

演習I・第13回

システム情報学部 中島涼輔
@K502
2025/7/11

長いプログラムコードのエラー修正のコツ

- ・ プログラムコードを最後まで書いて、エラーが出たらじっとプログラムコードを眺める



```
▶ a = 1 + 2
  b = 3 + 4
  c = 5 + 6
  total = a + b + c
print(total)

↳ File "/tmp/ipython-input-14-2323259924.py", line 4
      total = a + b + c
                  ^
IndentationError: unexpected indent
```

- ・ 作業を分割して、途中途中でうまく行っているか確認しながらプログラムコードを書き進める(困難は分割せよ)

– print 文

– コメントアウト

– ?マーク(Google Colab. 限定)

```
▶ a = 1 + 2 # ← ここまでOK
print(a)

↳ 3
```

```
▶ a = 1 + 2 # ← ここまでOK
# b = 3 + 4
# c = 5 + 6
# total = a + b + c
# print(total)

print(a)

↳ 3
```

```
▶ a = 1 + 2
# b = 3 + 4
# c = 5 + 6
# total = a + b + c
# print(total)

a?

↳
```

```
ヘルプ ×

Type: int
String form: 3
Docstring:
int([x]) -> integer
int(x, base=10) -> integer
```

スケジュール

第1回:ガイダンス(4/11)	第8回:文字列(5/30)
第2回:システム設定(4/18)	第9回:リスト(6/13)
第3回:Python概要(4/25)	第10回:関数(6/20)
第4回:変数・型(5/2)	第11回:高階関数(6/27)
第5回:条件分岐(5/9)	第12回:クラス(7/4)
第6回:繰り返し(5/16)	第13回:継承・UNIX(7/11)
第7回:オブジェクト(5/23)	第14回:ライブラリ(7/18)
	第15回:まとめ(7/25)

- シラバスから若干変更

今日の内容

- ・継承(教科書 6-3)
- ・UNIXの初步

中島が参考にした本



クラスの復習



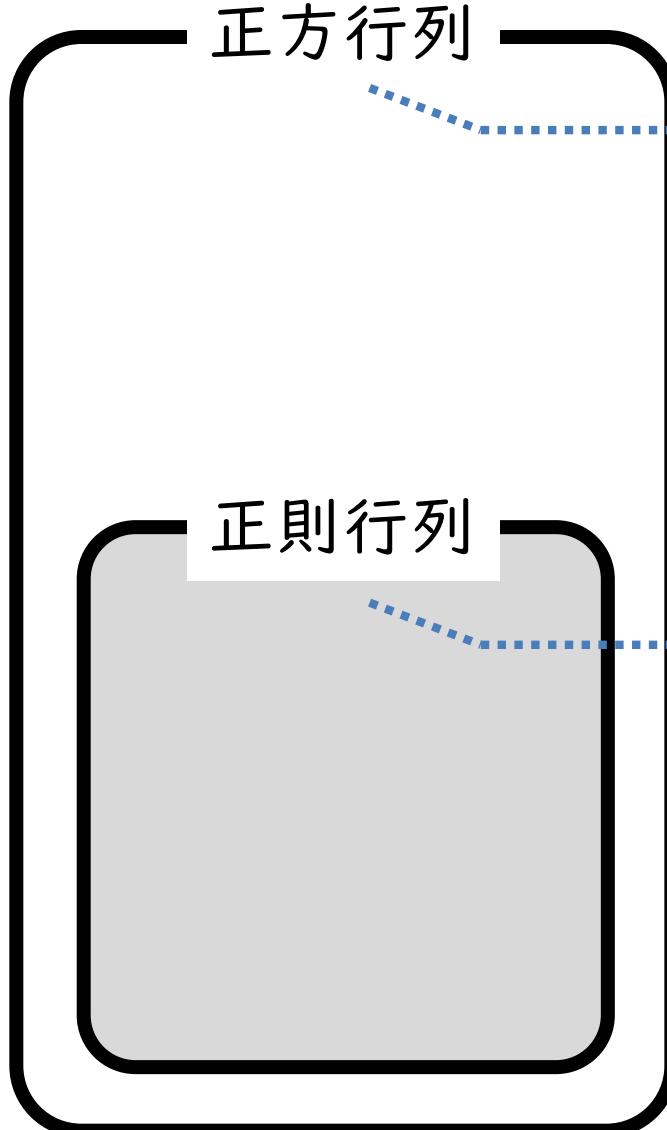
```
class Matrix2by2:  
    # 初期化メソッド (インスタンス変数を設定)  
    def __init__(self, matrix):  
        self.a = matrix[0][0]  
        self.b = matrix[0][1]  
        self.c = matrix[1][0]  
        self.d = matrix[1][1]  
  
    # メソッド  
    def print_matrix(self):  
        print(f'{self.a} {self.b}')  
        print(f'{self.c} {self.d}')  
  
    def trace(self):  
        return self.a + self.d  
  
    def determinant(self):  
        return self.a * self.d - self.b * self.c  
  
# インスタンス生成  
mat1 = Matrix2by2([[1, 2], [3, 4]])  
# インスタンス変数の参照  
print(mat1.a)  
# メソッドの呼び出し  
mat1.print_matrix()
```



1
1 2
3 4

情報(インスタンス変数)
と機能(メソッド)を一つの
オブジェクトにまとめる

継承とは



- 成分
- ✓ ト雷斯
- ✓ 転置
- ✓ ランク
- ✓ 行列式

Matrix2by2 クラス

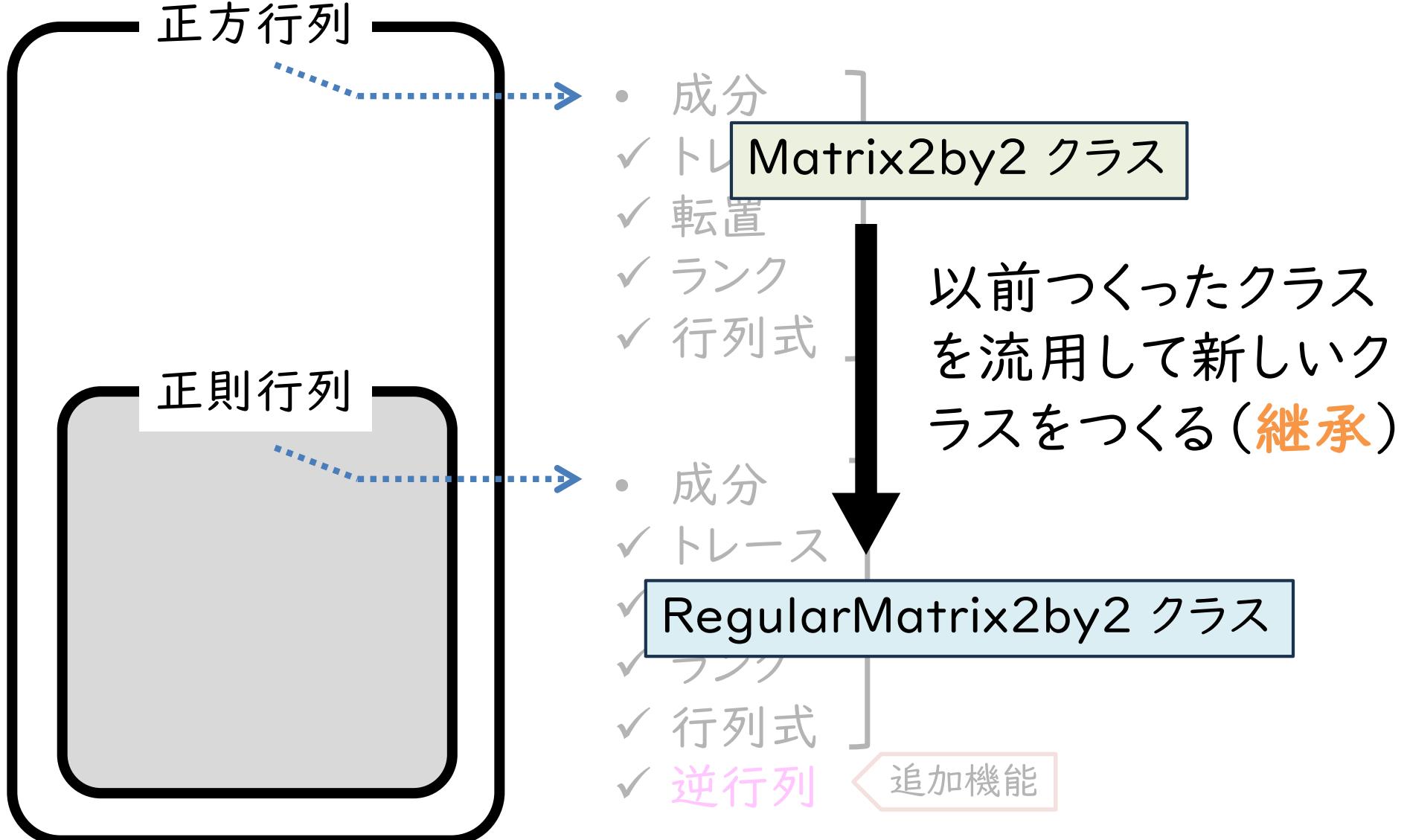
- 成分
- ✓ トレス
- ✓ 転置
- ✓ ランク
- ✓ 行列式
- ✓ 逆行列

RegularMatrix2by2
クラス

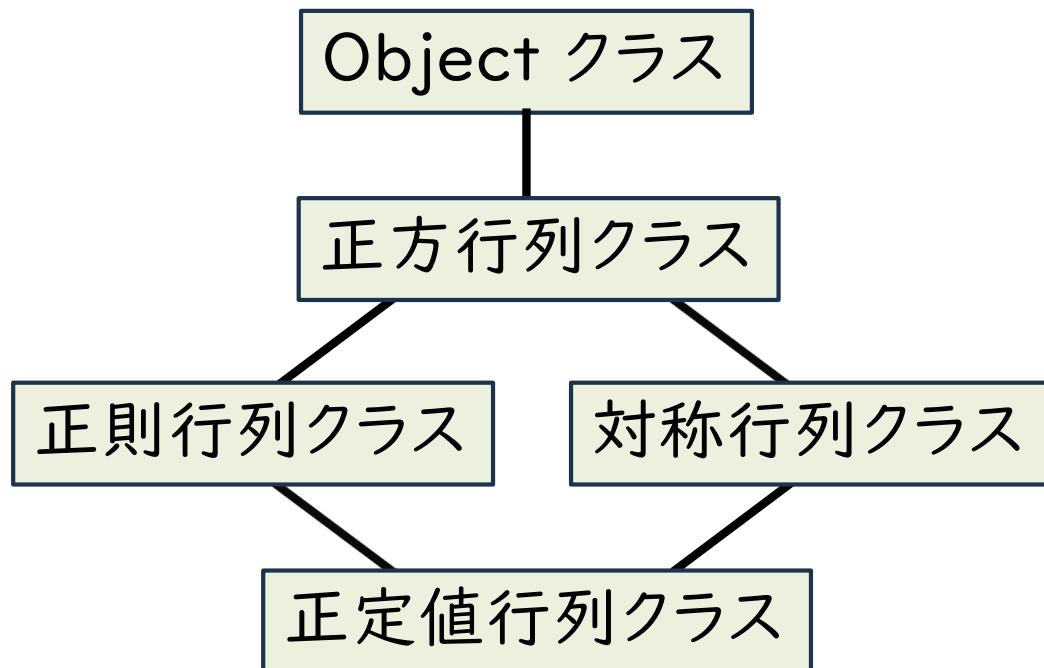
同じ

追加機能

継承とは



継承とは



- 前回授業の方法でつ
くったクラスは Object
クラスを継承
- 継承は階層的に行える
 - [↑] スーパークラス
(親クラス、基底クラス)
 - [↓] サブクラス (子クラ
ス、派生クラス)
- 複数のクラスを継承で
きる (多重継承)

継承の仕方(1)

```
▶ class Matrix2by2:  
    # 初期化メソッド（インスタンス変数を設定）  
    def __init__(self, matrix):  
        self.a = matrix[0][0]  
        self.b = matrix[0][1]  
        self.c = matrix[1][0]  
        self.d = matrix[1][1]  
  
    # メソッド  
    def print_matrix(self):  
        print(f'{self.a} {self.b}')  
        print(f'{self.c} {self.d}')  
  
    def trace(self):  
        return self.a + self.d  
  
    def determinant(self):  
        return self.a * self.d - self.b * self.c  
  
class RegularMatrix2by2(Matrix2by2):  
    pass # 何も処理をしないという文（とりあえず）  
  
# インスタンス生成  
reg_mat_1 = RegularMatrix2by2([[1, 2], [3, 4]])  
# インスタンス変数の参照  
print(reg_mat_1.a)  
# メソッドの呼び出し  
reg_mat_1.print_matrix()  
  
→ 1  
 1 2  
 3 4
```

- クラス定義のときに、クラス名の後ろに「**(スーパークラスの名前)**」とすると継承できる
- 継承すると、スーパークラスの初期化メソッドやメソッドがそのまま引き継がれる

継承

初期化メソッドが引き継がれている

メソッドも引き継がれている

継承の仕方(2) 機能の追加

```
▶ class Matrix2by2:  
    # 初期化メソッド (インスタンス変数を設定)  
    def __init__(self, matrix):  
        self.a = matrix[0][0]  
        self.b = matrix[0][1]  
        self.c = matrix[1][0]  
        self.d = matrix[1][1]  
  
    # メソッド  
    def print_matrix(self):  
        print(f'{self.a} {self.b}')  
        print(f'{self.c} {self.d}')  
  
    def trace(self):  
        return self.a + self.d  
  
    def determinant(self):  
        return self.a * self.d - self.b * self.c  
  
class RegularMatrix2by2(Matrix2by2):  
    def inverse(self):  
        det = self.determinant()  
        inv_mat = [[self.d/det, -self.b/det], [-self.c/det, self.a/det]]  
        return RegularMatrix2by2(inv_mat)  
  
reg_mat_1 = RegularMatrix2by2([[1, 2], [3, 4]])  
reg_mat_1.inverse().print_matrix()  
  
mat1 = Matrix2by2([[1, 2], [3, 4]])  
mat1.inverse().print_matrix()
```

- サブクラスの定義に追加したい機能(メソッド)を記述するだけ

```
→ -2.0 1.0  
1.5 -0.5  
--  
AttributeError Traceback (most recent call last)  
/tmp/ipython-input-19-811352951.py in <cell line: 0>()
```

スーパークラスは inverse メソッドをもっていないので、エラーになる

練習

のために準備したけど時間が足りないのでやめた

Matrix2by2クラスを継承して、対称行列のクラスをつくり、
2つの実固有値を計算するメソッドを追加してください（重解
は気にしなくて良い） モジュール(import文)の使用は禁止。 $\sqrt{a} \rightarrow (a)^{**0.5}$

```
class Matrix2by2:  
    def __init__(self, matrix):  
        self.a = matrix[0][0]  
        self.b = matrix[0][1]  
        self.c = matrix[1][0]  
        self.d = matrix[1][1]  
  
    def print_matrix(self):  
        print(f'{self.a} {self.b}')  
        print(f'{self.c} {self.d}')  
  
    def trace(self):  
        return self.a + self.d  
  
    def determinant(self):  
        return self.a * self.d - self.b * self.c
```

答えの例(他でもOK)

```
class SymmetricMatrix2by2(Matrix2by2):
    def eigenvalues(self):
        tr = self.trace()
        det = self.determinant()
        lambda1 = (tr + (tr**2 - 4*det)**0.5)/2
        lambda2 = (tr - (tr**2 - 4*det)**0.5)/2
        return lambda1, lambda2

sym_mat_1 = SymmetricMatrix2by2([[1, 2], [2, 1]])
print(sym_mat_1.eigenvalues())
```

⇒ (3.0, -1.0)

$$\det(A - \lambda E) = \lambda^2 - \text{tr}(A)\lambda + \det(A) = 0$$

(注) プログラムで固有値を求めるとき、実際は数値計算に特化した方法で解きます(べき乗法、QR法など)

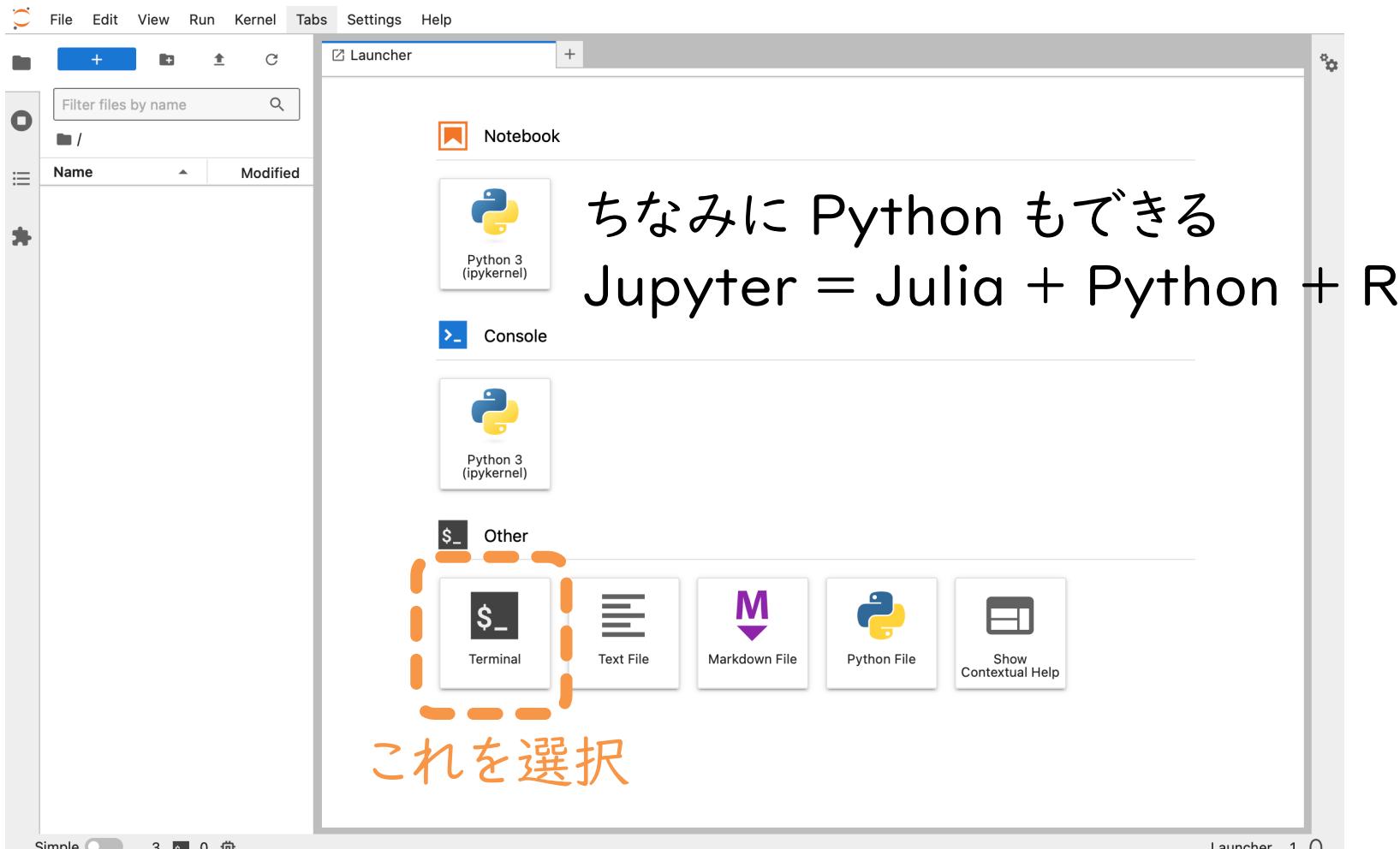
出席テスト

- BEEF+に載せています
 - Google Colaboratory にコピペして、試しながら解答しても良いです（ネット検索も可）
 - 全問正解するまで受験してください
-

出席テストが終わった人（今日の課題は12:30に公開）

- BEEF+ のマニュアルなどを見ながら、教育用計算機システムに接続しておいてください
- Macの場合、Safari は非推奨です（マニュアルの7月10日改訂版を参照）

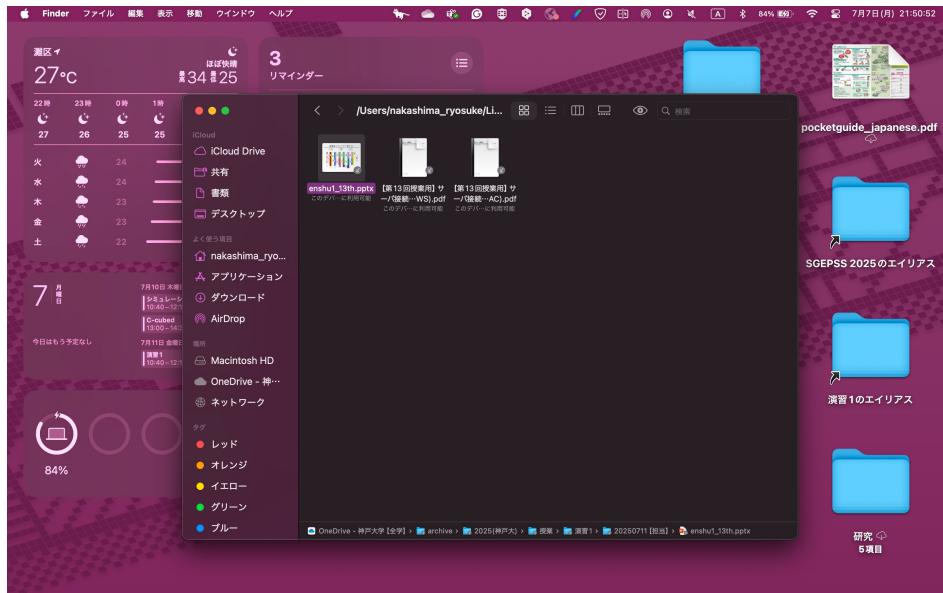
JupyterHub に接続できましたか？



接続できない人、時間がかかりそうな人は <https://webvm.io> にアクセス

GUI と CLI

グラフィカル ユーザ インターフェース (GUI)



マウスやトラックパッドを使って、アイコンやボタンなど視覚的な要素をクリックし、ファイルやソフトウェアを操作する

コマンドライン インターフェース (CLI)

```
Last login: Mon Jul 7 21:57:32 on ttys001
=====
nakashima_ryosuke@MacBook-Pro ~ % ls
Applications           Library
Desktop                Movies
Documents              Music
Downloads              OneDrive - 神戸大学【全学】
git_kobe               Pictures
git_local              Public
git_public
=====
nakashima_ryosuke@MacBook-Pro ~ %
```

文字によるコマンドで、ファイルやソフトウェアを操作する

計算機サーバー、
Webサーバーなどは
CLIが普通

OSとは

- オペレーティング システム(OS)
 - Windows、macOS、UNIX、Linux、Android、iOSなど
 - CPU、メモリ、ストレージ、入出力装置などのハードウェアを制御し、アプリケーションソフトウェアを実行する環境を整える
- Unix系OS
 - マルチユーザー、マルチタスクを前提に設計
 - UNIXから派生 → macOS、Linux(Ubuntu、Debian、Fedora、CentOSなど)など
- 教育用計算機システムは Rocky Linux

カーネルとシェル

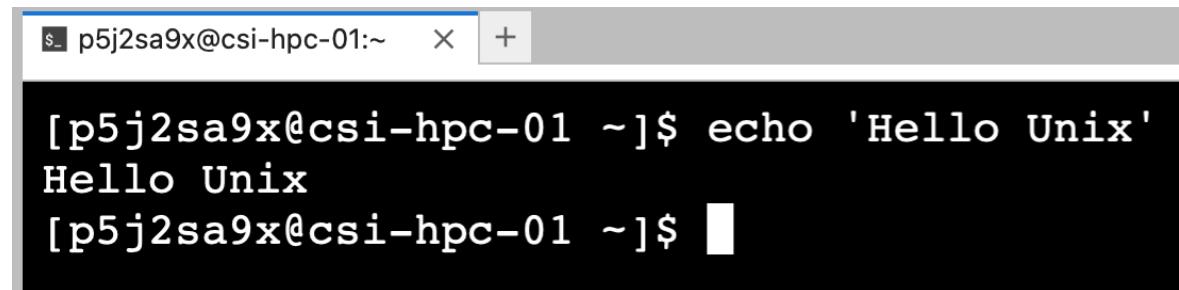
- ・ カーネル(核)：OSの中心となる部分
 - ・ シェル(殻)：ユーザーが入力したコマンドをカーネルに伝える(bash, zshなど)
- ✓ 今日は、このUNIXコマンドの初步を勉強する
- ・ プロンプト：コマンドを入力するところ
 - ・ コマンドの中斷・強制終了「Ctrl + C」
 - コマンド間違い、応答が遅すぎたら

A screenshot of a terminal window. The title bar shows the session details: \$ p5j2sa9x@csi-hpc-01:~. The window contains the following text:
[p5j2sa9x@csi-hpc-01 ~]\$ sleep 10000
^C
[p5j2sa9x@csi-hpc-01 ~]\$ █

(注) WebVM で Ctrl + C
も効かない場合はブラウザを
再読み込みしてください

echo (= Python の print 文)

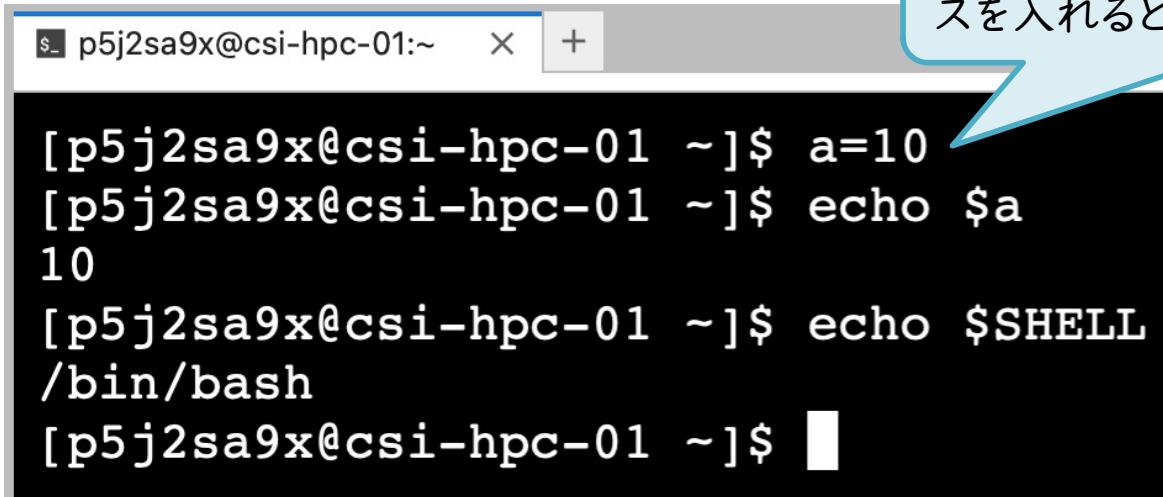
- 「'」や大文字アルファベットが入力できないときは
2回連打してみてください



```
p5j2sa9x@csi-hpc-01:~ $ echo 'Hello Unix'  
Hello Unix  
[p5j2sa9x@csi-hpc-01 ~]$
```

- 変数の参照: 「\$ + 文字」(「\$ + 大文字」は環境
変数の参照)

「=」の前後や「\$」の後ろにスペースを入れるとエラーになるので注意



```
p5j2sa9x@csi-hpc-01:~ $ a=10  
[p5j2sa9x@csi-hpc-01 ~]$ echo $a  
10  
[p5j2sa9x@csi-hpc-01 ~]$ echo $SHELL  
/bin/bash  
[p5j2sa9x@csi-hpc-01 ~]$
```

Python もできます

- **python** : Python の対話モード(インタラクティブシェル)を起動する

```
p5j2sa9x@csi-hpc-01:~ +  
[p5j2sa9x@csi-hpc-01 ~]$ python  
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:2  
7:36) [GCC 11.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information  
.  
>>> 1+2  
3  
>>>
```

(注) WebVM は Python2系しか使えなさそう?

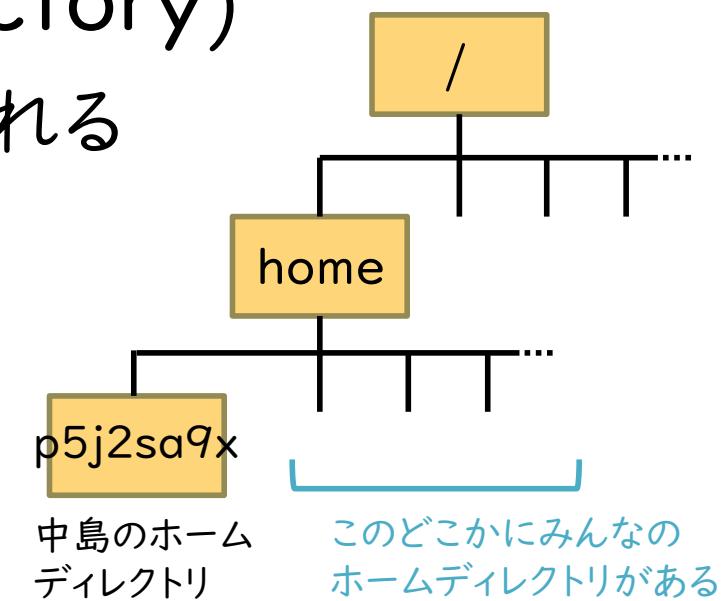
- **exit()** で終了

```
>>> exit()  
[p5j2sa9x@csi-hpc-01 ~]$
```

ファイルのツリー構造

- ディレクトリ (GUIでいうところのフォルダ)
- pwd** (print working directory)
 - 今自分がいる場所を教えてくれる

```
$ p5j2sa9x@csi-hpc-01:~ × + 
[p5j2sa9x@csi-hpc-01 ~]$ pwd
/home/p5j2sa9x
[p5j2sa9x@csi-hpc-01 ~]$
```

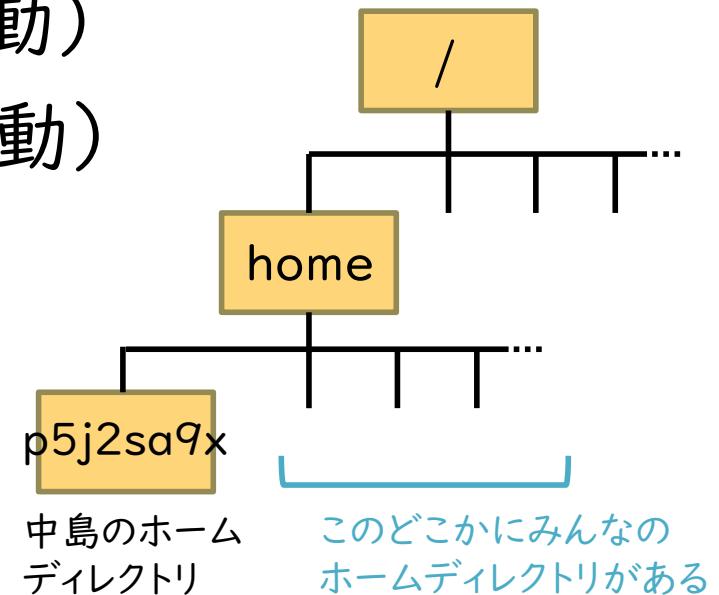


- カレント ディレクトリ ('.') : 今いる場所
- ルート ディレクトリ (先頭の '/')
- ホーム ディレクトリ ('~') : ログインしたときに、最初にいる場所。基本的にこの中で作業

ディレクトリの移動

- **cd** (change directory)
 - cd . (カレントディレクトリに移動(何もしない))
 - cd / (ルートディレクトリに移動)
 - cd ~ (ホームディレクトリに移動)

```
$ p5j2sa9x@csi-hpc-01:~ x +  
[p5j2sa9x@csi-hpc-01 ~]$ cd .  
[p5j2sa9x@csi-hpc-01 ~]$ pwd  
/home/p5j2sa9x  
[p5j2sa9x@csi-hpc-01 ~]$ cd /  
[p5j2sa9x@csi-hpc-01 /]$ pwd  
/  
[p5j2sa9x@csi-hpc-01 /]$ cd ~  
[p5j2sa9x@csi-hpc-01 ~]$ pwd  
/home/p5j2sa9x  
[p5j2sa9x@csi-hpc-01 ~]$
```



絶対パス/相対パス

- **絶対パス**: ルートディレクトリを起点として、ファイルやディレクトリの位置を指定する方法

– 例) `cd /bin`

ルートディレクトリの直下にあるbinディレクトリに移動

- **相対パス**: カレントディレクトリを起点として、ファイルやディレクトリの位置を指定する方法

– 例) `cd ../../..../bin`

カレントディレクトリの**親ディレクトリ**（「..」、1つ上のディレクトリのこと）のさらに親ディレクトリの直下にあるbinディレクトリに移動（カレントディレクトリがホームディレクトリの場合）

– 最初の「./」は省略できる

ディレクトリの中身を見る

- **ls** (list)
 - ls . (カレントディレクトリの中身を見る、「.」は省略できる)
 - ls / (ルートディレクトリの中身を見る)
 - ls ~ (ホームディレクトリの中身を見る)
 - ls .. (親ディレクトリの中身を見る)
 - 中身を見るディレクトリの指定に絶対/相対パスが使用可

```
[p5j2sa9x@csi-hpc-01 ~]$ ls .
[p5j2sa9x@csi-hpc-01 ~]$ ls /
afs    data   home2  manual   proc    srv      usr
ap     dev    home3  media    root    swapfile  var
bin    etc    lib     mnt     run    sys
boot   home   lib64   opt     sbin    tmp
[p5j2sa9x@csi-hpc-01 ~]$ ls ~
[p5j2sa9x@csi-hpc-01 ~]$ ls ..
1805013t  2305084t  241x026x  2495011t  2545159t
1825003t  2305098t  241x035x  2495025t  254x013x
```

コマンドのオプション

- オプション：「ハイフン + 文字」で動作を指定
 - **ls -l** (詳しい情報付きの ls)

```
[p5j2sa9x@csi-hpc-01 ~]$ ls .. -l
合計 48
drwx----- 2 1805013t student    96  4月   3 10:25 1805013t
drwx----- 2 1825003t student    76  4月   3 10:25 1825003t
drwx----- 2 1855055t student    96  4月   3 10:25 1855055t
drwx----- 2 1925062t student    96  4月   3 10:25 1925062t
```

- **ls -a** (隠しファイルも表示できる ls)

```
[p5j2sa9x@csi-hpc-01 ~]$ ls ~ -a
.                                .cache          .local
..                               .config          .npm
.Xauthority          .emacs           .python_history
.bash_history        .ipynb_checkpoints .ssh
.bash_logout         .ipython          .viminfo
.bash_profile        .jupyter         .virtual_documents
.bashrc              .lesshst
```

隠しファイルとは、
「.」で始まるファイル
のこととて、主に設定
ファイルです

パーミッション(ファイル権限)

- パーミッション: ファイルやディレクトリの読み(r)、書き(w)、実行(x)の権限
 - ls -l で確認できる

他の人のホームディレクトリは見ることができない

```
[p5j2sa9x@csi-hpc-01 ~]$ ls -la
合計 276
drwxr-xr-x  11 p5j2sa9x staff  4096 7月   8 15:36 .
drwxr-xr-x  735 root     root 225280 7月   3 11:44 ..
-rw-----  1 p5j2sa9x staff    100 7月   8 01:08 .Xauthority
-rw-----  1 p5j2sa9x staff   1637 7月   8 03:21 .bash_history
-rw-r--r--  1 p5j2sa9x staff     18 4月   30 2024 .bash_logout
-rw-r--r--  1 p5j2sa9x staff    248 2月   18 19:41 .bash_profile
-rw-r--r--  1 p5j2sa9x staff    629 3月   28 15:57 .bashrc
drwxr-xr-x  3 p5j2sa9x staff     26 7月   8 01:09 .cache
```

所有者・グループ・他の人

- chmod (change mode): パーミッション変更
 - 使い方は省略

ディレクトリの作成

- これから先の演習は、自分のホームディレクトリに戻ってから行ってください `cd ~`
- **mkdir** (make directory)
 - 例) `mkdir test`

「./」が省略された書き方

カレントディレクトリにtestディレクトリを作成

– 例) `mkdir ~/test/test2`

ホームディレクトリの直下のtestディレクトリにtest2ディレクトリを作成

```
$ p5j2sa9x@csi-hpc-01:~ × +  
[p5j2sa9x@csi-hpc-01 ~]$ ls  
test  
[p5j2sa9x@csi-hpc-01 ~]$ ls test  
test2  
[p5j2sa9x@csi-hpc-01 ~]$ █
```

(リダイレクトを用いた) ファイルの作成

- リダイレクト「>」: コマンドの出力をファイルに保存

– 例) `echo 'a = 1' > a.txt`

文字データのみのファイル

「a = 1」という文字が書かれたa.txtというテキストファイルをカレントディレクトリに作成

```
$ p5j2sa9x@csi-hpc-01:~ × +  
[p5j2sa9x@csi-hpc-01 ~]$ echo 'a = 1' > a.txt  
[p5j2sa9x@csi-hpc-01 ~]$ echo 'print(a)' > b.txt  
[p5j2sa9x@csi-hpc-01 ~]$ ls  
a.txt b.txt test  
[p5j2sa9x@csi-hpc-01 ~]$
```

ファイルの表示と連結

- **cat** (concatenate)

– 例)

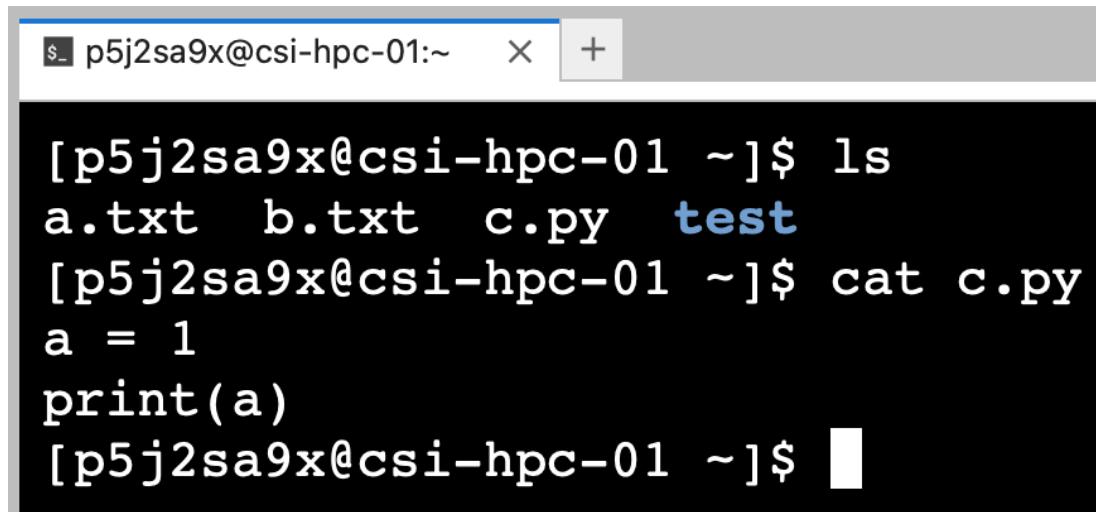
```
[p5j2sa9x@csi-hpc-01 ~]$ cat a.txt  
a = 1
```

カレントディレクトリのa.txtに書かれているテキストを表示

– 例)

```
cat a.txt b.txt > c.py
```

カレントディレクトリのa.txtとb.txtに書かれているテキストを連結して、リダイレクトでc.pyというファイルを作成



A screenshot of a terminal window titled 'p5j2sa9x@csi-hpc-01:~'. The window shows two command-line sessions. The first session runs 'ls' and lists files 'a.txt', 'b.txt', 'c.py', and 'test'. The second session runs 'cat c.py', which outputs the contents of the file 'c.py': 'a = 1' followed by a 'print(a)' statement. The terminal has a light gray background and a dark gray border.

```
[p5j2sa9x@csi-hpc-01 ~]$ ls  
a.txt b.txt c.py test  
[p5j2sa9x@csi-hpc-01 ~]$ cat c.py  
a = 1  
print(a)  
[p5j2sa9x@csi-hpc-01 ~]$ █
```

ファイルやディレクトリのコピー

- **cp** (copy)

- 例) **cp a.txt copy_a.txt**

カレントディレクトリのa.txtを複製して、copy_a.txtという名前で保存

- 例) **cp b.txt test/**

カレントディレクトリのb.txtを複製して、testディレクトリの中に保存

```
p5j2sa9x@csi-hpc-01:~ +  
[p5j2sa9x@csi-hpc-01 ~]$ ls  
a.txt  b.txt  c.py  copy_a.txt  test  
[p5j2sa9x@csi-hpc-01 ~]$ ls test  
b.txt  test2  
[p5j2sa9x@csi-hpc-01 ~]$ █
```

ファイルやディレクトリの移動、名前の変更

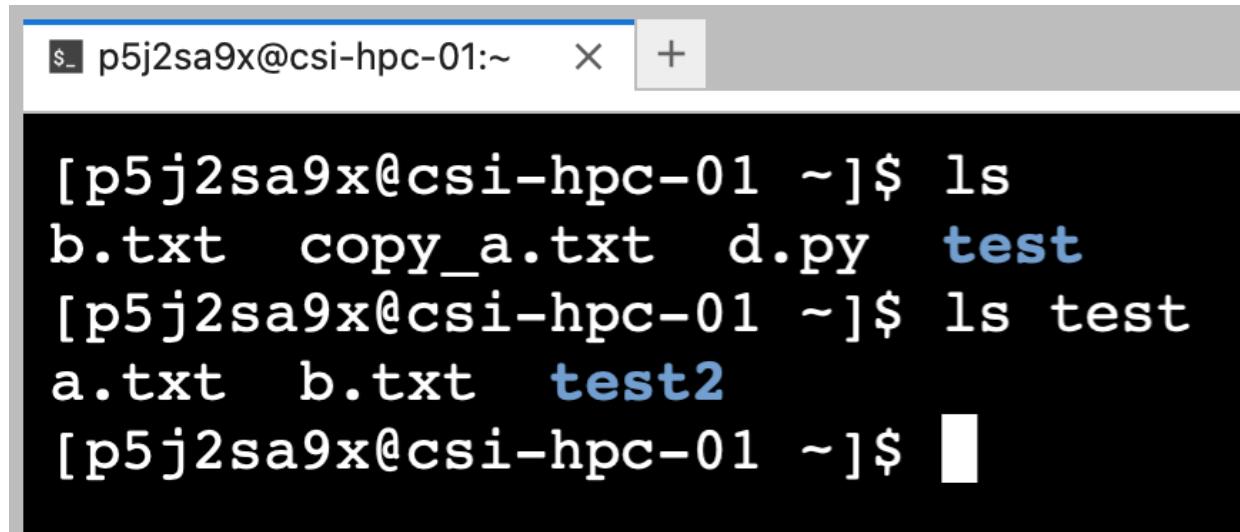
- **mv** (move)

- 例) **mv a.txt test/**

カレントディレクトリのa.txtを、testディレクトリに移動

- 例) **mv c.py d.py**

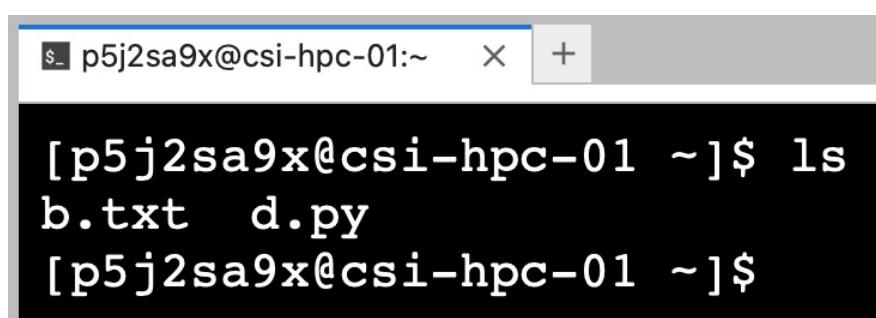
カレントディレクトリのc.pyをカレントディレクトリのd.pyへ
移動(ファイル名を変更したことに対応)



The screenshot shows a terminal window with a light gray header bar containing the text '\$ p5j2sa9x@csi-hpc-01:~' and two small icons. The main area is a black terminal window displaying the following command-line session:

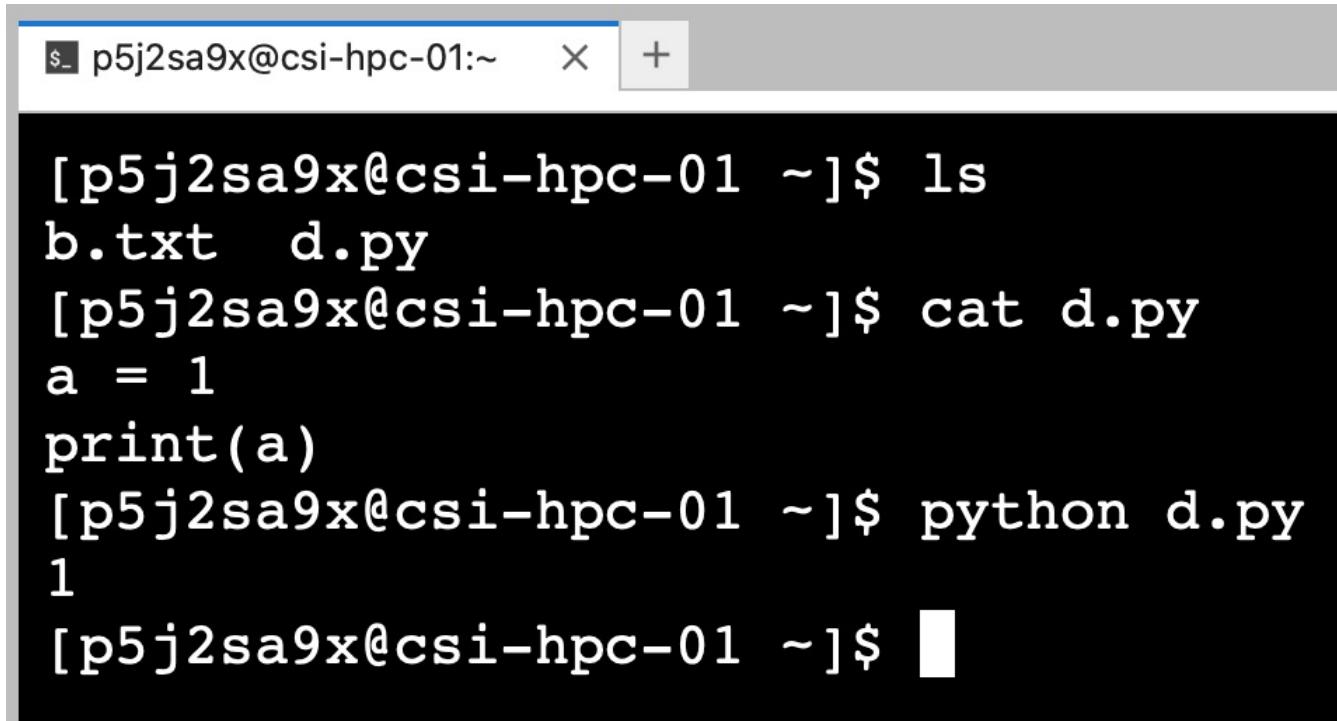
```
[p5j2sa9x@csi-hpc-01 ~]$ ls
b.txt copy_a.txt d.py test
[p5j2sa9x@csi-hpc-01 ~]$ ls test
a.txt b.txt test2
[p5j2sa9x@csi-hpc-01 ~]$ █
```

ファイルやディレクトリの削除

- **rm** (remove)
 - 危険⚠ 「ゴミ箱」はなく、すぐ削除される
 - 例) **rm copy_a.txt**
カレントディレクトリのcopy_a.txtを削除
 - 例) **rm -r test**
カレントディレクトリのtestディレクトリを丸ごと削除 超危険⚠
 - 手が滑って or タイプミスで、消すつもりではなかったファイルやディレクトリを消さないように注意

ファイルに保存したPythonプログラムコードの実行

- Pythonプログラムコードの拡張子「**.py**」
- 対話モードのときに使用したpythonコマンドの後ろに .py ファイルを指定すると、そのプログラムコードを実行できる



A screenshot of a terminal window titled "p5j2sa9x@csi-hpc-01:~". The terminal shows the following session:

```
[p5j2sa9x@csi-hpc-01 ~]$ ls  
b.txt d.py  
[p5j2sa9x@csi-hpc-01 ~]$ cat d.py  
a = 1  
print(a)  
[p5j2sa9x@csi-hpc-01 ~]$ python d.py  
1  
[p5j2sa9x@csi-hpc-01 ~]$ █
```

時間の都合で扱えなかつた初步的な内容

- chmod の使い方（パーミッションの変更）
- ワイルドカード、パス名展開
 - 複数ファイルや複数ディレクトリを指定する
- テキストエディタ（vi、vim、emacs など）
 - 今回の授業では、リダイレクトでテキストファイルを作成したが、普通はテキストエディタで編集する
- パイプ、コマンド置換、シェルスクリプト
 - 複数のUNIXコマンドを組み合わせて、退屈な作業を自動化する
- SSH接続、公開鍵・秘密鍵の作成
 - 学部2~3年の授業では、教育用計算機システムにSSHで接続するらしい

出席テストと課題

- 課題 (12:30にBEEF+に公開されます)
 - 提出期限: 7/13(日)の 23:59 まで
 - 演習1の授業時間後に再度、教育用計算機システムに接続し、以下のコマンドを実行している様子が全て写ったスクリーンショット画像1枚を提出してください（システムに接続できない人は、WebVMでも可）
 1. date コマンド
 2. hostname コマンド
 3. echo コマンドとリダイレクトを用いて、1~3行程度の .py ファイルを作成し、そのPythonプログラムコードを実行
 4. その他何か授業で扱っていないUNIXコマンドやオプション自分で調べて1つ実行
 - ファイル名の指定なし。拡張子はImageMagickで手軽にjpgやpngに変換できるものだけ可とする

テキストエディタ(vi)

- 例) `vi test.txt`
 - カレントディレクトリのtest.txtを編集、test.txtがなければ新規作成
- ノーマルモード  or  ← 画面左下がこんな感じ
- 挿入モード  ← 画面左下がこんな感じ
- ノーマルモードのときに「`i`」で入力モードへ切り替え
- 入力モードのときに「`escキー`」でノーマルモードへ切り替え
- 入力モードのときだけ、テキストを編集できる
 - カーソルはマウスで動かない
 - 矢印キーで移動「↑」「→」「↓」「←」
- 保存して終了：ノーマルモードで「`:wq`」

課題の答えの例

```
s_ p5j2sa9x@csi-hpc-01:~  × +  
[p5j2sa9x@csi-hpc-01 ~]$ date  
2025年 7月 8日 火曜日 18:57:14 JST  
[p5j2sa9x@csi-hpc-01 ~]$ hostname  
csi-hpc-01  
[p5j2sa9x@csi-hpc-01 ~]$ echo 'print("Hello Python on Unix")' > enshul_13.py  
[p5j2sa9x@csi-hpc-01 ~]$ python enshul_13.py  
Hello Python on Unix  
[p5j2sa9x@csi-hpc-01 ~]$ touch test.txt  
[p5j2sa9x@csi-hpc-01 ~]$ █
```

問一

オブジェクト指向プログラミングに最も関係ないものを1つ選んでください。

ダック・タイピング（ポリモーフィズム）

ラムダ式

正解

インスタンス

クラス

カプセル化

継承

隠蔽

RPGゲームをつくるために、次のようなゲームキャラクターを表すクラスを定義しました。

```
class Character:
    def __init__(self, name, hp):
        self.name = name
        self.hp = hp

    def attack(self, monster):
        print(f'{self.name}の通常攻撃!')
        monster.hp -= 10

class Mage(Character):
    def __init__(self, name, hp, mp):
        super().__init__(name, hp)
        self.mp = mp

    def attack(self, monster):
        if self.mp >= 10:
            print(f'{self.name}の魔法攻撃!')
            self.mp -= 10
            monster.hp -= 20
        else:
            print(f'{self.name}はMPが足りない!')
            super().attack(monster)
```

次に、これらのクラスのインスタンスを3つ、以下のように生成しました。うりぼうずは魔法使いという設定でMageクラスのインスタンスにしています。

```
char1 = Character('神大うりぼー', 100)
char2 = Mage('うりぼうず', 50, 10)
monster = Character('スライム', 50)
```

その後、以下のようなプログラムコードでRPGゲームの戦闘シーンをつくりました。

```
while ((char1.hp > 0) and (char2.hp > 0)) or (monster.hp > 0):
    for char in [char1, char2, monster]:
        print(f'* {char.name}のHP: {char.hp}')
        print('')

    for char in [char1, char2]:
        char.attack(monster)

    if monster.hp <= 0:
        print(f'勝利した!')
        break

    if char1.hp > 0:
        monster.attack(char1)
    else:
        monster.attack(char2)

    if (char1.hp <= 0) and (char2.hp <= 0):
        print('全滅した!')
        break

    print('-'*10)
```

以上のプログラムコードを実行したとき、出力結果として正しいものはどれでしょうか？

◆◆◆ ヒント ◆◆◆

メソッドのオーバーライド（教科書 p. 195-198）に関する問題です。

スーパークラスで定義されたメソッドと同じ名前のメソッドがサブクラスにあったとき、サブクラスのメソッドで上書きされます。このことを「オーバーライド」と呼びます。オーバーライドは初期化メソッドでも起こります。メソッドや初期化メソッドをオーバーライドするとき、共通部分が多い場合には、スーパークラスのメソッドや初期化メソッドを流用できると便利です。スーパークラスのメソッドや初期化メソッドをサブクラスから呼び出すには「super().」の後に「メソッド名」や「__init__」と記述します。

- * 神大うりぼーのHP: 100
- * うりぼうずのHP: 50
- * スライムのHP: 50

神大うりぼーの通常攻撃!

うりぼうずの魔法攻撃!

スライムの通常攻撃!

* 神大うりぼーのHP: 90
* うりぼうずのHP: 50
* スライムのHP: 20

神大うりぼーの通常攻撃!
うりぼうずはMPが足りない!
うりぼうずの通常攻撃!
勝利した!

正解

問3

インターネットのような公衆のネットワーク上に仮想的な専用の回線をつくり、安全に通信できるようにする技術を何というか？

UNIX

KNOSSOS

JupyterHub

VPN

正解

FortiClient