

計算物理学II

第2回 コマンド操作とデータの可視化

秋山 進一郎

2025年10月10日

授業日の確認

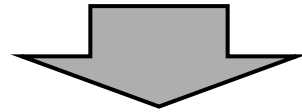
- ・ 全10回
 - ・ 第1回：10月3日（金）
 - ・ 第2回：10月10日（金）
 - ・ 第3回：10月17日（金）
 - ・ 第4回：10月24日（金）★
 - ・ 第5回：10月31日（金）
 - ・ 第6回：11月14日（金）★
 - ・ 第7回：11月21日（金）
 - ・ 第8回：12月5日（金）★
 - ・ 第9回：12月12日（金）
 - ・ 第10回：12月19日（金）★
- ・ ★の付いた授業にてレポート課題を配布予定

今日の授業の目標

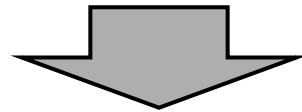
- **引き続き, Terminalの操作に慣れていく**
 - **コマンド操作**
 - **図の作成 (matplotlib, Gnuplot)**

毎回の授業の流れ

“compphy2”に移動する



“git fetch”と“git merge”で演習資料を入手



“lecture*”に移動し, 演習開始

本日の演習内容

- コマンドを使ったファイル操作について
 - 前回の復習といくつかの新しいコマンド (cp, mv, rm…)
- matplotlibによる図の作成 / Gnuplotによる図の作成
 - 関数のプロット
 - 数値データのプロット
 - スクリプトの活用

前回学んだコマンドを使って以下を実行しましょう

- “ex_command”というディレクトリを作る

```
akiyama@:~/compphys2/lecture2$ mkdir ex_command
```

- “ex_command”へ移動

```
akiyama@:~/compphys2/lecture2$ cd ex_command/  
akiyama@:~/compphys2/lecture2/ex_command$
```

- “test.txt”というファイルを作成

```
akiyama@:~/compphys2/lecture2/ex_command$ touch test.txt
```

- “test.txt”を好きなエディタ（vi, VS code, …）を使って編集してみよう
 - 「test」と入力して保存し、エディタを閉じる

- 編集内容の確認

```
akiyama@:~/compphys2/lecture2/ex_command$ cat test.txt  
test
```

ファイルのコピー：cpコマンドを使ってみる

- ・ファイルのコピーはcpコマンドで行う
 - ・“test.txt”を“cp_test.txt”という名前でコピーする
 - ・コマンドの構造は, “cp コピーしたいファイルのパス コピー先のファイルのパス”
 - ・パスの説明は後で. まずは以下のコマンドを実行してみましょう

スペース

```
akiyama@:~/compphys2/lecture2/ex_command$ cp ./test.txt ./cp_test.txt
```

- ・lsを使って“cp_test.txt”が作成されたことを確認せよ
- ・catを使って“cp_test.txt”の中身を確認し, コピーされていることを確認せよ

ファイルのコピー：cpコマンドを試してみる

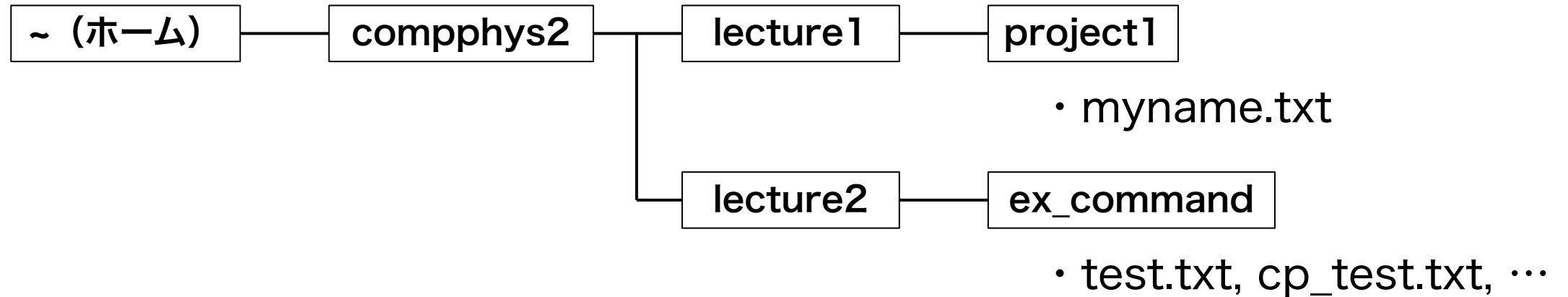
- ・ ファイルのコピーはcpコマンドで行う
 - ・ “./”はカレントディレクトリ（今自分がいるディレクトリ）を意味する
 - ・ “./test.txt”は「カレントディレクトリの中のtest.txt」を意味する
 - ・ カレントディレクトリ内での操作では“./”は省略可能
 - ・ 試しに以下のコマンドも実行せよ

```
akiyama@:~/compphys2/lecture2/ex_command$ cp test.txt cp2_test.txt
```

- ・ lsを使って“cp2_test.txt”が作成されたことを確認せよ
- ・ catを使って“cp2_test.txt”の中身を確認し、コピーされていることを確認せよ

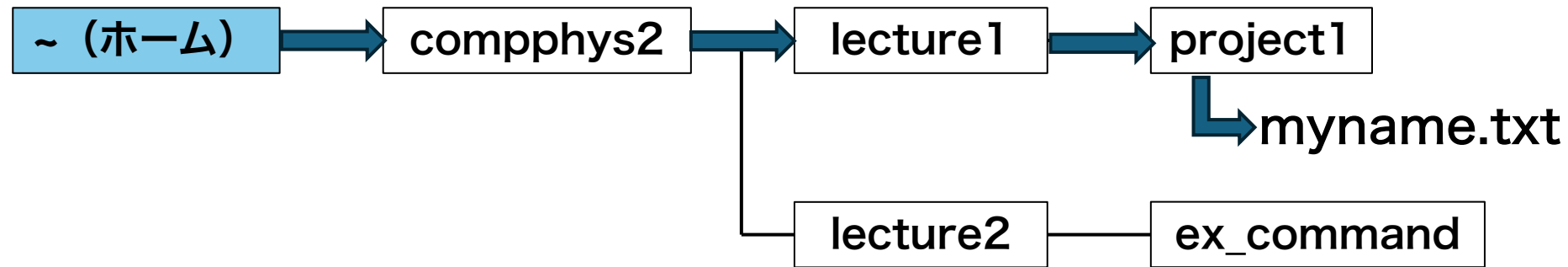
ディレクトリを跨ぐファイルのコピー

- ・ 前回作成した“myname.txt”を今いるディレクトリにコピーしよう
- ・ 現状のディレクトリ構造は以下のようにになっている



方法1：絶対パスを使う

- ・先ほど出てきた「パス」には二種類ある（絶対パスと相対パス）
- ・絶対パスとは, ディレクトリのツリー構造のトップを起点とした記述形式



・ test.txt, cp_test.txt, ...

- ・絶対パスによるname.txtの記述
 - ・ “~/compphys2/lecture1/project1/name.txt”
 - ・ 「ホームの中のcompphys2の中のlecture1の中のproject1の中のname.txt」
という意味である

方法1：絶対パスを使う

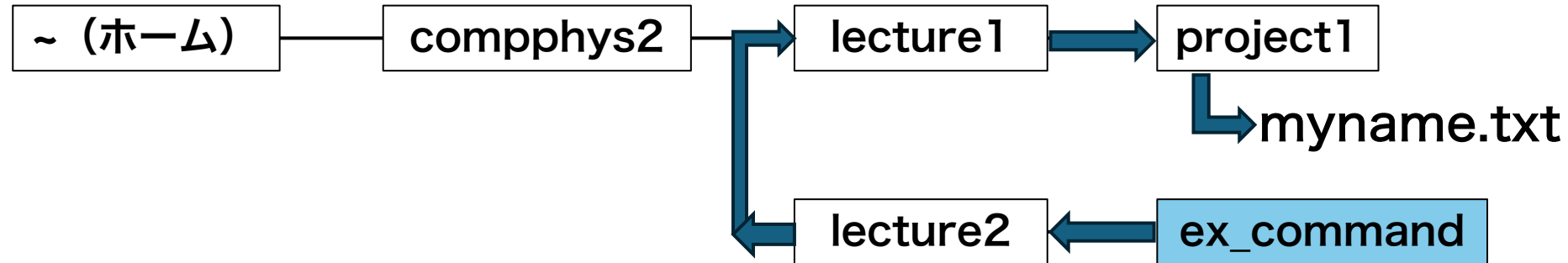
- “cp コピーしたいファイルのパス コピー先のファイルのパス”でコピーする
 - コピー後のファイル名はmyname.txtのままとする
 - コピー後のファイル名を変えない場合, コピー先のファイルのパスの部分は, 単にカレントディレクトリ (./) とする

```
akiyama@~/compphys2/lecture2/ex_command$ cp ~/compphys2/lecture1/project1/myname.txt ./
```

- lsを使って“myname.txt”がカレントディレクトリにコピーされたことを確認せよ
- catを使って“myname.txt”の中身を確認し, 正しくコピーされていることを確認せよ

方法2：相対パスを使う

- ・ 相対パスとは, カレントディレクトリを起点とした記述形式



- ・ 相対パスによるmyname.txtの記述
 - ・ “../../lecture1/project1/myname.txt”
 - ・ “../”はカレントディレクトリから見て一つ上のディレクトリ (lecture2)
 - ・ “../../”はカレントディレクトリから見て二つ上のディレクトリ (compphys2)
 - ・ 「二つ上のディレクトリの中のlecture1の中のproject1の中のmyname.txt」という意味
- ・ test.txt, cp_test.txt, ...

方法2：相対パスを使う

- “cp コピーしたいファイルのパス コピー先のファイルのパス”でコピーする
 - コピー後のファイル名はcp_myname.txtとする

```
akiyama@~/compphys2/lecture2/ex_command$ cp ../../lecture1/project1/myname.txt ./cp_myname.txt
```

- lsを使って“cp_myname.txt”がカレントディレクトリにコピーされたことを確認せよ
- catを使って“cp_myname.txt”の中身を確認し, 正しくコピーされていることを確認せよ

cpコマンドの注意点

- ・ 上書きに注意

- ・ 先ほどコピーしてきたcp_myname.txtを編集し, 入力されている文字を全て削除し, 保存しましょう
- ・ 保存したらターミナルに戻り, catを使ってcp_myname.txtを見てみましょう
 - ・ 何も書かれていないため, 以下のようになります

```
akiyama@~/compphys2/lecture2/ex_command$ cat cp_myname.txt  
akiyama@~/compphys2/lecture2/ex_command$
```

- ・ この状態で, もう一度以下のコマンドを実行し, myname.txtをすでにコピーしてきたcp_myname.txtと同じ名前でコピーしてみましょう

```
akiyama@~/compphys2/lecture2/ex_command$ cp ../../lecture1/project1/myname.txt ./cp_myname.txt
```

- ・ catを使ってcp_myname.txtを見てみましょう
- ・ cp_myname.txtが上書きされたのを確認できましたか? cpする時は気をつけましょう

絶対パスと相対パスの使い分け

- ・ 絶対パスはカレントディレクトリに依存しない記述形式
 - ・ いわば各ディレクトリや各ファイルの住所
 - ・ リンクを貼る時には便利
- ・ 相対パスはカレントディレクトリに依存した記述形式
 - ・ 実際の作業を進める時には相対パスをよく使う

カレントディレクトリの絶対パスの取得：pwdを使う

- ・カレントディレクトリの絶対パスを知りたい場合はpwdコマンドを使う

```
akiyama@:~/compphys2/lecture2/ex_command$ pwd  
/home/akiyama/compphys2/lecture2/ex_command  
akiyama@:~/compphys2/lecture2/ex_command$
```

- ・ここではホームディレクトリは“~/”ではなく“/home/”で表示されている
- ・pwdで自分の居場所を確認できる

ディレクトリのコピー：cpコマンドを使う

- ・ 今度は, 前回作成した“project1”（ディレクトリ）を今いるディレクトリにコピーしよう
 - ・ 相対パスによるコピー

```
akiyama@:~/compphys2/lecture2/ex_command$ cp ../../lecture1/project1 ./
```

- ・ このコマンドを実行すると…

```
akiyama@:~/compphys2/lecture2/ex_command$ cp ../../lecture1/project1 ./  
cp: -r not specified; omitting directory '../../lecture1/project1'  
akiyama@:~/compphys2/lecture2/ex_command$
```

- ・ lsを使ってproject1がコピーされているかどうかを見てみよう
- ・ Terminalは何と言っているのでしょうか？（次ページ）

ディレクトリのコピー：cpコマンドを使う

```
akiyama@~/compphys2/lecture2/ex_command$ cp ../../lecture1/project1 ./  
cp: -r not specified; omitting directory '../../lecture1/project1'  
akiyama@~/compphys2/lecture2/ex_command$
```

- 「-rがない」と言っている
- ディレクトリをコピーする方法
 - “cp **-r** コピーしたいディレクトリのパス コピー先のディレクトリのパス”
 - 素朴に下記のコマンドを打っても良いが、↑キーを一回押すと直前に打ったコマンドが出てくる. そこに-rを書き加えても良い

```
akiyama@~/compphys2/lecture2/ex_command$ cp -r ../../lecture1/project1 ./
```

- lsを使ってproject1がコピーされているかどうかを見てみよう

コマンドに対するオプション

- ・ コマンドには様々なオプションが用意されている
 - ・ オプションを指定することでコマンドの機能が拡張される
- ・ 先ほどの場合だと, “-r”がcpコマンドに対するオプション
- ・ 各コマンドに用意されているオプションは“コマンド **--help**”で閲覧できる
 - ・ 以下のコマンドを実行し, cpにどのようなオプションがあるか見てみよう

```
akiyama@:~/compphys2/lecture2/ex_command$ cp --help
```

その他のよく使うオプション一例

- lsコマンド

- 一度ホームディレクトリに戻りましょう（その場でcdする）
- ホームディレクトリでlsを実行

```
akiyama@:~$ ls  
compphys2 Desktop dev Documents Downloads Dropbox intel Music Pictures Public slurm-24.11.3 snap Templates Videos work
```

- 今度はlsに-laオプションをつけて実行してみましょう

```
akiyama@:~$ ls -la
```

- 表示形式が変わり, lsでは見えていなかったディレクトリやファイルが見えたらOK
- -laは, -l（詳細情報表示）と-a（隠しファイルを含む全てを表示）というオプションを合わせたもの. よく使う
- 隠しファイルとは名前が“.”から始まるもの

ファイルの移動：mvコマンドを使う

- ・まずホームから“ex_command”のディレクトリまで移動しましょう
 - ・以下の手順で, “cp2_test.txt”をlecture2ディレクトリ直下へ移動させましょう
 - ・ファイルを移動するにはmvコマンドを使う
 - ・コマンドの構造は, “mv 移動したいファイルのパス 移動先のファイルのパス”
- ```
akiyama@~/compphys2/lecture2/ex_command$ mv ./cp2_test.txt ../
```
- ・「カレントのcp2\_test.txtを一つ上のディレクトリへ移動する」の意味
  - ・lsを使って, ex\_commandの中にcp2\_test.txtが無いことを確認せよ
  - ・lecture2ディレクトリに移動してlsせよ. cp2\_test.txtがあるはず
  - ・cpとmvの役割の違いを理解しましょう

# ファイル名の変更：mvコマンドを使う

- ・ mvコマンドを応用するとファイル名の変更が可能
- ・ lecture2ディレクトリに移動させた“cp2\_test.txt”を“test.txt”という名前に変更してみましょう
- ・ ファイル名の変更とはファイルの移動の一種

```
akiyama@:~/compphys2/lecture2$ mv ./cp2_test.txt ./text.txt
```

- ・ lsを使って, cp2\_test.txtがtest.txtになったことを確認せよ

## ディレクトリの移動：mvコマンドを使う

- ・ 今度は, “ex\_command”の中にある“project1”ディレクトリを“lecture2”直下へ移動させてみましょう
- ・ 相対パスを使えば, わざわざ“ex\_command”に入らずに次のようにできます

```
akiyama@:~/compphys2/lecture2$ mv ./ex_command/project1 ./
```

- ・ lsを使って, project1がlecture2直下に移動されたことを確認せよ
- ・ cpでディレクトリのパスを扱うには-rオプションが必要だったが, mvでは不要

# ファイルやディレクトリの削除：rmコマンドを使う

- ・ ファイルやディレクトリの削除は, rmコマンドを使う
  - ・ rmで削除すると元には戻せないなので, 実行時は要注意！
- ・ lecture2直下にあるtest.txtを削除してみます

- ・ -iオプション：削除前に本当に削除して良いか確認する

- ・ 次のコマンドを実行

```
akiyama@~/compphys2/lecture2$ rm -i text.txt
rm: remove regular file 'text.txt'? y
```

- ・ Terminalが削除していいか確認してくる
  - ・ 削除する場合: yを押してEnterを押す（今回はこちら）
  - ・ 削除する場合: nを押してEnterを押す
- ・ lsでtest.txtが消えたことを確認せよ



# ファイルやディレクトリの削除：rmコマンドを使う

- ・ディレクトリの削除は, rmコマンドに-rオプションをつける
  - ・-iオプションと組み合わせる場合は-riとする (-ir, -r -iでもOK)
- ・lecture2直下にあるproject1を削除してみます
  - ・次のコマンドを実行
    - akiyama@~/compphys2/lecture2\$ rm -ri project1
    - rm: descend into directory 'project1'? y
  - ・Terminalが削除していいか確認してくる
    - ・ディレクトリ配下に入るか尋ねてきた場合もyを押してEnter
    - ・その場合, 配下のファイルを削除するか逐一尋ねられるので, 逐一yを押してEnter
    - ・配下のファイルが全て削除されたら, 最後にディレクトリを削除するか尋ねてくるので, yを押してEnter
- ・lsでproject1が消えたことを確認せよ

# 本日の演習内容

- ・ コマンドを使ったファイル操作について
  - ・ 前回の復習といくつかの新しいコマンド (cp, mv, rm…)
- ・ matplotlibによる図の作成 / Gnuplotによる図の作成
  - ・ 関数のプロット
  - ・ 数値データのプロット
  - ・ スクリプトの活用

# matplotlibとは？

- Pythonでグラフを描くためのパッケージ
- 特によく使うのが, matplotlib.pyplotモジュール
  - MATLAB-likeな環境でグラフ描画ができる
  - MATLAB（マトラボ）は数値解析向けのソフトウェア/プログラミング言語の一つ
- 公式のUser manualは[ここ](#)から
- Pythonによるプログラミングについては後半の演習で詳しく学びます
  - 今回の演習では, サンプルスクリプトをベースに, matplotlibの使用感を体験しましょう

# Pythonスクリプトファイル（詳しくは後日の演習で）

- Pythonインタプリタには二種類ある
  - ① 対話モード
  - ② Pythonスクリプトファイルを使う方法
- ②の方法では、スクリプトファイルのパスを引数にして、python3コマンドを実行する
  - Pythonスクリプトファイルの拡張子は.py
- 以降では②の方法を使っていきます

# 関数をプロットしてみよう

- ・まずはlecture2の直下に“ex\_matplotlib”というディレクトリを作りましょう

```
akiyama@~/compphys2/lecture2$ mkdir ex_matplotlib
```

- ・lsコマンドでディレクトリが作られたことを確認しましょう

```
akiyama@~/compphys2/lecture2$ ls
data ex_matplotlib src
```

- ・“src”というディレクトリの中に, “sample\_plot\_function.py”というPythonスクリプトが格納されています. このスクリプトをex\_matplotlibへcpしましょう

```
akiyama@~/compphys2/lecture2$ cp ./src/sample_plot_functions.py ./ex_matplotlib/
```

- ・cpできたら, ex\_matplotlibへ入ります

```
akiyama@~/compphys2/lecture2$ cd ex_matplotlib/
akiyama@~/compphys2/lecture2/ex_matplotlib$
```

# サンプルスクリプトの説明1

- cpしたPythonスクリプトをVS codeで開いてみましょう
- 以下でこのスクリプトの意味をパーツごとに簡単に見ていきます
- 繰り返しになりますが, 詳細は今後の演習で学んでいきますので, 今は眺めてもらうだけでOKです

```
sample_plot_functions.py X
home > akiyama > compphys2 > lecture2 > ex_matplotlib > sample_plot_functions.py
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def make_plot(cxs, cys, dxs, dys, output_figname):
5 plt.rc('text', usetex=True)
6 plt.rc('font', family='serif')
7 plt.figure(figsize=(8, 6))
8 plt.title('Sample', fontsize=20)
9 plt.xlabel(r'x', fontsize=20)
10 plt.ylabel(r'$f(x)$', fontsize=20)
11 #plt.xlim(0, 2.0*np.pi)
12 #plt.ylim(-1.0, 1.0)
13 plt.plot(cxs, cys, 'r-', label='function A')
14 plt.plot(dxs, dys, 'b--', label='function B')
15 plt.legend()
16 #plt.grid(True)
17 #plt.show()
18 plt.savefig(output_figname)
19 plt.close()
20
21 cx = [0.1*i for i in range(100)]
22 cy = [np.sin(x) for x in cx]
23
24 dx = [0.1*i for i in range(100)]
25 dy = [np.cos(x) for x in dx]
26
27 make_plot(cx, cy, dx, dy, "functions.pdf")
```

## サンプルスクリプトの説明2

- ・最初に, このスクリプトで使用するモジュールをimportしています
  - ・“as”以下でそれぞれのモジュールに略称を設定しています
  - ・略称は自由に設定できますが, matplotlib.pyplotは“plt”とするのが通例です

```
1 import matplotlib.pyplot as plt
2 import numpy as np
```

## サンプルスクリプトの説明3

- ・次に, グラフ描画用の「関数」を定義しています

```
4 def make_plot(cxs, cys, dxs, dys, output_figname):
```

- ・ make\_plotという「関数」は二つの関数を一枚のグラフにプロットします
- ・ make\_plotを使うには, 5つの引数を渡します
  - ・ cxs, cys: プロットしたい関数のxとyの値
  - ・ dxs, dys: プロットしたい別の関数のxとyの値
  - ・ output\_figname: 出来上がったグラフを保存する際のファイル名



## サンプルスクリプトの説明4

- make\_plotという「関数」の中身を見ていきましょう
  - plt.\*\*という文で, importしたmatplotlib.pyplotモジュールの機能を使っています
- はじめに, グラフで使うフォントやグラフの大きさを設定しています

```
5 plt.rc('text', usetex=True)
6 plt.rc('font', family='serif')
7 plt.figure(figsize=(8, 6))
```

- 続く三行で, グラフのタイトルやx軸, y軸のラベルを設定しています
  - l.9, 10の `r'$*****$'` により, TeXの表記が使えます (TeXについては次回の演習で)

```
8 plt.title('Sample', fontsize=20)
9 plt.xlabel(r'x', fontsize=20)
10 plt.ylabel(r'$f(x)$', fontsize=20)
```

## サンプルスクリプトの説明5

- ・ 行頭に#が付いた文は「コメントアウト」されています
  - ・ コメントアウトされた文は実行されません
  - ・ スクリプトに関する説明文や「今は使わないが後で使うかもしれない機能」などを書いておくためによく使われます
- ・ ここでコメントアウトされているのは, x軸, y軸の範囲を設定する機能です

```
11 #plt.xlim(0,2.0*np.pi)
12 #plt.ylim(-1.0,1.0)
```

## サンプルスクリプトの説明6

- l.13とl. 14でグラフ描画を行なっています
  - plt.plotの最初の引数がxの値, 次の引数がyの値です
  - 三番目の引数でグラフの色や線のスタイルを設定しています
  - 最後の引数はグラフの凡例 (legend) を設定しています
- l.15で設定したlegendを有効にし, グラフに表示されるようにしています

```
13 plt.plot(cxs, cys, 'r-', label='function A')
14 plt.plot(dxs, dys, 'b--', label='function B')
15 plt.legend()
```

- l.18では出来上がったグラフをファイル名を指定して保存しています
- l.19でプロット作業を終了します

```
18 plt.savefig(output_figname)
19 plt.close()
```

## サンプルスクリプトの説明7

- l.13とl. 14でグラフ描画を行なっています
  - plt.plotの最初の引数がxの値, 次の引数がyの値です
  - 三番目の引数でグラフの色や線のスタイルを設定しています
  - 最後の引数はグラフの凡例 (legend) を設定しています
- l.15で設定したlegendを有効にし, グラフに表示されるようにしています

```
13 plt.plot(cxs, cys, 'r-', label='function A')
14 plt.plot(dxs, dys, 'b--', label='function B')
15 plt.legend()
```

- l.18では, 作ったグラフをファイル名を指定して保存し, l.19でプロット作業を終了します

```
18 plt.savefig(output_filename)
19 plt.close()
```

- ここまでが「関数」make\_plotの定義部分です

## サンプルスクリプトの説明8

- ・ 1.21以下で, プロットしたい関数を用意し, make\_plotに受け渡しています
  - ・ この部分がこのサンプルスクリプトの主要な部分です

```
21 cxs = [0.1*i for i in range(100)]
22 cys = [np.sin(x) for x in cxs]
23
24 dxs = [0.1*i for i in range(100)]
25 dys = [np.cos(x) for x in dxs]
26
27 make_plot(cxs, cys, dxs, dys, "functions.pdf")
```

- ・ cxsは0, 0.1, 0.2, ..., 99.9の100個の数値です
- ・ cysは上で作ったcxsの各点xにおける $\sin(x)$ の数値です
- ・ dxs, dysでは $\cos(x)$ を用意しています
- ・ 作ったグラフは“functions.pdf”という名前で, このスクリプトを実行したのと同じディレクトリに保存されます

# サンプルスクリプトを実行しよう

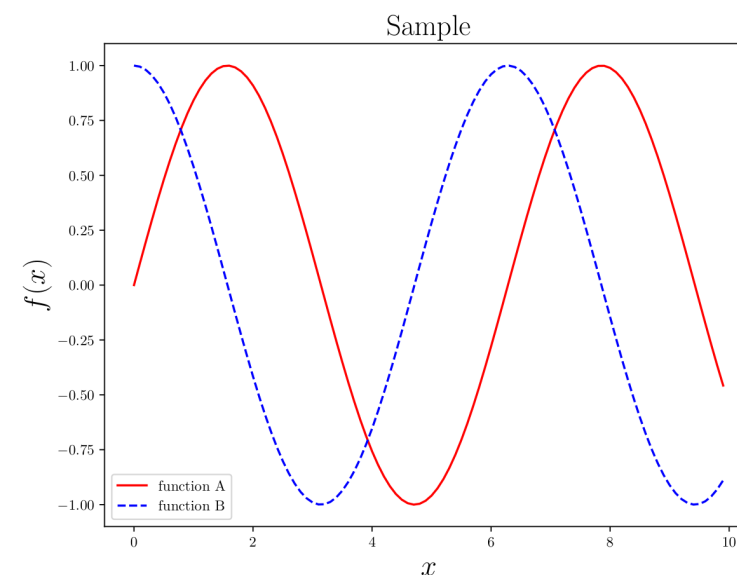
- では, サンプルスクリプトを動かしてみましょう
  - やり方はとても簡単. 以下のコマンドを実行するだけ

```
akiyama@~/compphys2/lecture2/ex_matplotlib$ python3 sample_plot_functions.py
```

- 実行後, lsでディレクトリの中を確認しましょう

```
akiyama@~/compphys2/lecture2/ex_matplotlib$ ls
functions.pdf sample_plot_functions.py
```

- 出来上がったfunctions.pdfを見てみましょう
  - 右図が見えればOK



## コメントアウトされた機能も使ってみよう

- ・もう一度, サンプルスクリプトをVS codeで開き, コメントアウトされた文章を有効にしてみましょう
  - ・例えば, l.16のコメントアウト（#）を外してみます

```
16 plt.grid(True)
```

- ・このままスクリプトを実行すると, 同じファイル名でグラフが保存されるので, 先ほど作った図が消えてしまいます
  - ・例えば, 保存ファイル名を以下のように変えておきましょう

```
27 make_plot(cxs, cys, dxs, dys, "functions_grid.pdf")
```

- ・では, もう一度スクリプトを実行し, 出来上がったグラフを見てみましょう
  - ・先ほどのグラフとの違いは分かりましたか？
- ・他のコメントアウトも外してみて, どんな変化が見られるか確認してみましょう

# データを読み込んでmatplotlibでプロットしてみよう

- ・ 今度は, 別ファイルとして保存されている数値データを読み込んで可視化してみましょう
  - ・ 用意されたサンプルを使っていきます
- ・ 以下の二つのファイルを先ほど作ったex\_matplotlibへcpしましょう
  - ・ /lecture2/src/sample\_plot\_data.py
  - ・ /lecture2/data/noisy\_data.dat
  - ・ 相対パスを活用しましょう



# サンプルスクリプトの説明1

- cpしたsample\_plot\_data.pyをVS codeで開いてみましょう
  - スクリプトの構造は先ほどのsample\_plot\_function.pyと同じ
- plot\_datという「関数」がinput\_filenameというデータを読み込み, output\_fignameという名前でプロットしたグラフを保存します
  - plot\_datは三列の数値データを読み込み, 一列目をx, 二列目をy, 三列目をyの値に対する誤差として, プロットします

```
4 def plot_dat(input_filename, output_figname):
5 data = np.loadtxt(input_filename)
6 x, y, yerr = data[:, 0], data[:, 1], data[:, 2]
```

## サンプルスクリプトの説明2

- ・l.13のplt.errorbarでyの値に対するエラーバー付きのプロットを作成しています

```
13 plt.errorbar(x, y, yerr=yerr, fmt='o', color = 'b', capsize=3)
```

- ・サンプルスクリプトの大半がplot\_datという「関数」の定義になっています
  - ・このスクリプトの主要な部分はl.19のみ

```
19 plot_dat("noisy_data.dat", "noisy_data.pdf")
```

- ・このスクリプトと同じディレクトリに置かれた“noisy\_data.dat”を読み込み,  
“noisy\_data.pdf”の名前で保存します

# サンプルデータの説明

- これからプロットするnoisy\_data.datをcatで見ておきましょう
  - この数値データはGaussianにノイズを加えて作成したものです
  - 一列目がx, 二列目がy, 三列目が誤差です

```
akiyama@~/compphys2/lecture2/ex_matplotlib$ cat noisy_data.dat
-10.00000000 -0.00883169 0.01200000
-9.59183673 -0.00013210 0.01159184
-9.18367347 0.00852781 0.01118367
-8.77551020 -0.01266085 0.01077551
-8.36734694 0.00999309 0.01036735
-7.95918367 0.01186520 0.00995918
-7.55102041 -0.01066796 0.00955102
-7.14285714 0.00144074 0.00914286
-6.73469388 0.01217763 0.00873469
-6.32653061 0.00121590 0.00832653
-5.91836735 0.00187981 0.00791837
-5.51020408 0.00254904 0.00751020
-5.10204082 0.01528427 0.00710204
-4.69387755 0.01076080 0.00669388
```

# サンプルスクリプトを実行しよう

- ・では, サンプルスクリプトを動かしてみましょう
  - ・以下のコマンドを実行するだけ
  - ・実行後, lsでディレクトリの中を確認しましょう

```
akiyama@~/compphys2/lecture2/ex_matplotlib$ python3 sample_plot_data.py
akiyama@~/compphys2/lecture2/ex_matplotlib$ ls
functions.pdf noisy_data.dat noisy_data.pdf sample_plot_data.py sample_plot_functions.py
akiyama@~/compphys2/lecture2/ex_matplotlib$
akiyama@~/compphys2/lecture2/ex_matplotlib$
```

- ・出来上がったnoisy\_data.pdfを見てみましょう

# ぜひmatplotlibを色々な場面で活用してください

- matplotlibは今後の演習やレポート課題でも活用していきます
  - 実際の数値計算は次のような流れになることが多いです
    - ① プログラムの作成（コーディング）
    - ② 作成したプログラムによる数値計算
    - ③ 数値計算の結果をプロットして可視化
    - ④ 結果に関する考察
- ③のステップはとても重要です

# 本日の演習内容

- ・ コマンドを使ったファイル操作について
  - ・ 前回の復習といくつかの新しいコマンド (cp, mv, rm…)
- ・ matplotlibによる図の作成 / Gnuplotによる図の作成
  - ・ 関数のプロット
  - ・ 数値データのプロット
  - ・ スクリプトの活用

# Gnuplotとは？

- ・ 入力された数式やデータをグラフ化してくれるソフトウェア
- ・ 二種類の形式（使い方）がある
  - ・ 対話型
  - ・ バッチ処理形式型（推奨）
- ・ 公式のUser manualは[ここ](#)から
  - ・ User manualは300ページ以上ある
  - ・ 知りたい情報があれば、ページ内検索で拾う

# 対話型

- gnuplotコマンドで起動
  - 一行ずつgnuplotに命令を出していく
  - あまり使わない
    - 理由は後で
- 例えば, “plot x\*\*2”や“plot sin(x)”と打ってEnterを押してみましよう
  - 図が出てくればOK. 確認後, 図は閉じてよい
- “exit”と打ってEnterを押すとTerminalに戻ってくる
  - Terminalに戻りましょう

```
akiyama@~/compphys2/lecture2$ gnuplot

G N U P L O T
Version 5.4 patchlevel 2 last modified 2021-06-01

Copyright (C) 1986-1993, 1998, 2004, 2007-2021
Thomas Williams, Colin Kelley and many others

gnuplot home: http://www.gnuplot.info
faq, bugs, etc: type "help FAQ"
immediate help: type "help" (plot window: hit 'h')

Terminal type is now 'unknown'
gnuplot>
```



# バッチ処理形式

- gnuplotへのコマンドをリストしたファイルを用意し, それを読み込ませて図を作成する
  - このようなファイルのことをスクリプトと呼ぶ
  - スクリプトは普通の.txtファイルでもいいが, gnuplotのスクリプトであることを明確にする目的で, .gplや.pltもよく使われる
- まず, lecture2の直下に“ex\_gnuplot”というディレクトリを作り, そこへ移動しましょう
- lecture2/src/sample.gplがサンプルとして用意したgnuplotのスクリプトです
  - “sample.gpl”を“ex\_gnuplot”へコピーしましょう

# 演習で使う数値データの説明

- lecture2/data/population.datがこれから可視化する数値データです
  - “population.dat”も“ex\_gnuplot”へコピーしましょう
- “population.dat”には、大正9年から平成27年までの全国、及び各都道府県の人口が格納されています（5年ごと）
  - “population.dat”は、[ここ](#)からダウンロードできる国勢調査のオープンデータに基づいています
- catを使い、“population.dat”を見てみましょう

The screenshot shows the e-Stat website interface. At the top, there's a header with the e-Stat logo and navigation links. Below the header, there's a search bar with filters for 'File', 'Census', 'Time Series Data', and 'CSV format'. The search results show a table with columns for 'Government Statistics Name', 'Government Statistics Code', and 'Survey Overview'. The first row is for the 'Census' (国勢調査) with code 00200521. The overview text states that the census is conducted every 5 years and provides data on population, household composition, and foreign residents.

| 政府統計名   | 国勢調査                                                                                                                                                                                                                   | 詳細 |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 政府統計コード | 00200521                                                                                                                                                                                                               |    |
| 調査の概要   | <p>国勢調査は、日本に住んでいるすべての人と世帯を対象とする国の最も重要な統計調査で、5年ごとに実施されます。国勢調査から得られる日本の人口や世帯の実態は、国や地方公共団体の行政において利用されることはもとより、民間企業や研究機関でも広く利用され、そのような利用を通じて国民生活に役立てられています。</p> <p>国勢調査では、年齢別の人口、家族構成、働いている人や日本に住んでいる外国人などの結果を提供しています。</p> |    |

出典：「政府統計の総合窓口(e-Stat)」、調査項目を調べる－国勢調査「男女別人口－全国、都道府県（大正9年～平成27年）」

# 演習で使う数値データの説明

・ “population.dat” の構造は次のようになっています

・ 1列目：都道府県コード

・ 2列目：都道府県名

・ 3列目：元号

・ 4列目：和暦（年）

・ 5列目：西暦（年）

・ 6列目：人口（総数）

・ 7列目：人口（男性）

・ 8列目：人口（女性）

```
akiyama@~/compphys2/lecture2/data$ cat population.dat
0 Japan Taisho 9 1920 55963053 28044185 27918868
1 Hokkaido Taisho 9 1920 2359183 1244322 1114861
2 Aomori Taisho 9 1920 756454 381293 375161
3 Iwate Taisho 9 1920 845540 421069 424471
4 Miyagi Taisho 9 1920 961768 485309 476459
5 Akita Taisho 9 1920 898537 453682 444855
6 Yamagata Taisho 9 1920 968925 478328 490597
7 Fukushima Taisho 9 1920 1362750 673525 689225
8 Ibaraki Taisho 9 1920 1350400 662128 688272
9 Tochigi Taisho 9 1920 1046479 514255 532224
10 Gunma Taisho 9 1920 1052610 514106 538504
11 Saitama Taisho 9 1920 1319533 641161 678372
12 Chiba Taisho 9 1920 1336155 656968 679187
13 Tokyo Taisho 9 1920 3699428 1952989 1746439
14 Kanagawa Taisho 9 1920 1323390 689751 633639
```

# 特定の数値データを抜き出す

- “population.dat”から特定の情報を抜き出し, gnuplotで可視化してみましょう
  - 例題：「横軸に西暦, 縦軸に日本の総人口を取って, 総人口の推移を見よう」
- ここで, Terminalから使える便利なコマンドを紹介します
  - **grep**コマンド=あるファイルから特定の文字列を含んだ行だけを抜き出す
  - “grep”**抜き出したい文字列** **ファイル (のパス)** ”
- 以下のコマンドを実行してみましょう

```
akiyama@~/compphys2/lecture2/ex_gnuplot$ grep 'Japan' ./population.dat
```

- Japanを含む行だけ抜き出されたらOK

## 特定の数値データを抜き出し, 保存する

- ・ さらに, 次のようにgrepコマンドを使うと, 抜き出した結果を新たに保存できます
  - ・ “grep **‘抜き出したい文字列’** **ファイル (のパス)** **>>** **保存ファイル (のパス)** ”
- ・ 以下のコマンドを実行してみましょう (↑キーを押すと便利)

```
akiyama@~/compphys2/lecture2/ex_gnuplot$ grep 'Japan' ./population.dat >> ./japan.dat
```

- ・ catを使い, “japan.dat”を見てみよう

# スクリプトを使ってgnuplotに図を書かせる

- ・ 先ほどcpした“sample.gpl”をcatで見てください
- ・ gnuplotに読ませると、各行が上から順番に実行される
- ・ #以下はコマンドとして読み込まれない（コメントアウト）

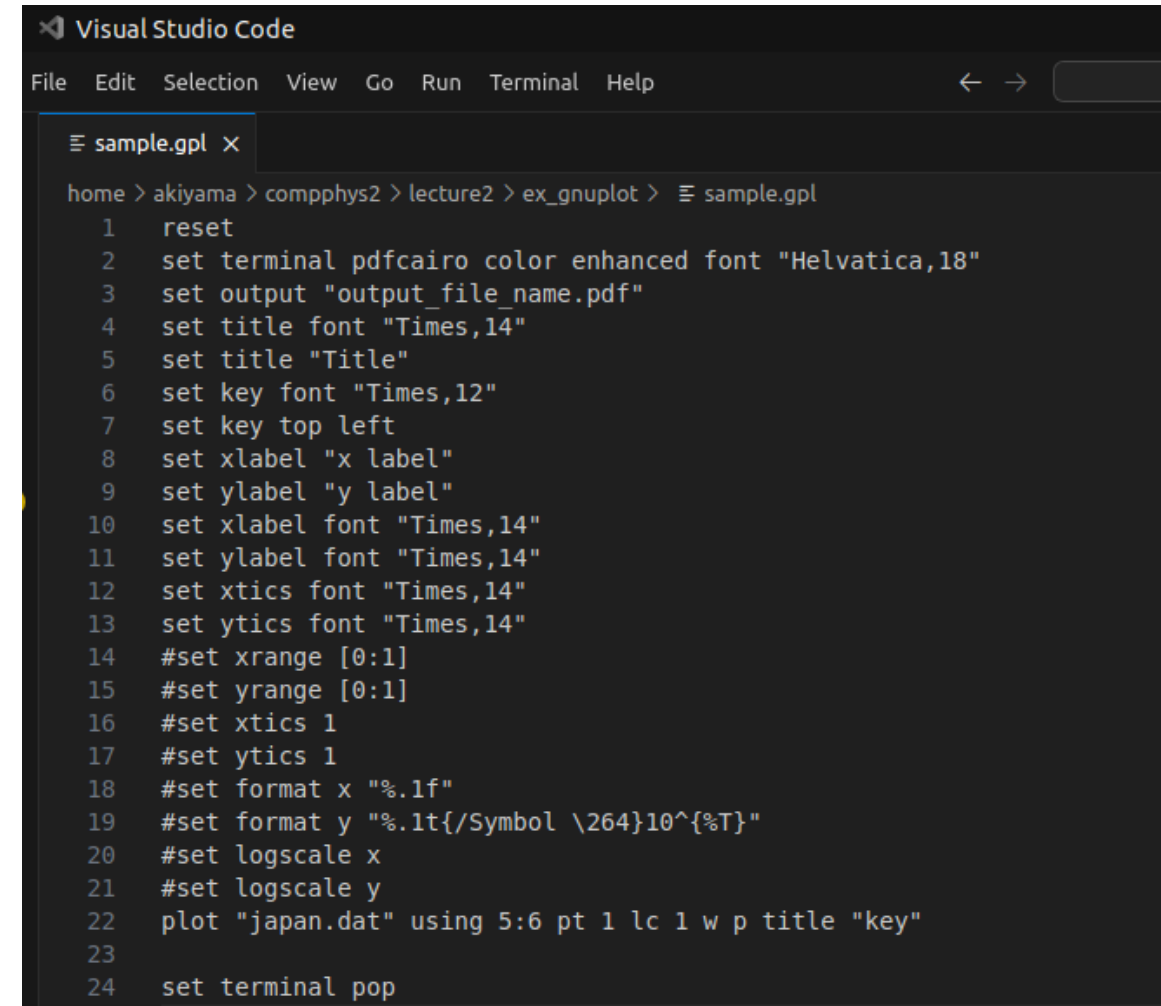
```
akiyama@~/compphys2/lecture2/ex_gnuplot$ cat sample.gpl
reset
set terminal pdfcairo color enhanced font "Helvetica,18"
set output "output_file_name.pdf"
set title font "Times,14"
set title "Title"
set key font "Times,12"
set key top left
set xlabel "x label"
set ylabel "y label"
set xlabel font "Times,14"
set ylabel font "Times,14"
set xtics font "Times,14"
set ytics font "Times,14"
#set xrange [0:1]
#set yrange [0:1]
#set xtics 1
#set ytics 1
#set format x "%.1f"
#set format y "%.1t{/Symbol \264}10^{%T}"
#set logscale x
#set logscale y
plot "input_file_name" using x:y pt 1 lc 1 w p title "key"

set terminal pop
```

# スクリプトを編集する

- ・ 横軸に西暦, 縦軸に日本の総人口を取って, 総人口の推移を見よう
  - ・ エディタでsample.gplを開く
  - ・ “plot”から始まる行(l.22)を編集する
  - ・ 読み込むファイルはjapan.dat
  - ・ x軸に使うのは5列目, y軸に使うのは6列目
- ・ 右のようにl.22を編集したら, 以下のようにgnuplotにスクリプトを読ませる
  - ・ “gnuplot **スクリプトファイル名**”

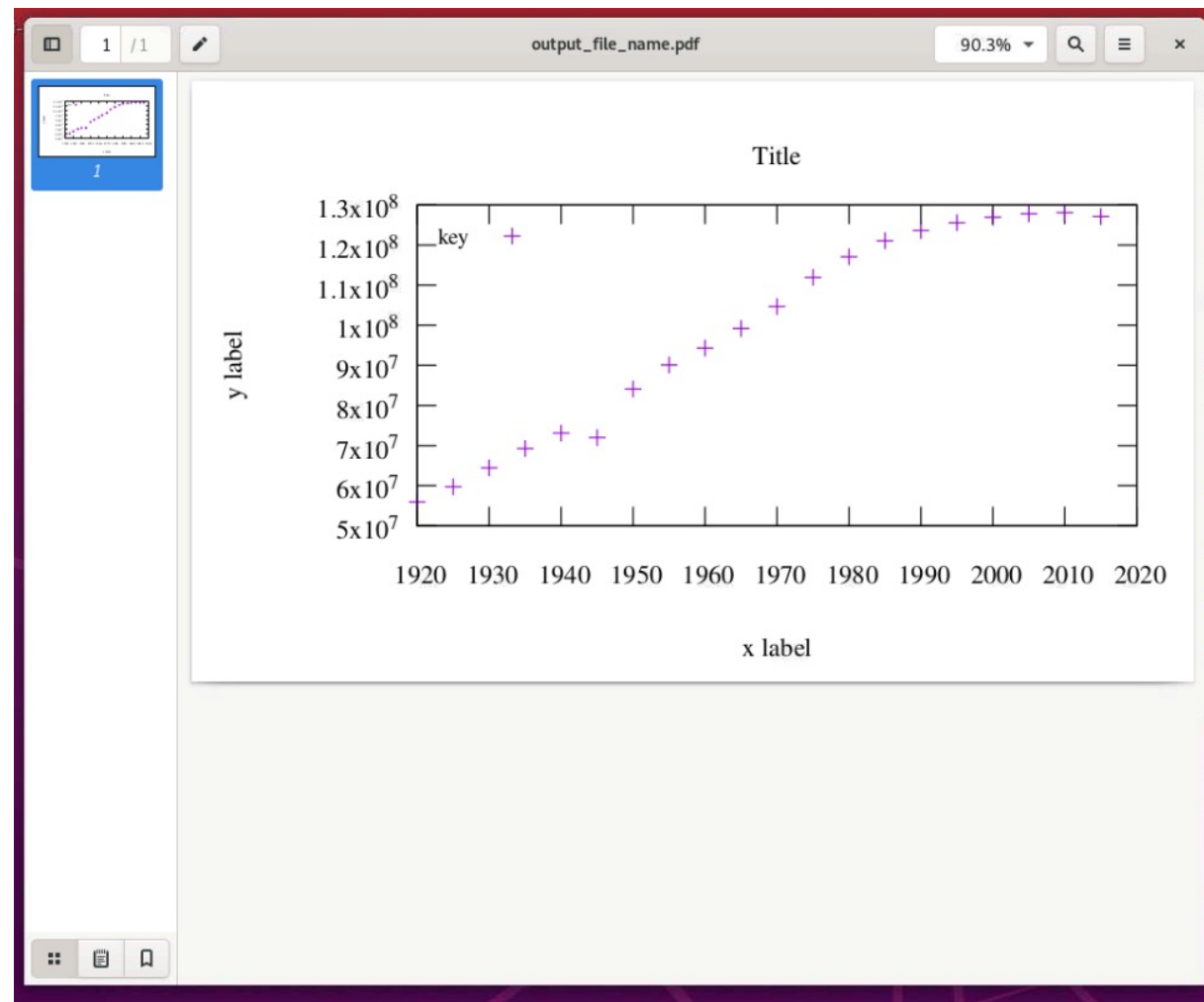
```
akiyama@:~/compphys2/lecture2/ex_gnuplot$ gnuplot sample.gpl
```



```
Visual Studio Code
File Edit Selection View Go Run Terminal Help
sample.gpl x
home > akiyama > compphys2 > lecture2 > ex_gnuplot > sample.gpl
1 reset
2 set terminal pdfcairo color enhanced font "Helvetica,18"
3 set output "output_file_name.pdf"
4 set title font "Times,14"
5 set title "Title"
6 set key font "Times,12"
7 set key top left
8 set xlabel "x label"
9 set ylabel "y label"
10 set xlabel font "Times,14"
11 set ylabel font "Times,14"
12 set xtics font "Times,14"
13 set ytics font "Times,14"
14 #set xrange [0:1]
15 #set yrange [0:1]
16 #set xtics 1
17 #set ytics 1
18 #set format x "%.1f"
19 #set format y "%.1t{/Symbol \264}10^{%T}"
20 #set logscale x
21 #set logscale y
22 plot "japan.dat" using 5:6 pt 1 lc 1 w p title "key"
23
24 set terminal pop
```

# 作成した図の確認

- “output\_file\_name.pdf”というファイルが出来たはず
- lsで確認せよ
- xdg-openで図を開いてみよ
- 右のような図ができましたか？





# バッチ処理形式が推奨される理由

- ・ スクリプトを見返せば, 図の作成に用いた数値データ（ファイル）が何だったか分かる
  - ・ 対話型ではこうした記録は残らない
  - ・ 「この図は何をプロットしたんだっけ？」という事態を防ぐことができる
  - ・ 図の再現性を担保することにつながる

## 図を改善していきましょう 1/2

- “sample.gpl”を適宜編集し, 図の見栄えを改善しましょう
  - “Title”, “x label”, “y label”, “key”に相応しい名前をつけましょう
  - 出力ファイル名を変更しましょう (何の図か分かるような名前をつけましょう)
  - 図のシンボルや色は整数値で指定されます
    - 違う整数値に変えてみましょう
- 総人口に加えて, 男性人口 (7列目), 女性人口 (8列目) の推移も可視化してみましょう
  - 1.22で, カンマ (,) で区切りながら付け足していけばOK

```
plot "japan.dat" using 5:6 pt 1 lc 1 w p title "key", "japan.dat" using 5:7 pt 2 lc 2 w p title "key",
```

- “key”にも相応しい名前をつけましょう

## 図を改善していきましょう 2/2

- “sample.gpl”を適宜編集し, 図の見栄えを改善しましょう
  - 配布したスクリプトではいくつかの行がコメントアウトしてあります
  - コメントアウトを外して (“set”直前の#を削除して) 軸の書式を変えてみましょう

## ここまで終わったら

- ・ お疲れ様でした
- ・ 時間に余裕のある人は, “population.dat”から好きな都道府県データを抜き出し, 人口の推移を可視化してみましょう
- ・ [宿題] gnuplotで使用可能なシンボル, 色と整数値の対応をネットで調べてみる