ICE計算機システムガイダンス およびPython概要

名古屋大学 情報基盤センター
情報基盤ネットワーク研究部門
基盤ネットワーク研究グループ
嶋田 創

概要

- Python概要
- ICE計算機システムガイダンス
  - GNOMEデスクトップ
  - 端末エミュレータ
  - Visual Studio Code (VS Code)
  - (Emacs)

軽量プログラミング言語

スクリプト言語の一種で、特に、プログラムを構成するコードの取り回しに優れるもの

- 特徴
  - 変数のタイプに自由度がある(動的型付け)
    - 普通の言語は事前に「整数」、「小数(浮動小数点型)」、「文字列」などを決めないとだめ
    - ただし、必要に応じて変換処理は必要
  - 文字列処理(切り出し、置換、連結、など)が強力
  - 逐次機械語に変換されながら実行するので処理速度は遅め(インタプリタ型)
- Perl/PHP/Pythonなど名前がPで始まる物が多いため、P言語と呼ばれることもある
- 他の2学科の講義ではRubyを利用

プログラミング言語Pythonと特長

- インデントをベースにプログラム構造を組む
  - インデント(字下げ): 字下げの量が同じ所は同じレベルのプログラム構造に所属(後述するループとか条件分岐とか)
- 「1つのことを複数の手段で実現できる」ことをあまり良しとしていない
  - 誰が書いても同じような記述になる →他のプログラムを参考にしやすい
- 軽量言語の中では比較的後発 →改良されている点が多い
- 新しいアルゴリズムなどのライブラリが早く実装されることが多い(機械学習とか深層学習とか)
  - TensorflowとかPyTorchとかChainerとかCaffeとかScikit-learnとか

Pythonプログラムの例

```
from easygui import *
import random

counter = 0
number = int(random.random() * 100)
message = 'I generated random number from 0 to 99. Estimate it and enter it!'
message_small = 'Too small! Estimate it again and enter it!'
message_large = 'Too large! Estimate it again and enter it!'
window_title = 'Large / small game'
in_key = enterbox(message, window_title)
in_key = int(in_key)

while 1 :
    counter = counter + 1
    if number < in_key :
        in_key = enterbox(message_large, window_title)
        in_key = int(in_key)
    elif number > in_key :
        in_key = enterbox(message_small, window_title)
        in_key = int(in_key)
    else :
        final_message = 'True! The number is %d. Total trial is %d.' % (number, counter)
        msgbox (final_message)
        break
```

Python 2とPython 3

- 世の中はとっくにPython 3に移行している
  - 現在の最新は2024/10リリースのPython 3.13
- でも、Python 2がかなり長期間(20年近く)使われていたため、Python 2の情報もWebなどにしぶとく残っている
  - Python 2は2020/1でサポート終了
- Python 3でいろいろと書式が代わったものもあります
  - 例: 文字列を表示するprintコマンド(関数)
    - Python 2: print "Hello world"
    - Python 3: print ("Hello world!")

→外部資料を参照する時には、Python 2時代の資料ではないか要注意

ICE計算機におけるPython2と3

- Python 2.7とPython 3.9がインストールされている
  - /usr/bin/pythonがPython 2.7、/usr/local/bin/python3がPython 3.9
  - ICE計算機のCentOS7系列(2014年リリース)は保守的が売りなため
- 初期状態で、シェルのコマンドラインから「python」とだけ打つと、2.7の方が呼び出されてしまう

  →alias python='python3'を設定しておいた方が良い
  - バージョンの確認: シェルから「python --version」と入力
- パッケージ(ライブラリ等)管理のpipは3側しか存在しない
  - 本来管理者権限が必要なツールなので、普通に新規にパッケージをインストールしようとすると怒られる
  - --userオプションをつけると$HOME/.localにパッケージを入れることができる

Python(プログラム)の実行

たいていはスクリプトを書いて1もしくは2の形で実行(2を推奨)

1. シェルからPythonインタプリタとプログラムを呼び出す
   - 例:「python3 hoge.py」の形で実行
2. Pythonプログラム冒頭にインタプリタ呼び出し文を追加し、プログラム自体に実行属性を追加後、シェルから呼び出す
   - プログラム冒頭(1行目)に「#!/bin/env python3」の行を追加
   - シェルから「chmod +x hoge.py」の形で実行属性を追加
   - シェルから「./hoge.py」の形で実行
3. シェルからpythonインタプリタを対話モードで呼び出してプログラムを1行1行入力して実行する
   - シェルから「python3」だけを入力すると対話モードで呼び出せる
     - そのまま数字と演算子(+ - * /)などを入力して(関数)電卓として利用も可能

Python関係のおすすめ参考書

- クジラ飛行机著, "実践力を身につけるPythonの教科書 第2版," マイナビBOOKS, 2024年9月.
  - この講義で強くお勧めする参考書
  - 入門者向け書籍(**プログラミング経験の無い人は購入を強く勧める**)
  - ある程度プログラミング言語の利用経験がある人は、こちらよりもより深い内容を扱っている書籍がお勧め
- 柴田淳著, "みんなのPython 第4版," SBクリエイティブ, 2016年12月.
  - 機械学習などの発展的な話も多い書籍
- Bill Lubanovic著, "入門 Python 3 第2版," オライリー・ジャパン, 2021年3月.
  - プログラミングやシステム構築の書籍として有名なO'Reilly Books
  - 技術的な詳細などについても、細かく正確に記されている

概要

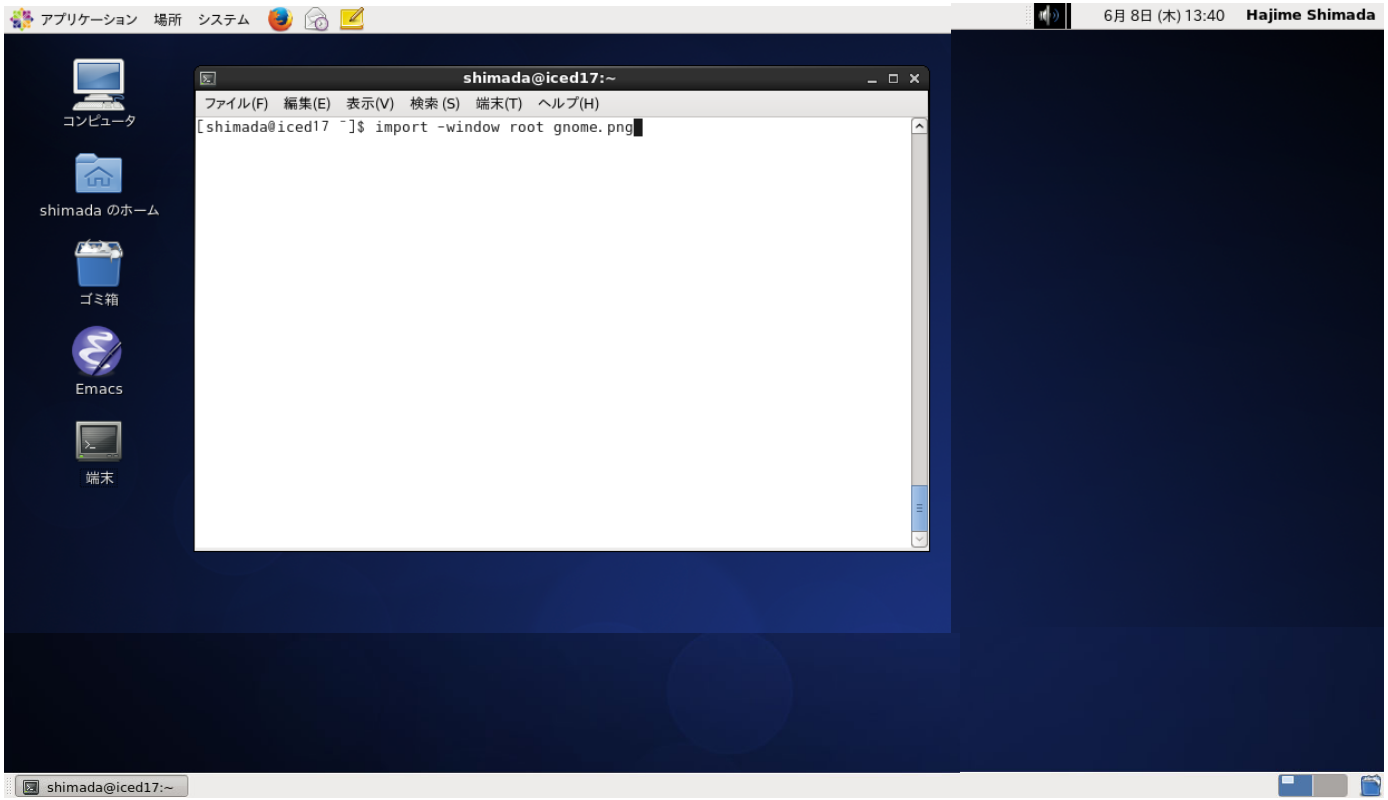
- Python概要
- ICE計算機システムガイダンス
  - GNOMEデスクトップ
  - 端末エミュレータ
  - Visual Studio Code (VS Code)
  - (Emacs)

ICE計算機システムについて復習

- 端末の仕様
  - Linux (CentOS 7.8) (多分みなさんの在学中に次期システムに)
  - デスクトップはGNOME
- 端末台数
  - 295演習室: 70台程度
    - 基本的に講義時以外は端末の電源は入っていない
    - CAD演習室の端末が利用者で埋まっている場合は、283号室(計算機準備室)にいる技官の雨宮さんにお願いして電源を入れてもらう
  - CAD演習室: 20台程度
    - 24時間いつでも使える
    - IB館南棟の解錠時間は8:00-20:00なので、時間外はカードリーダに学生証をかざして認証して建物に入る

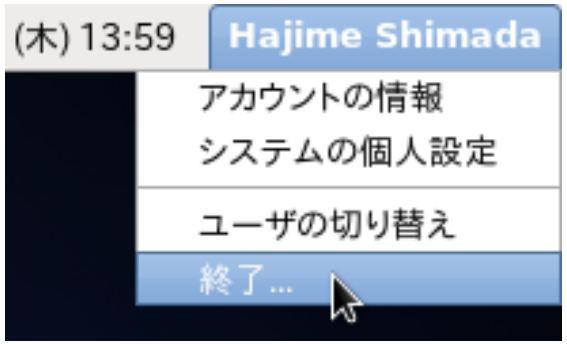
GNOMEデスクトップ

- Linuxのデスクトップ環境の1つ
  - Linuxはユーザがいろいろなデスクトップ環境を選べる(KDEなど)
  - ただし、現状のICEで使えるのはGNOMEだけ

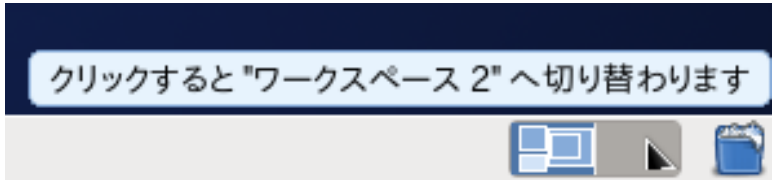


GNOMEデスクトップ操作の概要

- アプリケーション起動は左上の階層メニューより
  - 入れて欲しいアプリケーションがあれば技官の方まで要望を
- 設定も左上の階層メニューより
- ログアウトは右上より「終了」
- 起動中のアプリケーションは下にリストアップされる
  - (Shift +)Alt + Tabで順番に切替可能
- 仮想デスクトップ
  - Ctrl + Alt + 矢印キーで移動
  - Shift + Alt + 矢印キーで選択中ウィンドウを別ワークスペースに移動



ログアウト



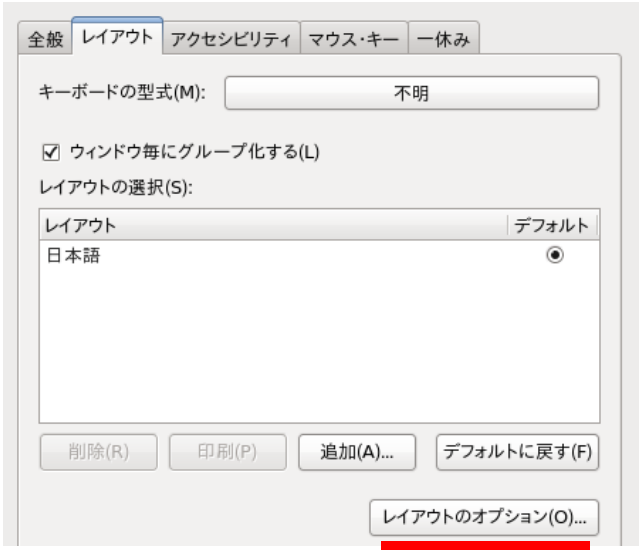
仮想デスクトップマネージャ
(画面右下)

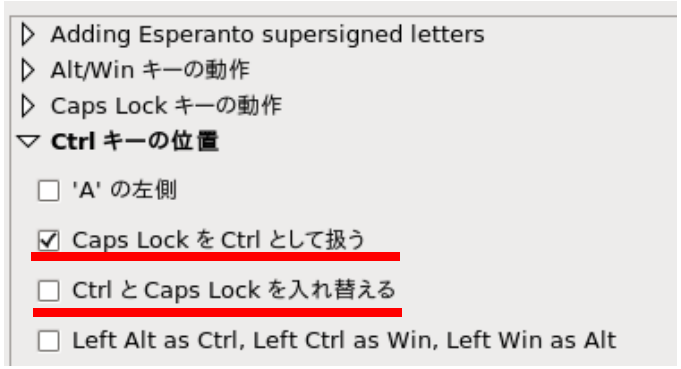
日本語入力の初期設定

- これまたLinuxは自由度が無駄に高い
  - 変換中の文字の画面への表示方法やウィンドウへの渡し方を司る「インプットメソッド(入力メソッド)」
  - かな漢字変換を司る日本語入力エンジン(インプットメソッド)
  - 注: 括弧内がGNOMEの日本語表記ですが、訳が変です
- 最初に設定しないと日本語が利用できません
  - Firefoxを立ち上げるとICEのページが出てくるはずなので、そこから「初めてのユーザー（新2年生、編入生）の皆さんへ」のリンクを選択
    - 入力メソッドにibus、インプットメソッドにmozcを設定
      - ICEのページだとインプットメソッドにAnthyを設定と書いてあるが情報が古い
  - 設定したら、Firefoxの検索窓などで「全角/半角」キーを押して日本語入力ができるか確認

キー入力の設定

- Ctrlキーを多様するので、Ctrlキーを使いやすいようにしましょう
  - システム→設定→キーボード→レイアウトのオプションで設定可能
    - オススメ: Caps LockキーをCtrlキーに変更
- さくさくプログラムを書けるよう、ブラインドタッチを練習しましょう
  - 記号や数字なども含めて練習
  - 入力速度測定サイトとかで数字を見ると練習の励みになります
    - http://typing.twi1.me/
    - http://zty.pe/





この講義で使うアプリケーション

アイコンをデスクトップや上部のバーに設置しておくと便利

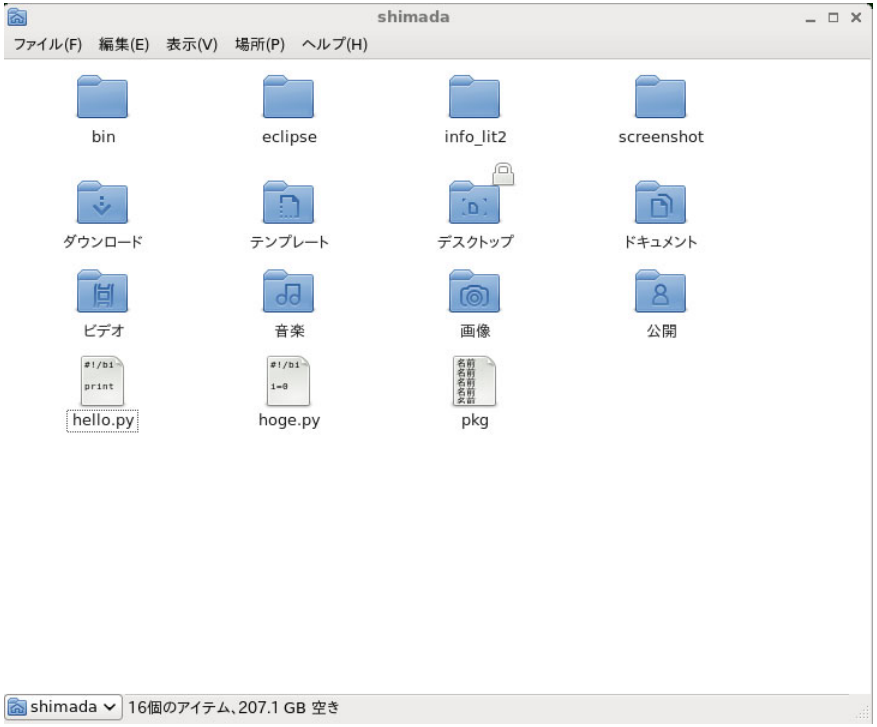
- ウェブブラウザ Firefox
- ファイルマネージャ nautilus
  - (シェルからのファイル管理に慣れて欲しいのでなるべく使わないで)
  - nautilusからPythonプログラムをダブルクリックするとgeditというエディタが立ち上がるが、**使ってはいけない**
- 端末エミュレータ gnome-terminal
  - シェルからのPythonプログラムの実行
- 統合開発環境 Visual Studio Code (VS Code)
  - Pythonプログラムの記述
  - Emacs, Vimや, Eclipseもあるので、自己責任で好きな環境で
    - どのサーバでもまず入っているvi系は一度利用を試みてもらいたい
    - 以前はEmacsを基本としていた

Firefox

- Add-onでの拡張が便利なウェブブラウザ
- メニューバーはAltキーを押すと出る

ファイルマネージャ nautilus

- Windowsのエクスプローラなどと同様の形で操作できる
- 個人用データはファイルサーバにあるので、そこをトップにしたショートカットが準備されている
- Pythonプログラムをダブルクリックするとgeditというエディタが立ち上がるが、**使ってはいけない**

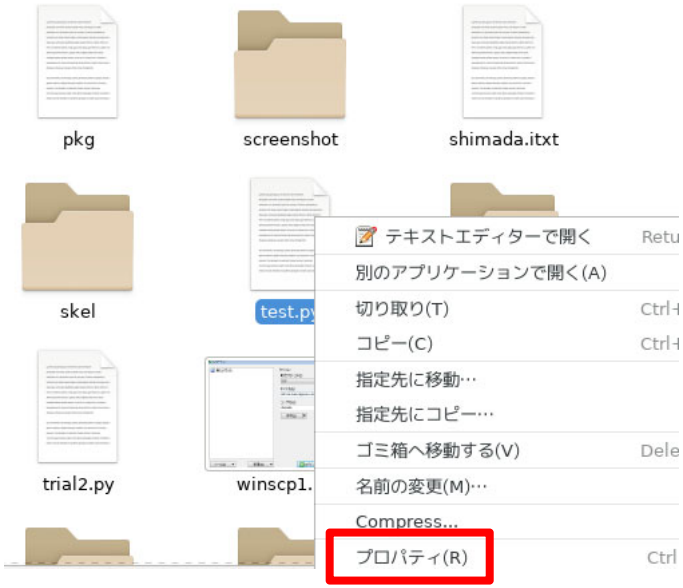


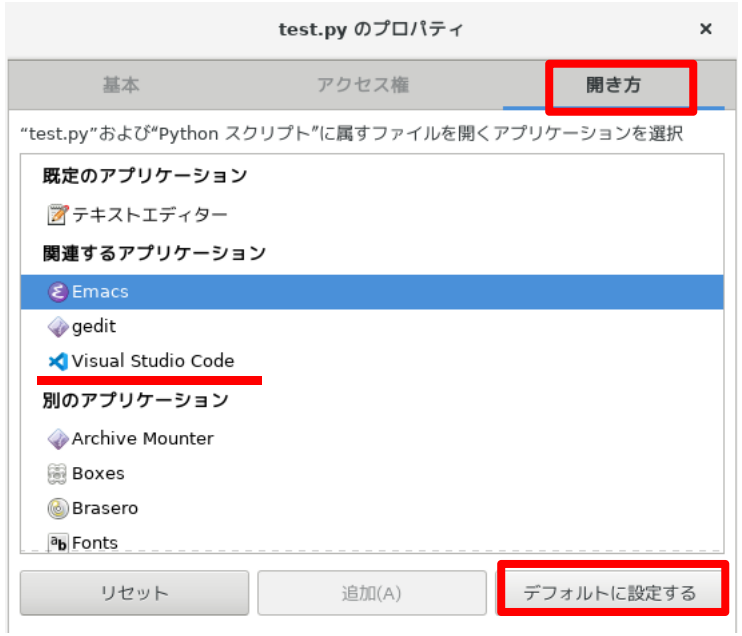
ファイル
マネージャ
nautilus

デスクトップ上の
ショートカット

nautilusでのファイルとアプリの関連付け変更

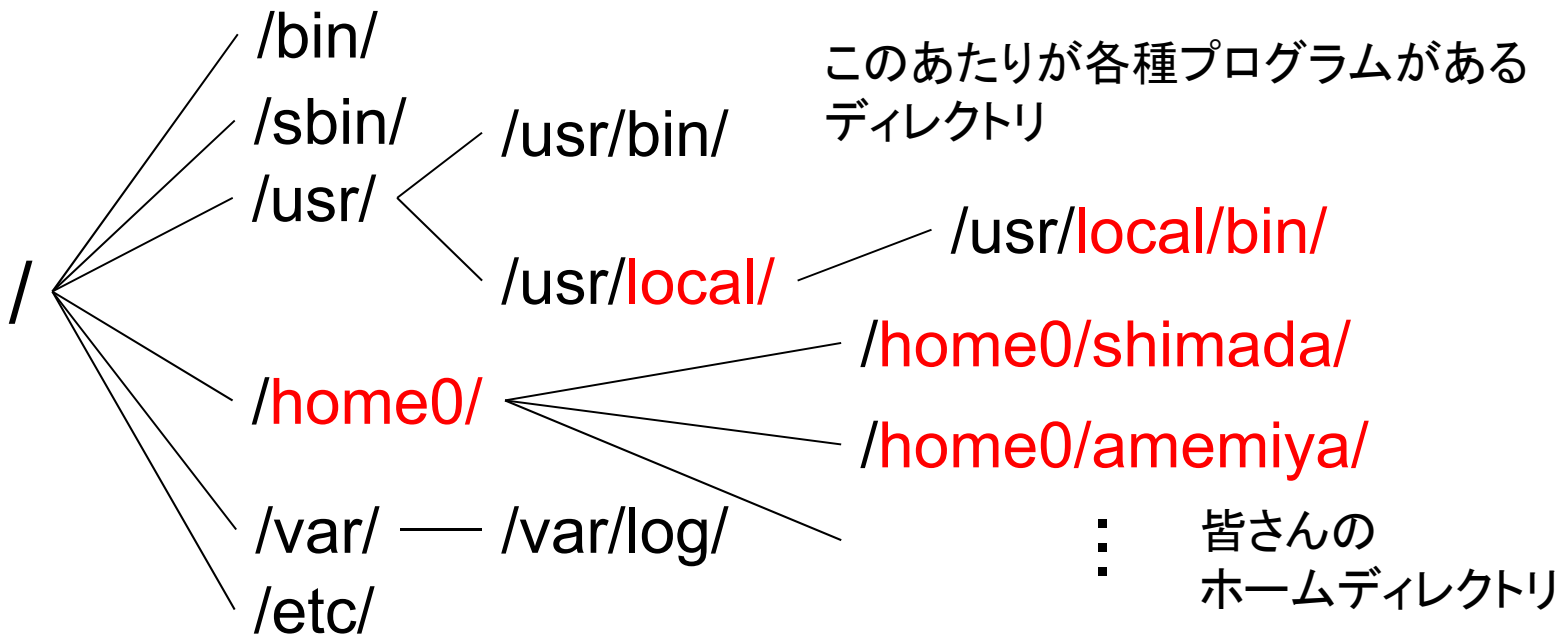
1. .pyファイルを選択して右クリックメニューを開く
2. 「プロパティ」を選択
3. 「開き方」のタブを選択
4. 「Visual Studio Code」を選択して「デフォルトに設定する」ボタンを押す





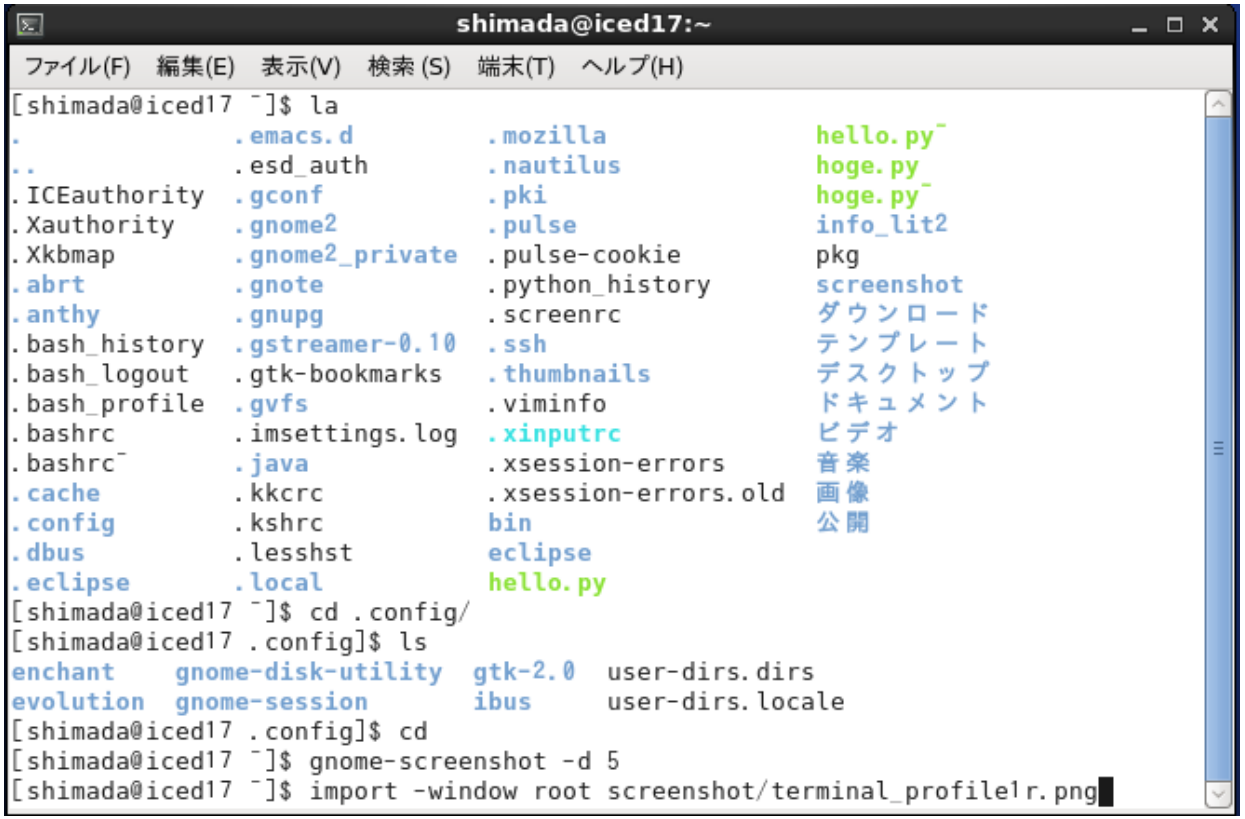
Linuxのディレクトリ構造

- ルートディレクトリ(/)からの木構造を取る
- USBメモリやファイルサーバのディスクなどはディレクトリを接ぎ木する形で接続
  - ICEでは赤文字の部分はファイルサーバに存在



端末(ターミナル)エミュレータ

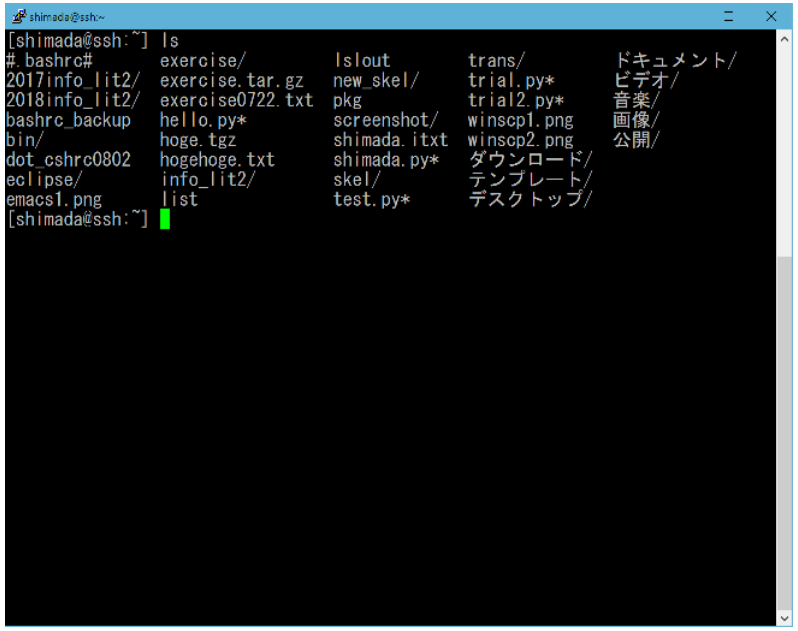
- プログラム、ファイル、オプションを指定して処理を実行可能
- ターミナルでコマンドを受け付けているのはシェル(shell)
  - OSから見て、ユーザの指示を受け付ける外殻なのでshell



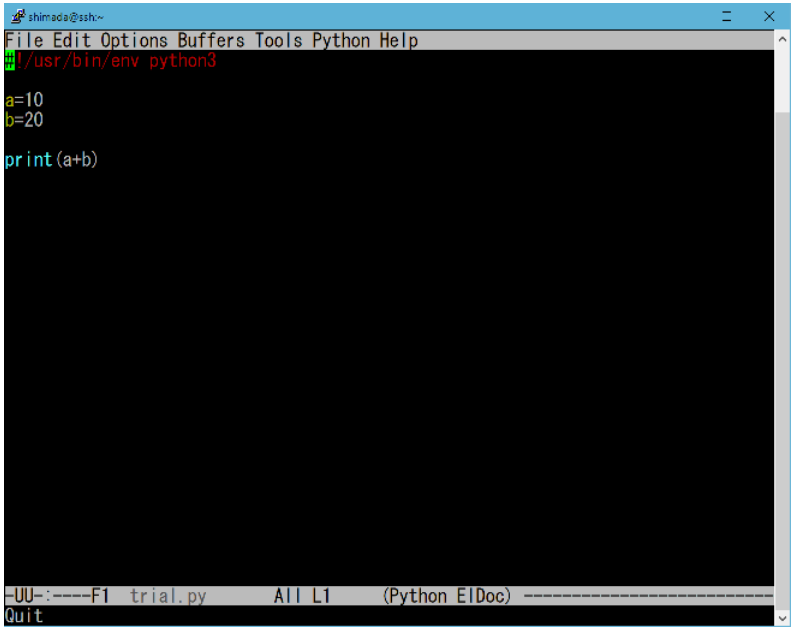
SSHで遠隔でICE計算機のターミナルに接続した時の操作(1/2)

- やり方は、TACTのお知らせにある「外部からICE計算機の端末エミュレータ(+Emacs)利用方法…」を参照して下さい
- 「emacs -nw」でターミナル内でEmacs起動
- Emacs動作中に「Ctrl+z」で中断したターミナルに戻る
- 中断したターミナルは「fg」や「%数字(中断中ジョブ番号)で呼び出し

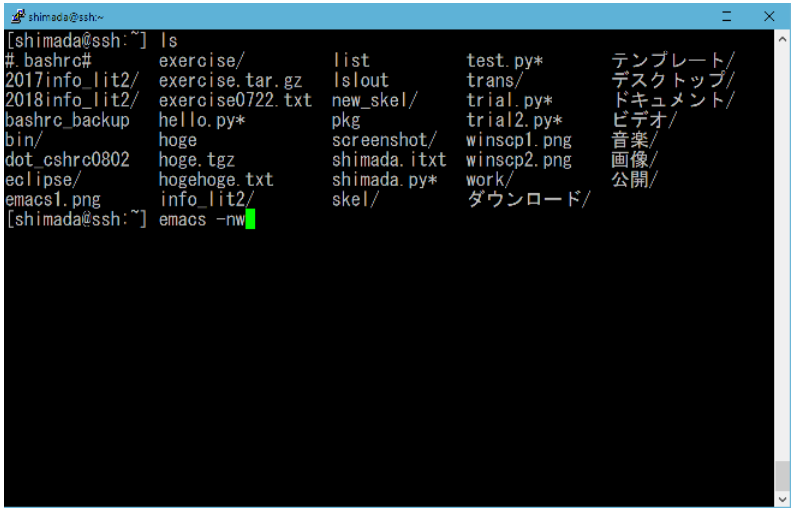
puttyで接続した時のターミナル



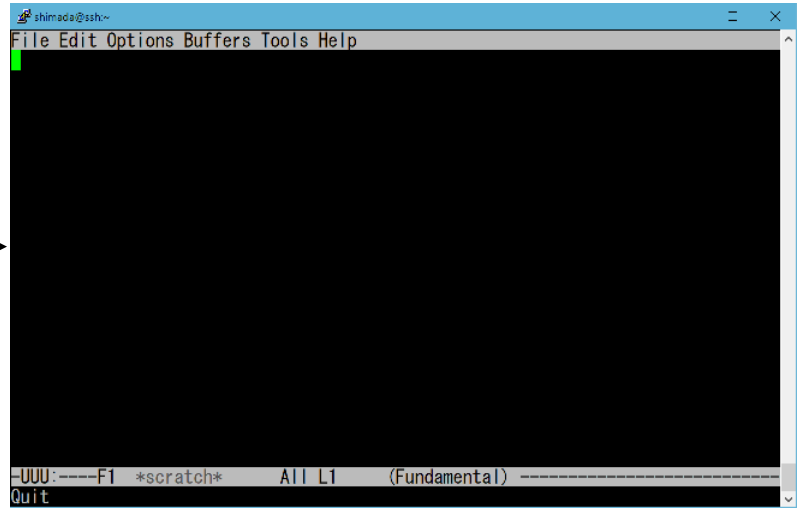
ターミナル上で立ち上げたEmacs



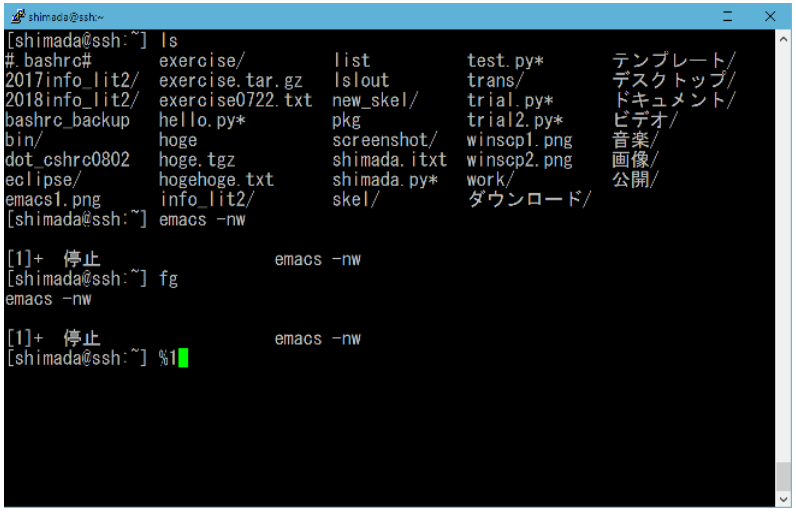
SSHで遠隔でICE計算機のターミナルに接続した時の操作(2/2)



Emacs
立ち上げ



Ctrl+z



「fg」もしくは「%1」

- %の後の数字は中断時に表示されたジョブ番号
- ターミナル上から「jobs」で再確認可能

シェル

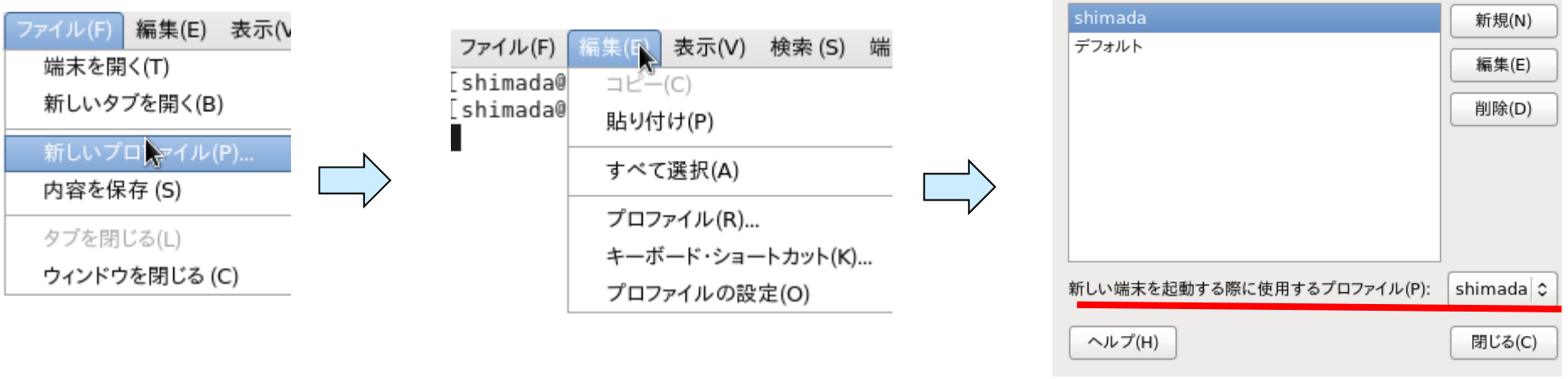
- コマンドラインから処理を実行する機構は他のOSにもある
  - WindowsのコマンドプロンプトやPowerShell
  - Mac OSのターミナル
- シェルにもいろいろ種類がある
  - 標準ではLinux標準のbash(Bone Again shell)
  - ICEへインストール済みではcsh(tcsh)がある
  - ypchshコマンドで変更できます
- シェル上のコマンドやプログラムの連続実行処理をプログラミング可能(シェルスクリプト)

```
[shimada@iced17 ~]$ cd .config/
[shimada@iced17 .config]$ ls
enchant    gnome-disk-utility  gtk-2.0  user-dirs.dirs
evolution  gnome-session       ibus     user-dirs.locale
[shimada@iced17 .config]$ cd
[shimada@iced17 ~]$ gnome-screenshot -d 5
[shimada@iced17 ~]$ import -window root screenshot/terminal_profile1r.png
```

端末エミュレータの便利な使い方

- タブを使うと複数のシェルを1つのウィンドウで開けます

- 端末エミュレータをカスタマイズするならプロファイルを作成
  - プロファイルを設定した上で「新しく端末を起動する際に…」で指定



シェルでのショートカット

- Tabキーで入力の補完が効く
  - 例:「pyt」とまで入力してTab　→後ろが補完されて「python」となる
  - 候補が複数ある場合、2回Tabを押すと候補がリストアップされる
    - 空白状態でTabを2回押すと候補ファイルがリストアップされる
    - 例: pythonと入力してTab　→pythonで始まるコマンドがリストアップ
- 上下左右矢印キーやBackspaceキーで入力を編集可能
  - 上下は過去の入力履歴の参照
    - 前に実行したコマンドの一部を改変して実行可能
  - Ctrl + p(prev)/n(next)/f(forward)/b(backward)で矢印とおなじになるショートカットもある
    - Meta(Alt)+p/n/f/bで1単語ごとに移動するショートカット

シェルからよく利用するコマンド(=プログラム) (1/6)

略称が変なのは左右の手で分担して打ちやすくするため

- ls(LiSt): 現在のディレクトリ内のファイルやサブディレクトリの表示
  - ファイル名などを指定すれば、それだけを表示
    - ワイルドカードも使用可能(例: ls *.pyで.pyで終わるファイルのみを表示)
  - オプション -l は更新日時なども表示(例: ls -l)
  - オプション -aは隠しファイルも表示(例: ls -a)
  - オプション -Fはディレクトリや実行ファイルに添字追加(例: ls -F)
- rm(ReMove): ファイルを消す(例: rm hoge.txt)
  - ワイルドカードも利用可能(例: rm *.txt)
  - オプション -iをつけるとyes/noを聞いてくれる(例: rm -i hoge.txt)
- touch: 空ファイルを作成する(他の使い方の方が重要だが)

シェルからよく利用するコマンド(=プログラム) (2/6)

- mv(MoVe): ファイルやディレクトリの移動や名前変更
  - 別ディレクトリ中(fuga)に移動: mv hoge.py fuga/
  - 別名に変更: mv hoge.py fuga.py
    - すでにその名前のファイルがあると上書きになる点に注意
- cd(Change Directory): 他のディレクトリに移動(例: cd デスクトップ)
  - 何も指定しないとホームディレクトリ(/home0/xxxx/)に移動する
    - ホームディレクトリは$HOMEとか~とかでも表される
  - 上のディレクトリは「..」で指定できる(例: cd ..)
  - 複数の移動も一度にできる
    - 例: cd ../..
    - 例: cd ../../usr/bin
  - ルートディレクトリからの絶対位置でも移動先を指定可能(例: cd /home0/shimada)

シェルからよく利用するコマンド(=プログラム) (3/6)

- mkdir(MaKe DIRectory): ディレクトリを作成(mkdir hoge)
- rmdir(ReMove DIRectory): ディレクトリを削除(rmdir hoge)
  - ディレクトリの中身が空でないと削除できない
    - 「中身が空なのに消せない」と思ったら、中に隠しファイルがあったり…
  - ディレクトリの中身ごと消す時には「rm -r ディレクトリ名」を利用
- pwd(Print Working Directory): 現在のディレクトリを表示
- less: テキストファイルを端末エミュレータ上に表示(例: less hoge.txt)
  - 上/下矢印キー(もしくはj/kキー)で上下にスクロール可能
  - スペース/bキーで1ページ単位で進む/戻ることも可能
  - /キー(前方検索)や?キー(後方検索)で検索をかけれる
    - 次候補はnキー、前候補はpキー

シェルからよく利用するコマンド(=プログラム) (4/6)

- alias: コマンドなどのショートカット文字列を作成(例: alias py="python3")
  - オプションも含めて設定可能(例: alias ll="ls -l")
  - 再帰的な設定も可能(例:「alias py="python"」+「alias python="python3"」で「py="python3"」になる)
  - 「alias」だけ入力してリターンを押すと、現在のalias設定一覧を見れる
- unalias: aliasの解除(例: unalias py)
- source: シェルに設定ファイルを読み込ませる(例: source .bashrc)
  - ピリオド1文字で代理も可能(例: . .bashrc)

シェルからよく利用するコマンド(=プログラム) (5/6)

- grep: テキストファイルから文字列一致を検索
  - 例: grep print hoge.py →hoge.pyというファイルからprintという文字が含まれている行を表示
  - 結果は行単位で表示される
  - ワイルドカードを使って複数ファイルを検索可能
    - 例: grep print *.py (.pyで終わるファイルそれぞれから....)
    - 「どのファイルで一致したか」も表示してくれる
- find: ファイルやディレクトリの検索(例: find ./ -name "*.py")
  - 最初の./は「現在のディレクトリ」を示し、それより下を検索
  - -nameの後ろに検索キーワードを指定
  - 「ある日時より前/後」「サイズxx以上/以下」「ディレクトリのみ」「一致した物に別コマンド適用(削除など)」など多彩なオプションを利用可能

シェルからよく利用するコマンド(=プログラム) (6/6)

- du(Disk Usage): ファイズサイズの合計を調査
  - 現在のディレクトリから下を調べる(ディレクトリごとの小計あり)
  - ディレクトリを指定することも可能(例: du ダウンロード/)
- chmod(CHange MODe): ファイルの属性の変更
  - よくやるのは実行属性の付与(例: chmod +x hoge.py)
- man(MANnual): コマンドのマニュアルを表示(例: man ls)
  - オプション-kをつけるとコマンド説明に指定文字が入った検索を実行(例: man -k grep)
- which: コマンドの実体が何なのか(どこのディレクトリのどのファイルか)を表す(例: which ls)
  - aliasをごちゃごちゃ設定したりして訳が分からなくなった時に便利

コマンドの実行とパイプ

- コマンドを「|」でつなぐことで他のプログラムに実行結果を直接渡せる
  - 例: ls | less　→lsの結果をlessに受け渡す = lsの結果をスクロールさせたり検索をかけれたり
  - 例: ls | grep　→lsの結果をgrepに渡して特定のキーワードを含む物のみ表示
- 「>」の後ろにファイル名を記すことで、コマンドの実行結果(標準出力)をファイルに出力可能
  - 例: ls > filelist　→lsの結果をflielistというファイルに出力
  - 「>>」の形では既存のファイルがあればそれに追記
- 「< 」の後ろにファイル名を記すことで、ファイルの中身をコマンドへの標準入力として入力可能

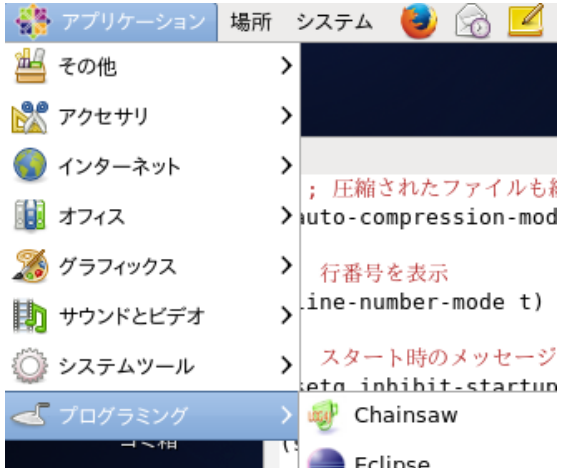
bashの設定ファイル

- bash起動時にはホームディレクトリの.bashrcを設定ファイルとして読む
  - 編集したら. .bashrcで読み込んで反映(個々のシェルごとに)
  - 編集失敗して焦ることもあるので、適時バックアップ用コピーは取っておく
  - 「.」で始まるファイルはLinuxでは隠しファイルになって見えないので注意
  - ホームディレクトリには他にも色々と設定ファイルが存在している
- 「#」以降の文字はコメントになる
- 不便な点気になったらちまちまとカスタマイズを追加すると良い

```
# .bashrc

alias rm="rm -i"
alias cp="cp -i"
alias mv="mv -i"

alias ls="ls -F"
alias la="ls -a"
alias ll="ls -l"
alias lla="ls -la"

alias python="python3"
alias py="python"

alias xv="display"

alias lv="less"
alias em="emacs -nw"
alias sdr="screen -d -r"

PATH=${HOME}/bin:$PATH
export PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

-:---   .bashrc          All L1     (Shell
Indentation setup for shell type bash
```

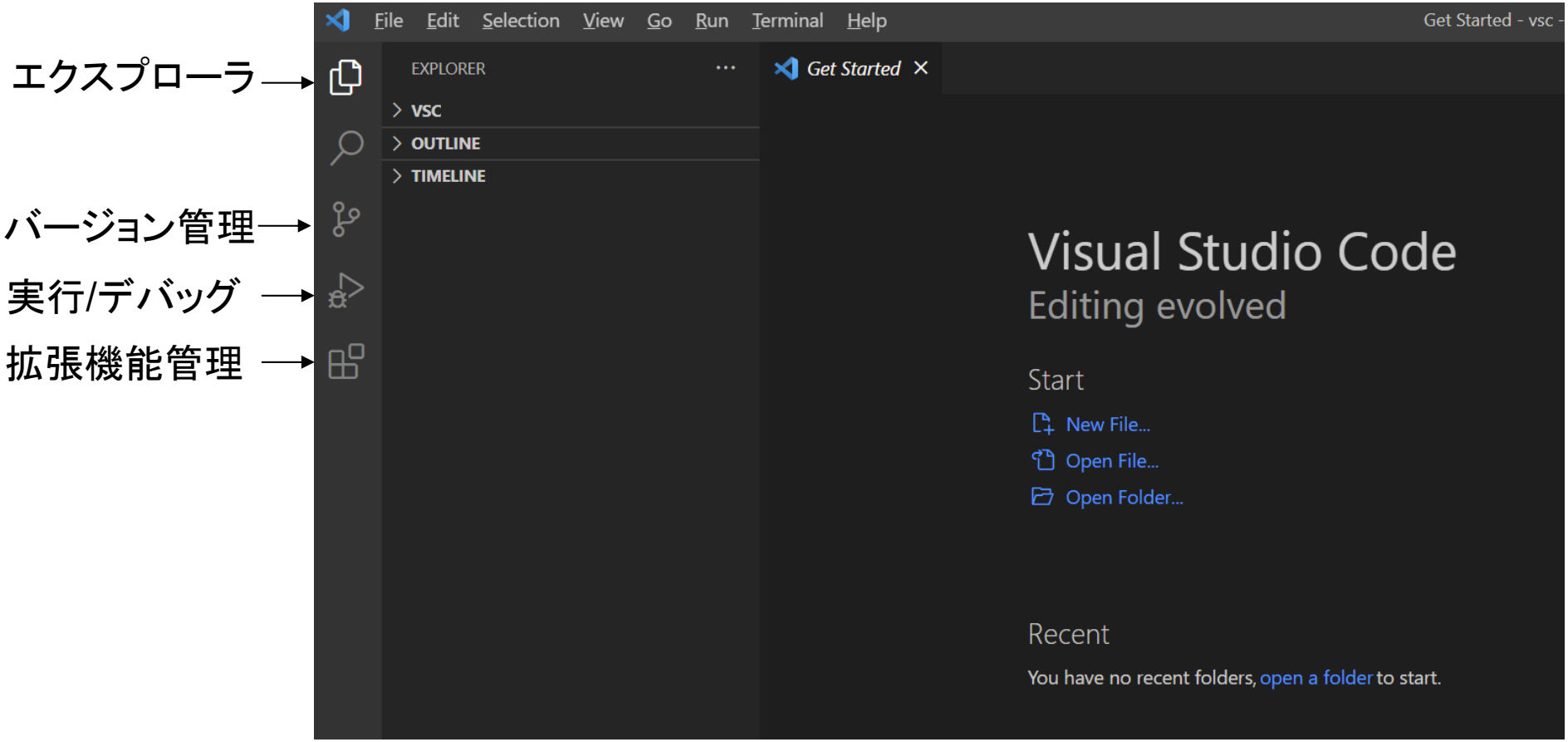
Visual Studio Code(1/8)

- 統合開発環境(IDE: Integrated Development Environment)
  - プログラムの開発に必要なものを詰め込んだもの
  - エディタ、シェル、デバッガ、バージョン管理などを含む
  - 多数の言語/OSに対応する物もあれば、特定の言語/OSのみ対応も
  - 機能拡張パックを追加できるものも
- Visual Studio Code (VS Code)はIDEの一種
  - 機能拡張が多くて様々な言語/OSで使いやすい
  - 他に有名なIDEはEclipse, Xcode, Android Studioなど
- ICEでの起動方法:
  アプリケーション→プログラミング
  →Visual Studio Code



Visual Studio Code(2/8)

- 初起動時はシンプル
  - メニューなどの表記は英語のみ、プログラミング言語パック無し



Visual Studio Code(3/8)

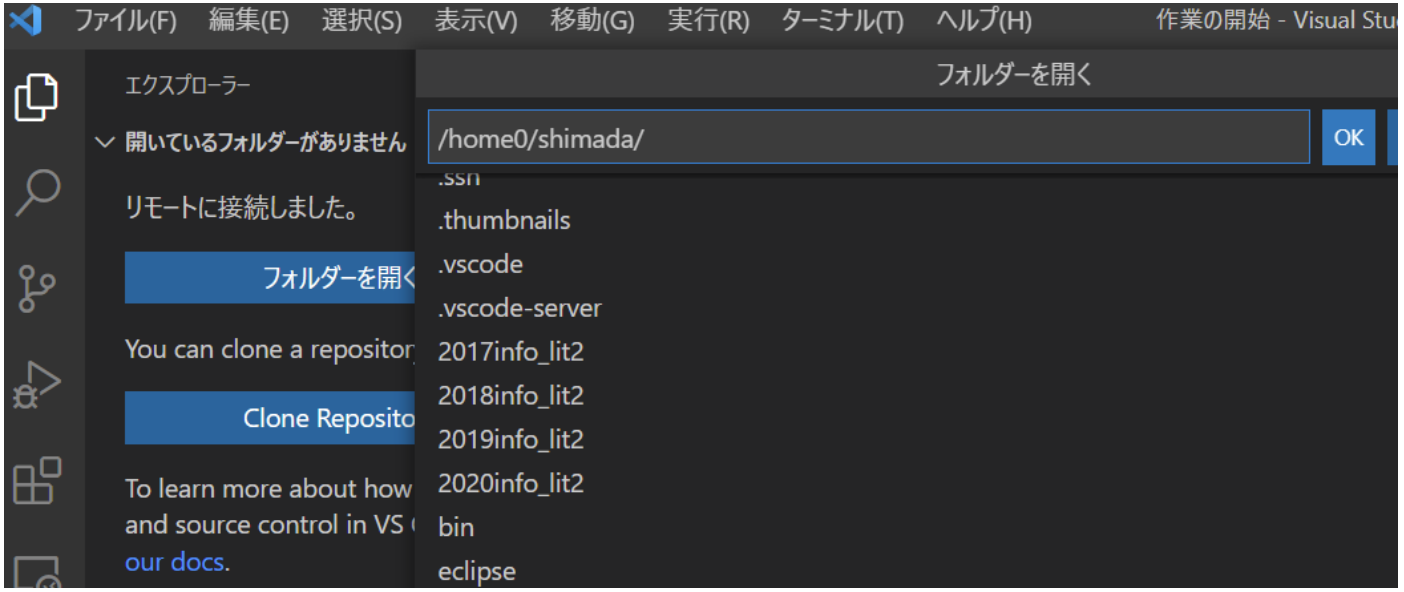
- まずは日本語化
  - 拡張機能で"Japanese"で検索 →Japanese Language Pack for VS Codeを入れる
  - VS Codeを再起動

Visual Studio Code(4/8)

- Python言語パックを入れる
  - 拡張機能を"Python"で検索 →Python Extension for Visual Studio Codeを入れる(まず一番上に出る)
    - 紛らわしいの多いので詳細説明や提供者を要確認

Visual Studio Code(5/8)

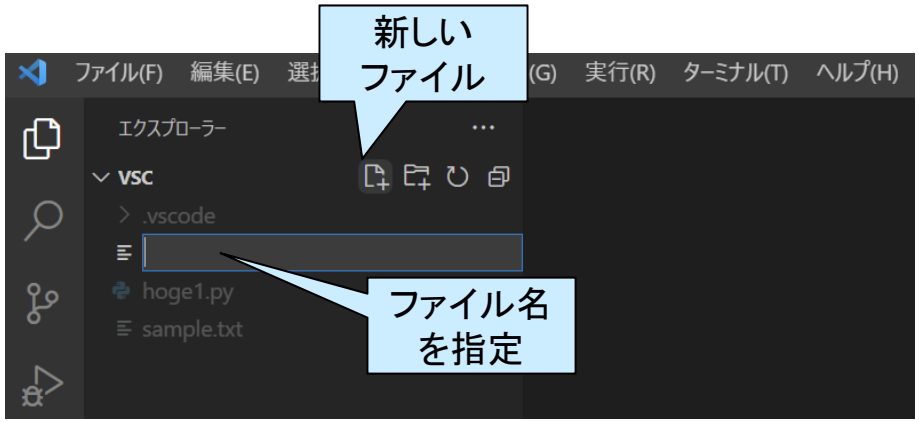
- エクスプローラを開くと、コードを置くディレクトリの指定を依頼される
- 適当に本講義用のディレクトリを作った上で指定
  - ホーム直下を指定することも可能だがおすすめしない
  - 複数のディレクトリを指定して、左のエクスプローラのウィンドウで切り替え可



Visual Studio Code(6/8)

コード書き

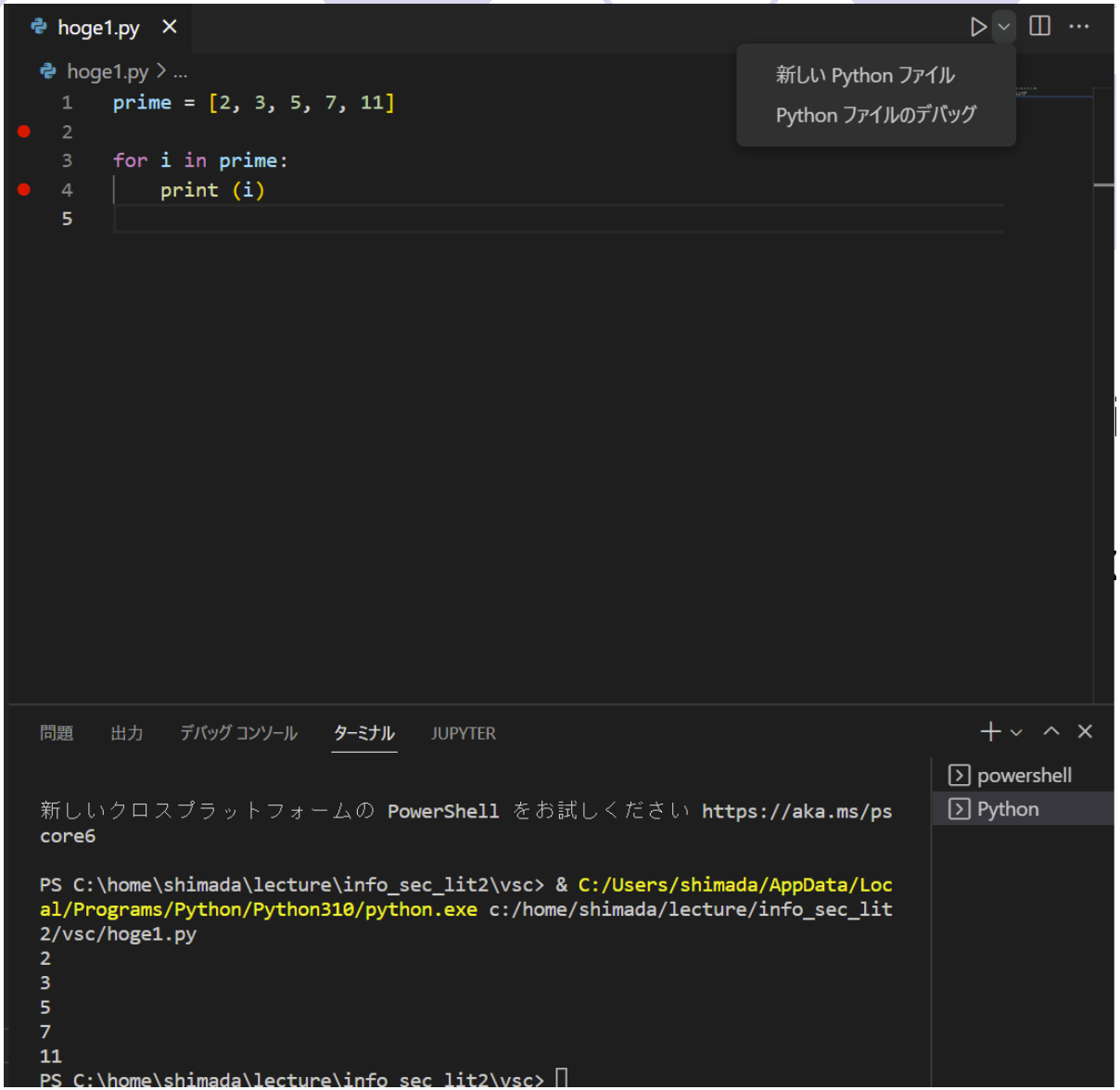
- エクスプローラで作業するディレクトリを選択
- 「新しいファイル」のアイコンを押すとファイル名を聞かれる
  - .pyで終わるファイルを作るとPythonコード扱いにされる
    - テキストエディタがそういうモードに入る
  - .txtで終わるファイルを作るとテキストファイル扱いにされる



Visual Studio Code(7/8)

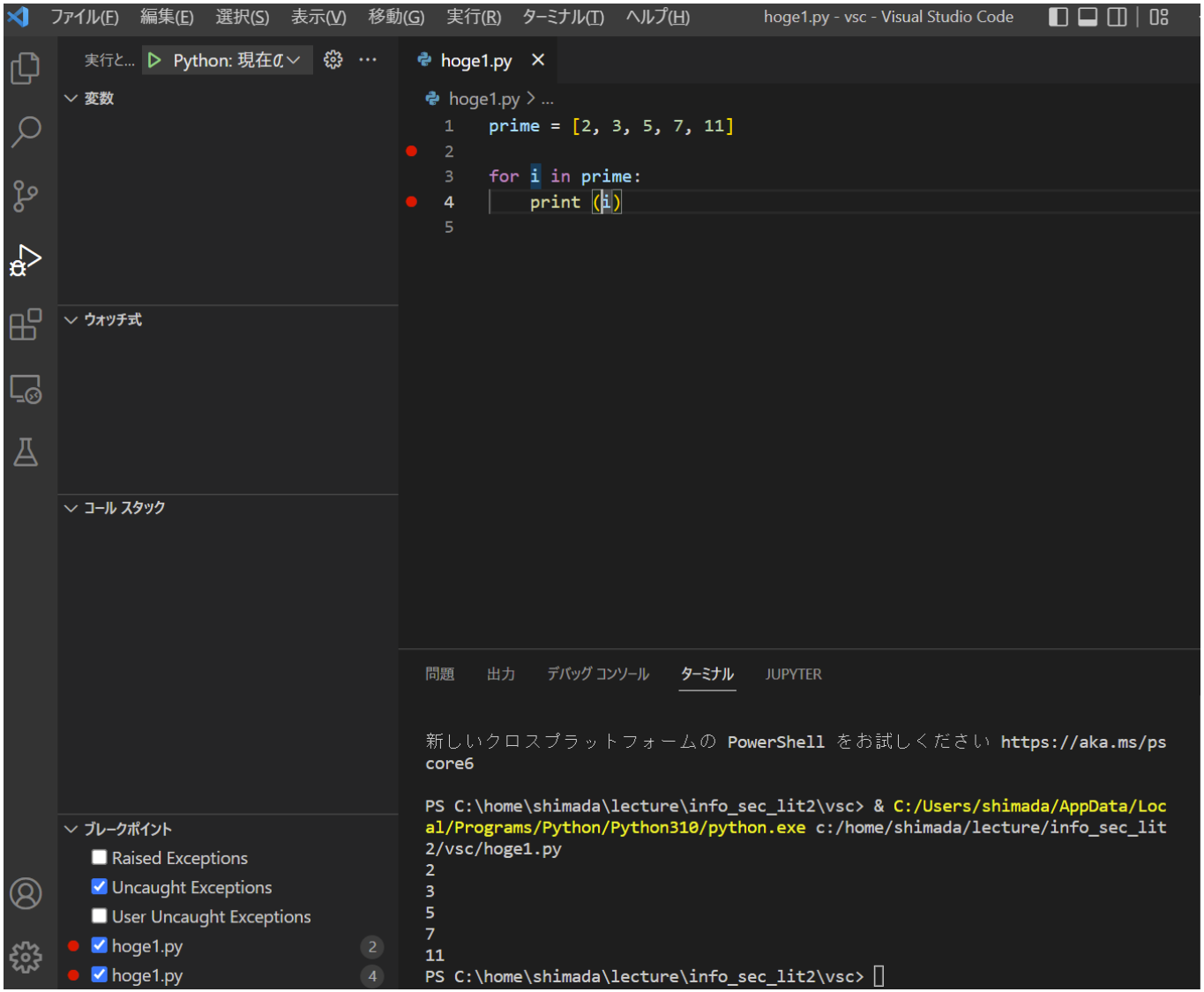
実行

- 右上の再生ボタンを押すと実行されるが、CS学科なら自分でターミナルのウィンドウのシェルから実行してほしい



Visual Studio Code(8/8)

実行とデバッグから、変数表示、ブレークポイント設定、ステップ実行などをやりつつのデバッグ実行も可能



Visual Studio CodeのTips

- 自分でローカルに環境を作るなら、Python Japanの「Visual Studio CodeでPython入門」のコンテンツがお勧め

  https://www.python.jp/python_vscode/windows/index.html
- 参考情報はいっぱいあるので、積極的に検索することを勧めるが、ちゃんと評判の良いサイトの情報を優先的に見ること
  - 例: @ITの「Visual Studio Codeで始めるPythonプログラミング」
    https://atmarkit.itmedia.co.jp/ait/series/10063/

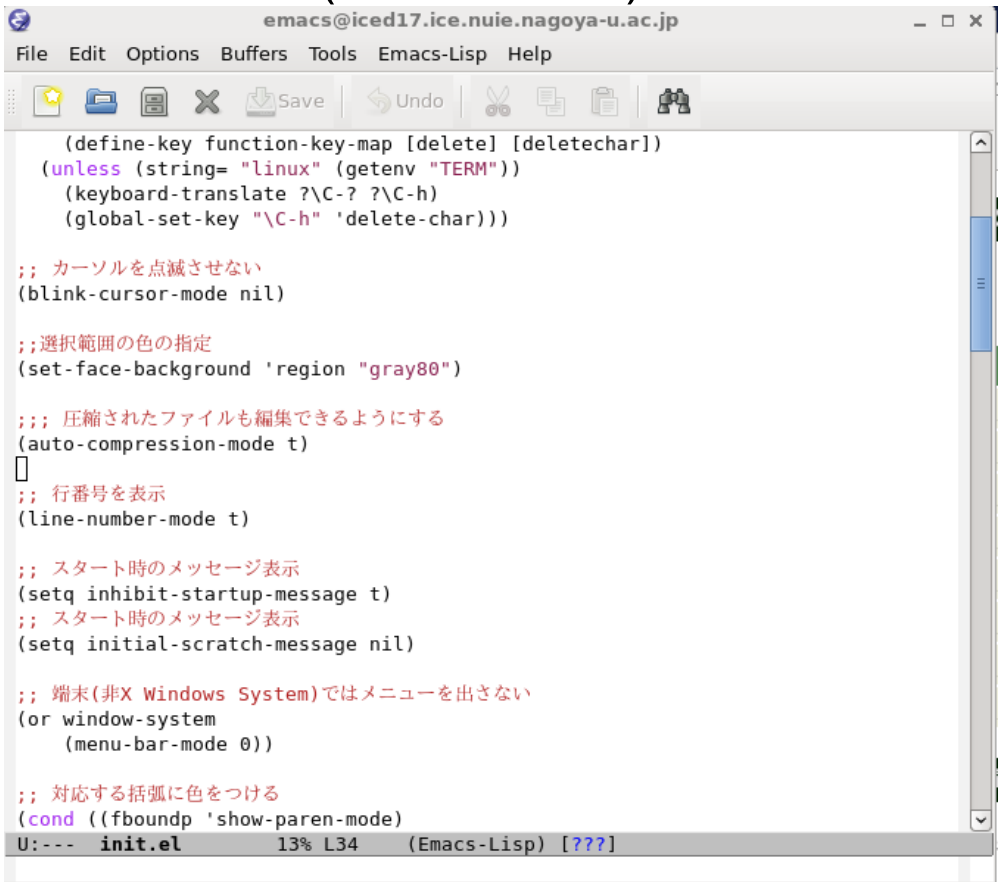
遠隔でのICE計算機の利用

- SSHを用いて外部から接続して利用可能
  - 細かなやり方はICE計算機のページ(講義ページよりリンク)の「プロバイダ経由でICEを使う方法」を参照(学内専用ページ)
  - SSHクライアントを使って端末エミュレータを利用可能
    - emacsはオプション-nwをつけて端末エミュレータ内部で起動
  - screen(GNU screen)を使えば、接続が切れても再接続可能
    - 複数の端末エミュレータを1ウィンドウで利用可能にもなる
  - おすすめSSHクライアント: putty, Tera Term(TTSSH)
- VS Code上からSSHでICE上のコードを遠隔で編集&実行(詳細は別資料)
  - 拡張機能SSH - Remoteをインストールして設定して利用
  - ICE上のコードを手元のVS Code上で違和感なく編集可能
  - 個人的には、シェル上での実行は別SSHクライアントに分業する方が便利

Emacs(高機能テキストエディタ)

(以下、以前はEmacsの利用を基本としていた時の資料を参考までに)

- カスタマイズに優れた高機能テキストエディタ
- テキストエディタ宗教戦争の当事者の片方(もう片方はVi)



Emacsの特長

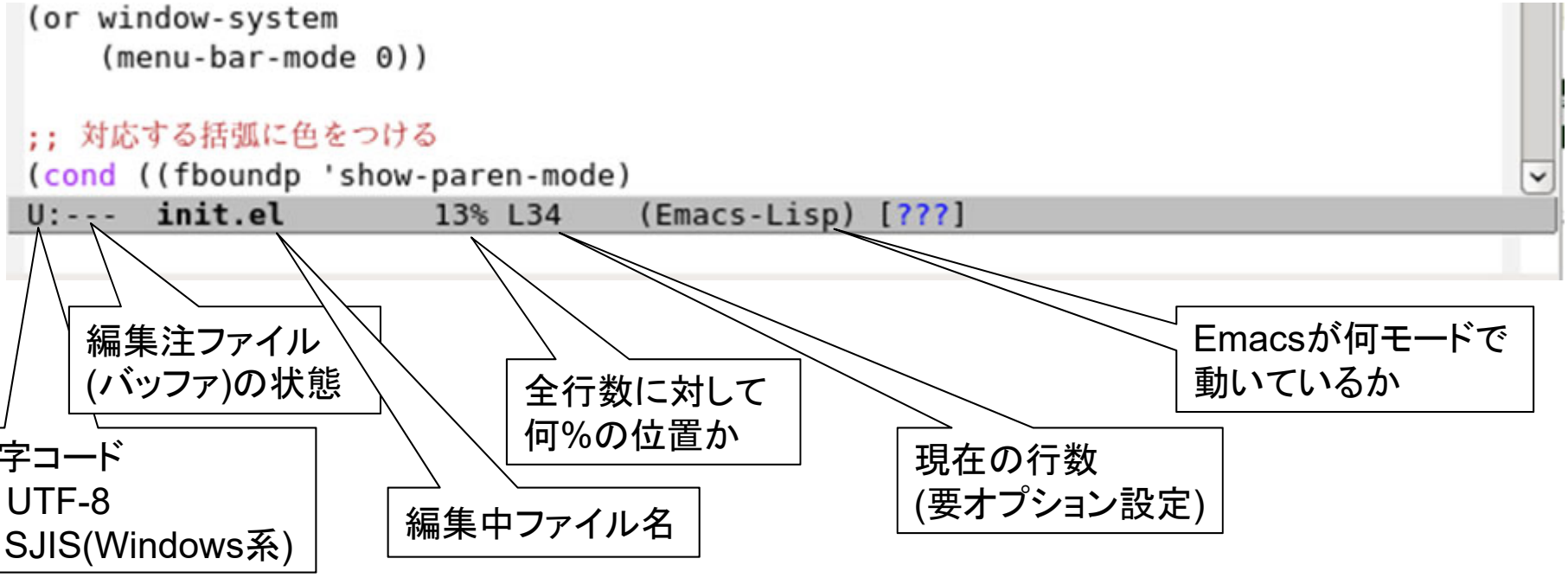
- Emacs Lispでいろいろカスタマイズ可能
  - 設定ファイルはホームディレクトリから.emacs.d/init.el
  - Lispという言語に触れる良い機会とも言える
- 各種ファイルの編集モードを持つ
  - 各プログラミング言語のキーワードやコメントアウトをハイライト可能
    - コメントアウト: 先頭にコメントアウト指定記号などをつけてプログラムとして解釈しないようにする
  - Tabキーでインデント(字下げ)を行ってプログラムを見やすくできる
    - Tabキーでのインデントが予想と違うことでミスが見つかることも
    - Pythonではインデントが構文に関わるので、Tabキー複数押しで「どのレベルのインデントを行なうか」選択できる
- 外部プログラム呼び出したりすることで、メール読み書きやTwitter読み書きも可能

Emacsカスタマイズ

- ホームディレクトリ下の.emacs.dディレクトリの下のinit.elを編集してカスタマイズ
  - こちらも編集失敗して焦ることがあるので適時バックアップを
  - 嶋田が利用しているサンプルを講義ページに置いてあるので、必要な部分をコピーしたりして利用すること
  - xxx-modeな設定はAlt+x→xxx-modeで有効化/無効化可能
  - 「;」の文字以降はコメント扱いになる
- カスタマイズに関する変数の現在値を知りたい場合はAlt+x→eval-expression→変数名を入力
- カスタマイズ情報はEmacsWikiにいっぱいある
  - https://www.emacswiki.org/emacs?interface=ja
    (検索キー: emacswiki)

Emacsのウィンドウの説明

- メニューバーやアイコンバーなどは分かりやすいので割愛
- ステータス行(モードライン)



- 最下行は操作結果の表示や入力(開くファイル名の入力)などで利用

Emacsのキーバインド(1/5)

複数のキーを順番で押す物が多い

- Ctrl+x→Ctrl+f: ファイルを開く(最下行にファイル名を入力)
  - ここでもTabキーで補完が効くし、候補一覧も出る
- Ctrl+x→Ctrl+s: 上書き保存
- Ctrl+x→Ctrl+w: 名前をつけて保存
- Ctrl+x→Ctrl+c: Emacsを終了
- Ctrl+g: 作業中のコマンドのキャンセル
- Ctrl+¥: 日本語入力の開始
  - GNOMEの日本語入力とごっちゃになって混乱しないように
- Ctrl+_もしくはCtrl+x→u: アンドゥ

Emacsのキーバインド(2/5)

- Ctrl+b/f: 1文字前/後に移動
- Alt+b/f: 1単語前/後に移動
- Ctrl+a/e: 行頭/行末に移動
- Ctrl+p/n: 1行前/後に移動
- Ctrl+v / Alt+v(Esc→v): 1ページ後/前に移動
- Esc→数字→コマンド: コマンドを数字の回数だけ繰り返す(無効な例外もある)
  - 例: Esc→10→Ctrl+n　→10行後に移動
  - 「Esc→100→0」で0を100文字入力なんてことも可能
- Alt+</>: 文頭/文末に移動
- Alt+g→g: 指定行番号に移動
- Ctrl+l: 現在の行を上下中心に持ってくる(複数押しで…)

Emacsのキーバインド(3/5)

- Ctrl+スペースもしくはCtrl+@: 範囲選択開始
- Ctrl+w: 選択範囲をカット
- Alt+w: 選択範囲をコピー
- Ctrl+y: カットorコピーされた物をペースト
- Ctrl+s: 前方検索
- Ctrl+r: 後方検索
- Alt+%: 置換(置換前、置換後の文字を入力)
  - yes/noを聞かれるのでy/n/!で応答(!で無条件全置換)
- Ctrl+d: カーソルキーの場所の文字を削除
- Ctrl+h: カーソルキーの1文字前の文字を削除
- Ctrl+k: カーソルキーから行末までを削除
  - 空行だと行自体を削除

Emacsのキーバインド(4/5)

- Ctrl-x→Ctrl-b: 編集中のファイル(バッファ)一覧
  - 上下移動でバッファを選択して、リターンキーで選択可能
- Ctrl-x→b: 名前を指定してバッファに切替
  - 初期状態で1つ前に編集したバッファ名が入っている
- Ctrl+x→2: ウィンドウの上下分割
- Ctrl+x→3: ウィンドウの左右分割
- Ctrl+x→o: 他のウィンドウへ移動
- Ctrl+x→1: 分割終了(現在のウィンドウだけ表示)

Emacsのキーバインド(5/5)

Emacsキーバインドに関するTips

- Alt+x→xxxx-mode: xxxx-modeに入る
  - 通常はファイルのサフィックス(「.py」などの末尾)を見て各モードに入るが、サフィックスの無いファイル編集時に利用
  - 行頭に行番号を付けるlinum-modeなど、他のモードと併用可能なモードの起動にも利用可能
  - 他にも種々のショートカット指定されていない機能の呼び出し可能
    - 選択範囲のインデントを行なうindent-regionとか、機能や変数の説明を出すaproposとか
- 初期設定ファイルの編集でキーバインドもカスタマイズ可能
- 一連のキー入力をキーボードマクロとして繰り返し実行可能