基礎プログラミングI

第3回　メソッド、値の型変換

メディア情報コース
平居　悠（ひらい　ゆたか）

到達目標

計算機上での情報の取り扱い方の基礎の理解

1. コンピュータに指示を送る基本を理解する。
2. 文字列や数値を基本とした値の概念とそれを格納する変数の仕組みを理解する。
3. 変数一つで集合を表す概念を理解する。
4. ２進数を基本としたコンピュータの内部表現について基本を理解する。
5. 現実社会の簡単な問題を扱うプログラムを作成できるようになる。

前回の目標

値と変数を理解し入出力を利用するプログラムを作成できるようになる。

前回の復習

値：
プログラムの中でやり取りされるデータ
変数：
値を入れる箱のようなもの
制御構造：
プログラムの実行順序などを変える仕組み

今回

第１回	４月９日	プログラミングの基礎
第２回	４月１６日	変数・制御構造
第３回	**４月２３日**	**メソッド、値の型変換**
第４回	４月３０日	確認テスト
第５回	５月１４日	集合処理１（配列）
第６回	５月２１日	集合処理２（CSVとデータ処理）
第７回	５月２８日	集合処理３（ハッシュ）
第８回	６月１１日	正規表現
第９回	６月１８日	計算機の内部表現
第１０回	６月２５日	スタイルとデバッグ、実用的なプログラム
第１１回	７月　２日	作品の公開
第１２回	７月　９日	チーム課題作成
第１３回	７月１６日	チーム課題発表
第１４回	日程未定	期末試験

今回の目標

入力値により異なる処理を行う
プログラムが作成できる

導入課題

チーム内で2人もしくは3人組を作り、以下から1つ問題を選んで協力して解答せよ。解答はs4基礎プロI(G, H)の「#03 導入課題」に指示通り書き込む。ただし、チーム内のペアごとに異なる問題を解答すること。書き込みは全員行うこと。

1. **メソッドの例を3つ挙げよ。**
2. **値の型変換とは何か。**
3. **x＝“2.72”.to_iとした時、xの値はいくつか？**

タイピング練習スケジュール

第１回　　ホームポジション
第２回　　ローマ字
第３回　　英語初級
第４回　　日本国憲法（trr試験 ）
第５回　　ホームポジション
第６回　　ローマ字
第７回　　英語初級
第８回　　日本国憲法（trr試験）
第９回　ホームポジション
第１０回　ローマ字
第１１回　英語初級
第１２回　日本国憲法（trr試験、合格スコア150）

trr起動方法

1. ブラウザを起動し、https://www.koeki-prj.org/trr/ に繋ぐ。
2. 学籍番号（Cは大文字、省略なし８桁）を入力する。
3. Koeki MAILに届いたパスコードをPasscode: 欄に入力する。

ホームポジション


左手でタイプするキー
右手でタイプするキー
小指
薬指
中指
人差指
人差指
中指
薬指
小指
漢字
Back space
Tab
Enter
Caps Lock
Shift
Shift
Ctrl
Fn
Alt
Space
Alt
Fn
Ctrl
左手の人差指から小指までのホームポジション
両手の親指のホームポジション
右手の人差指から小指までのホームポジション

今回の内容

1. 足し算プログラム復習
2. 入出力を行うプログラムの例
3. マッチ棒取りゲーム

今回の内容

1. 足し算プログラム復習
2. 入出力を行うプログラムの例
3. マッチ棒取りゲーム

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

sum変数を0にする

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了はq): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は%dです\n", sum)
end
printf("合計は%dです\n", sum)
```

対応するendまでを
ひたすら繰り返す

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

「足したい数 (終了は q):」と標準エラー出力（画面出力のこと）に出力する。

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了はq): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は%dです\n", sum)
end
printf("合計は%dです\n", sum)
```

数値を入力させline変数に代入

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

入力したものが“q”ならば…

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

whileループを抜ける

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

ifでの場合分けはここで終わり

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

lineを整数に直したものをsumに加算

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

while文内のsumの値を整数値で出力

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

whileに対応するend

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

sum = 0
while true
 STDERR.print "足したい数(終了は q): "
 line = gets.chomp
 if line == "q" then
  break
 end
 sum += line.to_i
 printf("これまでの合計は %d です\n", sum)
end
printf("合計は %d です\n", sum)
```

最終的なsumの値を
整数値で出力

コメントの書き方

#記号以降の部分を**コメント**という。プログラムの実行に影響を与えないので自由な文を書くことができる。わかりやすいコメントを入れながらプログラムを作る習慣を付けよう。

例：

```
sum = 0 # sum変数を0とする。
```

今回の内容

1. 足し算プログラム復習
2. 入出力を行うプログラムの例
3. マッチ棒取りゲーム

入出力を行うプログラム例

double.rbというファイルを作成し、右のプログラムを書いて実行してみよう。

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

while true
 print "好きなことは(qで終了)?: "
 koto = gets
 if koto == "q\n" then
  break
 end
 printf("%sだ、", koto)
 printf("%sだ楽しいな。\n",
koto.chomp)
end
```

出力例

```
% ./double.rb
好きなことは(qで終了)?: 祭
祭
祭
だ、祭だ楽しいな。
好きなことは(qで終了)?: q
```

「祭」と入力した場合の説明

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

while true
 print "好きなことは(qで終了)?: "
 koto = gets
 if koto == "q\n" then
  break
 end
 printf("%sだ、", koto)
 printf("%sだ楽しいな。\n",
koto.chomp)
end
```

getsメソッドが入力した「祭」を受け取り“祭\n”という文字列を作って返す。“祭\n”がkoto変数に代入される。

「祭」と入力した場合の説明

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

while true
  print "好きなことは(qで終了)?: "
  koto = gets
  if koto == "q\n" then
    break
  end
  printf("%sだ、", koto)
  printf("%sだ楽しいな。\n",
koto.chomp)
end
```

「祭」＋「改行文字(\n)」が出力される

「祭」と入力した場合の説明

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

while true
  print "好きなことは(qで終了)?: "
  koto = gets
  if koto == "q\n" then
    break
  end
  printf("%sだ、", koto)
  printf("%sだ楽しいな。\n",
koto.chomp)
end
```

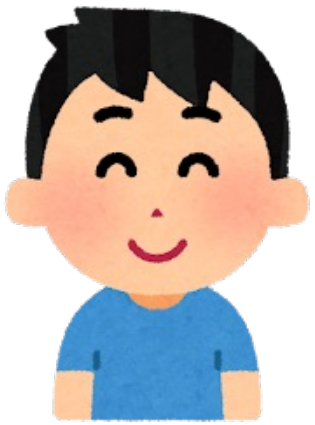
文字列をchompすることで、改行文字を削除して出力

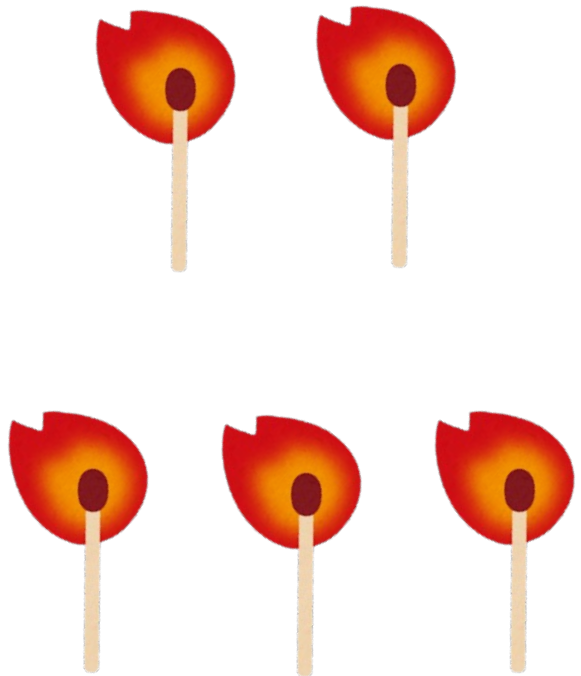
今回の内容

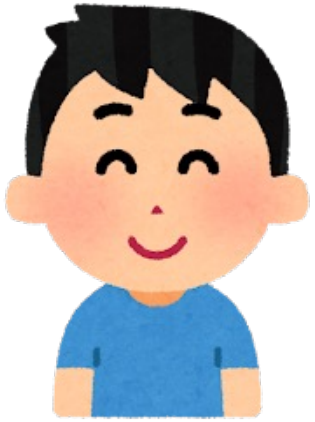
1. 足し算プログラム復習
2. 入出力を行うプログラムの例
3. マッチ棒取りゲーム

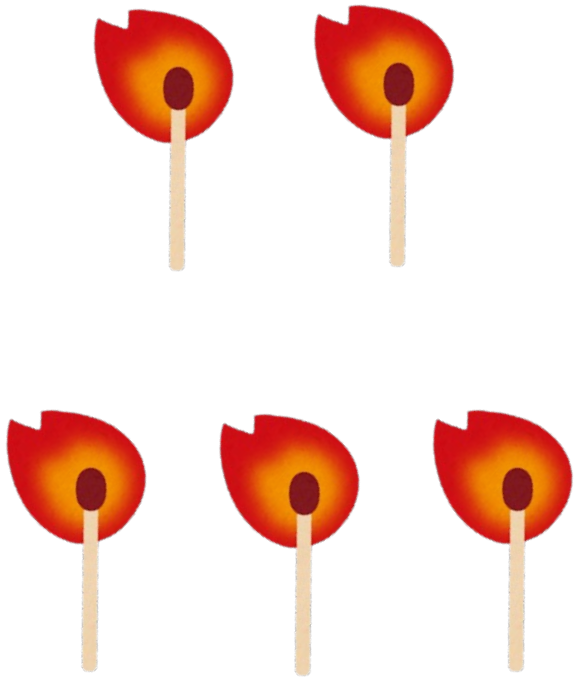
マッチ棒取りゲーム

1から3本のマッチ棒を順に取る。
最後の1本を取った方が負け。

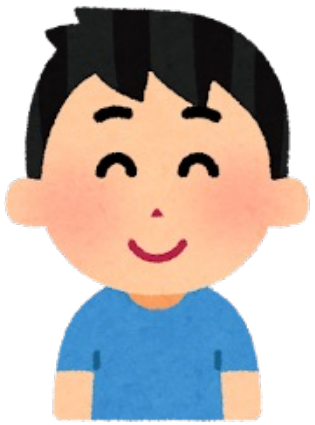


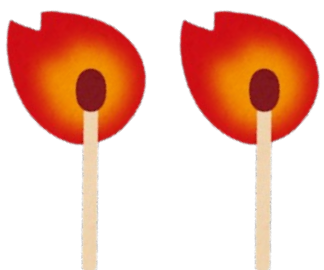


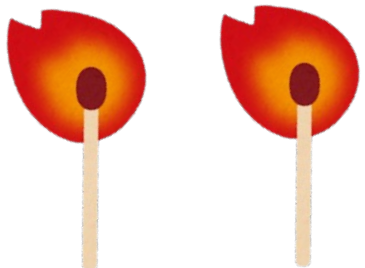


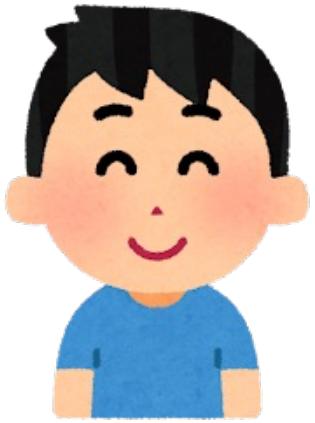


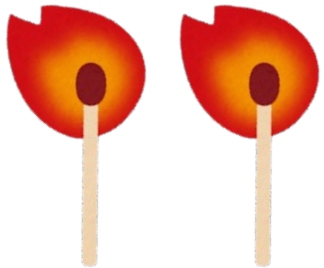
あなたの番：マッチ棒が5本あります。何本取りますか？

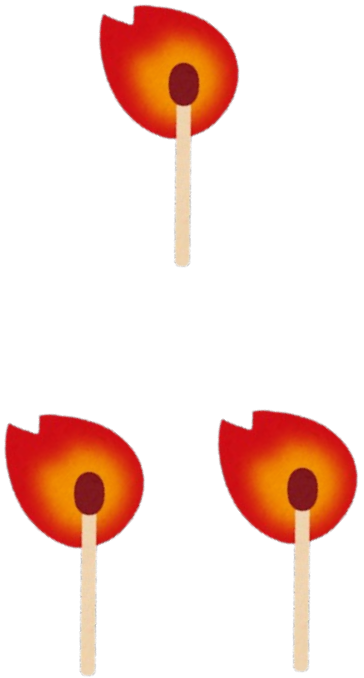




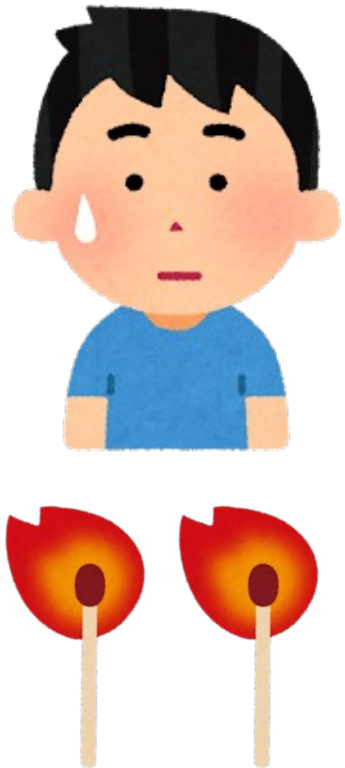




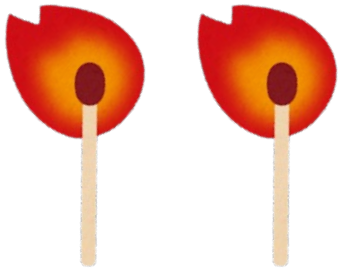


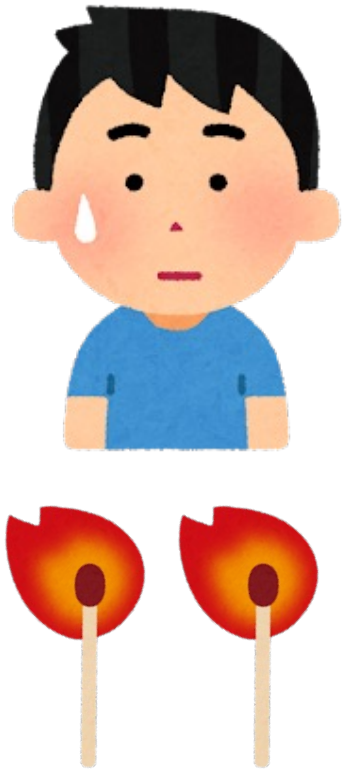


わたしの番：マッチ棒が3本あります。

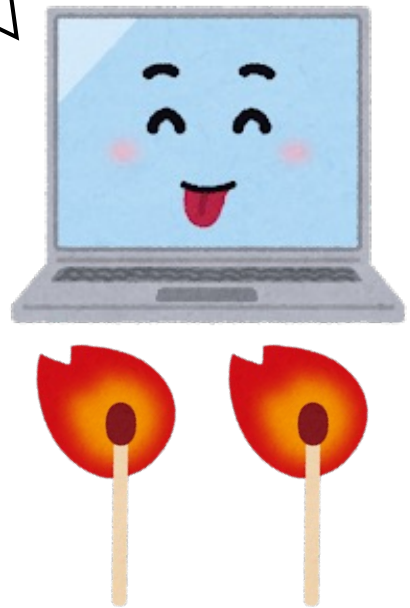


わたしは2本取りました。





あなたの番：マッチ棒が1本あります。何本取りますか？




最後の1本を取りました。あなたの負けです！やあい

マッチ棒取りゲームのプログラム

右のプログラム match.rbを書いて実行してみよう。

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

match = 17

print "1から3本の範囲でマッチ棒を取ります。最後の1本を取った方の負け!\n"

while match > 0
 printf("あなたの番: マッチ棒が%d本あります\n", match)
 print "何本取りますか?(1..3): "
 toru = gets.chomp.to_i
 if toru < 1 || toru > 3 then
  print "1から3の範囲で取って下さい.\n"
  redo
 end
 match -= toru
 if match < 1 then
  print "最後の1本を取りました。あなたの負けです! やあい\n"
  break
 end
 printf("わたしの番: マッチ棒が%d本あります\n", match)
 com_toru = 3-toru+1
 printf("私は%d本取りました。\n", com_toru)
 match -= com_toru
 if match < 1 then
  print "最後の1本を取りました。わたしの負けです! くやちいい!!\n"
  break
 end
end
```

授業内課題: TA試問

match.rbのそれぞれの行についてその意味をTAに説明する。

今回の目標

入力値により異なる処理を行う
プログラムが作成できる

今回の内容

1. 足し算プログラム復習
2. 入出力を行うプログラムの例
3. マッチ棒取りゲーム

まとめ

メソッド：
あらかじめ決められた処理を行ってくれるもの
値の型変換：
あるデータ型を別のデータ型に変更すること

課題

match.rbは

- 17本でゲーム開始
- 取れるのは1から3本
- 最後の1本を取った人が負け
- コンピュータが必勝

というプログラムである。これを改造したsupermatch.rbを作成せよ。次ページどれか1つを選択する。いずれの場合もゲーム開始時のマッチ棒の本数はよく考えて決めること。

課題

1. ゲームを起動したら、1度に取れる最大本数をプレイヤーが好きな数に指定できるようにする（ただしコンピュータ必勝にする）。
2. ゲーム起動後、先手か後手かをプレイヤーが選べるようにする（ただしコンピュータ必勝にする）。
3. ゲーム起動後、最初の本数をプレイヤーが選べるようにする（ただしコンピュータ必勝にする）。
4. 上記全てを自由に指定できるようにする（コンピュータが負ける可能性がある）。
5. 構造は同じだがルールの全く違う新しいゲームにする（内容自由、ファイル名自由）。

レポート提出方法

s4基礎プロIG, Hの「#3 提出課題 (課題3)」に以下の通り書き込む。

1. **学籍番号　氏名**
2. **どういうプログラムを作るか（何番を選択したか）**
3. **プログラムの説明（どのように考えてどこをどう修正したか重要な行を引用しつつ8行以内で説明する）**
4. **参考文献（AI利用は不可）**
5. **quiz ruby-4の結果**
6. **感想**

以上を本文エリアに書き、作成したプログラムsupermatch.rbとプログラムを動かした結果（画面）ファイルscreen.txtを添付すること。

締め切り：4月27日 (月)

実行結果ファイル作成法

プログラムを実行した画面はテキストファイルに保存する。例えば、supermatch.rbを実行し、screen.txtに保存する場合は以下のようにする。

1. プログラムが提出できる状態まで完成させ実行する。
2. 念の為結果ファイル (screen.txt)を削除する。

```
rm -f screen.txt
```

3. 作成したプログラムを以下のように実行する。

```
script -c 'ruby supermatch.rb' screen.txt
```

これらの手順で作成したファイル (screen.txt)をs4に添付する。

2回以上の実行結果をまとめたいときは、「追記」を意味する-aオプションを指定する。

```
script -a -c 'ruby supermatch.rb' screen.txt
```

次回

第１回	４月９日	プログラミングの基礎
第２回	４月１６日	変数・制御構造
第３回	４月２３日	メソッド、値の型変換
第４回	**４月３０日**	**確認テスト**
第５回	５月１４日	集合処理１（配列）
第６回	５月２１日	集合処理２（CSVとデータ処理）
第７回	５月２８日	集合処理３（ハッシュ）
第８回	６月１１日	正規表現
第９回	６月１８日	計算機の内部表現
第１０回	６月２５日	スタイルとデバッグ、実用的なプログラム
第１１回	７月　２日	作品の公開
第１２回	７月　９日	チーム課題作成
第１３回	７月１６日	チーム課題発表
第１４回	日程未定	期末試験

事前課題

1. 第2回解説動画(https://youtu.be/o300Tdk3Su0)を視聴する。
2. 第3回解説動画(https://youtu.be/p2YHv7YtdNQ)を視聴する。
3. ターミナルでquiz ruby-4を3回以上行い、最も早くクリアした記録をs4基礎プロIG, Hの「#04 事前課題 (ruby-4)」に貼り付ける。

締切：4月29日 (水)