonpy_compositiondescriptor_tutorial_and_test%20\(1\).ipynb](https://github.com/M-asaki-K/XenonPy-study/blob/master/Xenonpy_compositiondescriptor_tutorial_and_test%20(1).ipynb)



The screenshot shows a web browser displaying the XenonPy documentation page for "Descriptor calculation". The page has a sidebar on the left with navigation links: "USER DOCUMENTATION", "Copyright", "Installation", "Features", "Tutorial" (selected), "Data access", "Descriptor calculation" (expanded), "Compositional descriptors", "Structural descriptors", "Advance", "Visualization", "Random NN models", "Pre-trained Model Library", and "Read the Docs". The main content area is titled "Descriptor calculation" and includes a link to "Edit on GitHub". The text explains that XenonPy provides a general interface for descriptor calculation and lists 15 built-in featurizers. A table titled "Summary of built-in featurizers" is shown below.

Featurizer	Type	Description
Counting	Composition	Encoding number of compounds elements in to
WeightedAverage	Composition	Weighted average (abbr: ave): $f_{ave,i} = w_A^* f_{A,i}$
WeightedVariance	Composition	Weighted variance (abbr: var): $f_{var,i} = w_A^* (f_{A,i})$
WeightedSum	Composition	Weighted sum (abbr: sum): $f_{sum,i} = w_A f_{A,i} +$
GeometricMean	Composition	Geometric mean (abbr: gmean): $f_{gmean,i} = w_A^{1/w}$

- 基本的にはweb page上のコマンドをひたすら実行するだけ
- 上記だとつまらないので、別のデータセットでも出来るか試しました
- 私のgithubに掲載済み、バンドギャップ予測の結果は次回動画にて！