プログラミング演習 I
– 第 1、2 回　プログラムに慣れる –

1　演習の進め方

教科書を参照しながら、演習をすすめよう。(演習*) の下にプログラムが書かれている場合は、プログラムを実行し、実行結果を確かめよう。(演習*) の下に文章が書かれている場合は、教科書等を参照し、調べよう。

各演習に対応するように、ファイル名を考えて付けて残しておくようにしよう。(例えば、ex1-1.rb など) 復習するときに便利である。

2　プログラムをつかってみよう

CGI プログラムを使ってみよう。

演習 1-1　`http://klis.tsukuba.ac.jp/klib/Subjects/ProgI/example2014/` にアクセスして、プログラムを実行してみよう。

プログラムをダウンロードし、実行してみよう。

演習 1-2　`http://klis.tsukuba.ac.jp/klib/Subjects/ProgI/example2014/` にアクセスして、プログラムをダウンロードして、実行してみよう。

注意事項

プログラミング演習 I のテキスト内の\(バックスラッシュ) は、キーボードの￥(エンマーク) を使って入力する。

3　Rubyをとにかく使おう その1

ここでは、簡単なプログラムの入力と実行を繰り返してもらい、プログラミングに慣れることを目標とする。プログラムを1行毎に理解する前に、基本的な用語の確認とプログラムでよく使う命令文の確認を行おう。ここでは、全ての事項を完全に覚える必要はない。(もちろん、きちんと理解するに越したことはない。)まずは、全体の雰囲気をつかんでほしい。

どのようなプログラムを記述すると、下記に示したことを行うことが出来るのだろうか?
まずは、演習1-3から点線内に示されているプログラムを実行し、また教科書を使って調べながらすすめていこう。各演習毎に別々のファイルを作成すること。ファイル名は、( )内に示したように演習番号と対応付けておくとよい。

プログラミング演習I前半で行う内容の一部

■　文字列を表示するには?
- □　改行するには?
- □　日本語を表示するには?

■　演算を行うには?
- □　3　足す　2
- □　3　割る　2

■　変数って何?

■　キーボードを使って値を入力するには?
- □　入力した値を保存しておきたい
- □　文字列を入力するには?
- □　数値を入力するには?

■　型変換って何?
- □　文字列 `9999` を　数値 `9999` に変換するには?
- □　小数を扱うには?

教科書

楽しいRuby 第4版 p.7(楽しいRuby 第3版 p.8)- プログラムの解説

演習 1-3　(ex1-3.rb)

```
------------------------------------------------
print("Hello, Ruby.\n")
------------------------------------------------
```

演習 1-4

文字列オブジェクトとは何か
`\n`　は、何を表すか
`print` メソッドとは何か
引数とは何か

教科書

p.8(p.10)- 文字列　(記述方法によって、なぜ結果が異なるのか説明できるようにしよう。)

演習 1-5　(ex1-5.rb)

```
------------------------------------------------
print("Hello, \nRuby\n!\n")
------------------------------------------------
```

演習 1-6　(ex1-6.rb)

```
------------------------------------------------
print("Hello, \"Ruby\"!\n")
------------------------------------------------
```

演習 1-7　(ex1-7.rb)

```
------------------------------------------------
print("Hello \\ Ruby!\n")
------------------------------------------------
```

演習 1-8　(ex1-8.rb)

```
------------------------------------------------
print('Hello,\nRuby\n!\n')
------------------------------------------------
```

演習 1-9

演習 1-5 から演習 1-8 の表示結果が異なる理由を説明しよう。

演習 1-10　以下のプログラム中の「文章を入力してみよう」と書かれている部分を書き換えよう。

```
------------------------------------------------
print("文章を入力してみよう")
------------------------------------------------
```

教科書

p.11(p.12)- メソッドの呼び出し

演習 1-11

```
------------------------------------------------
print("Hello, ", "Ruby", ".", "\n")
------------------------------------------------
```

演習 1-12　演習 1-11 と表示結果が同じになることを確認しよう。

```
------------------------------------------------
print("Hello, ")
print("Ruby")
print(".")
print("\n")
------------------------------------------------
```

教科書

p.14(p.15)- 日本語の表示

演習 1-13　下記のプログラムを実行してみよう。エラーが表示されたら、一行目に `#encoding: Shift_JIS` を追加して保存した後、実行してみよう。

```
------------------------------------------------
print("いづれの御時にか女御更衣あまたさぶらひたまいけるなかに\n")
print("いとやむごとなき際にはあらぬがすぐれて時めきたまふありけり\n")
------------------------------------------------
```

演習 1-14　「日本語を扱う場合の注意」を確認しておこう。
以後、テキスト内で日本語を含むプログラム例には、全て先頭行に `#encoding: Shift_JIS` を追加すること。

教科書

p.175(p.175)- 算術演算

演習 1-15　算術演算を行うときに使う、+ - / * の意味を調べよう。

教科書

p.17(p.19) - 四則演算（四則演算を行ってみよう。結果が同じになるか確かめよう。）

演習 1-16

```
------------------------------------------------
print(1.0 + 2.0, "\n")
print(2.0 * 3.0, "\n")
print(5.0 - 8.0, "\n")
print(9.0 / 2.0, "\n")
------------------------------------------------
```

演習 1-17

```
-----------------------------------------------
print(1 + 2, "\n")
print(2 * 3, "\n")
print(5 - 8, "\n")
print(9 / 2, "\n")
-----------------------------------------------
```

演習 1-18　演習 1-16 と演習 1-17 の計算結果が異なる理由を調べよう。
(ヒント：整数と浮動小数点数の違いは？)

教科書

p.18(p.20)- 数学的な関数

演習 1-19

```
-----------------------------------------------
print("sin(3.1415) = ", Math.sin(3.1415), "\n")
print("sqrt(10000) = ", Math.sqrt(10000), "\n")
-----------------------------------------------
```

教科書

p.228(p.177)- Math モジュール

演習 1-20　Math モジュールのメソッドにはどのようなものがあるか調べよう。

演習 1-21　1 行目と 2 行目の表示される値は同じ？　異なる場合はその理由を考えよう。

```
-----------------------------------------------
print(5 * (12 - 8) - 15, "\n")
print(5 * 12 - 8 - 15, "\n")
-----------------------------------------------
```

演習 1-22　文字列の足し算を行うと、どのような結果が得られるか試そう。

```
-----------------------------------------------
print("I like" + " apple pie.", "\n")
-----------------------------------------------
```

教科書

p.19(p.21)- 変数

演習 1-23　「変数名 = オブジェクト」は何を意味するのか調べよう。

教科書

p.69(p.67)- 変数名のつけ方

演習 1-24　変数名のつけ方を調べよう。また、プログラミング演習版コーディング規約（演習ページ内）を参照して、コーディング規約を確かめておこう。

演習 1-25　変数が示すオブジェクトが変わることを確認しよう。

```
------------------------------------------------
my_String = "楽しい Ruby"
print(my_String, "\n")

my_String = "愉快な Ruby"
print(my_String, "\n")

my_String = "面白い Ruby"
print(my_String, "\n")
------------------------------------------------
```

演習 1-26　2,3,4 行目の表示結果は同じ？　異なる場合はその理由を考えよう。

```
------------------------------------------------
myString = "楽しい Ruby"
print(myString, "\n")
print(myString, "\n")
print("myString", "\n")
------------------------------------------------
```

演習 1-27　変数を使って、文字列を表示するプログラムを作成してみよう。

教科書

p.229(p.179) 数値型の変換

演習 1-28　1,3,5 行目の表記と結果の違いを見比べよう。

```
-----------------------------------------------
print(12 + 12)
print("\n")
print("12" + "12")
print("\n")
print("12 + 12")
print("\n")
-----------------------------------------------
```

演習 1-29

```
-----------------------------------------------
print("整数を入力してください\n")
x = gets.chomp
print("入力した整数は", x, "\n")
print("整数を入力してください\n")
y = gets.chomp
print("入力した整数は", y, "\n")
print("入力した整数の和は", x + y, "\n")
-----------------------------------------------
```

演習 1-30　演習 1-29 の表示結果の理由を考えよう。

演習 1-31　gets と chomp について調べよう。

演習 1-32　読みやすいプログラムを書く方法を調べよう。

p.21(p.23)「コメントを書く」を参照し、コメントの書き方を調べよう。
p.123(p.114)「読みやすいプログラムを書こう」を参照し、改行や空白の使い方を調べよう。
プログラミング演習版 Ruby コーディング規約 (演習ページ内) を参照し、プログラムを記述する際のポイントを確認しよう。

演習 1-33　以下のプログラムはエラーになる。なぜだろうか？

```
-----------------------------------------------
var1 = 2
var2 = "5"
print(var1 + var2, "\n")
-----------------------------------------------
```

演習 1-34　演習 1-33 との違いを説明せよ。

```
-----------------------------------------------
var1 = 2
var2 = "5"
print(var1.to_s + var2, "\n")
-----------------------------------------------
```

演習 1-35

```
-----------------------------------------------
var1 = 2
var2 = "5"
print(var1 + var2.to_i, "\n")
-----------------------------------------------
```

演習 1-36　演習 1-29 を数値の計算にするには、プログラムをどのように修正したらよいか考え、試そう。

演習 1-37

```
-----------------------------------------------
print("15".to_f, "\n")
print("99.999".to_f, "\n")
print("99.999".to_i, "\n")
-----------------------------------------------
```

演習 1-38　p.229(p.179)「数値型の変換」を参照し、型変換とは何か調べよう。

4　Rubyをとにかく使おう その2

これまでに、簡単なプログラムの入力と実行を繰り返しを行ったので、だいぶプログラミングに慣れてきたと思う。そこで、今回の演習からは、プログラムを単に実行するだけではなく、プログラム1行ごとの意味を理解することを目標とする。

どのようなプログラムを記述すると、下記に示したことを行うことが出来るのだろうか?
まずは、点線内に示されているプログラムを実行し、また教科書を使って調べながらすすめていこう。

プログラミング演習I 前半で行う内容の一部

■　単語のつづりをチェックするには？
　□　正しい場合には OK と表示し、間違っている場合は NG と表示するには？

■　同じ処理を何度も繰り返すには？
　□　こんにちはＲｕｂｙさん！　を　画面上に 100 行表示するには？
　□　0 から 100 までの数を画面上に表示し、最後に総和を表示するには？

■　変数って何？
　□　文字列や数値を記憶させておきたい場合はどうする？

■　5 人分の名前を記憶させておくには？
　□　その 5 人の名前を表示するには？

■　果物の種類と価格を記憶させておくには？
　□　りんご 130 円、オレンジ 80 円、バナナ 30 円　というデータを記憶させておくには？
　□　記憶させておいた果物の種類と価格を表示するには？

教科書

p.24(p.25)- 条件判断 if

演習 1-39 （ ex1-39.rb ）キーボードから入力した英単語が正しいがチェックするプログラム

```
-----------------------------------------------
print("本を英語で書くと？入力してください\n")
name = gets.chomp
if name == "book"
  print("正解！\n")
else
  print("不正解! 正解は、 book だよ。\n")
end
-----------------------------------------------
```

演習 1-40 （ ex1-40.rb ）

```
-----------------------------------------------
print("本を英語で書くと？入力してください\n")
name = gets.chomp
if name != "book"
  print("不正解! 正解は、 book だよ。\n")
else
  print("正解！\n")
end
-----------------------------------------------
```

教科書

p.26(p.27) 繰り返し while

演習 1-41 （ ex1-41.rb ） You > と表示されたら文章を入力し、Enter キーを押す。（bye と入力するまで入力と出力を繰り返すプログラム）

```
-----------------------------------------------
command = ""
print("Ruby > ", "こんにちは Ruby です", "\n")
while command != "bye"
  print("You  > ")
  command = gets.chomp
  print("Ruby > ", command, "\n")
end
-----------------------------------------------
```

教科書

p.19(p.21)- 変数

演習 1-42

```
-----------------------------------------------
name = "ベートーベン"
birth = 1770
print(name, "は、", birth, "年に生まれた\n")
-----------------------------------------------
```

演習 1-43

```
-----------------------------------------------
name = "シューベルト"
birth = 1797
print(name, "は、", birth, "年に生まれた\n")
-----------------------------------------------
```

演習 1-44

```
-----------------------------------------------
name_a = "ベートーベン"
birth_a = 1770
print(name_a, "は、", birth_a, "年に生まれた\n")
name_b = "シューベルト"
birth_b = 1797
print(name_b, "は、", birth_b, "年に生まれた\n")
print(name_a, "は、", name_b, "の", birth_b - birth_a, "年前に生まれた\n")
-----------------------------------------------
```

教科書

p.30(p.34)- 配列

演習 1-45　配列とは何か、また要素、インデックスとは何か調べ、図を使ってまとめよう。

演習 1-46

```
-----------------------------------------------
name = ["ベートーベン", "シューベルト"]
birth = [1770, 1797]

print(name[0], "\n")
print(name[1], "\n")

print(name[0], "は、", birth[0], "年に生まれた\n")
print(name[1], "は、", birth[1], "年に生まれた\n")
print(name[0], "は、", name[1], "の", birth[1] - birth[0], "年前に生まれた\n")
-----------------------------------------------
```

演習 1-47

```
-----------------------------------------------
a = [1, 3, 5, 7]
total = a[0] + a[1] + a[2] + a[3]
print(total, "\n")
-----------------------------------------------
```

演習 1-48　配列 a の要素を全て足し合わせるプログラムを完成させよう。

```
-----------------------------------------------
a = [1, 3, 5, 7, 9, 11]
total = ????
print(total, "\n")
-----------------------------------------------
```

教科書

p.34(p.38)- 配列の大きさ

演習 1-49　配列の大きさを表示しよう。

```
-----------------------------------------------
name = ["ベートーベン", "シューベルト", "ショパン"]
print(name.size, "\n")
-----------------------------------------------
```

演習 1-50　配列の要素を表示してみよう。

```
-----------------------------------------------
name = ["ベートーベン", "シューベルト", "ショパン"]
print(name[0], "\n")
print(name[1], "\n")
print(name[2], "\n")
-----------------------------------------------
```

演習 1-51

```
-----------------------------------------------
name = ["ベートーベン", "シューベルト", "ショパン"]
i = 0
while i < 3
  print(i, ":", name[i], "\n")
  i = i + 1
end
-----------------------------------------------
```

演習 1-52　演習 1-51 のプログラムとどこか異なるか。表示結果は同じか？

```
-----------------------------------------------
name = ["ベートーベン", "シューベルト", "ショパン"]
i = 0
while i < name.size
  print(i, ":", name[i], "\n")
  i = i + 1
end
-----------------------------------------------
```

教科書

p.36(p.40)- ハッシュ

演習 1-53　ハッシュとは何か調べよう。

演習 1-54

```
-----------------------------------------------
birth_table ={"ベートーベン" => 1770, "シューベルト" => 1797}

print("ベートーベンの誕生年：", birth_table["ベートーベン"], "\n")
print("シューベルトの誕生年：", birth_table["シューベルト"], "\n")
-----------------------------------------------
```

演習 1-55

```
-----------------------------------------------
font_table = {"normal" => "+0", "small" => "-1", "big" => "+1"}
print(font_table["small"], "\n")
print(font_table["normal"], "\n")
print(font_table["big"], "\n")
-----------------------------------------------
```

演習 1-56

```
-----------------------------------------------
book_table = {"たのしい Ruby" => 2730, "プログラミング Ruby" => 3990}
print(book_table["たのしい Ruby"], "円\n")
print(book_table["プログラミング Ruby"], "円\n")
-----------------------------------------------
```

演習 1-57

```
-----------------------------------------------
font_table = {"normal"=> "+0", "small" => "-1", "big" => "+1"}
font_table.each { |key, value|
   print(key, value, "\n")
}
-----------------------------------------------
```

演習 1-58

```
-----------------------------------------------
book_table = {"たのしい Ruby" => 2730, "プログラミング Ruby" => 3990}
book_table.each { |key, value|
   print(key, value, "\n")
}
-----------------------------------------------
```

演習 1-59　演習 1-58 のハッシュ　book_table のキーと値を書き換えて実行してみよう。

演習 1-60　ハッシュtango のキーには日本語を保存し、値には英語を保存してみよう。（例： キーとしてネコ、値として cat を保存する。）

```
-----------------------------------------------
tango = {"???"=>"???", "???"=>"???","???"=>"???","???"=>"???"}
tango.each { |key, value|
   print("日本語：",key,  "英語：",value, "\n")
}
-----------------------------------------------
```