

プログラミング言語 第一回

担当: 篠沢 佳久
櫻井 彰人

平成29年度: 春学期

1

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)
 - 高度プログラミング(4年春, 森田先生)

2

クラス分け

3

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

4

クラス分け②

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

5

講義のガイダンス

講義の目的, 進め方

6

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学ぶ
 - 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
 - 関数(メソッド)

7

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
- プログラミングが一人でできることを目的
 - 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

8

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
 - 標準入出力, ファイル入出力
 - 関数(メソッド)

9

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
 - 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

10

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

11

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

12

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
 - 初級者は無視してよい

13

進め方:

- (繰り返しになりますが)Rubyを使う
 - 実習を多く行ないます
- 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

14

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

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
 - 3回を予定
 - 講義の最終回(7/17)に最終課題を行ないます
 - 海の日ですが、講義日です
 - 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(3回)+最終課題の成績から判定

17

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3回のレポートは必ず提出して下さい
 - また最終回のみは必ず出席して下さい

18

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)

19

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

20

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

21

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

22

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

23

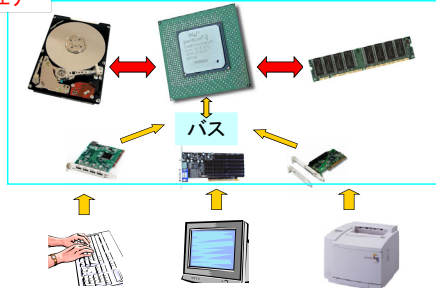
プログラムとは③

- 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

24

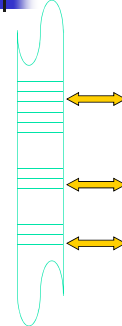
コンピュータとは

ハードウェア



25

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働かせるための基本ソフトウェア
 - 例: Windows 7/8/10, UNIX, Linux, FreeBSD, Solaris, Tron, Mac OS
- アプリケーション
 - 例: 表計算、文書作成、プレゼンテーション作成、ブラウザ
- ユーザ作成プログラム
 - 例: 「こんにちは」プログラム

26

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(プログラム言語)

27

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)

28

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

29

どんなものがあるか?

- 高級 (high-level) 言語
 - 実行方法による分類
 - コンパイラ言語
 - Ex. C, 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
 - EX. Lisp
- アセンブリ言語・機械語

30

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

31

プログラミング実習

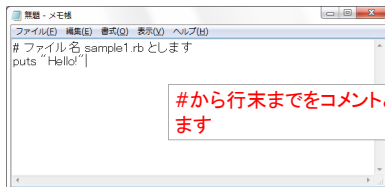
Rubyプログラムの作成手順

32

"Hello!" のRubyプログラム

作成するプログラム

```
# ファイル名 sample1.rb とします  
puts "Hello!"
```



33

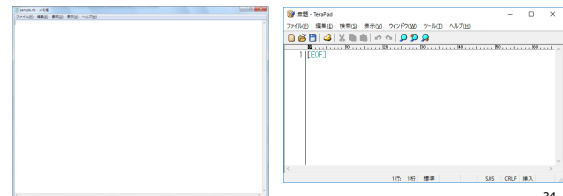
"Hello!" のRubyプログラム

プログラム

→ テキストエディタで記述する

メモ帳

TeraPad



34

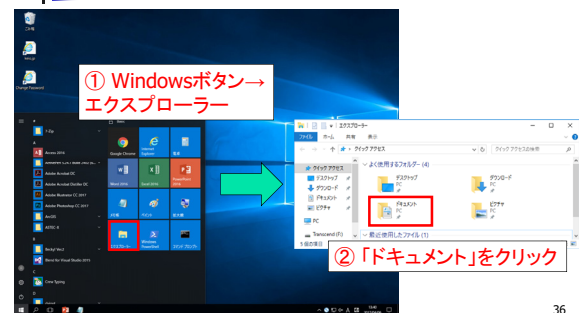
まずは、やってみよう

- 皆さんの「ドキュメント」は、日吉のPCでは、Zドライブになっています
- そこに、Ruby という名のフォルダを作ってください



35

日吉ITCの場合 (OSはWindows10)




36

日吉ITCの場合 (OSはWindows10)



37

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ではフォルダという
- WindowsのGUIでは  のように見えるもの

38

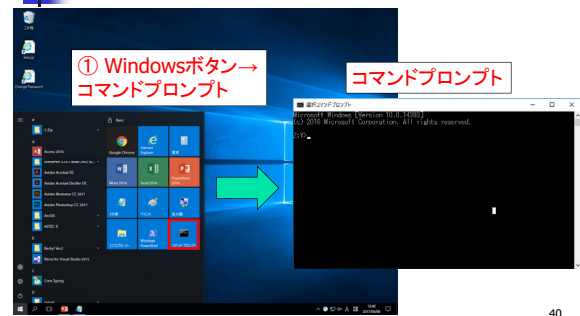
どうすれば、プログラムを書いたことになるの？

- 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の話は、
- 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

Enterキー

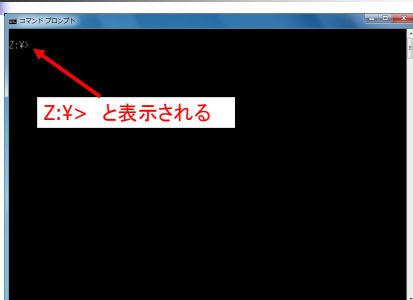
39

コマンドプロンプトの起動



40

コマンドプロンプトの画面



41

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

42

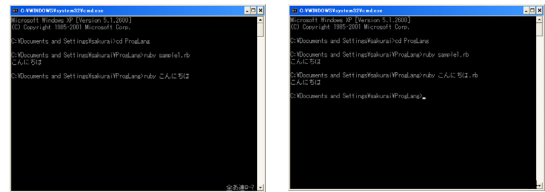
ファイル

- 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め
- 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

43

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



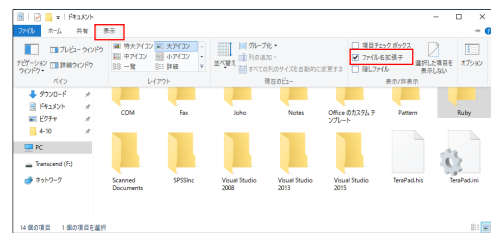
拡張子

- 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加するのが普通
- 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、MS-Word を起動したいなら ff.docx 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

45

拡張子の表示方法 (Windows10)

「表示」→「ファイル名拡張子」をクリック

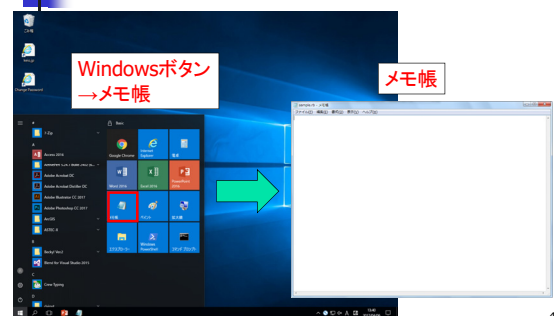


46

プログラムの書き方その① (「メモ帳」を用いる場合)

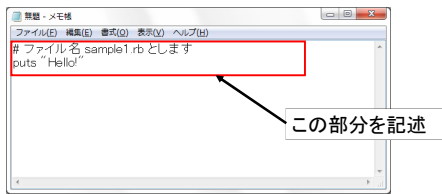
47

エディター(メモ帳)の起動



48

プログラムの記述



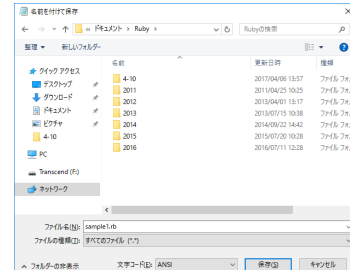
日本語以外は半角文字で書いて下さい
 全角の空白は使わないで下さい
 "" (ダブルクォート)は半角文字で書いて下さい

" 2 ふ

49

プログラムの保存①

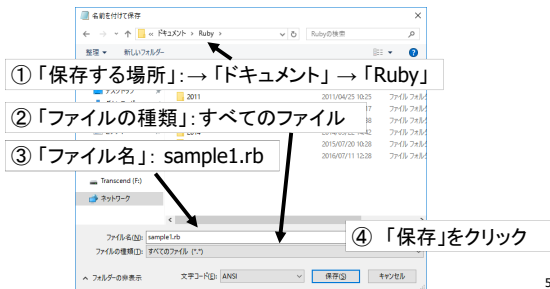
- メニューバーの「ファイル」→「名前を付けて保存」



50

プログラムの保存②

- メニューバーの「ファイル」→「名前を付けて保存」

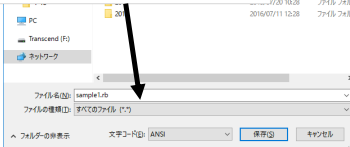


51

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」

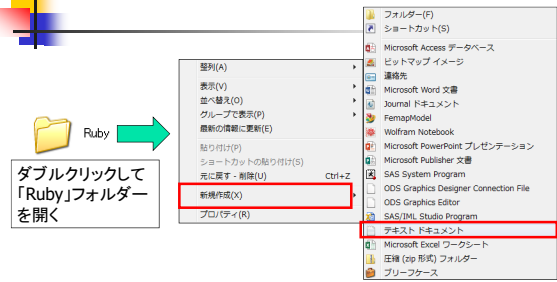
重要!
 「ファイルの種類」:「すべてのファイル」
 忘れると「txt」という拡張子が自動的に付けられます
 の選択を忘れないこと



52

プログラムの書き方その② (「メモ帳」を用いる場合)

プログラムの記述方法①



ダブルクリックして「Ruby」フォルダーを開く

「Ruby」のフォルダー内で右クリック→「新規作成」→「テキストドキュメント」

53

54

プログラムの記述方法②

ファイル名の変更
「新しいテキストドキュメント.txt」
から
「sample1.rb」
に変更する

「sample1.rb」は半角文字として下さい

55

プログラムの記述方法③

ファイル名を変更すると...

名前の変更
拡張子を変更すると、ファイルが使えなくなる可能性があります。
変更しますか?
はい(Y) いいえ(N)

「はい(Y)」をクリック→
ファイル名が変更される

56

ファイル名の変更方法

- 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)問題はおこらない(はず)

57

エディターの起動①

sample1.rbを右クリック→「送る」→「Notepad」*

*sample1.rbを右クリック→「Edit」でも良い

58

エディターの起動②

タイトルが「無題」から変わる

メモ帳を起動 → 「sample1.rb」のアイコンをメモ帳にドラッグ

59

プログラムの書き込み①

① この部分を記述

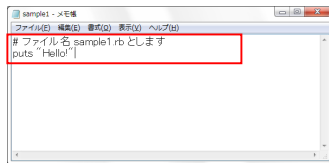
日本語以外は半角文字で書いて下さい
全角の空白は使わないで下さい
" (ダブルクォート)は半角文字で書いて下さい

2 ふ

60

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



```
sample1 - テキスト  
[ファイル(F)] 編集(E) 書式(O) 表示(V) ヘルプ(H)  
# ファイル名 sample1.rb とします  
puts "Hello!"
```

書き終わったら、上書き保存を行なう

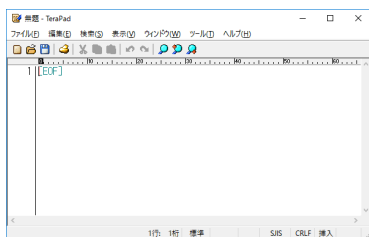
- ② メニューバーの「ファイル」→「上書き保存」

61

プログラムの書き方その③ （「TeraPad」を用いる場合）

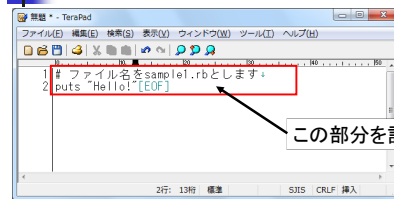
エディターの起動

- 「Windowsボタン」→「TeraPad」→「TeraPad」



63

プログラムの記述



この部分を記述

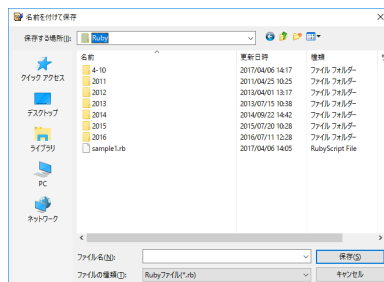
日本語以外は半角文字で書いて下さい
全角の空白は使わないで下さい
" " (ダブルクォート)は半角文字で書いて下さい



64

プログラムの保存①

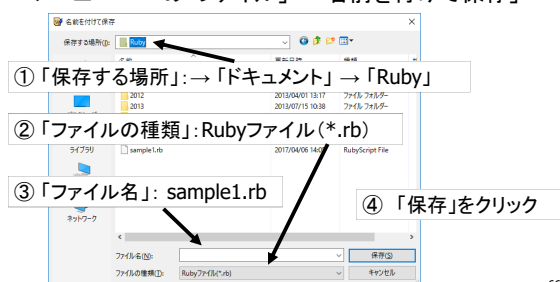
- メニューバーの「ファイル」→「名前を付けて保存」



65

プログラムの保存②

- 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

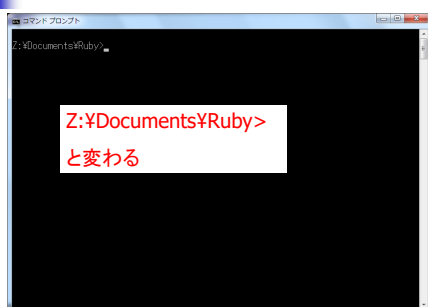
67

Rubyフォルダーへの移動①



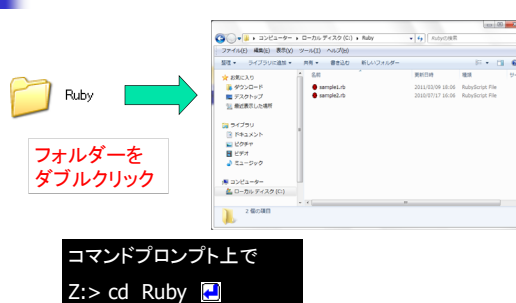
68

Rubyフォルダーへの移動②



69

コマンドプロンプト①



70

コマンドプロンプト②

Z:¥Documents¥ruby> dir と入力

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

Z:¥ruby のディレクトリ

2017/04/03 17:47	<DIR>	.
2017/04/03 17:47	<DIR>	..
2017/04/03 18:15		51 sample1.rb
2017/04/03 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

dir
フォルダ内のファイル名を表示

71

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

72

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 10/8/7/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

73

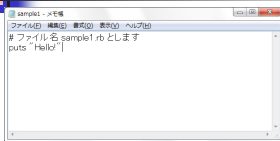
Rubyプログラムの実行

- 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

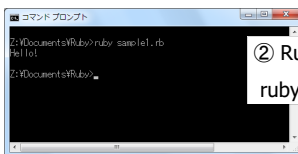


74

Rubyプログラムの実行方法のまとめ



① プログラム
テキストエディタで記述



② Rubyプログラムの実行
ruby Rubyプログラム

75

プログラムが動かない場合

エラーメッセージについて

76

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からなあ」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からなあ』」と考えると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

77

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: { } [([])] }

78

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

79

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
 - ただし、文字コードを指定しなければならない(次週以降説明します)

80

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

81

82

実はファイル名「keio.rb」の「i」が全角文字だった

83

84

どこが間違っているでしょうか①

```
# Keio3.rb  
puts "Hello!"
```

```
Z:¥Documents¥Ruby>ruby -Ks Keio3.rb  
Keio3.rb:2:in `<main>': undefined method `puts' for  
main:Object (NoMethodError)  
Did you mean? puts  
              putc
```

85

どこが間違っているでしょうか①

```
Z:¥Documents¥Ruby>ruby -Ks Keio3.rb  
Keio3.rb:2:in `<main>': undefined method `puts' for  
main:Object (NoMethodError)  
Did you mean? puts  
              putc
```

```
# Keio3.rb  
puts "Hello!"
```

半角ではなく全角の空白となっている

86

どこが間違っているでしょうか②

```
# Keio4.rb  
puts "Hello!"
```

```
Z:¥Documents¥Ruby>ruby -Ks Keio4.rb  
Keio4.rb:2: unterminated string meets end of file
```

87

どこが間違っているでしょうか②

```
Z:¥Documents¥Ruby>ruby -Ks Keio4.rb  
Keio4.rb:2: unterminated string meets end of file
```

```
# Keio4.rb  
puts "Hello!"
```

全角文字の「」

88

どこが間違っているでしょうか③

```
Z:¥Documents¥Ruby>ruby -Ks Keio5.rb  
Keio5.rb:1:in `<main>': uninitialized constant Keio5  
(NameError)
```

全角文字

```
# Keio5.rb  
puts "Hello!"
```

89

繰り返しになりますが...

- 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
→ ただし、文字コードを指定しなければなら
ない(次週以降説明します)

90

練習問題

91

他の例題①

(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)

四則演算を行なうRubyプログラム

```
# 四則演算 # 以降はコメントです
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"
```

```
Z:\Documents\Ruby>ruby sample2.rb
a+b=9
a-b=1
a*b=20
a/b=1
```

92

他の例題②

日本語を出力するRubyプログラム

```
# coding: Windows-31J
puts "こんにちは"
Z:\Documents\Ruby>ruby sample3.rb
こんにちは
```

```
# coding: Windows-31J
puts "春の"
puts "うららの"
puts "隅田川"
Z:\Documents\Ruby>ruby sample4.rb
春の
うららの
隅田川
```

圧縮ファイルの展開方法

94

圧縮ファイルの展開方法①

①右クリック*

②「すべて展開」を選択

展開したいフォルダーを指定

③「展開」をクリック

*ダブルクリックではファイルは解凍されないので注意

95

圧縮ファイルの展開方法②

展開されたフォルダー

展開されたファイル

96

Rubyに関する情報

97

バージョンでの違い

- Rubyには複数のバージョン(最新のバージョンは2.4.1)があり、少々違いがあります
- 日吉ITCのPCにインストールされているのは2.4系(2.4.0)のバージョンです

98

Ruby 関連サイト(2017年4月現在)

- 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」→「Windows版Rubyバイナリ」
 - 本文中の「ActiveScriptRuby」→「Ruby-2.4.0 Microsoft Installer Packages」
- <http://www.namaraii.com/rubytips/> も便利です

99

参考書

- 各種出ています。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

- Ruby MSI Packages
 - <http://www.artonx.org/data/asr/> から最新版(Ruby-2.4.0 Microsoft Installer Packages)をダウンロードして下さい
 - 環境変数の設定はインストールするPCによって異なります

102

Ruby のインストール②

ダブルクリック

Ruby 2.4-x64 セットアップ ウィザードへようこそ

インストール フォルダの選択

103

Ruby のインストール③

インストールの確認

インストールが完了しました。

104

Rubyの実行

- 「Windowsボタン」→「Ruby-2.4-x64」
- 「Ruby-2.4 console」*

Rubyのバージョンの確認
> ruby -v

*「Ruby-2.4(Administration) console」でもよい

105

環境変数(Path)の設定

- ただし、このままではITCのPCと同様に、コマンドプロンプトからはrubyコマンドは実行できない
- ITCのPCと同様にコマンドプロンプトからrubyコマンドを実行したい場合
 - Pathの設定が必要
 - Pathとは？

106

Pathの設定例① (Windows10の場合)

コンピュータ

システムのプロパティ

設定の変更

107

Pathの設定例②

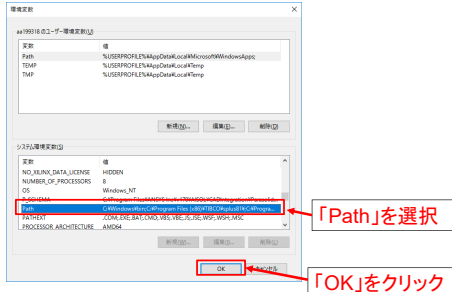
詳細設定

環境変数

「環境変数」をクリック

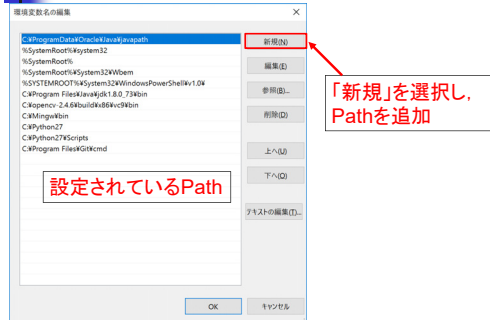
108

Pathの設定例③



109

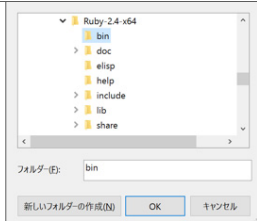
Pathの設定例④



110

Pathの設定例⑤

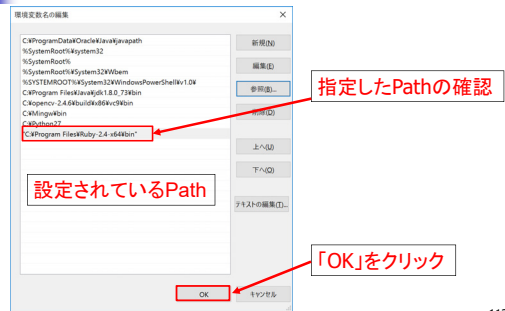
「Program Files」→「Ruby-2.4-x64」
→「bin」を指定*



*インストールの際に指定した
フォルダ名によって異なる

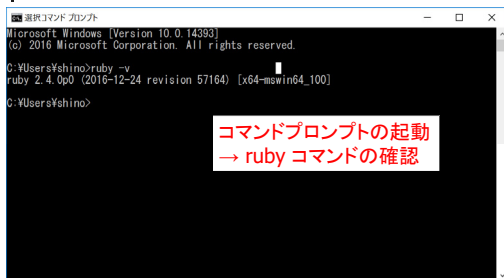
111

Pathの設定例⑥



112

Rubyの実行



113