

プログラミング言語 第一回

担当: 篠沢 佳久
櫻井 彰人

平成23年度: 春学期

1

クラス分け

2

クラス分け①

- プログラミング言語は二つの教室で同時に行います
 - 703教室(50人収容)
 - 704教室(100人収容)
- どちらの教室も同じ内容の講義をします

3

クラス分け②

- 703
 - までの学籍番号の学生
- 704
 - 以降の学籍番号の学生
 - 管理工学科2年生以外の学生

4

講義のガイダンス

講義の目的, 進め方

5

この講義の目指すもの Part1

- プログラミングの基礎を理解
 - プログラミングの基礎知識を中心に
 - プログラムとは
 - プログラムの実行とは
 - 命令とデータ
 - 判断と分岐
 - プログラミングの構造と実行制御

6

この講義の目指すもの Part2

- プログラミングという行為
 - 書く、テストする、使う
- プログラミングがひとりのできるように
 - アルゴリズム
 - データ構造
- プログラミング言語とは
- プログラミングの基本をプログラム言語Rubyを通して学ぶ

7

この講義の目指すもの Part3

- Ruby言語でプログラムが作れるように
 - 基本的な演算
 - 制御構造
 - 条件式
 - 繰り返し
 - 配列
 - 関数(メソッド)
 - 標準入出力, ファイル入出力

8

Rubyとは何か?



- Ruby: まつもとゆきひろ氏による、便利さと容易さを兼ね備えたオブジェクト指向スクリプト言語
 - まつもとゆきひろ: <http://www.rubyist.net/~matz/>
 - スクリプト言語: 動作内容を、台本 (Script) のように記述するための、簡易的なプログラミング言語の総称
 - かなり簡単に(周辺環境が)インストールできる
 - 皆さんのコンピュータで実習ができる
 - かなり簡単にプログラムできる
 - 初心者にも容易に学習できる
 - 結構まともに動くプログラムも書ける
 - Ruby on Rails により、Webアプリが容易に書ける
 - そして、Ruby が有名になった

9

この講義では

- 演習をできる限り行います。
 - そのためには、実は、Ruby プログラムを実行するシステムとして irb (interactive Ruby) をよく用います。
 - irb は対話的に Ruby プログラムを実行するもので、ちょっと実習をするには、適しているのです。

10

この講義で目指せたら

- もう少し先に行くと
 - ファイル処理
 - DB処理
 - Web アプリケーション
 - 日本語処理

11

内容に関する注意

- 基本的(初歩的)なことに注力する
- ただし、ところどころ細かい話もする
 - 少し深いことを知りたい方への追加
 - 疑問に対する答えとして
 - 初級者は無視をしてよい

12

進め方:

- (繰り返しになりますが) Ruby を使う
 - 特に irb を用いて実習を多く
- ある事例(課題)を考える
 - ある動作をする「プログラム」
 - もちろん、簡単版

13

方針

- 多くのサンプルプログラムを用意します
 - 講義では全て話すことができません
 - 自習(復習)もして下さい
- 練習問題を多く行ないます

14

管理工学科におけるプログラミングの講義

- 本講義(2年春)
 - Rubyを対象として、プログラミングの基礎を中心に
- Java言語, オブジェクト指向
 - ソフトウェア工学(2年秋, 飯島先生)
 - ソフトウェア工学実習(3, 4年春, 飯島先生)
- プログラミングの応用
 - 管理工学実験演習Ⅱ 計算機(COM)実験(3年通年)

15

実習について

- この講義では理解を深めるために実習を交えて行ないます。
- 教室... 日吉ITC 地下一階
 - 703(50人収容)
 - 704(100人収容)
 - どちらも同じ講義内容

16

成績について①

- 成績のつけかた
 - 講義以外の時間にレポートを作成
 - 5回程度を予定
 - 最後の一回は、講義の最終回に行ないます(必ず出席して下さい)
 - 講義中の演習問題(平常点)
 - 平常点+レポートの成績から判定

17

最終回の課題

- 講義時間内に課題を行ないます
- 必ず出席して下さい
- 早慶戦が行われた場合 → 7/18
- 早慶戦が行われなかった場合 → 7/11

18

成績について②

- Rubyでプログラムが書ける(自信のある)人は、授業に出席しなくてもレポートさえ出せば単位がとれる
 - 予め申告することが条件
 - ただし最終回のみは必ず出席して下さい

19

講義に関する情報

- 講義資料のURL
 - <http://www.sakurai.comp.ae.keio.ac.jp/class.html>
- 教員, TAへの質問
 - 電子メール
 - 直接質問(アポイント必要)

20

プログラムとは

プログラミングの必要性
プログラムとプログラム言語

21

なぜプログラミング？

- 他の講義・実験・演習、卒論に必要
- 必要な技術
- 知っておくべき技術
- 論理的思考力の訓練

22

プログラムとは①

- 日常使う「プログラム」はどのような意味か？
- すなわち、手順・動作を記した書類
 - 書類といっても、紙に書かれているわけではない

23

プログラムとは②

- コンピュータにおいて用いる「プログラム」とは？
- コンピュータが行う動作を
 - 事細かに
 - 逐一記述したもの

24

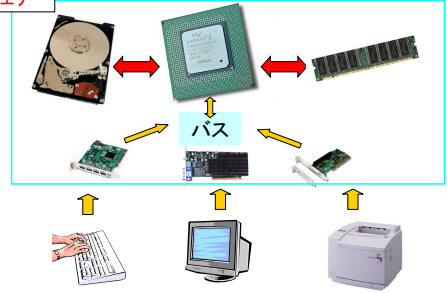
プログラムとは③

- コンピュータの「記憶装置」に蓄えられている
 - メモリ: 普通はコンピュータの中に隠されている。
 - 内容を持ち運びたいときに、USBメモリとかCD-RとかDVD-Rとかいったものにコピーする
- すなわち、プログラムは「ソフトウェア(軟件)」
 - ハードウェア(硬件)ではない
 - つまり、触って感じる物ではない

25

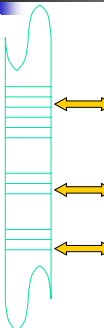
コンピュータとは

ハードウェア



26

プログラム プログラム プログラム

- 
- OS: プログラムの実行の制御や、ハードウェアの制御と管理など、コンピュータを安全にそして効率良く動かせるための基本ソフトウェア。
 - 例: Windows 2000/XP/Vista/7, UNIX, Linux, FreeBSD, Solaris, Tron
 - アプリケーション
 - 例: 表計算、文書作成、Java
 - ユーザ作成プログラム
 - 例: 「こんにちは」プログラム

27

プログラムは何語で書くか

- 「書類」だから、記述する言語が必要
 - 言語: 意味のある文字列
- 日本語や英語がだめなことは、勿論
 - なぜか?
- コンピュータが分かる言語?
 - 比喩が過ぎる。コンピュータは意味は分からないから。
 - コンピュータが、文字列から自分がすべき動作に変換できればよい
- コンピュータ用の言語を作ればよい

28

それをプログラム言語という

- コンピュータは、メモリのどこかに書いてある「命令」を自分の動作に変換すればよい。
 - この「命令」の構成規則が言語
 - この変換規則は言語ではない
 - コンピュータ(機械)にとっては言語(かな?)なので、機械語といったりする
 - この変換規則の例:
 - 01100 → 出力電圧を5Vに
 - 某神経細胞on → 右手親指曲る (人間の脳)

29

プログラミング言語とは

- 人間の思いをコンピュータに伝える言葉
 - といったって相手はコンピュータですから
 - 人間の言葉より、機械の言葉にずっと近い。ということは
 - 硬い。すなわち、規則にやかましい
 - 手書き文字ではない。すなわち、キーボード入力

30

どんなものがあるか？

- 高級 (high-level) 言語
 - 実行方法による分類
 - コンパイル言語
 - Ex. Java
 - インタプリタ言語
 - Ex. Ruby
 - 概念による分類
 - 命令型言語
 - Ex. Ruby, Java
 - 関数型言語
 - 分類にならない分類
 - オブジェクト指向言語
 - Ex. Ruby, Java
 - アセンブリ言語・機械語

31

Ruby の長所・短所

- 長所
 - 始めやすい
 - インストールが簡単
 - プログラムもその実行も簡単
 - 一行から始められる
 - (実は隠れた長所がたくさんあります。急成長中)
- 短所
 - 「作法」「行儀」が学びにくい
 - 個性が非常に強い
 - 高速な実行に向かない
 - 大きなプログラムが作りにくい

32

プログラミング実習

Rubyプログラムの作成手順

33

今年度からのITCのアカウント

- 日吉ITC情報ネットワークアカウント
 - 旧アカウント
 - fr000000
- ITCアカウント
 - 新アカウント
 - ua000000

二つのアカウントではドキュメントの位置(パス)に違いがあります

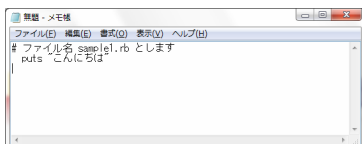
34

“こんにちは”のRubyプログラム

これより右側の文字は、Ruby は無視します。

```
# ファイル名 sample1.rb とします  
puts "こんにちは"
```

#から行末までをコメントと見なします。



```
sample1.rb  
# ファイル名 sample1.rb とします  
puts "こんにちは"
```

35

“こんにちは”のRubyプログラム

プログラム
→ テキストエディタで記述する

36

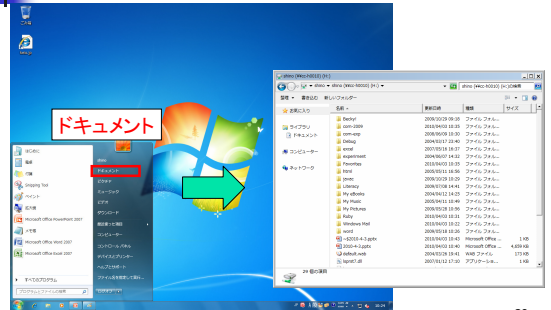
まずは、やってみよう

- 皆さんの「ドキュメント」は、日吉のPCでは、HDドライブになっています(旧アカウント)
(新アカウントの場合はZドライブ)
- そこに、Ruby という名のフォルダを作ってください。



37

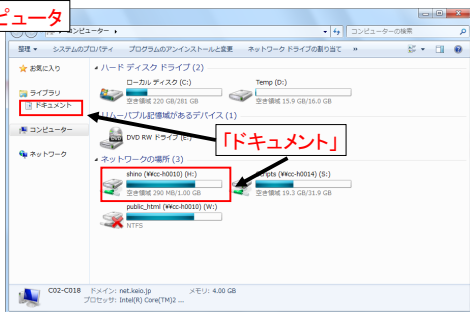
日吉ITCの場合 (OSはWindows7)



38

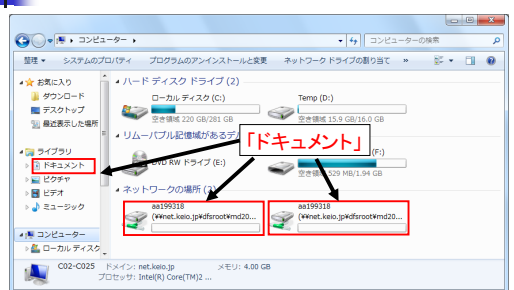
「ドキュメント」のフォルダー (旧アカウントの場合)

コンピュータ




39

「ドキュメント」のフォルダー (新アカウント)





40

ディレクトリ/フォルダとは

- ハードディスクやCD-ROMなどの記憶装置において、ファイルを分類・整理するための保管場所。
- UNIXやMS-DOSではディレクトリといい、MacintoshやWindowsではフォルダいう。
- Windows の GUI では  のように見えるもの

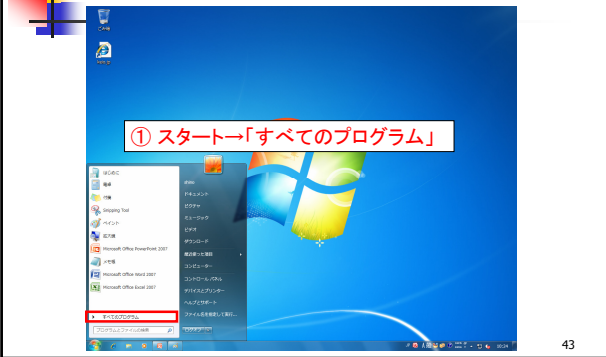
41

どうすれば、プログラムを書いたことになるの？

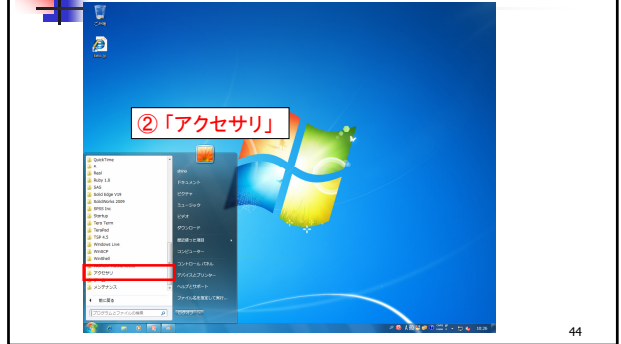
- Ruby 言語の場合
 - ①「**コマンドプロンプト**」というプログラムを起動して行う
 - ②メモ帳(でなくてもいいが)で、プログラムを書く(キーボードから入力する)
 - ③ファイルにセーブ(ハードディスクに入れること)
 - 仮に sample1.rb (全て小文字)という名前だとして以下は、
 - ④「**コマンドプロンプト**」上で `ruby sample1.rb`  と入力。
 - ⑤ エラーがなければ結果が得られる 

42

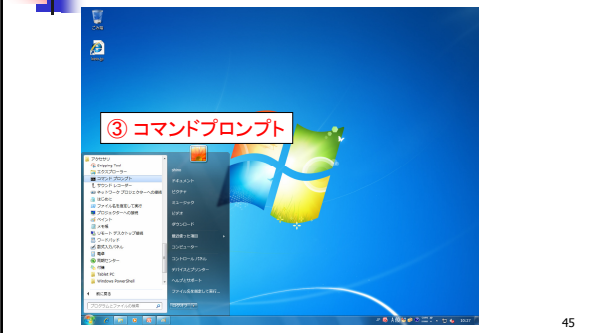
コマンドプロンプトの起動①



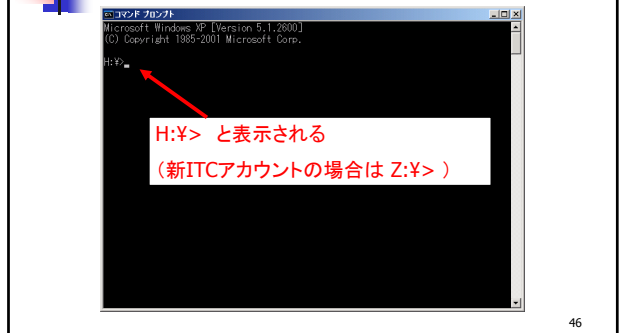
コマンドプロンプトの起動②



コマンドプロンプトの起動③



コマンドプロンプトの画面



プログラムの書き方

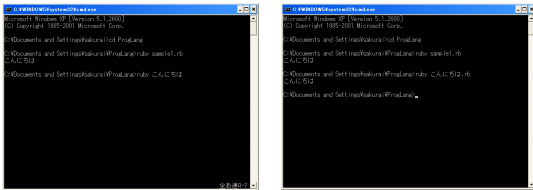
- 二つの作成手順を紹介します
- 初心者はファイルの「拡張子」で混乱します
- どちらの方法でもよいので慣れて下さい

ファイル

- ハードディスクやCD-ROMなどの記憶装置に記録されたデータのまとまり。
- OS(Windows OSなど)は記憶装置上のデータをファイル単位で管理する
- プログラムはファイルに記述する

ファイル名

- 識別のために、ファイルにつけられた名前。一つのディレクトリでは、一名一ファイル
- Windows は、大文字・小文字を区別しない。
- 日本語Windowsでは、かな・カナ・漢字も使える。
 - 入力は、Alt + 半角/全角。



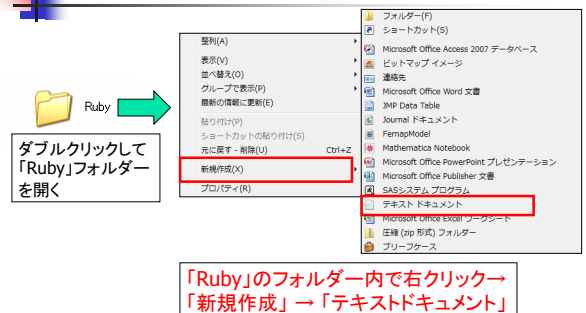
拡張子

- ファイル名の末尾にファイルの種類をあらわす「拡張子」と呼ばれる数文字のアルファベットを付けるのが普通
- ただし、Windows が拡張子を(真剣に！)見るのは、ファイル・アイコンがダブルクリックされたとき
 - ダブルクリックしたときに、メモ帳を起動したいなら ff.txt と、MsWord を起動したいなら ff.doc とする
- Rubyプログラムの場合は「rb」という拡張子を必ずつける

50

プログラムの書き方その①

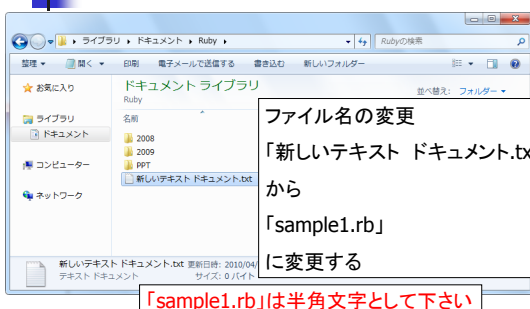
プログラムの記述方法①



51

52

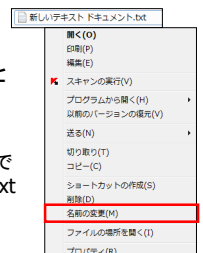
プログラムの記述方法②



53

ファイル名の変更方法①

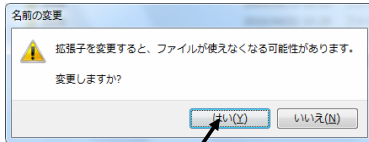
- ファイルを選択 → 右クリック → 「名前の変更(M)」
- ファイルの名前を sample1.rb としてください。
 - 半角文字
 - 今回の講義では、拡張子(この例でいえば(.rb)は.rbでなくても(.txtでも)問題は起こらない(はず)。



54

ファイル名の変更方法②


ファイル名を変更すると

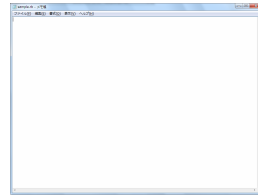


「はい(Y)」をクリック→ファイル名が変更される

55

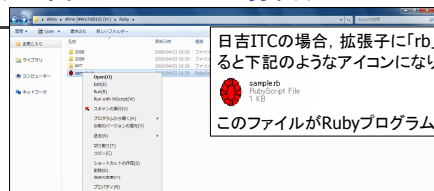
エディターの起動①

- sample1.rbが  というアイコンであれば、ダブルクリックしてください。
- メモ帳が立ち上がるでしょう、きっと。



56

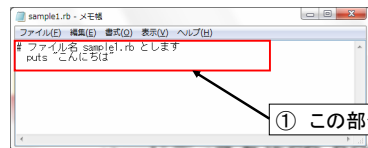
エディターの起動② (日吉ITCのPCの場合)



Rubyプログラム「sample1.rb」を右クリック→「Edit」

57

プログラムの書き込み①



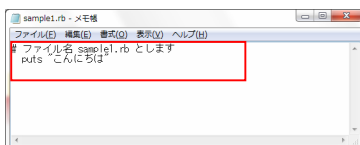
日本語以外は半角文字で書いて下さい
全角の空白は使わないで下さい
"" (ダブルクォート)は半角文字で書いて下さい

"
2 ふ

58

プログラムの書き込み②

作成したファイルがRubyプログラム



書き終わったら、上書き保存を行なう

② メニューバーの「ファイル」→「上書き保存」

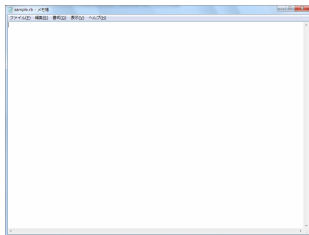
59

プログラムの書き方その②

60

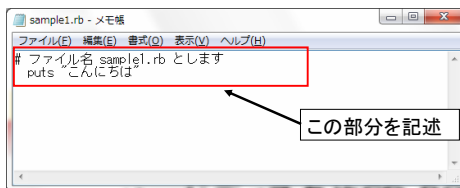
エディターの起動

- 「スタートボタン」→「すべてのプログラム」
→「アクセサリ」→「メモ帳」



61

プログラムの記述



日本語以外は半角文字で書いて下さい
全角の空白は使わないで下さい
" " (ダブルクォート)は半角文字で書いて下さい

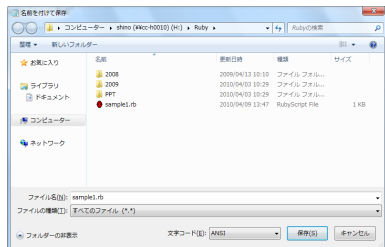
”

2 ふ

62

プログラムの保存①

- メニューバーの「ファイル」→「名前を付けて保存」



63

プログラムの保存② (旧アカウントの場合)

- メニューバーの「ファイル」→「名前を付けて保存」

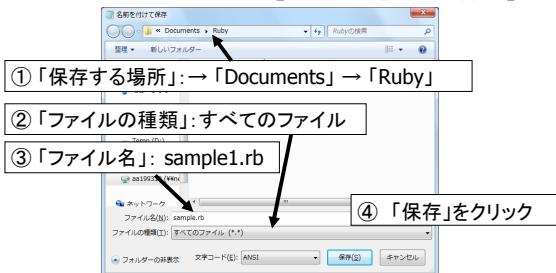


*ライブラリドキュメント→Rubyでも可能

64

プログラムの保存② (新アカウントの場合)

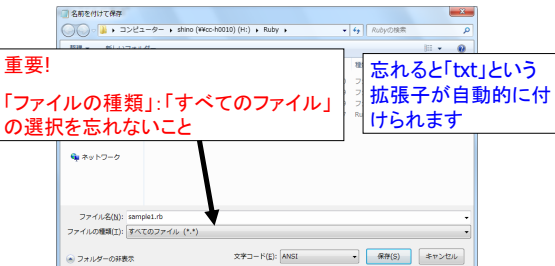
- メニューバーの「ファイル」→「名前を付けて保存」



65

プログラムの保存③

- メニューバーの「ファイル」→「名前を付けて保存」

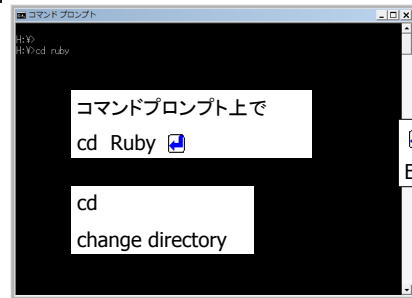


66

Rubyプログラムの実行

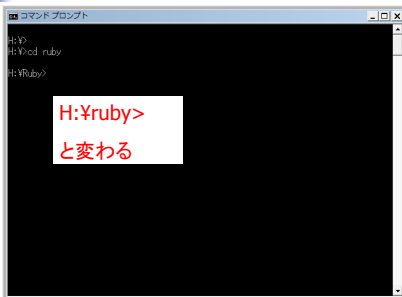
67

Rubyフォルダーへの移動① (旧アカウントの場合)



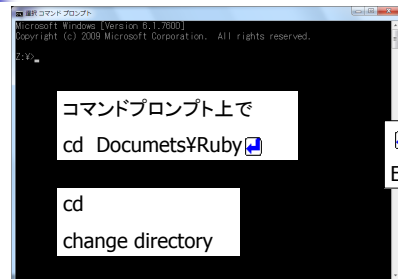
68

Rubyフォルダーへの移動② (旧アカウントの場合)



69

Rubyフォルダーへの移動① (新アカウントの場合)



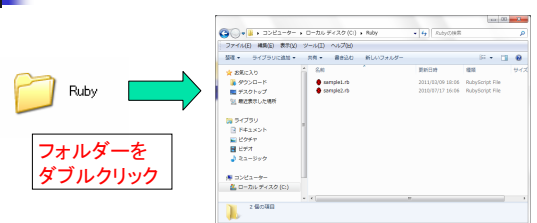
70

Rubyフォルダーへの移動② (新アカウントの場合)



71

コマンドプロンプト①



コマンドプロンプト上で
H:> cd Ruby

72

コマンドプロンプト②

```
C:\Ruby>dir
ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 74B7-7992 です

C:\Ruby のディレクトリ
dir
フォルダ内のファイル名を表示

2008/04/03 17:47 <DIR>
2008/04/03 17:47 <DIR> ..
2008/04/03 18:15          51 sample1.rb
2008/03/03 16:14       103 sample41.rb
2 個のファイル          154 バイト
2 個のディレクトリ 229,682,393,088 バイトの空き領域
```

73

GUI (graphical user interface)

- 表示として、グラフィックスを用いたユーザインタフェース。入力は、マウスやそれと類似した装置を用いる。
- パソコンでは、Macintosh が使い始めた。
- 今では、これが常識

74

CLI or CUI (command line user interface)

- 表示として、文字列を用いたユーザインタフェース
- 入力はキーボードを用いる。
- 入力するものは、コンピュータに対するコマンドであり、行(ライン)単位に入力する。入力する場所をコマンドラインという。コマンドの実行結果はコマンド入力直後に表示する。画面を使い切ると、スクロールする。
- Windows 7/Vista/XP/2000 では、コマンドプロンプトという言葉が用いられる
 - コマンドプロンプトは、本来は、コマンドラインの先頭にコンピュータが書く文字である

75

Rubyプログラムの実行

- ruby とは Ruby プログラムを実行するコマンド
 - 指定されたファイルの中身を見て、それに従った動作をする

```
G:\Ruby>ruby sample1.rb
こんにちは
G:\Ruby>
```

ruby sample1.rb

Ruby プログラムの実行
ruby Rubyプログラム

76

Rubyプログラムの実行方法のまとめ

```
プログラム名
ファイル名 sample1.rb とします
puts "こんにちは"
```

① プログラム
テキストエディタで記述

```
G:\Ruby>ruby sample1.rb
こんにちは
G:\Ruby>
```

② Ruby プログラムの実行
ruby Rubyプログラム

77

プログラムが動かない場合

エラーメッセージについて

78

コンピュータは忠実である

- 言われたとおりに、実行する
- 規則通りに書かれていない場合は、実行せずに、エラーメッセージを出力する
- 書かれたように読む
 - 決して、「きっとこう書きたかったのだろうなあ」と考えて読むことはしない
 - 勿論、「『きっとこう書きたかったのだろうなあ』と読んで読む」ようにプログラムを書けば、そう書いた範囲で「考えて読む」ようにはなる

79

プログラム構文上の大原則

- 括弧(広い意味での括弧です)は、開いたら、必ず閉じる。
 - Ruby での例外:「#」で始まるコメント(プログラムと関係のない書き込み)は、改行(そして改行のみ)が閉じる記号
- 複数種の括弧が混じるときには、互いに交錯してはならない
 - 例: { ([]) }
 - 誤例: { } { ([]) }

80

空白について

- Ruby にとって、空白は区切り文字。連続する空白は一つの空白と同じ。しかし、
- Ruby が空白とみなす空白は1バイトコードの空白だけ。2バイトコードの空白は Ruby にとっては空白ではない。
 - よく読んでください。決して、禅問答ではありません。
- どちらの空白かは、人間がみて区別しにくいので、ちょっと目には訳の分からないこと、しかし、よく考えれば分かることが起こる

81

半角文字と全角文字

- プログラムは半角文字で書く
- ただし例外もあります
- # の後はコメントであり、この後は全角文字を使用してもよい
- " " の中は全角文字を使用してもよい

82

文字について

- 日本語 Windows が取り扱う文字には、1バイトコード(所謂半角文字)と2バイトコード(所謂全角文字)とがある。
- 昔は、本当に、半角と全角で表示されていたので分かりやすかったが、今では、プロポーショナルフォントなどを用いるので、分かりにくい。
 - 例: A A と並べれば分かるが KEIO keio
- コンピュータはちゃんと区別するからやっかいだ

83

The screenshot shows a Windows command prompt window with the following text:

```
G:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Ysakurai>G:
G:>cd Ruby
G:\Ruby>ruby Keio.rb
ruby: No such file or directory -- Keio.rb (LoadError)
```

Below the command prompt, there is an error message box that says: 「Keio.rb」というファイルがないというエラーが表示

Another box points to the file list in the command prompt, saying: Keio.rb というファイルがあるはず...

The file list in the command prompt shows:

```
Keio.rb
comp1212
comp1213
comp1214
```

84

実はファイル名「keio.rb」の「i」が全角文字だった

```
G:\Ruby> dir
2007/03/25 21:59 <DIR> .
2007/03/25 21:59 <DIR> ..
2007/03/25 21:47      19 Keio.rb
2007/03/25 21:47      19 Kei o.rb
                2 個のファイル      38 バイト
                2 個のディレクトリ 1,877,010,688 バイトの空き領域
G:\Ruby>
```

「i」が半角文字のファイル

Keio.rb ← 「i」が半角文字のファイル

Kei o.rb ← 「i」が全角文字のファイル

ファイル名に英語の全角文字をつけないこと！

```
# Keio1.rb
puts "こんにちは"
```

```
G:\Ruby> ruby Keio1.rb
こんにちは
G:\Ruby> ruby Keio2.rb
Keio2.rb:1: Invalid char `¥201' in expression
Keio2.rb:1: syntax error, unexpected $undefined, expecting $end
puts "こんにちは"
^
G:\Ruby>
```

```
# Keio2.rb
puts "こんにちは"
```

全角の空白が含まれている

どこが間違っているでしょうか①

```
# Keio3.rb
puts "こんにちは"
```

```
C:\Ruby> ruby Keio3.rb
Keio3.rb:2: Invalid char `¥202' in expression
Keio3.rb:2: Invalid char `¥223' in expression
```

どこが間違っているでしょうか②

```
C:\Ruby> ruby Keio3.rb
Keio3.rb:2: Invalid char `¥202' in expression
Keio3.rb:2: Invalid char `¥223' in expression
```

```
# Keio3.rb
puts "こんにちは"
```

半角ではなく全角のsとなっている

どこが間違っているでしょうか③

```
# Keio4.rb
puts "こんにちは"
```

```
G:\Ruby> ruby Keio4.rb
Keio4.rb:2: unterminated string meets end of file
G:\Ruby>
```

どこが間違っているでしょうか④

```
G:\Ruby> ruby Keio4.rb
Keio4.rb:2: unterminated string meets end of file
G:\Ruby>
```

```
# Keio4.rb
puts "こんにちは"
```

全角文字

```
# Keio5.rb
puts "こんにちは"
```

```
G:\Ruby>ruby Keio5.rb
Keio5.rb:2: Invalid char `¥201' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥261' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥361' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥311' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥277' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥315' in expression
Keio5.rb:2: unterminated string meets end of file
Keio5.rb:2: warning: parenthesize argument(s) for future version
G:\Ruby>
```

91

```
G:\Ruby>ruby Keio5.rb
Keio5.rb:2: Invalid char `¥201' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥261' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥361' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥311' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥277' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥315' in expression
Keio5.rb:2: unterminated string meets end of file
Keio5.rb:2: warning: parenthesize argument(s) for future version
G:\Ruby>
```

```
# Keio5.rb
puts "こんにちは"
```

全角文字

92

```
C:\Ruby>ruby Keio6.rb
Keio6.rb:1: Invalid char `¥201' in expression
Keio6.rb:1: Invalid char `¥224' in expression
Keio6.rb:2: Invalid char `¥201' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥261' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥361' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥311' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥277' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥315' in expression
Keio6.rb:2: unterminated string meets end of file
Keio6.rb:2: warning: parenthesize argument(s) for future version
```

```
# Keio6.rb
puts "こんにちは"
```

全角文字

93

繰り返しの注意ですが...

- プログラムは半角文字で書く
- 特に全角の空白には注意する
- # の後はコメントであり, 全角文字を使用してもよい
- " " の中は全角文字を使用してもよい

94

練習問題

他の例題

(同じようにプログラミングしてみてください。ファイル名は自由につけても結構です)

```
# sample12.rb
print "春の"
print "うららの"
puts "隅田川"
```

ファイル名

```
H:\Ruby>ruby sample12.rb
春のうららの隅田川
G:\Ruby>
```

```
# sample13.rb
print "春の" + "うららの"
puts "隅田川"
```

```
H:\Ruby>ruby sample13.rb
春のうららの隅田川
G:\Ruby>
```

```
# sample14.rb
puts "春の"
puts "うららの"
puts "隅田川"
```

print と puts
表示させる命令で
すが違いは次回以
降に説明します

```
H:\Ruby>ruby sample14.rb
春の
うららの
隅田川
H:\Ruby>
```

95

他の例題

四則演算を行なうRubyプログラム

```
# sample15.rb ← ファイル名
a=5
b=4
print "a+b=", a+b, "\n"
print "a-b=", a-b, "\n"
print "a*b=", a*b, "\n"
print "a/b=", a/b, "\n"
```

```
H:¥ruby>ruby sample15.rb
a+b=9
a-b=1
a*b=20
a/b=1
```

97

Rubyに関する情報

98

Ruby 関連サイト

- Official site:
<http://www.ruby-lang.org/ja/>
 - マニュアル
 - 「ドキュメント」→本文中の「リファレンスマニュアル」
 - Rubyのインストール
 - 「ダウンロード」→「各環境用バイナリ」
 - 本文中の「ActiveScriptRuby」または「Ruby-mswin32」
- <http://www.namaraii.com/rubytips/> も便利です

99

参考書

- 各種出ています。ご自分の気に入ったものでよいと思います。on-line文書もあります。
 - UNIXプログラミング「Ruby入門」
<http://www.lab.ime.cmc.osaka-u.ac.jp/~kiyo/pub/lecture/unixpro/ruby/>

100

Rubyのインストール

個人PCへのRubyのインストール

101

Ruby のインストール①

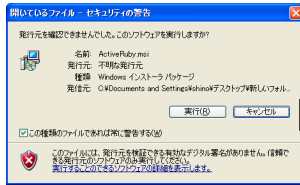
- ActiveScriptRuby
 - もっとも簡単。ただし、administrator 権限が必要(自分のPCならOK)
- ActiveScriptRuby をインストールする
 - <http://www.artonx.org/data/asr/>

102

ActiveScriptRubyのインストール①



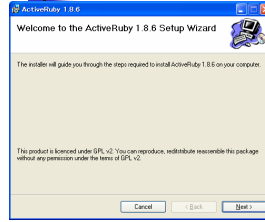
ダウンロードしてきたファイルをダブルクリック



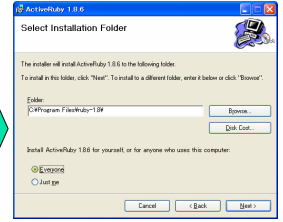
「実行」をクリック

103

ActiveScriptRubyのインストール②



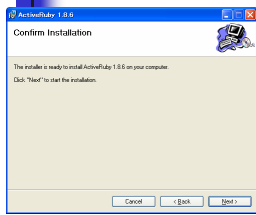
「Next」をクリック



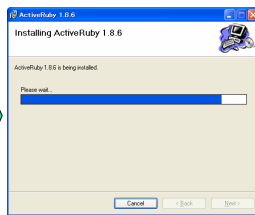
「Next」をクリック

104

ActiveScriptRubyのインストール③



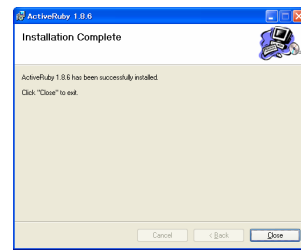
「Next」をクリック



インストール中...

105

ActiveScriptRubyのインストール④



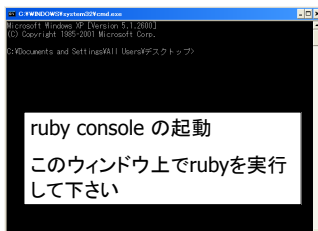
インストール終了

106

ActiveScriptRuby上での実行方法



ダブルクリック



日吉ITCとはコマンドプロンプトを起動した時のドライブ、フォルダが異なるの注意して下さい

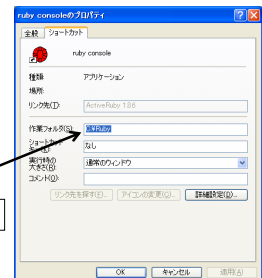
107

作業フォルダの変更①

Ruby という名前のフォルダでプログラムを実行したい場合



右クリック→プロパティ

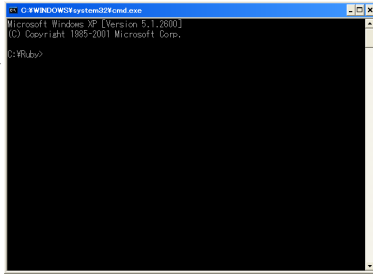


「C:¥Ruby」に変更

108

作業フォルダの変更②

ruby console
ショートカット
のアイコン
を
ダブルクリック



109

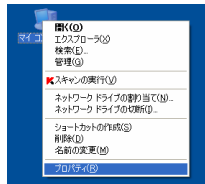
コマンドプロンプト上からの実行

- ITCと同様に、コマンドプロンプト上から実行する場合
→環境変数を設定する必要がある
- 以降の設定は間違えると、コンピュータの設定を壊してしまう場合があります
- 「環境変数」「Path」といった語を調べてから(理解してから)行なって下さい

110

環境変数の設定① (OSがWindows XPの場合)

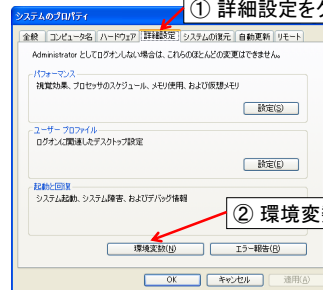
マイコンピュータ→右クリック→「プロパティ」



111

環境変数の設定②

① 詳細設定をクリック

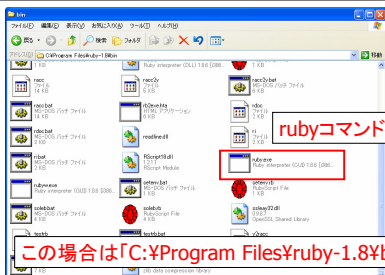


② 環境変数をクリック

112

環境変数の設定③

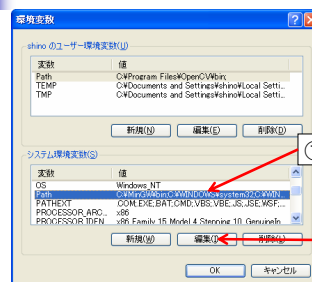
rubyコマンドがインストールされているフォルダを調べる



113

環境変数の設定④

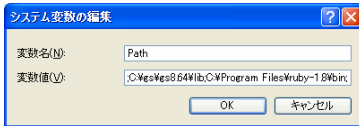
① 「Path」を選択



② 「編集」をクリック

114

環境変数の設定⑤

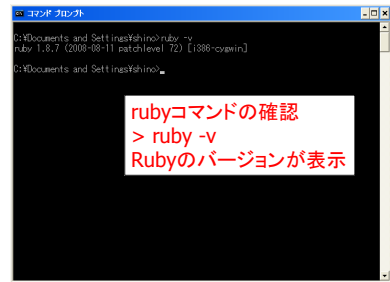


末尾に「;C:\Program Files\Ruby-1.8\bin」と追加
→「OK」をクリック

末尾に追加して下さい
その際、「;」で前に書かれているもの(Path)と区切って下さい

115

環境変数の設定⑥



116