

# 化学のための Python プログラミング入門

黒崎宏太  
医療分子解析学 博士課程

# GUIとCUI

---

- GUI (Graphical User Interface)  
人間の直感的な操作とPCの動きが連動している
- CUI (Character-based User Interface)  
コマンドを打ってPCを操作していく



# CUIに慣れよう！

---

- CUIに慣れることのメリット
  1. ネットで検索して出てくる情報がほとんどCUI
  2. 煩わしいクリック戦争からの脱却
  3. 自動化してコーヒーブレイク

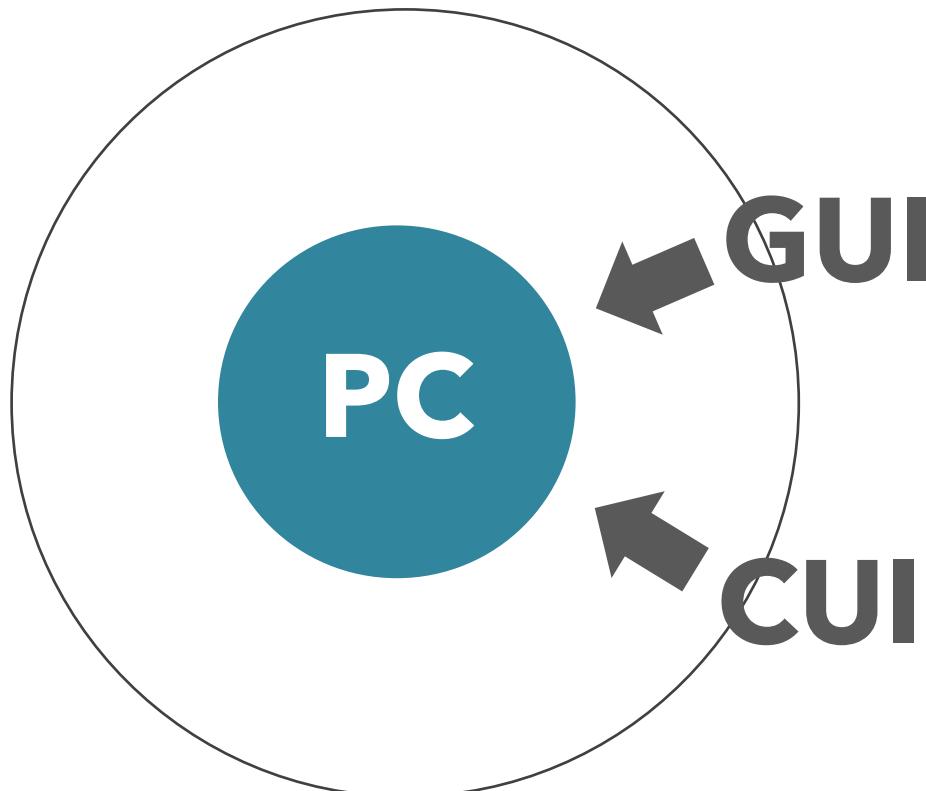


# コマンドとプログラムの違い

---

- ・コマンド
  - ・OSを操作するために出すCUIの指令
    - ・クリックしていたものを文字指令に置き換えたもの
    - ・例) シェルスクリプト
- ・プログラム
  - ・予め仕組んでおいた計算指令
    - ・例) Python, Java, C++, R, etc...

# PCをターミナルから動かすイメージ



# プログラミングやるならMac?

---

- Macの利点・欠点

- 利点) GUIとCUIが連動していて直感的
- 利点) Linuxと似ている
- 欠点) ユーザが少ない

- Windowsの利点・欠点

- 利点) CUIがさっぱりわからなくとも操作できる
- 欠点) コマンドが独自的すぎて情報が少ない

# WindowsでLinuxコマンドを使う

---

- WSL (Windows subsystem for Linux)
  - WindowsからLinuxのターミナル操作ができる！

# WSLの導入

---

1. 「Windows の機能の有効化または無効」から「Linux 用 Windows サブシステム」を有効化
2. 「Windows の機能の有効化または無効」から「仮想マシンプラットフォーム」を有効化
3. 管理者権限で Windows Power Shell を起動  
以下のコマンドを打つ 成功すると以下のコメントが出る

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux,VirtualMachinePlatform
```

>> Linux 用 Windows サブシステムには、ディストリビューションがインストールされていません。  
ディストリビューションは Microsoft Store にアクセスしてインストールすることができます:  
<https://aka.ms/wslstore>

4. Microsoft store からUbuntu **20.04**をインストール

# 続き

---

5. 管理者権限でWindows Power Shellを起動  
以下のコマンドを打つ

```
wsl --set-version Ubuntu-20.04 2
```

6. Ubuntu 20.04を起動してユーザ名とパスワードを設定する

# ターミナルをかっこよくする

---

1. zshというシェルをインストールする

```
sudo apt-get install zsh
```

2. Zshをデフォルトシェルに変更する

```
which zsh
```

```
chsh -s /usr/bin/zsh
```

3. ターミナルを再起動

# 続き

---

4. <https://qiita.com/kinchiki/items/57e9391128d07819c321> を参照

5. 上の内容の要約. 以下のコマンドを打つ

```
git clone --recursive https://github.com/sorin-  
ionescu/prezto.git "${ZDOTDIR:-$HOME}/.zprezto"
```

6. 終わったら以下のコマンドを打つ

```
setopt EXTENDED_GLOB for rcf file in "${ZDOTDIR:-  
$HOME}/.zprezto/runcoms/^README.md(.N); do ln -s "$rcf"  
"${ZDOTDIR:-$HOME}/.${rcf:t}" done
```

7. 最後にターミナルの設ファイルを編集する

```
sudo vi ~/.zpreztorc
```

# 続き

---

8. 下のようになっている部分のテーマを書き換える

```
zstyle ':prezto:module:prompt' theme 'sorin'
```



```
zstyle ':prezto:module:prompt' theme 'pure'
```

9. シンタックスハイライティングとオートサジェストを有効化する

```
# Set the Prezto modules to load (browse modules).
# The order matters.
zstyle ':prezto:load' pmodule \
    'environment' \
    'terminal' \
    'editor' \
    'history' \
    'directory' \
    'spectrum' \
    'utility' \
    'completion' \
    'syntax-highlighting' \
    'autosuggestions' \
    'prompt' \
```

Two white double-headed arrows pointing between 'syntax-highlighting' and 'autosuggestions' in the code snippet.

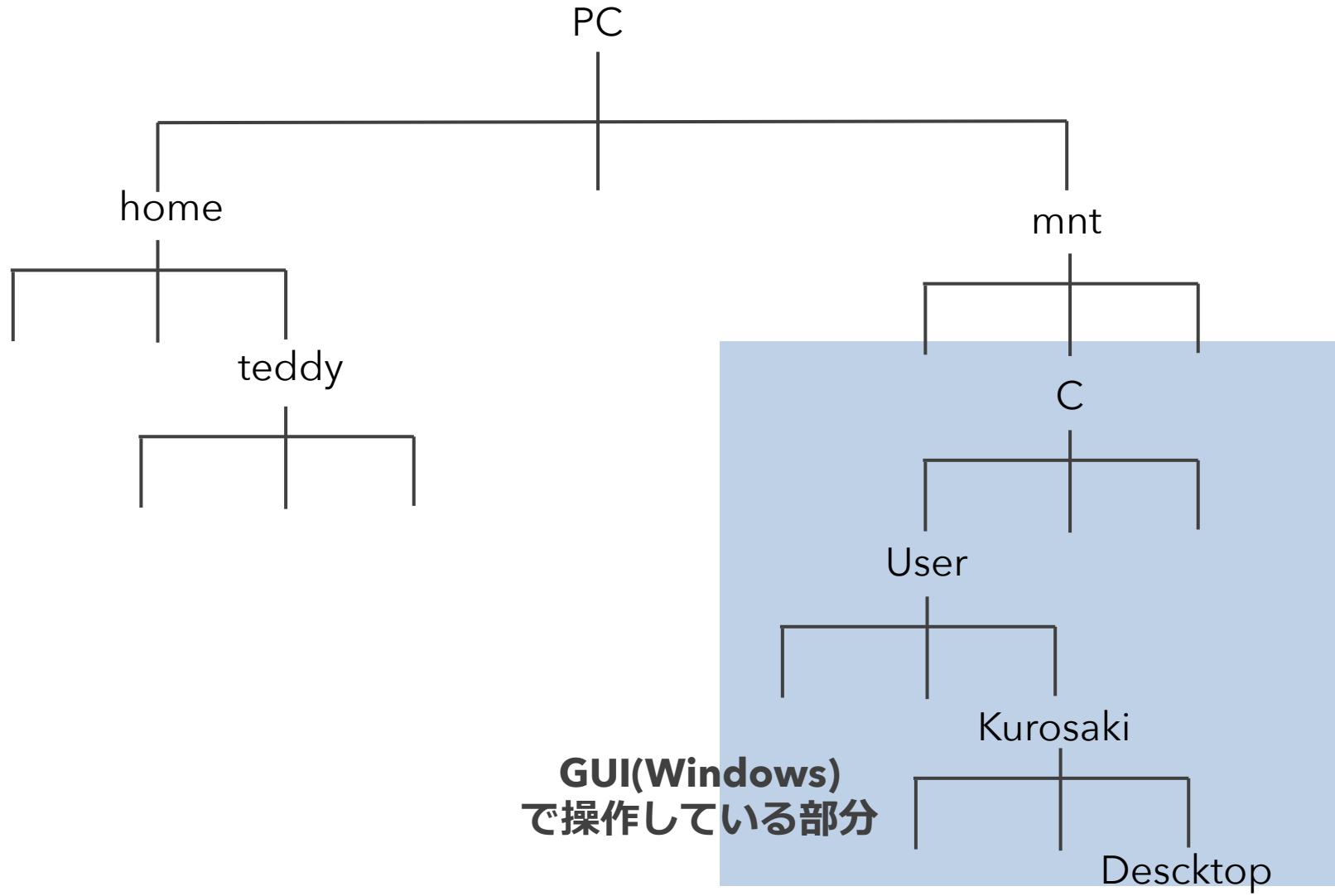
# 続き

---

10. 「Escape」 → :wq とうつ → 「Enter」で保存
11. ターミナルを再起動する

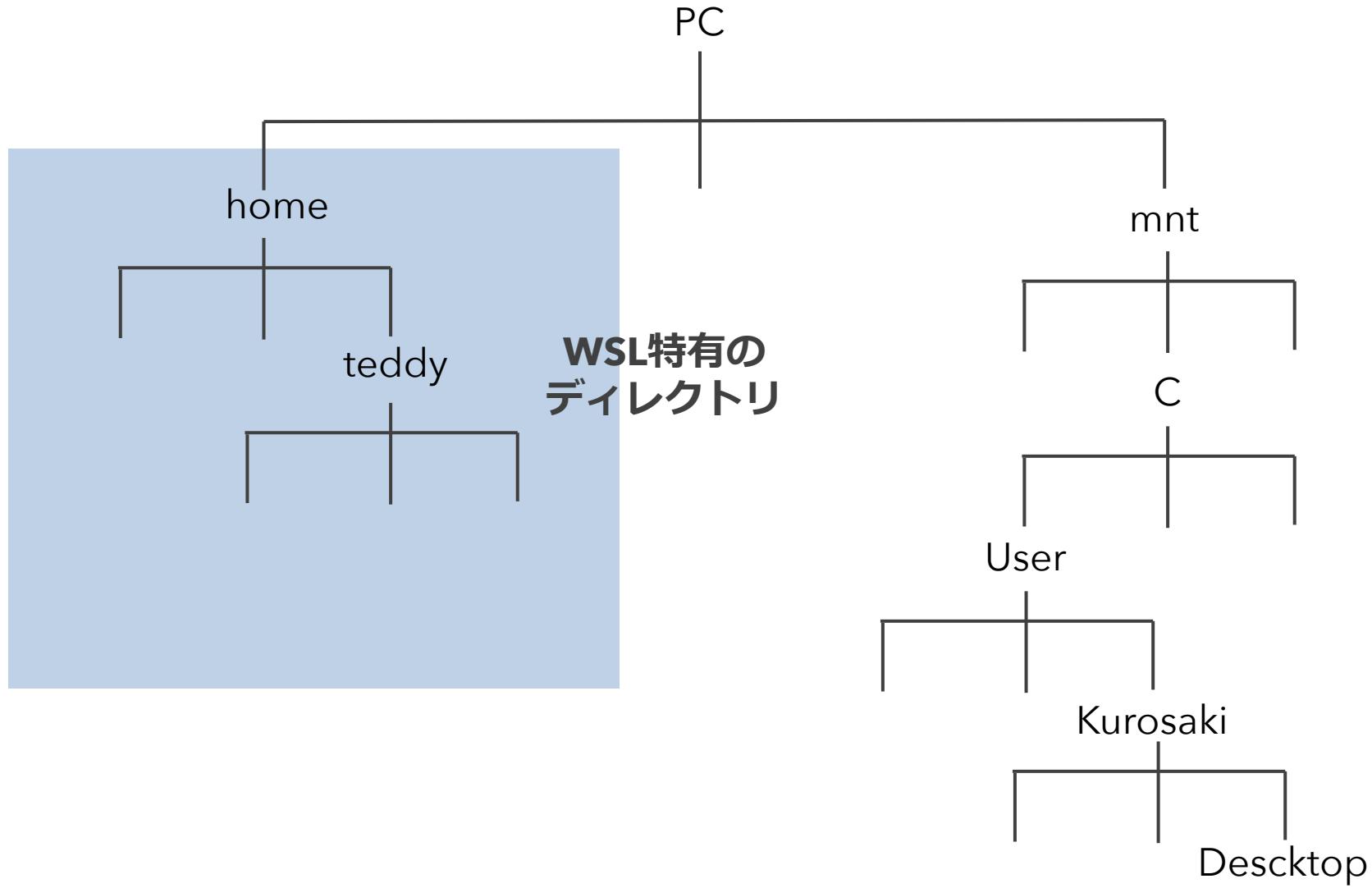
# ターミナルを少しいじる

- ディレクトリ構成

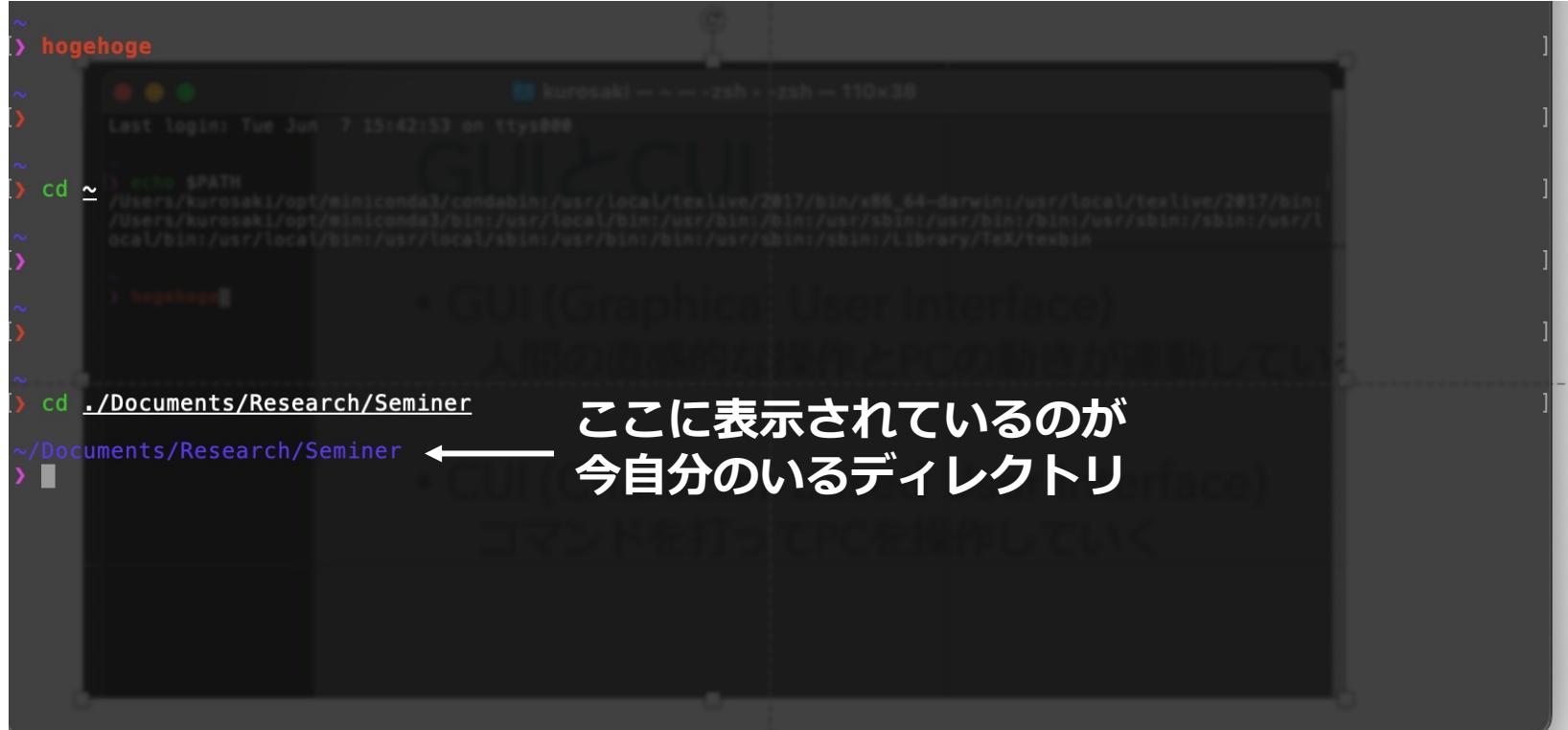


# ターミナルを少しいじる

- ディレクトリ構成



# ターミナルを少しいじる



A screenshot of a macOS terminal window titled "kurosa(k) --- zsh - zsh - 110x38". The window shows the following command history:

```
~ > hoge hoge
~ >
~ > cd ~
~ > echo $PATH
~ /Users/kurosa(k)/opt/miniconda3/condabin:/usr/local/textlive/2017/bin/x86_64-darwin:/usr/local/textlive/2017/bin:
~ /Users/kurosa(k)/opt/miniconda3/bin:/usr/local/bin:/usr/bin:/usr/sbin:/usr/bin:/usr/sbin:/usr/l
ocal/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/TeX/texbin
~ >
~ > hoge hoge
~ >
~ >
~ > cd ./Documents/Research/Seminer
~/Documents/Research/Seminer
```

A white arrow points from the text "今自分のいるディレクトリ" to the directory path in the terminal output. A white box highlights the path "Documents/Research/Seminer". To the right of the terminal window, Japanese text is overlaid:

ここに表示されているのが  
今自分のいるディレクトリ

# ターミナルを少しいじる

---

- ・ディレクトリを移動する(例)

```
cd ./Documents/
```

# ターミナルを少しいじる

---

- ・今いるディレクトリを確認する

```
pwd
```

# エディタをインストールする

- エディタって何?
  - プログラミングをしやすくする編集ソフト

The screenshot shows the Microsoft Code Editor interface. The title bar says "chapter1.ipynb — tutorials". The left sidebar shows a file tree with a folder "TUTORIALS" containing "notebook", ".gitignore", and "README.md". The notebook itself has two sections:

- RDkitで分子構造を描写する**: Contains a note about imports and a warning about using the import statement.
- ライブラリをインポートする**: Shows Python code for importing urllib.request, pandas, rdkit, and rdkit.Chem.

Below the notebook, there's a terminal window showing git push output:

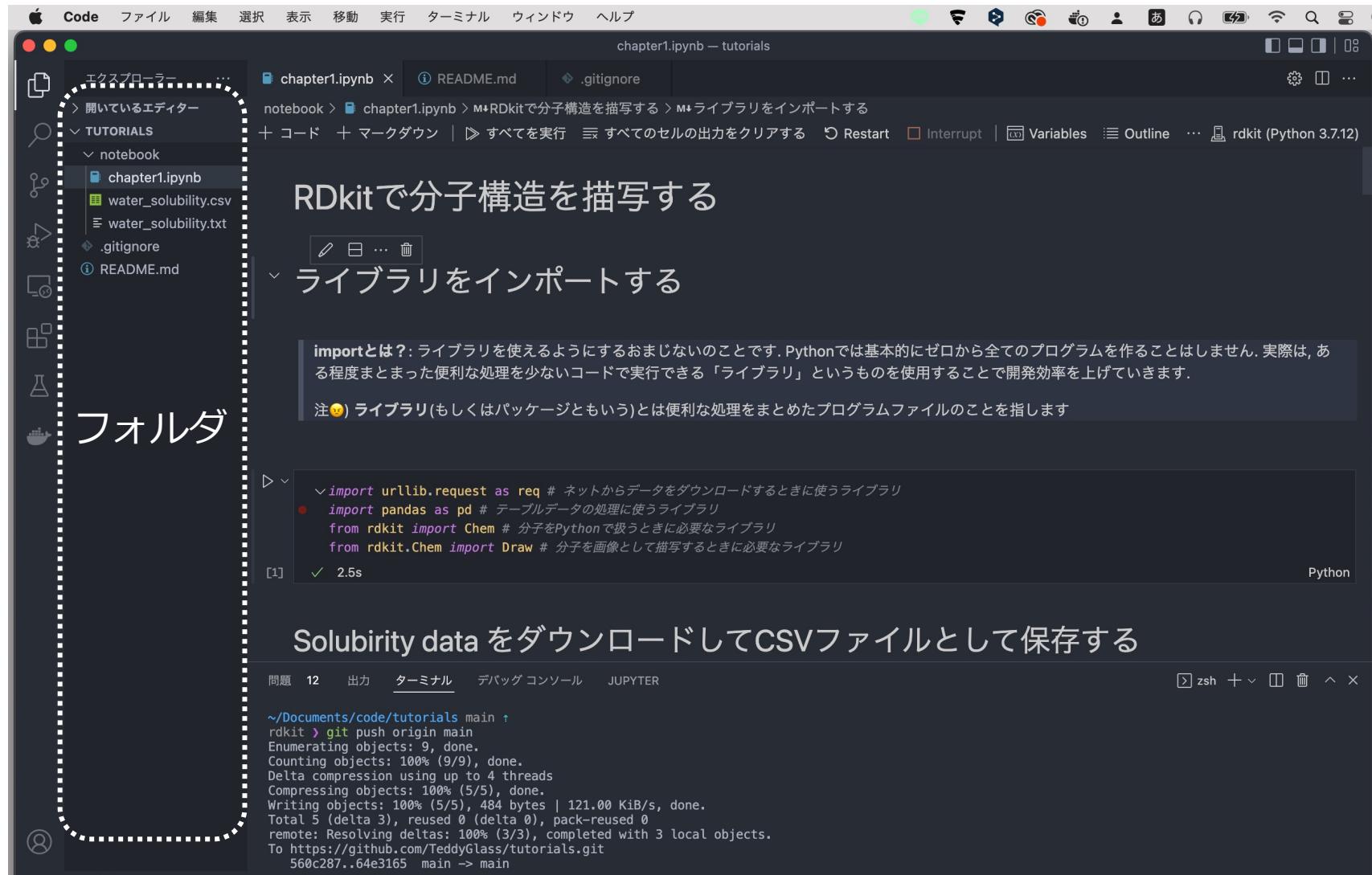
```
~/Documents/code/tutorials main ✘
rdkit > git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 484 bytes | 121.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/TeddyGlass/tutorials.git
  560c287..64e3165 main -> main
```

# VScodeをインストールする

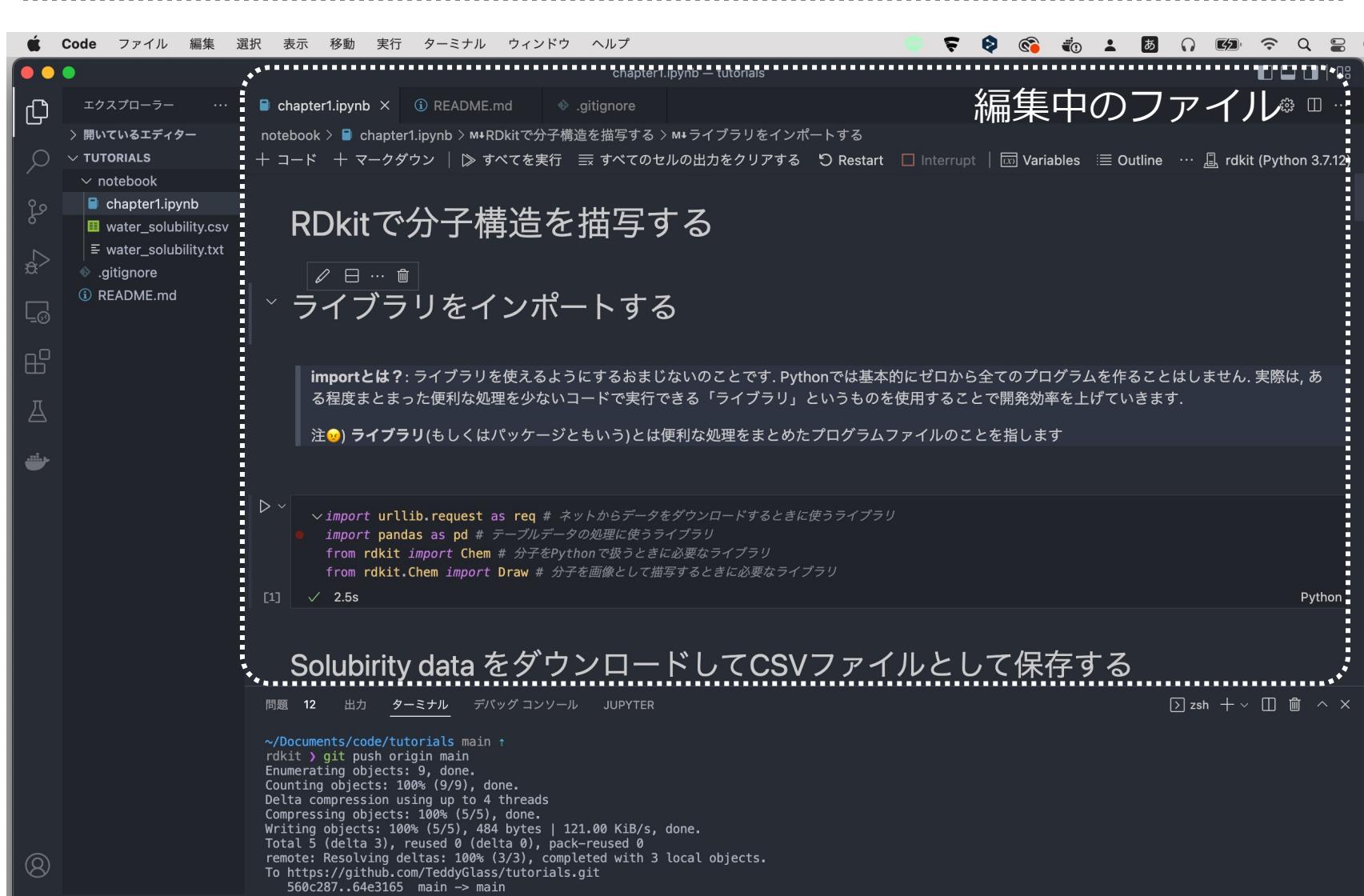
---

- ・[公式サイト](#)からダウンロードしてインストール

# Vscodeの説明



# Vscodeの説明



# Vscodeの説明

The screenshot shows the VS Code interface with the following components:

- Left Sidebar:** Shows the file tree with a folder named "TUTORIALS" containing "notebook" and "chapter1.ipynb".
- Top Bar:** Shows the menu bar (Code, ファイル, 編集, 選択, 表示, 移動, 実行, ターミナル, ウィンドウ, ヘルプ) and various status icons.
- Central Area:** A Jupyter notebook cell titled "RDkitで分子構造を描写する" (Describing molecular structures with RDkit). The cell content is: "ライブラリをインポートする".

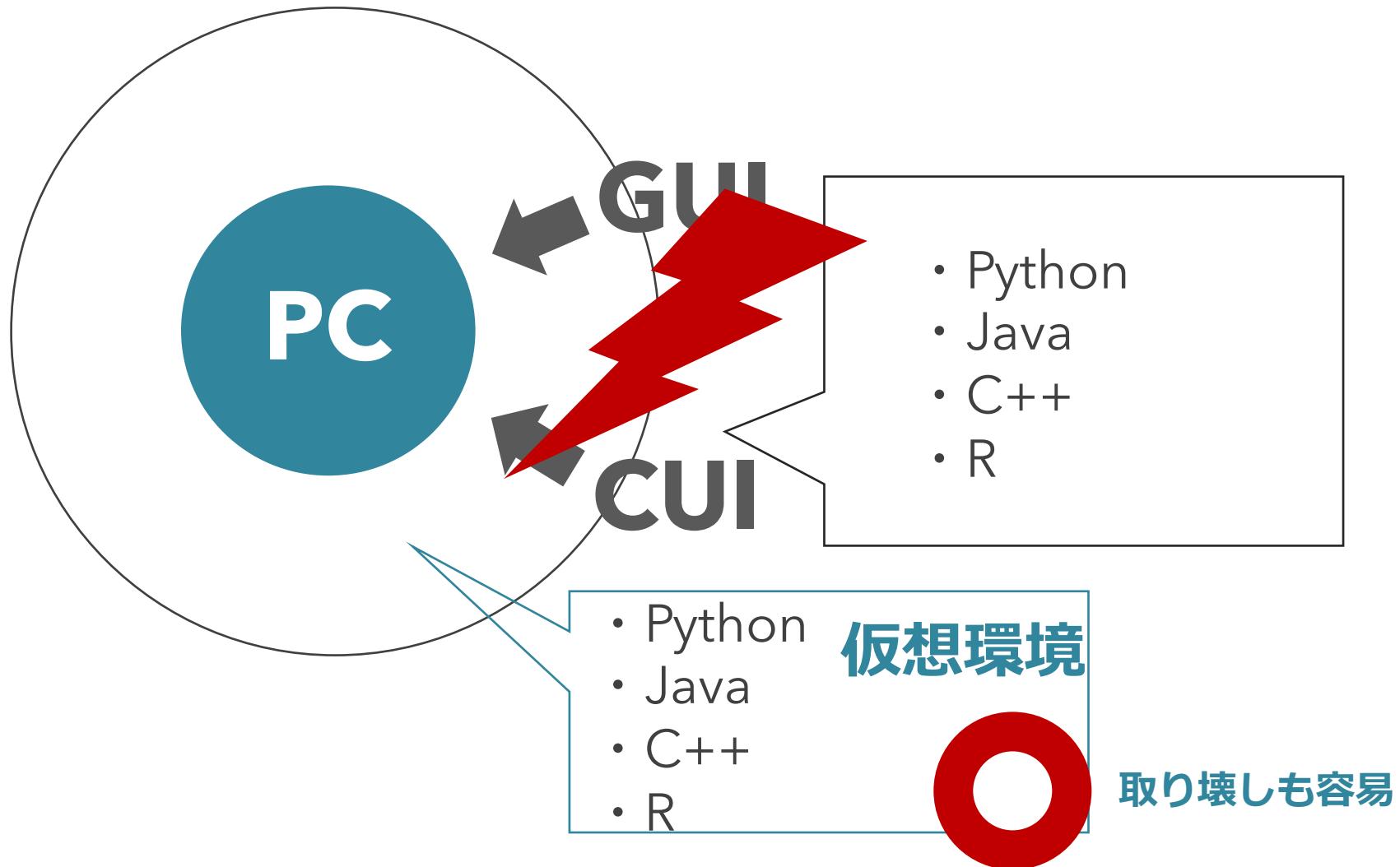
**Text:** importとは?: ライブラリを使えるようにするおまじないのことです。Pythonでは基本的にゼロから全てのプログラムを作ることはしません。実際は、ある程度まとまった便利な処理を少ないコードで実行できる「ライブラリ」というものを使用することで開発効率を上げていきます。

**Note:** ライブラリ(もしくはパッケージともいう)とは便利な処理をまとめたプログラムファイルのことを指します

```
import urllib.request as req # ネットからデータをダウンロードするときに使うライブラリ
import pandas as pd # テーブルデータの処理に使うライブラリ
from rdkit import Chem # 分子をPythonで扱うときに必要なライブラリ
from rdkit.Chem import Draw # 分子を画像として描写するときに必要なライブラリ
```
- Bottom Area:** A terminal window titled "ターミナル" (Terminal) showing command-line output:

```
~/Documents/code/tutorials main $ git push origin main
rdkit > git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 484 bytes | 121.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/TeddyGlass/tutorials.git
```

# 仮想環境とは？



# Pythonの環境を整える

---

- Minicondaをインストールする
  - MinicondaとはAnacondaの軽量版
  - [Miniconda公式サイト](#)にアクセスする
  - Linux版のminicondaのリンクをコピーする

cd ~  
でホームディレクトリに移る

```
wget  
https://docs.conda.io/en/latest/miniconda.html#:~:text=Python%203.9,-Miniconda3%20Linux%2064%2Dbit,-73.1%20MiB
```

bash Miniconda~~~.sh

# Pythonの環境を整える

---

- conda コマンドを使えるようにする
  - cd ~
  - code .
  - .zshrcファイルを編集する
  - export PATH="/home/[user\_name]/miniconda3/condabin:\$PATH"
  - source ~/.zshrc

# Minicondaの使い方: 基本

---

- 仮想環境を作る

```
conda create -n env_name python=3.xx
```

- 仮想環境に入る(アクティベートする)

```
conda activate env_name
```



The screenshot shows a terminal window with the following text:

```
~/Documents/Research/Seminer
[> source activate rdkit
~/Documents/Research/Seminer
rdkit > █ ← 仮想環境の名前が表示される
```

A white arrow points from the text "仮想環境の名前が表示される" (The name of the virtual environment is displayed) to the terminal prompt "rdkit > █".

# Minicondaの使い方: 基本

---

- 仮想環境を壊す

```
conda remove -n env_name --all
```

# ライブラリについて

---

- ライブラリとは？

便利な処理をひとまとめにしたプログラム集のこと

# ライブラリについて

---

- ライブラリとは？

便利な処理をひとまとめにしたプログラム集のこと

座標から直線の長さを算出するプログラム

Ax = 1  
Ay = 2

長いし面倒臭い

Bx = 3  
By = 4

$d^2 = (Bx - Ax)^2 + (By - By)^2$

# ライブラリについて

- ・ライブラリとは？

便利な処理をひとまとめにしたプログラム集のこと

座標から直線の長さを算出するプログラム

Ax = 1  
Ay = 2

Bx = 3  
By = 4

$$d^2 = (Bx - Ax)^2 + (By - By)^2$$

長いし面倒臭い



distance([Ax, Ay], [Bx, By])

•  
•  
•

# GitHubに登録しよう

---

# 仮想環境にライブラリをインストールする

---

- rdkit (化学情報学系ライブラリ) のインストール  
仮想環境内で以下のコマンドを打つ

```
conda install -c conda-forge rdkit
```

「-」はオプションという  
コマンド実行時の細かい設定を指定できる  
condaコマンドで-cを設定すると,  
ライブラリのダウンロード先を指定できる

# 仮想環境にライブラリをインストールする

## ・インストールコマンドの調べ方

1. conda install ~~でGoogle検索してみる
2. anaconda.orgのサイトに飛んでコマンドをコピペ

A screenshot of a Google search results page. The search query 'conda install numpy' is entered in the search bar. Below the search bar, there are links for 'All', 'Images', 'Videos', 'News', 'Shopping', and 'More'. The search results show approximately 5,600,000 results found in 0.41 seconds. The top result is a link to 'Numpy :: Anaconda.org'.

**Numpy :: Anaconda.org**  
To install this package with conda run: `conda install -c anaconda numpy`. Description. NumPy is the fundamental package needed for scientific computing with ...

**Numpy - :: Anaconda.org**  
To install this package with conda run one of the following: `conda install -c conda-forge numpy` `conda install -c conda-forge/label/cf202003 numpy`

**Installing NumPy**  
NumPy can be installed with `conda`, with `pip`, with a package manager on macOS and Linux, or from source. For more detailed instructions, consult our Python ...

**People also ask :**

- How install NumPy package in Anaconda?
- How do I download NumPy using conda?
- Does conda include NumPy?
- How do I install NumPy?

A screenshot of the NumPy page on anaconda.org. The URL in the browser is <https://docs.scipy.org/doc/numpy/reference/>. The page shows the last upload was 2 days and 1 minute ago.

**Installers**

To install this package with conda run:

```
conda install -c anaconda numpy
```

**Description**

NumPy is the fundamental package needed for scientific computing with Python.

# RDkitでPythonコードを書いてみる

---

1. Vscodeを起動する
2. 適当に.ipynbファイルを作成する
3. VscodeのPython extensionとJupyter extensionを有効化する
4. .ipynbを開いて「shift」+「Enter」を押すとカーネル選択を迫られる
5. さっき作った仮想環境を選択
6. 以下の文を打つ→「shift」+「Enter」

```
from rdkit import Chem  
  
mol = Chem.MolFromSmiles("CCC")  
mol
```