

基礎プログラミングI

第1回：プログラミング・RUBYとは

KAROL NOWAKOWSKI
カール・ノヴァコフスキ

プログラミングとは？

コンピュータに実行してほしい作業に関する指示（**プログラム**）を記述すること

コンピュータは何語で指示を受ける？

デジタルコンピュータ

digital computer

“ThinkCentre S50” by dno1967b, CC BY 2.0



“LYF WATER 2 Smartphone” by Nairspecht, CC BY-SA 4.0

デジタルコンピュータ

Source: <https://www.blog-wajkomp.pl/>

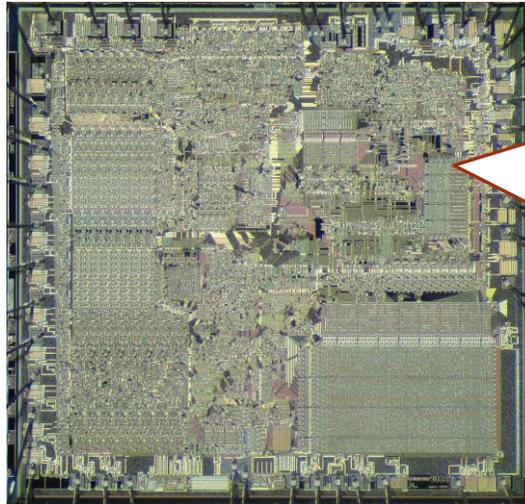


「AKAT-1」（デジタルではない、アナログコンピュータの例）

デジタルコンピュータ

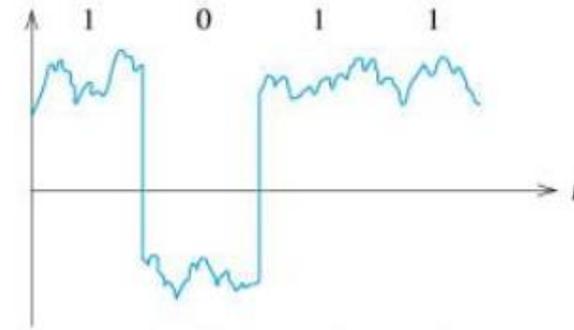
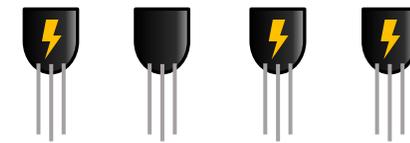
digital* computer

コンピュータの内部



"AMD 8088 die" by Pauli Rautakorpi, CC BY 3.0

トランジスタ



電圧



"ThinkCentre S50" by dno1967b, CC BY 2.0

* 語源 : digit (数字)

機械語 machine code

```
0011000010  
0100001101  
0011000101  
0100001110  
0010001101  
0110001110  
0100001111  
1110000000
```

意味 : 2 + 5

アセンブリ言語 assembly languages

Machine code	Assembly code	Description
001 1 000010	LOAD #2	Load the value 2 into the Accumulator
010 0 001101	STORE 13	Store the value of the Accumulator in memory location 13
001 1 000101	LOAD #5	Load the value 5 into the Accumulator
010 0 001110	STORE 14	Store the value of the Accumulator in memory location 14
001 0 001101	LOAD 13	Load the value of memory location 13 into the Accumulator
011 0 001110	ADD 14	Add the value of memory location 14 to the Accumulator
010 0 001111	STORE 15	Store the value of the Accumulator in memory location 15
111 0 000000	HALT	Stop execution

意味 : 2 + 5

高水準言語 high-level languages

例 : Ruby

```
number = 1 + 3
```

```
1 PI = 3.14 # 円周率
2 radius = 10 # 円の半径
3
4 puts 2 * radius # 直径を出力する
5 puts 2 * PI * radius # 円周を出力する
6 puts PI * radius * radius # 面積を出力する
7
```

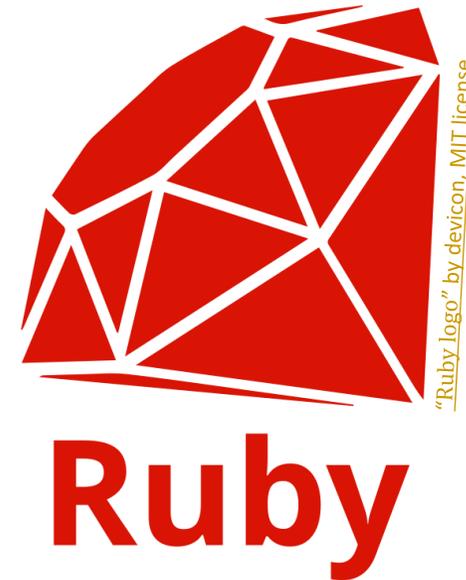
```
1 puts "What's your name?"
2 name = gets # 名前を入力する
3
4 puts "Hello, #{name}"
5
6 # 他の処理
7
8 puts "Bye, #{name}"
```

高水準言語

high-level languages

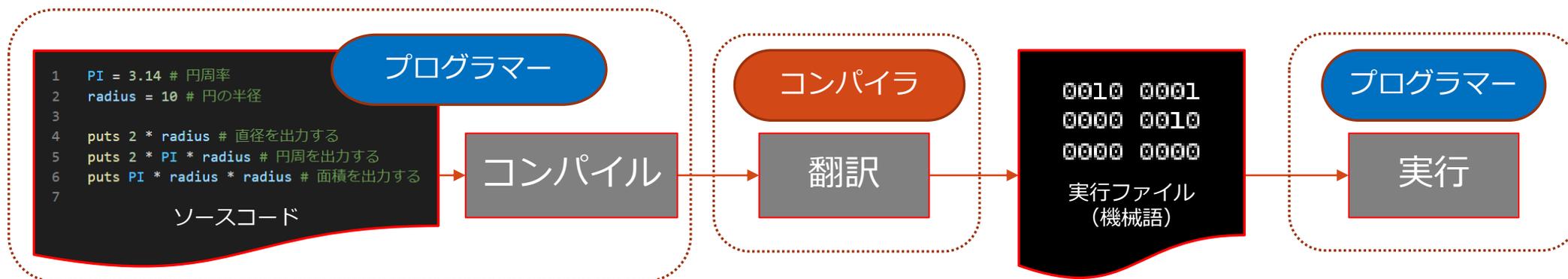
高水準言語の例：

Ada	Kotlin
Algol	Lisp
B	Lua
Basic	Objective-C
C	Pascal
C++	Perl
C#	PHP
Clojure	Prolog
Cobol	Python
Delphi	R
Eiffel	Rust
Fortran	Scala
Go	Smalltalk
Haskell	Swift
Java	SQL
JavaScript	Visual Basic

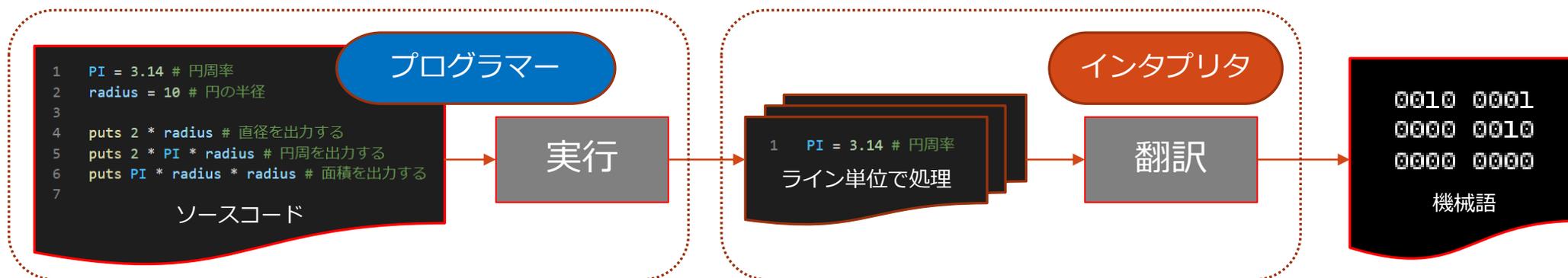


高水準プログラミング言語の2種類

1. コンパイル型 compiled languages



2. インタプリタ型 interpreted languages



高水準プログラミング言語の2種類

1. コンパイル型 compiled languages

例： C, C++, Java, Fortran

2. インタプリタ型 interpreted languages

例： Ruby, Python, JavaScript

高水準プログラミング言語の2種類

1. コンパイル型 compiled languages

例： C, C++, Java, Fortran

利点：（実行時の）処理速度；一度コンパイルすれば何度でも実行ファイルを使える

2. インタプリタ型 interpreted languages

例： Ruby, Python, JavaScript

利点： ソースコードを変更するたびにコンパイル作業を行う必要がない

ルビー Ruby

- 1993から開発され、1995年にはじめて公開された
- 開発者：松本行弘（まつもとゆきひろ）
- インタプリタ型
- オブジェクト指向
- Ruby on Railsという、人気のWebアプリケーション開発フレームワークで使われている

["Yukihiro Matsumoto" by Cep21](#)

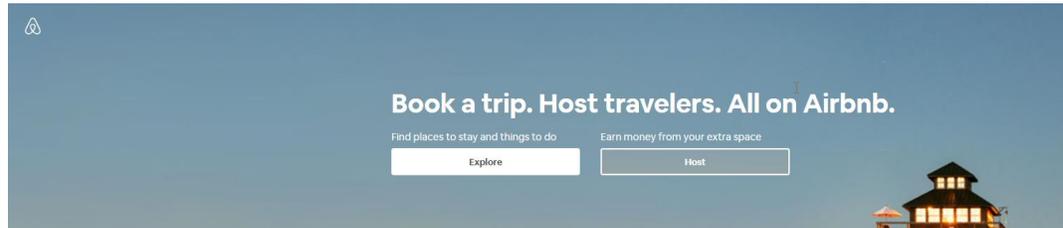


松本行弘さん

Rubyが活用されているところ

海外：

Airbnb（宿泊・民泊サービス）



Book unique homes and experiences

WHERE: Anywhere
CHECK-IN - CHECKOUT: mm/dd/yyyy - mm/dd/yyyy
GUESTS: 1 guest
Search

What guests are saying about homes in United States

★ United States homes were rated 4.8 out of 5 stars with 31,500,000+ reviews



★★★★★
Beautiful neighborhood, and convenient to many things in Houston. Beautiful property! The place was clean and everything well...



Michael
United States



★★★★★
Stewart's place was awesome! He was friendly and was there to greet us and get us all set up. He recommended a delicious dinner place an...



Nicole
United States



★★★★★
Everything was smooth and easy. Very nice place, very clean, conveniently located, and even has a very relaxing little yard. Would...



Dimitry
United States

Github（ソースコード管理・共有サービス）



Rubyが活用されているところ

日本国内：

価格.com（価格比較サイト）

The screenshot shows the homepage of price.com. At the top, there is a search bar with the text "何をお探しですか？ (メーカー、製品カテゴリ、製品名、型番...)" and a "検索" button. Below the search bar, there are several promotional banners and a grid of product categories. The categories include:

- パソコン: ノートパソコン, タブレット, ハードディスク, PCパーツ, 周辺機器
- 家電: テレビ, オーディオ, 生活・調理家電, 健康・美容家電, 冷蔵庫, 洗濯機
- カメラ: デジカメ, 一眼レフ, ミラーレス, レンズ, ビデオカメラ, メモリカード
- スマートフォン・携帯電話: iPhone, Androidスマホ, SIMフリー, 格安スマホ, カラーケー, ウェアラブル
- 格安SIMカード: 通話SIM, データSIM, スマホ代節約, MVNO, 料金比較, 人気ランキング
- プロバイド: 料金比較, 光回線, 光コラボ, WIMAX, インターネット接続, モバイル回線
- 自動車・バイク: 新車, 中古車, カー用品, バイク用品, 中古車査定, バイク査定, 車検見積もり
- 自動車保険: 一括見積もり, 満足度ランキング, 基礎知識, 選び方, 手続き, 業界動向
- モバイルデータ通信: モバイルルーター, モバイルWi-Fi, レンタルWi-Fi, 海外Wi-Fi
- 保険: 生命・医療・がん・学費・年金保険, 海外旅行・火災・バイク・ペット保険
- クレジットカード: ポイント還元率, 年会費無料, ETC, ゴールドカード, 海外旅行, マイル
- ローン: カードローン, 住宅ローン, 借り換え, 自動車・マイカー, バイクローン
- スポーツ: ゴルフ, 自転車, 釣具, サッカー, 野球, スポーツウェア, ダイエット
- アウトドア: キャンプ, バーベキュー, 登山, テント, 寝袋, ライト, 水筒
- 投資・資産運用・FX: 証券会社, 投資信託, FX, 暗号資産, 仮想通貨, ビットコイン

食べログ（飲食店検索サービス）

The screenshot shows the search results page on Tabelog for "酒田市のグルメ・レストラン". The search criteria are: 酒田市, キーワード [例:焼肉、店名、個室], 2022/5/31(火), 19:00, 2名. The results are filtered by "市区町村" (City/Town/Village) to "酒田市". The page shows a list of restaurants, with "オールデイダイニング 月のみち" (All Day Dining Tsuki no Michi) highlighted. The restaurant details include:

- ジャンル: イタリアン, 日本酒バー
- 評価: 3.12 (5 stars), 5住, 87件
- 価格帯: ¥3,000~¥3,999
- 営業時間: 無休

プログラミングと人工知能

最近話題になっているChatGPTといったAIモデルはプログラミングもできる。

プログラミングと人工知能

前年度講義の課題の一つをChatGPTに与えたら、完璧なプログラムを作成した。

1) 課題の説明

受けたテストの数とそれぞれのテストの点数（100点満点）をユーザーに入力してもらって、配列に保管し、それぞれのテストの平均点数との差と成績を出力するプログラム test_grades.rbを作成せよ。

成績の基準：

平均 9 0 以上	→ 秀
平均 8 0 点以上で 9 0 点未満	→ 優
平均 7 0 点以上で 8 0 点未満	→ 良
平均 6 0 点以上で 7 0 点未満	→ 可
平均 6 0 点未満	→ 不可

なお、ユーザーが入力した点数が100点を超えている場合は「100点以下の点数を入力してください」と表示し、前回の授業で紹介したredoを使って再度入力するようにしてください。

ヒント

2つのループが必要になる：

- 1) ユーザーがそれぞれのテスト点数を入力するループ
- 2) それぞれの点数の平均との差と成績を計算し表示するループ

実行結果の例：

```
sime(c1f0xxx)% ruby test_grades.rb
受けたテストの数を入力してください
3
1回目のテスト点数を入力してください (100点満点)
70
2回目のテスト点数を入力してください (100点満点)
80
3回目のテスト点数を入力してください (100点満点)
180
100点以下の点数を入力してください!
3回目のテスト点数を入力してください (100点満点)
100
```

点数	平均との差	成績
70	-13.33	良
80	-3.33	優
100	16.67	秀

2) ChatGPTの答え

```
以下はRubyで書かれたプログラムの前です。

loop do
  puts "受けたテストの数を入力してください"
  test_num = gets.to_i

  # 入力されたテストの数をもとに、配列を用意
  test_scores = []
  (1..test_num).each do |i|
    loop do
      puts "#(i)回目のテスト点数を入力してください (100点満点)"
      score = gets.to_i
      if score > 100
        puts "100点以下の点数を入力してください"
        redo
      else
        test_scores << score
        break
      end
    end
  end

  # 平均点を計算
  average_score = test_scores.sum / test_scores.size.to_f

  # 各テストの点数、平均点との差、成績を出力
  puts "\n\n点数 平均との差 成績"
  test_scores.each do |score|
    diff = (score - average_score).round(2)
    grade = case score
             when 90..100 then "秀"
             when 80...90 then "優"
             when 70...80 then "良"
             when 60...70 then "可"
             else "不可"
            end
    puts "%4d %11.2f %7s" % [score, diff, grade]
  end

  # もう一度実行するかどうかをユーザーに確認
  puts "\nもう一度実行しますか? (はい: y, いいえ: n)"
  answer = gets.chomp.downcase
  break if answer == "n"
end
```

3) 実行結果

```
Ruby
Run

1 loop do
2   puts "受けたテストの数を入力してください"
3   test_num = gets.to_i
4
5   # 入力されたテストの数をもとに、配列を用意
6   test_scores = []
7   (1..test_num).each do |i|
8     loop do
9       puts "#(i)回目のテスト点数を入力してください (100点満点)"
10      score = gets.to_i
11      if score > 100
12        puts "100点以下の点数を入力してください"
13        redo
14      else
15        test_scores << score
16        break
17      end
18    end
19  end

20
21 # 平均点を計算
22 average_score = test_scores.sum / test_scores.size.to_f
23
24 # 各テストの点数、平均点との差、成績を出力
25 puts "\n\n点数 平均との差 成績"
26 test_scores.each do |score|
27   diff = (score - average_score).round(2)
28   grade = case score
29            when 90..100 then "秀"
30            when 80...90 then "優"
31            when 70...80 then "良"
32            when 60...70 then "可"
33            else "不可"
34            end

> bundle exec ruby main.rb
受けたテストの数を入力してください
3
1回目のテスト点数を入力してください (100点満点)
100
100点以下の点数を入力してください
1回目のテスト点数を入力してください (100点満点)
124
100点以下の点数を入力してください
1回目のテスト点数を入力してください (100点満点)
100
2回目のテスト点数を入力してください (100点満点)
81
3回目のテスト点数を入力してください (100点満点)
60

点数 平均との差 成績
100 19.67 秀
81 0.67 優
60 -20.33 可

もう一度実行しますか? (はい: y, いいえ: n)
n
>
```

プログラミングと人工知能

- 今後はこれまで人間が行っていたコーディング作業の一部を自動化できる可能性が高い
- 従来のような形でプログラミングを行う人材の需要が減っていくだろう
- プログラミング以外のスキルを身につけるなど、自らの競争力を高める方法を今から考えるべき

プログラミングと人工知能

ただ、

- 人間に代わってAIがプログラムを書くようになるとしても、そのAIに指示を与える人間は全くプログラミングについて知識がなくても良いとは限らない
- 何かの仕事のマネジメントを適切に行うためには、その仕事の内容に関する知識が必要
- AIもミスをすることがある
 - 「バグ」と呼ばれる論理的な間違い
 - 一応動くけど効率が悪いコード
- そのような問題を見つけて、直すにはやはりプログラミングの知識を持つ人間が必要

プログラミングと人工知能

- プログラミングに必要な能力というのはRubyのようなプログラミング言語の知識だけではない
- プログラミングする = 計算機に指示を与える
- これまではRubyといった人工言語でその指示を記述していたが、今後は自然言語（例えば日本語）を使ってAIに指示を与える形になっていく
- プログラミング言語と比べれば自然言語には曖昧な表現が多いので気をつけないと期待していたような結果を得られないおそれがある

「田中、この書類を何とかして」

→ 具体的にどうすればいいかわかれていない。「処分してください」というふうに解釈されるかもしれない。

「田中、この書類を修正して」

→ どこをどのように修正すればいいか説明されていないので指示を与えた人が期待しているような結果にならないかもしれない。

「田中、この書類では会社の住所が間違っているよ。ホームページの住所に直して」

→ 具体的な説明があるので曖昧性が少ない。

プログラミングと人工知能

- プログラミングに必要な能力というのはRubyのようなプログラミング言語の知識だけではない
- プログラミングする = 計算機に指示を与える
- これまではRubyといった人工言語でその指示を記述していたが、今後は自然言語（例えば日本語）を使ってAIに指示を与える形になっていく
- プログラミング言語と比べれば自然言語には曖昧な表現が多いので気をつけないと期待していたような結果を得られないおそれがある
- **プログラミングを学ぶというのは、論理的かつ具体的に考えて実行してほしい仕事の正しい手順（どの作業をどの順番で行えばいいか）を決めるスキルを身につけることでもある**
- その能力はAIを活用したプログラミングでも役に立つはずである