2019年度「専修学校による地域産業中核的人材養成事業」

たのしいRuby 演習と解説（教員用）

札幌（北海道）をモデルとした地域創生のためのIT人材育成と企業連携推進事業

2019年度「専修学校による地域産業中核的人材養成事業」

たのしいRuby 演習と解説（教員用）

札幌（北海道）をモデルとした地域創生のためのIT人材育成と企業連携推進事業

目次

教材

『たのしいＲｕｂｙ』

第６版を使った

学習ポイントと演習課題

《教員用》

第 1 週　演習

学習ポイント

第 1 コマ： Ruby に触れる

（『たのしい Ruby』対応範囲: 第 1 章）
・開発環境の構築
・Ruby のコードを ruby コマンドと irb コマンドで実行する
・puts メソッドで”Hello, Ruby!”
・数値リテラルの基本（整数と小数）、四則演算、累乗

第 2 コマ：Ruby とあいさつをする

（『たのしい Ruby』対応範囲: 第 1 章）
・gets メソッドで入力された文字列を取得する・ローカル変数
・gets メソッド, print メソッド, 文字列リテラル内での式展開
・西暦を入力すると和暦を出力するプログラムを書く
・to_i メソッド、オブジェクトのクラスの説明（文字列クラスと数値クラス）
・p メソッド

第 3 コマ：条件分岐, スタックトレースの読み方

（『たのしい Ruby』対応範囲: 第 1 章, 第 10 章）
・if 文, unless 文, 比較演算子, true, false
・case 文
・予約語
・スタックトレースの読み方

演習課題

No	対応頁	演習
1-1	p24～33	次のように画面に表示するプログラムを作成しなさい。 開発環境の構築
1-2	p24～33	次のように画面に表示するプログラムを作成しなさい。ただし、プログラム中では puts メソッドは１回しか利用してはならない。 **第１コマ：Ruby に触れる** **開発環境の構築** **ruby コマンドと irb コマンド**
1-3	p24～33	次のように画面に表示するプログラムを作成しなさい。 **私は"Hello,Ruby!"とあいさつした**
1-4	p34～35	**「こんにちは、日本語」**と表示する Ruby プログラムを作成して実行し、文字化けやエラーとならずに実行できていることを確認しよう
1-5	p34～35	**「正確な、UTF-8 エンコードの日本語」**と表示する Ruby プログラムを作成して、ソースコードを UTF-8 エンコード以外のエンコーディングで保存した後に実行し、文字化けやエラーなどが起きてしまうことを確認しよう
1-6	p36～38	123 x 456 の答えを、演算子を使って計算して表示するプログラムを作成しなさい（ちなみに答えは 56088 になります）

No	対応頁	演習
1-7	p36～38	654321 ÷ 987 の答えを、演算子を使って計算して表示するプログラムを作成しなさい（ちなみに答えは 662 になります）
1-8	p26、 p36～38	123 x 456 の答えを irb コマンド上で演算子を使って計算して表示させなさい（ちなみに答えは 56088 になります）
1-9	p26、 p36～38	654321 ÷ 987 の答えを irb コマンド上で演算子を使って計算して表示させなさい（ちなみに答えは 662 になります）
2-1	p39～40	ある商品の名前（任意）を文字列として変数 product に代入し、print メソッドを使って**「この商品の名称は〇〇です」**（〇〇は変数 product の値）と表示させるプログラムを作成しなさい
2-2	p39～40	ある商品の価格（任意）を整数として変数 price に代入し、print メソッドを使って**「この商品の価格は〇〇円です」**（〇〇は変数 price の値）を表示させるプログラムを作成しなさい
2-3	p39～41	ある商品の名前（任意）を文字列として変数 product に、ある商品の価格（任意）を整数として変数 price に代入し、puts メソッドと#{～}を使って**「この商品の名称は〇〇で価格は△△円です」**（〇〇は変数 product、△△は変数 price の値）を表示させるプログラムを作成しなさい
2-4	p39～40	以下の内容を画面に表示するプログラムを作成しなさい。ただし表示には変数を利用して、「ルビィ」という文字列を直接繰り返し記述しないように工夫すること。また画面への表示には puts メソッドを利用すること **ルビィさんは 21 歳です** **ルビィさんの趣味は音楽鑑賞です** **ルビィさんは大学生です** **ルビィさんがこのプログラムを作りました**

No	対応頁	演習
2-5	p39〜41	以下の内容を画面に表示するプログラムを作成しなさい。ただし表示には変数を利用して、「商品 A」という文字列や、19800 という整数を直接繰り返し記述しないように工夫すること。また画面への表示には puts メソッドを利用すること 商品 A の定価は 19800 円です 商品 A の 50%オフで販売すると価格は 9900 円です 商品 A が定価で 3 個売れると売上は 59400 円です
2-6	p39〜40、 p372〜373 （第 17 章）	gets メソッドを使って氏名を入力し、「私の名前は○○です」（○○は gets メソッドを使って入力した氏名）と表示させるプログラムを作成しなさい。ただし、入力時の改行は chomp!メソッドを使って除去すること。
2-7	p39〜41、 p252〜253 （第 12 章） p372〜373 （第 17 章）	gets メソッドを使ってある商品の定価（任意）を入力し、その入力結果を整数に変換したうえで、その定価から 5 割引した割引後の価格を表示するプログラムを作成しなさい
2-8	p39〜41、 p252〜253 （第 12 章） p372〜373 （第 17 章）	gets メソッドを使って商品名と定価を入力し、その入力結果を利用して puts メソッドで以下の内容を表示するプログラムを作成しなさい。ただし表示例は商品名に「商品 A」、定価に「10000」と入力した想定である。 商品 A の定価は 10000 円です 商品 A の 50%オフで販売すると価格は 5000 円です 商品 A が定価で 3 個売れると売上は 15000 円です
3-1	p43〜44	irb コマンドを使って、評価結果が true（irb 上に「=> true」と表示される）となるような式（たとえば「1 == 1」のような比較演算子を使ったもの）を３種類考えて irb 上で結果を表示させなさい
3-2	p43〜44	irb コマンドを使って、評価結果が false（irb 上に「=> false」と表示される）となるような式（たとえば「1 == 2」のような比較演算子を使ったもの）を３種類考えて irb 上で結果を表示させなさい

No	対応頁	演習
3-3	p43～45	gets メソッドを使って整数を入力（入力した文字列を整数に変換）し、if 文を使って、その整数が 0 より大きければ「正しい値です」と表示するプログラムを作成しなさい
3-4	p43～45	gets メソッドを使って文字列を入力し、if 文を使って、その文字列が"Ruby"と等しければ「正しい名前です」と表示するプログラムを作成しなさい
3-5	p43～45	gets メソッドを使って整数を入力（入力した文字列を整数に変換）し、if 文を使って、その整数が 0 より大きければ「正しい値です」、そうでない場合は「不正な値です」と表示するプログラムを作成しなさい
3-6	p43～45	gets メソッドを使って文字列を入力し、if 文を使って、その文字列が"Ruby"と等しければ「正しい名前です」、そうでない場合は「不正な名前です」と表示するプログラムを作成しなさい
3-7	p43～45、 p91～92 (第 5 章)	gets メソッドを使ってある試験の点数を表す整数を入力（入力した文字列を整数に変換）し、if～elsif～else～end を使って、その点数が 80 点以上であれば「たいへんよくできました」、60 点以上であれば「よくできました」、それ以外であれば「ふつう」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、elsif について触れる機会がこの後のシラバスにないため、この時点で扱っておくとよいと思われます
3-8	p43～45、 p91～92 (第 5 章)	gets メソッドを使って文字列を入力し、if～elsif～else～end を使って、その文字列が"Ruby"と等しければ「正しい言語名です」、"Rails"と等しければ「正しいフレームワークです」、それ以外であれば「不正な名前です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、elsif について触れる機会がこの後のシラバスにないため、この時点で扱っておくとよいと思われます

No	対応頁	演習
3-9	p43～45、 p90～91 (第 5 章)	gets メソッドを使ってある試験の点数を表す整数を入力（入力した文字列を整数に変換）し、論理演算子と if 文を使って、その点数が０以上かつ１００以下であれば「正しい点数です」、そうでない場合は「不正な点数です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、elsif について触れる機会がこの後のシラバスにないため、この時点で扱っておくとよいと思われます
3-10	p43～45、 p90～91 (第 5 章)	gets メソッドを使って文字列を入力し、論理演算子と if 文を使って、その文字列が"Ruby"と等しいか、または"Rails"と等しければ「正しい名前です」、それ以外であれば「不正な名前です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、elsif について触れる機会がこの後のシラバスにないため、この時点で扱っておくとよいと思われます
3-11	p44、 p92～93 (第 5 章)	gets メソッドを使って整数を入力（入力した文字列を整数に変換）し、unless 文を使って、その整数が０より大きければ「正しい値です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、指導内容には列挙されているので、講義で触れておく必要があると思われます
3-12	p44、 p92～93 (第 5 章)	gets メソッドを使って文字列を入力し、unless 文を使って、その文字列が"Ruby"と等しければ「正しい名前です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、指導内容には列挙されているので、講義で触れておく必要があると思われます
3-13	p43～45、 p92～93 (第 5 章)	gets メソッドを使って整数を入力（入力した文字列を整数に変換）し、unless 文を使って、その整数が０より大きければ「正しい値です」、そうでない場合は「不正な値です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、指導内容には列挙されているので、講義で触れておく必要があると思われます

No	対応頁	演習
3-14	p43～45、 p92～93 (第 5 章)	gets メソッドを使って文字列を入力し、unless 文を使って、その文字列が"Ruby"と等しければ「正しい名前です」、そうでない場合は「不正な名前です」と表示するプログラムを作成しなさい ※シラバスでは第５章が範囲に入っていないが、指導内容には列挙されているので、講義で触れておく必要があると思われます

第２週　演習

学習ポイント

第 4 コマ：繰り返し, binding.irb

（『たのしい Ruby』対応範囲: 第 1 章）
・while 文, until 文
・binding.irb

第 5 コマ：メソッドを作ってみる

（『たのしい Ruby』対応範囲: 第 7 章）
・メソッドの定義方法
・可変長引数、キーワード引数

第 6 コマ：Array オブジェクトを使ってみる

（『たのしい Ruby』対応範囲: 第 2 章）
・配列リテラル
・Array#[], Array#[]=
・Array#each
・pp メソッド

演習課題

No	対応頁	演習
4-1	p46〜47	while 文を使って「たのしい Ruby」というメッセージを５回表示するプログラムを作成しなさい

No	対応頁	演習
4-2	p46～47	while 文を使って「たのしい Ruby　第〇版」というメッセージを第１版～第６版まで表示するプログラムを作成しなさい。実行結果は以下のようになります。 **たのしい Ruby　第１版** **たのしい Ruby　第２版** **たのしい Ruby　第３版** **たのしい Ruby　第４版** **たのしい Ruby　第５版** **たのしい Ruby　第６版**
4-3	p111～112 （第 6 章）	gets メソッドを使って文字列を入力し、入力した文字列が"Ruby"に等しくない間は繰り返し文字列を入力させ、"Ruby"に等しい場合は入力を繰り返さずに**「たのしかったです」**と表示するプログラムを作成しなさい。 ※シラバスでは第 6 章が範囲に入っていないが、until 文についてシラバスに挙げられているため、この時点で必要に応じて扱っておくとよいと思われます。

No	対応頁	演習
4-4	p111～112 （第 6 章）	until 文を使って「2020 年 1 月」から「2020 年 12 月」までを表示するプログラムを作成しなさい。実行結果は以下のようになります。 **2020 年 1 月** **2020 年 2 月** **2020 年 3 月** **2020 年 4 月** **2020 年 5 月** **2020 年 6 月** **2020 年 7 月** **2020 年 8 月** **2020 年 9 月** **2020 年 10 月** **2020 年 11 月** **2020 年 12 月** ※シラバスでは第 6 章が範囲に入っていないが、until 文についてシラバスに挙げられているため、この時点で必要に応じて扱っておくとよいと思われます。
4-5	該当箇所なし	問題 32 で作成したプログラム（**「たのしい Ruby」**を５回表示する）の while 文の最後（end の直前の行）に「binding.irb」メソッドの呼び出しを追加して実行してください。実行中に起動される irb 上で、ループに使用しているカウンタ変数の値を表示させたり、カウンタ変数の値を変更したりして、どのような動作になるかを確認しなさい
5-1	p122～123	times メソッドを使って、**「０時」**から**「２３時」**までを繰り返し表示するプログラムを作成しなさい。実行結果は以下のようになります。 **０時** **１時** **２時** **（略）** **２１時** **２２時** **２３時**

No	対応頁	演習
5-2	p122～123	times メソッドを使って、**「たのしい Ruby　第○版」**というメッセージを第１版～第６版まで表示するプログラムを作成しなさい。実行結果は以下のようになります。 **たのしい Ruby　第１版** **たのしい Ruby　第２版** **たのしい Ruby　第３版** **たのしい Ruby　第４版** **たのしい Ruby　第５版** **たのしい Ruby　第６版**
5-3	p121～123	「たのしい Ruby」p123 の説明を参考に、irb コマンドで足し算の演算を、「オブジェクト.メソッド名(引数)」の形式で記述しても正しく結果を得ることができることを受講者同士で確認しなさい。また、足し算以外の演算も同じ形式で記述、実行できるかどうかも確認しなさい。
5-4	p127	以下の順にプログラムを記述し、実行して結果を確認しよう。 ① 呼び出されると、puts メソッドで**「こんにちは」**と表示する処理を実行する greet メソッドを定義する。greet メソッドの引数はなしで良い。 ② puts メソッドで**「あいさつ」**と画面に表示する ③ greet メソッドを呼び出して実行する。引数は与えなくて良い。実行結果は以下の通りとなる。 **あいさつ** **こんにちは**

No	対応頁	演習
5-5	p127	以下の順にプログラムを記述し、実行して結果を確認しよう。 ① 呼び出されると、引数 message の内容を puts メソッドでそのまま表示する処理を実行する greet メソッドを定義する。greet メソッドには message という引数を定義する。 ② puts メソッドで「**朝のあいさつ**」と画面に表示する ③ greet メソッドを呼び出して実行する。引数には「**おはようございます**」という文字列を渡す。 ④ puts メソッドで「**昼のあいさつ**」と画面に表示する ⑤ greet メソッドを呼び出して実行する。引数には「**こんにちは**」という文字列を渡す。 ⑥ puts メソッドで「**夜のあいさつ**」と画面に表示する ⑦ greet メソッドを呼び出して実行する。引数には「**こんばんは**」という文字列を渡す。実行結果は以下の通りとなる。 **朝のあいさつ** **おはようございます** **昼のあいさつ** **こんにちは** **夜のあいさつ** **こんばんは**

No	対応頁	演習
5-6	p127	① 問題 5-5 で作成したプログラムに対し、greet メソッドに引数 count を追加して、以下のような処理が実行できるように書き換えよ（greet メソッドの処理内容） ・引数 message の内容を puts メソッドで、引数 count で与えられた回数繰り返し表示する →例えば、greet("Ruby",3)と呼び出した場合は、 **Ruby** **Ruby** **Ruby** と表示される ② 実行結果が以下の通りとなるよう、プログラムの記述を書き換えよ **朝のあいさつ** **おはようございます** **おはようございます** **おはようございます** **昼のあいさつ** **こんにちは** **こんにちは** **こんにちは** **こんにちは** **夜のあいさつ** **こんばんは** **こんばんは** **こんばんは** **こんばんは** **こんばんは**

No	対応頁	演習
5-7	p128～130	① 電子決済システムにおける決済額に対する還元ポイントを求めるメソッド cashless_point を定義しよう。メソッドの内容は以下の通りとする。 メソッド名：cashless_point 引数：price（決済額）、rate （ポイント還元率、単位は%） 戻り値：還元ポイント（小数点以下は切り捨て） 処理内容：return 文で、price*rate/100 を戻り値として返す ② gets メソッドで決済額とポイント還元率を入力し、cashless_point メソッドを実行してその戻り値を画面に表示しよう。実行結果は以下を参考にしよう。 **決済額を入力してください：19800** **ポイント還元率 (%)を入力してください：2019800** **円の還元ポイントは 3960 ポイントです**
5-8	p128～130	問題 5-7 で作成した cashless_point メソッドについて、「return」というキーワードを記述しなくても同じ戻り値が得られるようにするにはどうすればよいか。考えてプログラムを書き換え、同じ結果が得られることを確認しよう。
5-9	p128～130	問題 5-8 で変更した cashless_point メソッドについて、メソッドの最後に、「return」のみの文を１行追加したら、プログラムの実行結果がどう変わるかを確認しよう。また、その理由についても確認しよう。

No	対応頁	演習
5-10	p128	① 消費税込額を計算する shouhi_zei メソッドを定義しよう。 メソッドの内容は以下の通りとする。ただし、引数 rate にはデフォルト値として 10 を設定しておくこと。 メソッド名：shouhi_zei 引数：hontai（本体価格）、rate （消費税率、単位は%） 戻り値：税込額（小数点以下は切り捨て） 処理内容：return 文で、hontai*(100+rate)/100 を戻り値として返す ② gets メソッドで本体価格と消費税率を入力し、shouhi_zei メソッドを実行してその戻り値を画面に表示する。 実行結果は以下を参考にしよう。ただし、消費税率に 0 以下の値が入力された場合は、引数 rate を shouhi_zei メソッドに与えずに実行して結果を表示しよう 実行例１ **本体価格を入力してください：3980** **消費税率 (%)を入力してください：8** **本体価格 3980 円の税込額は 4298 円です** 実行例２ **本体価格を入力してください：3980** **消費税率 (%)を入力してください：0** **本体価格 3980 円の税込額は 4378 円です**
5-11	p133	問題 5-6 で作成した greet メソッドの引数をキーワード引数に変更しよう。 その際、デフォルト値をキーワード引数 message は”こんにちは”、キーワード引数 count は 3 に設定すること。 その上で、その後のメソッド呼び出しをキーワード引数に対応したものに変更しよう。 デフォルト値が利用できる場合は、そのキーワード引数の呼び出しは割愛すること。 また、キーワード引数は引数の記述順序が任意であるため、count、message の順にキーワード引数を指定しても問題なく実行できることを確認しよう。

No	対応頁	演習
6-1	p50〜52	irb 上で、以下の手順を実行しよう。 ① 変数 langs に"Ruby"、"Python"、"PHP"、"Java"、"JavaScript"の５つの文字列を格納した配列を代入する ② langs の中から、"Ruby"を取り出してその内容を表示させる ③ langs の中から、"PHP"を取り出してその内容を表示させる ④ langs に格納されている"JavaScript"を"ECMAScript"に変更する ⑤ langs の末尾(インデックスが 6 の場所)に"Kotlin"を追加する ⑥ langs 全体の内容を表示させる（[]やインデックス番号を指定せずに表示） ⑦ langs の配列の大きさを size メソッドで表示させる
6-2	p54〜55	配列を利用して以下のようなプログラムを作成しよう。 ①変数 langs に"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ② each メソッドを利用して、langs のすべての要素を puts メソッドで表示する表示結果は以下の通りとなる **Ruby** **Python** **PHP** **Java** **ECMAScript**
6-3	p54〜55	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 score に整数「39」「98」「75」「87」「72」「46」が格納された配列を代入する ② each メソッドを利用して、score のすべての要素を puts メソッドで「〇点」のように表示する表示結果は以下の通りとなる **39 点** **98 点** **75 点** **87 点** **72 点** **46 点**

No	対応頁	演習
6-4	p54～55	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 score に整数「39」「98」「75」「87」「72」「46」が格納された配列を代入する ② 点数の合計を格納するための変数 sum に 0 を代入する ③ each メソッドを利用して、score のそれぞれの要素を sum に加算する（複合代入演算子+=を利用すると簡単） ④ 点数の合計値を「合計：〇点」のように表示する ⑤ 点数の平均値を「平均：〇点」のように表示する表示結果は以下の通りとなる **合計：417 点** **平均：69.5 点**
6-5	p54～55	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 numbers に空の配列を代入する ② gets メソッドを利用して、整数を入力する ③ 入力した整数が 0 以上でない場合は、配列の末尾に入力した整数を代入（配列の末尾に追加）して、②の処理からを繰り返す ④ 入力した整数が 0 未満の場合は、②には戻らず⑤の処理に進む ⑤ 配列 numbers 内の値の平均値を求めて「平均値は〇です」と表示する

第３週　演習

学習ポイント

第 7 コマ：Hash オブジェクトを使ってみる

（『たのしい Ruby』対応範囲: 第 2 章）
・ハッシュリテラル
・シンボル
・Hash#[], Hash#[]=
・Hash#each, Hash#keys
・nil オブジェクト

第 8 コマ：String メソッドを使ってみる

（『たのしい Ruby』対応範囲: 第 3 章）
・String#slice, String#[]
・String クラスを使って入力をパースする

第 9 コマ：正規表現による文字列処理

（『たのしい Ruby』対応範囲: 第 16 章）
・正規表現リテラル
・Regexp.new
・String#slice(regexp), String#match, =~, String#scan, String#sub, String#gsub

演習課題

No	対応頁	演習
7-1	p56～58	irb 上で、以下の手順を実行しよう。 ① 変数 employee に次の内容が含まれるハッシュを代入する ・キー「:no」、オブジェクト「1001」（社員番号を示す） ・キー「:name」、オブジェクト「"ルビィ太郎"」（社員名を示す） ・キー「:salary」、オブジェクト「350000」（給与額を示す） ・キー「:dept」、オブジェクト「"営業部"」（部署名を示す） ② employee の中から、社員番号の値「1001」を取り出してその内容を表示させる ③ Employee の中から、社員名の値「ルビィ太郎」を取り出してその内容を表示させる ④ employee の中から、給与額の値「350000」を取り出してその内容を表示させる ⑤ employee の中から、部署名の値「営業部」を取り出してその内容を表示させる ⑥ employee の中の給与額の値を 350000 から 480000 に変更する ⑦ employee のハッシュ全体の内容を表示させる（[]やキーを指定せずに表示）
7-2	p58～59	以下の処理を行うプログラムを作成しなさい。 ① 変数 employee に次の内容が含まれるハッシュを代入する ・キー「:no」、オブジェクト「1001」（社員番号を示す） ・キー「:name」、オブジェクト「″ルビィ太郎″」（社員名を示す） ・キー「:salary」、オブジェクト「350000」（給与額を示す） ・キー「:dept」、オブジェクト「″営業部″」（部署名を示す） ② each メソッドを利用して、employee の内容をもとに以下のように表示する 表示結果 **no は 1001 です** **name はルビィ太郎です** **salary は 350000 です** **dept は営業部です**

No	対応頁	演習
7-3	p58～59	以下の処理を行うプログラムを作成しなさい。 ① 変数 employee に次の内容が含まれるハッシュを代入する ・キー「:no」、オブジェクト「1001」（社員番号を示す） ・キー「:name」、オブジェクト「"ルビィ太郎"」（社員名を示す） ・キー「:salary」、オブジェクト「350000」（給与額を示す） ・キー「:dept」、オブジェクト「"営業部"」（部署名を示す） ② gets メソッドを利用して、キーの文字列を入力し変数 key に代入する ③ 変数 key の値を利用して、ハッシュ employee からオブジェクトの値を取り出し、「キー【〇〇】のオブジェクトは【△△】です」（〇〇は変数 key の値、△△は取り出したオブジェクトの値）と表示する実行結果例は以下のようになる **キーを入力してください：salary** **キー【salary】のオブジェクトは【350000】です**

No	対応頁	演習
7-4	p58～59	以下の処理を行うプログラムを作成しなさい。 ① 変数 hash に空のハッシュを代入する ② gets メソッドを利用して、キーの文字列を入力し変数 key に代入する ③ gets メソッドを利用して、オブジェクトに該当する文字列を入力し変数 obj に代入する ④ 変数 hash にキーの値を key、オブジェクトを obj として格納する ⑤ p メソッドを利用して変数 hash の内容を表示する ⑥ 「さらに格納しますか？(y/n)」と表示する ⑦ gets メソッドを利用して、文字列を入力して変数 judge に代入する ⑧ 変数 judge の値が"y"もしくは"Y"に等しい場合は、②に戻って繰り返す ⑨ 上記以外の値が入力された場合はプログラムを終了する実行結果例は以下のようになる **キーを入力してください：name** **オブジェクトを入力してください：コーラ{ "name" => "コーラ" }** **さらに格納しますか？ (y/n) y** **キーを入力してください：price** **オブジェクトを入力してください：150{ "name" => "コーラ" , "price" => 150 }** **さらに格納しますか？ (y/n) n**

No	対応頁	演習
7-5	p329～330 (第 15 章)	以下の処理を行うプログラムを作成しなさい。 ① 変数 employee に次の内容が含まれるハッシュを代入する ・キー「:no」、オブジェクト「1001」（社員番号を示す） ・キー「:name」、オブジェクト「"ルビィ太郎"」（社員名を示す） ・キー「:salary」、オブジェクト「350000」（給与額を示す） ・キー「:dept」、オブジェクト「"営業部"」（部署名を示す） ② 配列の keys メソッドを利用して、キーの一覧を配列として受け取り、変数 keys_ary に代入する ③ each メソッドなどを使って、keys_ary の内容をすべて表示する 表示結果 **No** **Name** **Salaryd** **Ept** ※シラバスには第 15 章が含まれていないが、カリキュラムに列挙されているため、触れておく必要があるでしょう
7-6	p62	問題 7-3 で作成したプログラムに以下の処理を付け加えなさい ④ もし、上記③で得られたオブジェクトの値が nil のときは、代わりに「キー【〇〇】は存在しません」と表示する
8-1	p63～65	次の内容を実行するプログラムを作成しなさい。 ① 変数 profile に空のハッシュ {} を代入する ② profile にキーが「:name」、オブジェクトがコマンドライン引数 ARGV[0]、のペアを格納する ③ profile にキーが「:age」、オブジェクトがコマンドライン引数 ARGV[1]、のペアを格納する ④ profile にキーが「:job」、オブジェクトがコマンドライン引数 ARGV[2]、のペアを格納する ⑤ profile の内容を p メソッドで画面に表示する実行結果は以下のようになる（ファイル名を 8-1.rb とした場合） **（コマンドライン）** **ruby 8-1.rb ルビィ太郎 30 プログラマ** **（実行結果）** **{ :name => "ルビィ太郎" , :age => 30 , :job => "プログラマ" }**

No	対応頁	演習
8-2	p66～69	コマンドライン引数に指定されたファイル名のファイルを１行ずつ読み込んで画面に表示するプログラムを作成しよう。ただし、各行を表示する際には、「行番号:」（行番号は先頭行を１とする）を先頭につけて表示させること。
8-3	p69～70	コマンドライン引数に指定されたファイル名のファイルを１行ずつ読み込んで画面に表示するプログラムを作成しよう。ただし、表示するのは、先頭が"#"で始まる行のみとなるようにすること。
8-4	p71～73	ファイル「8-4_greet.rb」に以下の内容を記述しよう ①呼び出されると、puts メソッドで「こんにちは」と表示する処理を実行する greet メソッドを定義する。greet メソッドの引数はなしで良い。 上記と別のファイルを作成し、以下の内容を記述しよう ① require/require_relative メソッドのいずれかを使って、ファイル 8-4_greet.rb の内容を読み込む ② puts メソッドで「あいさつ」と画面に表示する ③ greet メソッドを呼び出して実行する。引数は与えなくて良い。実行結果は以下の通りとなる。 **あいさつ** **こんにちは**
9-1	p341～344	文字列のどこかに"y"が含まれている、という正規表現リテラル「/y/」を使って、"Ruby"、"Python"、”Java” がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう。 ※irb では、正規表現リテラルを変数に入れておくと、書き直しの手間が省けて便利（履歴を表示してもよいのですが）ということも触れてあげると良いでしょう また、クラスのレベルに合わせて、リテラルの表現（「/y/」の部分）は問題に示しておいてもよいし、受講者自身に考えさせても良いと思います
9-2	p344～345	文字列の行頭が"R"で始まっている、という正規表現リテラル「/^R/」を使って、"Rails"、”SUPER”、"URL" がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう。

No	対応頁	演習
9-3	p344～345	文字列の行末が”y" で終わる、という正規表現リテラル「/y$/」を使って、"Ruby"、”Spy”、"Python" がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう
9-4	p346～347	文字列の行頭がアルファベットの大文字で、続く文字がアルファベットの小文字である、という正規表現リテラル「/^[A-Z][a-z]/」を使って、"Ruby"、”COBOL”、"ECMAScript" がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう
9-5	p347～348	文字列の行頭が"R"で始まり、残りが任意の３文字で終わる、という正規表現リテラル「/^R...$/」を使って、”Ruby” 、”Rails”、”Red”がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう
9-6	p348～352	携帯電話番号を示す正規表現（数字３桁、ハイフン、数字４桁、ハイフン、数字４桁）を考えて、"090-1234-5678"、"03-9876-5432"、"0120-123-456"がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう
9-7	p348～352	文字列の行頭がアルファベットの大文字で始まり、残りが任意の文字数（０文字を含む）の英数字である、という正規表現を考えて、"WIndows10"、"macOS"、”VB”、"C"がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう
9-8	p348～352	文字列の行頭がアルファベットの大文字で始まり、残りが任意の文字数（１文字以上）の英数字である、という正規表現を考えて、"C++"、”C”、"VB"がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう

第４週　演習

学習ポイント

第 10 コマ：正規表現を使いこなす

（『たのしい Ruby』対応範囲: 第 16 章）
・キャプチャ
・メタ文字（行頭、行末、繰り返し、最短マッチなど）

第 11 コマ：プログラムを作ってみる

・ここまでに習得した知識をもとに、新しいプログラムを作る

第 12 コマ：Time クラスと Date クラス

（『たのしい Ruby』対応範囲: 第 20 章,第 11 章 4 節）
・オブジェクトの作り方: Time.now, Date.today, Time.new, Date.new
・演算子
・値を取り出すメソッド
・値を設定するメソッド
・Ruby リファレンスマニュアルの使い方

演習課題

No	対応頁	演習
10-1	p355～356	問題 9-4 で利用した正規表現「/^[A-Z][a-z]/」に、オプション「i」を付与した場合、"Ruby"、”COBOL”、"ECMAScript" がそれぞれ正規表現にマッチするかどうかを、irb を使って確認しよう

No	対応頁	演習
10-2	p356～358	問題 9-6 で利用した正規表現（携帯電話番号を示す正規表現）にキャプチャの( )を追加して、"090-1234-5678"をマッチングさせた後、先頭の３桁、真ん中の４桁、末尾の４桁をキャプチャ（$1、$2、$3）によって、それぞれ表示する手順を、irb を使って確認しよう
10-3	P356～358	問題 9-1 で利用した正規表現（「/y/」）で、"Python"をマッチングさせた後、マッチした部分より前の文字列($`)、マッチした部分そのものの文字列($&)、マッチした部分より後ろの文字列($')を表示する手順を、irb を使って確認しよう
10-4	p358～359	次の手順を実行するプログラムを作成しなさい。 ① 変数 word に"Password"という文字列を代入する ② sub メソッドを使って、word に含まれる"s"を"$"に置き換えた結果を、puts メソッドで表示する 上記が動作すると、表示結果は「Pa$sword」となるが、②の sub メソッドを gsub メソッドに変更すると、結果はどうなるかを確認しよう
10-5	p358～359	次の内容を実行するプログラムを作成しなさい。 ① 変数 word に"Password"という文字列を代入する ② gsub!メソッドを使って、word に含まれる"a"を"@"に置き換える ③ gsub!メソッドを使って、word に含まれる"s"を"$"に置き換える ④ gsub!メソッドを使って、word に含まれる"o（小文字のオー）"を"0（ゼロ）"に置き換える ⑤ word の内容を puts メソッドで画面に表示する 上記が動作すると、表示結果は「P@$$w0rd」となるが、②～④の gsub!メソッドを gsub メソッド（!をつけない）に変更すると、結果はどうなるかを確認しよう

No	対応頁	演習
10-6	p359～360	次の内容を実行するプログラムを作成しなさい。 ① 変数 mobile に"090-1234-5678"という文字列を代入する ② scan メソッドを使って、変数 mobile に含まれる先頭の３桁、真ん中の４桁、末尾の４桁を、それぞれ表示する 実行結果は以下のようになる **先頭の３桁：090** **真ん中の４桁：1234** **末尾の４桁：5678**
10-7	p361～363	gets メソッドを使って、電話番号（ハイフンを含む）を入力させ、正規表現で正しい電話番号かを判定し、正しい電話番号のときは**「入力を受け付けました」**、不正な電話番号のときは、**「再度入力してください」**と表示して入力をやり直すようなプログラムを作成しなさい。なお、この問題においては、電話番号は携帯電話番号・フリーダイヤル・固定電話のどの番号も正しい電話番号と考えよ
12-1	p427～429	次の内容を実行するプログラムを作成しなさい。 ① Time.now で生成した Time オブジェクトを変数 t に代入する ② Time クラスの year メソッド、month メソッド、day メソッドを使って**「2019 年 9 月 15 日」**のように表示しなさい ③ Time クラスの yday メソッドを使って、**「今日は 2019 年の 365 日中、〇日目です」**のように表示しなさい
12-2	p429	次の内容を実行するプログラムを作成しなさい。 ① gets メソッドを使って、年、月、日の値を入力し、適当な変数に格納します ② Time クラスの mktime メソッドを使って、①で入力した年月日を表す Time オブジェクトを作成する ③ 上記②で作成した日付を、year メソッド、month メソッド、day メソッドを使って**「2019 年 9 月 15 日」**のように表示しなさい

No	対応頁	演習
12-3	p430	次の内容を実行するプログラムを作成しなさい。 ① gets メソッドを使って、年、月、日の値を入力し、適当な変数に格納します ② Time クラスの mktime メソッドを使って、①で入力した年月日を表す Time オブジェクトを作成する ③ 上記②で作成した日付に 90 日(に等しい秒数)を足して、別の変数に代入する ④ Time クラスの year メソッド、month メソッド、day メソッドを使って**「2019 年 9 月 15 日の 90 日後は 2019 年 12 月 14 日です」**のように表示しなさい
12-4	p430～431	次の内容を実行するプログラムを作成しなさい ① Time.now メソッドを使って、現在日時を示す Time オブジェクトを作成する ② Time クラスの strftime メソッドを使って、"2019/9/15 11:30:45"のような文字列を取得し、**「本日は 2019/9/15 11:30:45 です」**のように画面に表示する
12-5	p435～437	次の内容を実行するプログラムを作成しなさい ① Date.today で生成した Date オブジェクトを変数 d に代入する ② Date クラスの strftime メソッドを使って、変数 d の日付を**「2019 年 9 月 15 日」**のように表示しなさい
12-7	p435～437	次の内容を実行するプログラムを作成しなさい ① gets メソッドを使って、年、月の値を入力し、適当な変数に格納します ② Date クラスの new メソッドを使って、①で入力した年月の末日を表す Date オブジェクトを作成する ③ Date クラスの year メソッド、month メソッド、day メソッドを使って、**「2016 年 2 月の末日は 29 日です」**のように表示しなさい

No	対応頁	演習
12-8	p436～437	次の内容を実行するプログラムを作成しなさい ① gets メソッドを使って、年、月、日の値を入力し、適当な変数に格納します ② Date クラスの new メソッドを使って、①で入力した年月日を表す Date オブジェクトを作成する ③ 上記②で作成した日付に 180 日を足して、別の変数に代入する ④ Date クラスの strftime メソッドを使って**「2019 年 9 月 15 日の 180 日後は 2020 年 3 月 13 日です」**のように表示しなさい

第５週　演習

学習ポイント

第 13 コマ：さまざまな条件分岐と繰り返しの構文

（『たのしい Ruby』対応範囲: 第 5 章,第 6 章）
・case と===メソッド
・each, for, while, until, loop とその使い分け
・break, next, redo

第 14 コマ：Array クラスを使いこなす

（『たのしい Ruby』対応範囲: 第 13 章）
・配列の作り方（Array.new, 配列リテラル, to_a, String#split など）
・要素の参照・追加: Array#[], []=, at, slice, push (<<), pop, shift, unshift
・複雑な要素の操作: +, -, &, |, concat, compact, uniq
・構造の変更: flatten, reverse, sort, sort_by
・破壊的メソッドと freeze

第 15 コマ：Enumerable, メソッドチェーン

（『たのしい Ruby』対応範囲: 第 13 章のコラム）
・Enumerable のメソッド、メソッドチェーン、tap メソッド
・Integer#step, Integer#upto などの Enumerator のインスタンスを返すメソッド
・Enumerator::Lazy

演習課題

No	対応頁	演習
13-1	p93～97	次の内容を実行するプログラムを作成しなさい ① gets メソッドを使って試験の点数を入力し適当な変数に格納する ② 範囲オブジェクト(たのしい Ruby p108 を参照)と case 構文を利用して、点数が 1～49 のときは**「不可」**、50～69 のときは**「可」**、70～89 のときは**「良」**、90～100 のときは「優」、それ以外のときは**「不正な点数です」**と表示する
13-2	p93～97	「たのしい Ruby」 p94、p95、p96 に掲載されているプログラムを、if～elsif～else 構文を使って同じ動作をするプログラムに書き換えなさい
13-3	p105～108	for 文を使って、**"たのしい Ruby"**と 10 回表示しなさい
13-4	p105～108	for 文を使って、12 時から 23 時までを表示するプログラムを作成しなさい。 表示結果は以下の通りとなる。 **12 時** **13 時** **14 時** **（略）** **22 時** **23 時**
13-5	p105～108	for 文を使って、1 から 100 までの整数をすべて足した合計（総和）を求めて表示しなさい（5050 と表示されれば正解です)。

No	対応頁	演習
13-6	p105～108	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs に"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ② for 文を使って、langs のすべての要素を puts メソッドで表示する 表示結果は以下の通りとなる **Ruby** **Python** **PHP** **Java** **ECMAScript**
13-7	p114	loop メソッドを使って以下のようなプログラムを作成しよう。 ① 変数 count に 0 を代入 ② loop メソッドのブロック内で以下を実行 ・変数 count の値を 1 加算する ・変数 count の値を print メソッドで表示する(数値の後に空白も表示すること) なお、このプログラムは実行するとそのままでは停止しないため、ある程度動作させたら、CTRL+C を入力して強制終了させること
13-8	p114～118	loop メソッドを使って以下のようなプログラムを作成しよう ①変数 count に 0 を代入 ②loop メソッドのブロック内で以下を実行 ・変数 count の値を 1 加算する ・変数 count の値を print メソッドで表示する(数値の後に空白も表示すること) ・変数 count が 99 に等しいときは break でループを終了する 表示結果は以下のようになる **1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99**

No	対応頁	演習
13-9	p114～118	loop メソッドを使って以下のようなプログラムを作成しよう ①変数 count に 0 を代入 ②loop メソッドのブロック内で以下を実行 ・変数 count の値を 1 加算する ・変数 count を 2 で割った余りが 0 のときは next を実行 ・変数 count の値を print メソッドで表示する(数値の後に空白も表示すること) ・変数 count が 99 に等しいときは break でループを終了する 表示結果は以下のようになる **1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99**
14-1	p265	次の内容を実行するプログラムを作成しなさい ① すべての要素に nil が代入されている要素数 10 の配列を生成し、変数 ary に代入する ② gets メソッドを使って 0 から 9 までの整数（配列のインデックス番号）を入力し変数 idx に代入する ③ 上記②で入力した整数が 0 から 9 の範囲外の場合は、②に戻って入力をやり直す ④ gets メソッドを使って配列に格納したい文字列を入力し変数 value に代入する ⑤ 配列 ary[idx] に value を代入 ⑥ 配列 ary の内容を p メソッドで表示 ⑦ 「続けますか？(y/n)」と print メソッドで表示 ⑧ y が入力されたら②に戻って繰り返す、それ以外のときはプログラムを終了する 実行例は以下の通り **インデックスを入力(0-9)：0** **格納したい値を入力：先頭** **["先頭",nil,nil,nil,nil,nil,nil,nil,nil,nil]** **続けますか？(y/n) y** **インデックスを入力(0-9)：5** **格納したい値を入力：６ばんめ** **["先頭",nil,nil,nil,nil,” ６ばんめ",nil,nil,nil,nil]** **続けますか？(y/n) n** **プログラムを終了します**

No	対応頁	演習
14-2	p265	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs に、%w を使って"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ② langs のすべての要素を p メソッドで表示する ③ 変数 frameworks に、%i を使って:Rails、:Django、:Laravel、:Spring、:React の５つのシンボルを格納した配列を代入する ④ framworks のすべての要素を p メソッドで表示する 表示結果は以下の通りとなる **["Ruby", "Python", "PHP", "Java", "ECMAScript"]** **[:Rails, :Django, :Laravel, :Spring, :React]**
14-3	p266	ハッシュと配列を利用して以下のようなプログラムを作成しよう。 ① 変数 fw_hash に、言語とフレームワークの組み合わせを示すハッシュ{Ruby: "Rails", Python: "Django", PHP: "Laravel"}を代入する ② fw_hash を p メソッドで表示する ③ fw_hash を to_a メソッドで配列に変換した結果を p メソッドで表示する 表示結果は以下の通りとなる **{"Ruby"=>"Rails", "Python"=>"Django", "PHP"=>"Laravel"}** **[["Ruby", "Rails"], ["Python", "Django"], ["PHP", "Laravel"]]**
14-4	p266	配列を利用して以下のようなプログラムを作成しよう ① 変数 fw_str に、フレームワークの名前を示す文字列"Rails/Django/Laravel/Spring/React"を代入する ② fw_str を split メソッドで配列に変換した結果を p メソッドで表示する。ただし、区切り文字は'/'とする 表示結果は以下の通りとなる **["Rails", "Django", "Laravel", "Spring", "React"]**

No	対応頁	演習
14-5	p267	問題 14-4 のプログラムの②～⑦を、[ ]を使わずに、at メソッドや slice メソッドを使った記述に書き換えたプログラムを作成しなさい。 表示結果は以下の通り（問題 14-4 と同じ）となる **["Python", "PHP", "Java"]** **["PHP", "Java", "ECMAScript"]** **["Ruby", "Python", "PHP"]** **"ECMAScript"** **"PHP"** **nil**
14-6	p270～271	配列を利用して以下のようなプログラムを作成しよう ① 変数 langs に、"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ② langs の、"Python","PHP","Java"を[n..m]の形式で"COBOL","FORTRAN","Pascal"に置き換える ③ langs の内容を p メソッドで表示する ④ langs の、"Pascal","ECMAScript"を[n,len]の形式で"Kotlin","TypeScript"に置き換える ⑤ langs の内容を p メソッドで表示する 表示結果は以下の通りとなる **["Ruby", "COBOL", "FORTRAN", "Pascal", "ECMAScript"]** **["Ruby", "COBOL", "FORTRAN", "Kotlin", "TypeScript"]**
14-7	p271	配列を利用して以下のようなプログラムを作成しよう ① 変数 langs に、"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ② langs の、"PHP"と"Java"の間に、"Perl","C#","Swift"を挿入する ③ langs の内容を p メソッドで表示する 表示結果は以下の通りとなる **["Ruby", "Python", "PHP", "Perl", "C#", "Swift", "Java", "ECMAScript"]**

No	対応頁	演習
14-8	p271	配列を利用して以下のようなプログラムを作成しよう ①変数 langs に、"Ruby"、"Python"、"PHP"、"Java"、"ECMAScript"の５つの文字列を格納した配列を代入する ②langs から values_at メソッドを使って、"Ruby","PHP","ECMAScript"だけを取り出してその結果を p メソッドで表示する 表示結果は以下の通りとなる **["Ruby", "PHP", "ECMAScript"]**
14-9	p272～274	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs1 に、"Ruby"、"Python"、"PHP"の３つの文字列を格納した配列を代入する ② 変数 langs2 に、"Java"、"Ruby"、"ECMAScript"の３つの文字列を格納した配列を代入する ③ 「共通集合」という文字列を puts メソッドで表示する ④ langs1 と langs2 の共通集合を p メソッドで表示する ⑤ 「和集合」という文字列を puts メソッドで表示する ⑥ langs1 と langs2 の和集合を p メソッドで表示する ⑦ 「差集合」という文字列を puts メソッドで表示する ⑧ langs1 と langs2 の差集合を p メソッドで表示する ⑨ 「結合」という文字列を puts メソッドで表示する ⑩ langs1 と langs2 を「+」で結合した配列を p メソッドで表示する 表示結果は以下の通りとなる **共通集合** **["Ruby"]** **和集合** **["Ruby", "Python", "PHP", "Java", "ECMAScript"]** **差集合** **["Python", "PHP"]** **結合** **["Ruby", "Python", "PHP", "Java", "Ruby", "ECMAScript"]**

No	対応頁	演習
14-10	p275～p278	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs に、"Ruby"、Python"、"PHP"を格納した配列を代入する ② langs の内容を p メソッドで表示する ③ langs の先頭に"Java" を加える ④ langs の内容を p メソッドで表示する ⑤ langs の末尾に"ECMAScript" を加える ⑥ langs の内容を p メソッドで表示する ⑦ langs の先頭の要素を取り出して(参照ではない)p メソッドで表示する ⑧ langs の末尾の要素を取り出して(参照ではない)p メソッドで表示する ⑨ langs の内容を p メソッドで表示する ⑩ langs の先頭の要素を参照して(取り出しではない)p メソッドで表示する ⑪ langs の末尾の要素を参照して(取り出しではない)p メソッドで表示する 表示結果は以下の通りとなる **["Ruby", "Python", "PHP"]** **["Java", "Ruby", "Python", "PHP"]** **["Java", "Ruby", "Python", "PHP", "ECMAScript"]** **"Java"** **"ECMAScript"** **["Ruby", "Python", "PHP"]** **"Ruby"** **"PHP"**

No	対応頁	演習
14-11	p278～279	配列を利用して以下のようなプログラムを作成しよう ① 変数 langs に、"Ruby"、Python"、"PHP"を格納した配列を代入する ② langs の内容を p メソッドで表示する ③ langs の末尾に << メソッドを使って"Java" を加える ④ langs の内容を p メソッドで表示する ⑤ langs と配列["ECMAScript","VisualBasic"] を「+」で結合した結果を p メソッドで表示する ⑥ langs の内容を p メソッドで表示する ⑦ langs と配列["ECMAScript","VisualBasic"] を concat メソッドで連結する ⑧ langs の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", "Python", "PHP"]** **["Ruby", "Python", "PHP", "Java"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic"]** **["Ruby", "Python", "PHP", "Java"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic"]**
14-12	p279～280	問題 14-11 のプログラムの②と③の間に、「langs.freeze」という文を追加して、再度実行すると、エラーとなって配列が変更できなくなっていることを確認しよう。 表示結果は以下のようになる **["Ruby", "Python", "PHP"]** **Traceback (most recent call last):** **freeze.rb:4:in `<main>': can't modify frozen Array (FrozenError)**

No	対応頁	演習
14-13	p281	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs に、配列 ["Ruby", nil,"Python", "PHP", nil,"Java", "ECMAScript",nil,nil, "VisualBasic","COBOL","Kotlin","HTML","CSS"]を代入する ② langs の内容を p メソッドで表示する ③ compact メソッドを使って langs から nil を取り除いた配列を p メソッドで表示する ④ compact!メソッドを使って langs から nil を取り除く ⑤ langs の内容を p メソッドで表示する ⑥ delete メソッドを使って、langs から要素"HTML"を取り除く ⑦ langs の内容を p メソッドで表示する ⑧ delete_at メソッドを使って、langs の一番末尾の要素を取り除く ⑨ langs の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", nil, "Python", "PHP", nil, "Java", "ECMAScript", nil, nil, "VisualBasic", "COBOL", "Kotlin", "HTML", "CSS"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic", "COBOL", "Kotlin", "HTML", "CSS"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic", "COBOL", "Kotlin", "HTML", "CSS"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic", "COBOL", "Kotlin", "CSS"]** **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic", "COBOL", "Kotlin"]**

No	対応頁	演習
14-14	p282	配列を利用して以下のようなプログラムを作成しよう。 ① 変数 langs に、配列 ["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic","COBOL","Kotlin","HTML","CSS"]を代入する ② langs の内容を p メソッドで表示する ③ delete_if メソッドを使って langs から文字がすべて大文字の要素を取り除く ④ langs の内容を p メソッドで表示する ⑤ reject!メソッドを使って、langs から文字"a"を含む要素を取り除く ⑥ langs の内容を p メソッドで表示する ⑦ slice!メソッドを使って、langs の一番末尾の要素（インデックスが-1）を取り除き、その結果（slice!によって取り除かれた要素）を p メソッドで表示する ⑧ langs の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", "Python", "PHP", "Java", "ECMAScript", "VisualBasic", "COBOL", "Kotlin", "HTML", "CSS"]** **["Ruby", "Python", "Java", "ECMAScript", "VisualBasic", "Kotlin"]** **["Ruby", "Python", "ECMAScript", "Kotlin"]** **"Kotlin"** **["Ruby", "Python", "ECMAScript"]**
14-15	p282	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", "Python", "PHP","Python", "Ruby", "PHP","PHP"]を変数 langs に代入する ② langs の内容を p メソッドで表示する ③ uniq!メソッドを使って langs から重複した要素をすべて取り除く ④ langs の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", "Python", "PHP", "Python", "Ruby", "PHP", "PHP"]** **["Ruby", "Python", "PHP"]**

No	対応頁	演習
14-16	p283	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", "Python", "PHP","Java", "ECMAScript"]を変数 langs に代入する ② map!メソッドを使って、langs の各要素をすべて大文字に変換する（文字列.upcase メソッドで大文字に変換できる） ③ langs の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", "Python", "PHP", "Java", "ECMAScript"]** **["RUBY", "PYTHON", "PHP", "JAVA", "ECMASCRIPT"]**
14-17	p284	配列を利用して以下のようなプログラムを作成しよう。 ① Array.new メソッドを使って、大きさが 5 で内容がすべて nil の配列を生成して変数 array に代入する ② array の内容を p メソッドで表示する ③ fill メソッドを使って、配列の要素すべてを"Ruby"に置き換える ④ array の内容を p メソッドで表示する ⑤ fill メソッドを使って、後半 3 つの配列の要素を"たのしい"に置き換える ⑥ array の内容を p メソッドで表示する 表示結果は以下のようになる **["Ruby", "Ruby", "Ruby", "Ruby", "Ruby"]** **["Ruby", "Ruby", "たのしい", "たのしい", "たのしい"]**

No	対応頁	演習
14-18	p284	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", ["Python", ["PHP",["Java", "ECMAScript"]]]] を変数 langs に代入する ② langs の内容を p メソッドで表示する ③ flatten!メソッドを使って、配列 langs を平坦化する ④ langs の内容を p メソッドで表示する ⑤ reverse メソッドを使って、配列の要素を逆順に並べ替えた結果を、p メソッドで表示する 表示結果は以下のようになる **["Ruby", ["Python", ["PHP", ["Java", "ECMAScript"]]]]** **["Ruby", "Python", "PHP", "Java", "ECMAScript"]** **["ECMAScript", "Java", "PHP", "Python", "Ruby"]**
14-19	p285	配列を利用して以下のようなプログラムを作成しよう。 ① 空の配列 [] を変数 nums に代入する ② nums の内容を p メソッドで表示する ③ nums に 0〜99 までの乱数を要素として 10 個追加する（0〜99 までの乱数は rand(100)で発生できる） ④ nums の内容を p メソッドで表示する ⑤ sort メソッドを使って、配列の要素をソートした結果を、p メソッドで表示する 表示結果は以下のようになる **[]** **[12, 41, 6, 99, 65, 54, 30, 90, 85, 99]** **[6, 12, 30, 41, 54, 65, 85, 90, 99, 99]**

No	対応頁	演習
14-20	p285	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", "Python", "PHP","Java", "ECMAScript"]を変数 langs に代入する ② langs の内容を p メソッドで表示する ③ sort_by メソッドを使って、langs を文字数の少ない順に並べ替えた配列を p メソッドで表示する（文字列の文字数は文字列.size メソッドで求められる） 表示結果は以下のようになる **["Ruby", "Python", "PHP", "Java", "ECMAScript"]** **["PHP", "Ruby", "Java", "Python", "ECMAScript"]**
14-21	p288	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", "Python", "PHP","Java", "ECMAScript"]を変数 langs に代入する ② langs に pop メソッドを実行し、結果を変数 item に代入する ③ puts メソッドで、item を downcase メソッドで小文字に変換した結果を表示する ④ langs が空になる(langs.pop が nil になる）まで上記②～③を繰り返す 表示結果は以下のようになる **ecmascript** **java** **php** **python** **ruby**

No	対応頁	演習
15-1	p292～294	配列を利用して以下のようなプログラムを作成しよう。 ① 空の配列 [] を変数 nums に代入する ② nums に 0～99 までの乱数を要素として 100 個追加する （0～99 までの乱数は rand(100)で発生できる） ③ 0～99 までの整数を gets メソッドを使って入力させる ④ count メソッドを使って、入力した数値と同じ値の要素の個数を表示する 表示結果は以下を参考にしよう **0～99 の整数を入力してください : 45** **値が 45 の要素は 3 個あります**
15-2	p292～294	配列を利用して以下のようなプログラムを作成しよう。 ① 空の配列 [] を変数 nums に代入する ② nums に 0～99 までの乱数を要素として 20 個追加する （0～99 までの乱数は rand(100)で発生できる） ③ nums の内容を p メソッドで表示する ④ detect メソッドを使って、nums の中で 3 の倍数(3 で割った余りが 0 に等しい)である最初の要素を表示する 表示結果は以下を参考にしよう **[97, 29, 24, 74, 76, 57, 64, 84, 59, 18, 41, 46, 84, 9, 92, 72, 72, 34, 43, 11]** **配列のうち、３の倍数である最初の要素は 24 です**

No	対応頁	演習
15-3	p292～294	配列を利用して以下のようなプログラムを作成しよう ① 空の配列 [] を変数 nums に代入する ② nums に 0～99 までの乱数を要素として 20 個追加する（0～99 までの乱数は rand(100)で発生できる） ③ nums の内容を p メソッドで表示する ④ find_all メソッドを使って、nums の中で偶数(2 の倍数)である要素すべてを配列として求め、p メソッドで表示する 表示結果は以下を参考にしよう **[91, 10, 1, 92, 39, 57, 53, 50, 56, 25, 68, 29, 39, 56, 37, 24, 87, 83, 44, 85]** **配列のうち、偶数の要素は以下の通りです** **[10, 92, 50, 56, 68, 56, 24, 44]**
15-4	p292～294	配列を利用して以下のようなプログラムを作成しよう。 ① 空の配列 [] を変数 nums に代入する ② 空の配列 [] を変数 result に代入する ③ nums に 0～9 までの乱数を要素として 10 個追加する（0～9 までの乱数は rand(10)で発生できる） ④ nums に対して each_slice(2)メソッドを使って、ブロックに|i,j|の２つのブロック変数をとり、その２つの値の合計値を、配列 results の末尾に<<メソッドを使って追加する ⑤ 配列 result の内容を、p メソッドを使って表示する 表示結果は以下を参考にしよう **[5, 6, 7, 0, 5, 5, 8, 9, 0, 1]** **配列の要素を２個ずつ合計した結果は以下の通りです** **[11, 7, 10, 17, 1]**

No	対応頁	演習
15-6	p292〜294	配列を利用して以下のようなプログラムを作成しよう。 ① 配列 ["Ruby", "Python", "PHP","Java", "ECMAScript"]を変数 langs に代入する ② each_with_index メソッドを使って、以下の表示例にあるような**内容を表示する** 表示結果は以下のようになる **0:Ruby** **1:Python** **2:PHP** **3:Java** **4:ECMAScript**
15-7	p292〜294	配列を利用して以下のようなプログラムを作成しよう ① 配列 ["Ruby", "Python", "PHP","Java", "ECMAScript"]を変数 langs に代入する ② inject メソッドを使って、それぞれの要素の頭文字（先頭の文字。文字列[0]で取得できる）をすべて連結した文字列を生成する ③ 上記②の結果を puts メソッドを使って表示する 表示結果は以下のようになる **RPPJE**

第６週　演習

学習ポイント

第 16 コマ：String クラスを使いこなす

（『たのしい Ruby』対応範囲: 第 14 章）
・文字列リテラルの詳細
・文字列長の取得
・文字列の分割、結合
・文字列の比較
・文字列の検索と置換
・文字コードの扱い

第 17 コマ：Hash クラスを使いこなす

（『たのしい Ruby』対応範囲: 第 15 章）
・Hash の作り方
・値の保存と取り出し
・値の削除
・キーや値の存在確認
・Hash のマージ
・キーと値の繰り返し
・オブジェクトの同一性判定

第 18 コマ：簡単なゲームを作る(1)

・復習、チーム開発の入門
  a) 数当てゲーム
  b) ペアで開発

演習課題

No	対応頁	演習
16-1	p298～299	次の内容を実行するプログラムを作成しなさい。 ① 「%Q」を利用して、「私は"たのしい Ruby"という本を読んでいます」という文字列を変数 message に代入する ② 変数 message を、puts メソッドを使って表示する
16-2	p298～299	次の内容を実行するプログラムを作成しなさい。 ① ヒアドキュメントを利用して、 **私は** **"たのしい Ruby"** **という本を** **読んでいます** という文字列を変数 message に代入する ② 変数 message を puts メソッドを使って表示する
16-3	p304	次の内容を実行するプログラムを作成しなさい。 ① "私は「たのしい Ruby」という本を読んでいます”という文字列を変数 message に代入する ② 変数 message を puts メソッドを使って表示する ③ length または size メソッドを使って、message の文字数を表示する ④ bytesize メソッドを使って、message のバイト数を表示する ⑤ empty?メソッドを使って、message の長さが 0 であるかどうかを表示する 表示例は以下の通り **私は「たのしい Ruby」という本を読んでいます** **23** **46** **False**

No	対応頁	演習
16-4	p304～305	次の内容を実行するプログラムを作成しなさい。 ① "私はたのしい Ruby という本を読んでいます”という文字列を変数 message に代入する ② 変数 message を、puts メソッドを使って表示する ③ [ ]メソッドを使って、「私」の部分だけを取り出し、puts メソッドで表示する ④ [ ]メソッドを使って、「たのしい Ruby」の部分だけを取り出し、puts メソッドで表示する ⑤ [ ]メソッドを使って、「読んでいます」の部分だけを取り出し、puts メソッドで表示する
16-5	p305～306	次の内容を実行するプログラムを作成しなさい。 ① 変数 str1 に "たのしい"という文字列を代入する ② 変数 str2 に"Ruby"という文字列を代入する ③ str1 と str2 を、連結した文字列を puts メソッドを使って表示する ④ << メソッドを使って、str1 に str2 をつなげる ⑤ str1 を、puts メソッドを使って表示する ⑥ concat メソッドを使って、str2 に str1 をつなげる ⑦ str2 を、puts メソッドを使って表示する
16-6	p310～311	次の内容を実行するプログラムを作成しなさい。 ① 変数 str に "たのしい Ruby"という文字列を代入する ② loop または while を使って、str が空文字（長さ０）になるまで以下③～④を繰り返す ③ str を、puts メソッドを使って表示する ④ chop!メソッドを使って str の末尾の文字を１文字削る 実行例は以下のようになる **たのしい Ruby** **たのしい Rub** **たのしい Ru** **たのしい R** **たのしい** **たのし** **たの** **た**

No	対応頁	演習
16-7	p311～312	次の内容を実行するプログラムを作成しなさい。 ① "私はたのしい Ruby という本を読んでいます”という文字列を変数 message に代入する ② 変数 message に”たのしい"が含まれるかどうかを、index メソッドで調べて、含まれる場合は、puts メソッドを使って**「たのしいは含まれます」**と表示する ③ 変数 message に"Ruby"が含まれるかどうかを、include?メソッドで調べ、含まれる場合は、puts メソッドを使って**「Ruby は含まれます」**と表示する
16-8	p313～314	次の内容を実行するプログラムを作成しなさい。 ① "私はたのしい JavaRuby という本を読んでいます”という文字列を変数 message に代入する ② slice!メソッドを使って、変数 message から”Java”という文字を取り除く ③ 変数 message の値を puts メソッドを使って表示する 実行結果は以下のようになる **私はたのしい Ruby という本を読んでいます**
16-9	p315～316	次の内容を実行するプログラムを作成しなさい。 ① ヒアドキュメントを利用して、 **私は** **たのしい Ruby** **という本を** **読んでいます** という文字列を変数 message に代入する ② each_line メソッドを使って、変数 message の各行ごとに"「"を先頭に、"」"を末尾につけて puts メソッドを使って表示する 実行結果は以下のようになる **「私は」** **「たのしい Ruby」** **「という本を」** **「読んでいます」**

No	対応頁	演習
16-10	p317～324	次の内容を実行するプログラムを作成しなさい。 ①　ヒアドキュメントを利用して、 **私 Python は** **たのしい RubyPython** **という Python 本を** **Python 読んでいます** という文字列を変数 message に代入する ② delete!メソッドを使って、変数 message から”Python”という文字を取り除く ③ each_line メソッドを使って、変数 message の各行を reverse メソッドを使って逆順に並べ替え、puts メソッドを使って表示する 実行結果は以下のようになる **は私** **ybuR いしのた** **を本ういと** **すまいでん読**
17-1	p325～327	次の内容を実行するプログラムを作成しなさい。 ① Hash クラスの new メソッドを利用して、新規に Hash オブジェクトを生成して変数 profile に格納する ② 変数 profile に対して、キーが「:name」、値が「"ルビィ太郎"」のペアを格納する ③ 変数 profile に対して、キーが「:age」、値が「30」のペアを格納する ④ pp メソッドを使って、変数 profile の内容を表示する 実行結果は以下のようになる **{:name=>"ルビィ太郎", :age=>30}**

No	対応頁	演習
17-2	p328～329	次の内容を実行するプログラムを作成しなさい。 ① Hash クラスの new メソッドを利用して、新規に Hash オブジェクトを生成して変数 profile に格納する ② 変数 profile に対して、キーが「:name」、値が「"ルビィ太郎"」のペアを store メソッドを使って格納する ③ 変数 profile に対して、キーが「:age」、値が「30」のペアを store メソッドを使って格納する ④ 変数 profile に対して、キーが「:address」、値が「"東京都千代田区"」のペアを store メソッドを使って格納する ⑤ 変数 profile に対して、キーが「:job」、値が「"プログラマ"」のペアを store メソッドを使って格納する ⑥ gets メソッドを使って、キーの名前を入力し、to_sym メソッドでシンボルに変換したものを、変数 key に格納する ⑦ 変数 profile から、fetch メソッドを使って、変数 key の値をキーとして値を取り出して表示する。ただし、登録されていないキーだった場合は、「未登録のキーが指定されました」と表示させるように fetch メソッドの引数を設定すること。 実行結果は以下のようになる **（登録済みのキーを指定した場合）** **キーを入力してください:name** **ルビィ太郎** **（未登録のキーを指定した場合）** **キーを入力してください:aaa** **未登録のキーが指定されました**

No	対応頁	演習
17-3	p329～330	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: 30,address: "東京都千代田区",job: "プログラマ"} を変数 profile に格納する ② profile に格納されているキーの一覧を、keys メソッドで取得し p メソッドで表示する ③ profile に格納されている値の一覧を、values メソッドで取得し p メソッドで表示する ④ profile に格納されているキーと値のペアの一覧を、to_a メソッドで取得し p メソッドで表示する 実行結果は以下のようになる **[:name, :age, :address, :job]** **["ルビィ太郎", 30, "東京都千代田区", "プログラマ"]** **[[:name, "ルビィ太郎"], [:age, 30], [:address, "東京都千代田区"], [:job, "プログラマ"]]**
17-4	p329～330	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: 30,address: "東京都千代田区",job: "プログラマ"} を変数 profile に格納する ② profile に格納されているキーと値の一覧を、each メソッドで取得しながら画面に表示する。ただし、each メソッドのブロックのブロック変数は１つ（配列として）で受け取ること。 実行結果は以下のようになる **name は ルビィ太郎 です** **age は 30 です** **address は 東京都千代田区 です** **job は プログラマ です**

No	対応頁	演習
17-5	p331～332	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: 30,address: "東京都千代田区",job: "プログラマ"} を変数 profile に格納する ② gets メソッドを使って、キーの名前を入力し、to_sym メソッドでシンボルに変換したものを、変数 key に格納する ③ key?/has_key?/include?/member?メソッドのいずれかを使って、入力されたキーが登録されているかを調べ、結果を画面に表示する ④ gets メソッドを使って、値を入力したものを変数 value に格納する。ただし、数字のみの内容が入力された場合は整数に変換して格納すること ⑤ value?/has_value?メソッドのいずれかを使って、入力された値が登録されているかを調べ、結果を画面に表示する 実行結果は以下のようになる **（登録済みのキーと値を指定した場合）** **キーを入力してください:name** **登録されています** **値を入力してください:プログラマ** **登録されています** **（未登録のキーと値を指定した場合）** **キーを入力してください:aaa** **未登録です** **値を入力してください:エンジニア** **未登録です**

No	対応頁	演習
17-6	p332～334	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: "30",address: "東京都千代田区",job: "プログラマ"} を変数 profile に格納する（※age の値は文字列にしておくこと） ② profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ③ delete メソッドを使って、profile からキーが:job のペアを削除する ④ profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ⑤ delete_if/reject!メソッドのいずれかを使って、profile から値に数字を含まないペアをすべて削除する ⑥ profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ⑦ delete メソッドを使って、profile からキーが:age のペアを削除する ⑧ profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ⑨ profile.empty?の結果を表示する 実行結果は以下のようになる **{:name=>"ルビィ太郎", :age=>"30", :address=>"東京都千代田区", :job=>"プログラマ"}** **profile の大きさは 4 です** **{:name=>"ルビィ太郎", :age=>"30", :address=>"東京都千代田区"}** **profile の大きさは 3 です** **{:age=>"30"}** **profile の大きさは 1 です** **{}** **profile の大きさは 0 です** **true**

No	対応頁	演習
17-7	p334～336	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: 30,address: "東京都千代田区",job: "プログラマ"} を変数 profile に格納する ② profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ③ clear メソッドを使って、profile の内容を初期化する ④ profile の内容を pp メソッドで表示した後、profile の大きさを、size/length メソッドのいずれかを使って表示する ⑤ profile.empty?の結果を表示する 実行結果は以下のようになる **{:name=>"ルビィ太郎", :age=>30, :address=>"東京都千代田区", :job=>"プログラマ"}** **profile の大きさは 4 です** **{}** **profile の大きさは 0 です** **true**
17-8	p336	次の内容を実行するプログラムを作成しなさい。 ① Hash オブジェクト { name: "ルビィ太郎",age: 30 }を変数 h1 に格納する ② Hash オブジェクト { address: "東京都千代田区",job: "プログラマ"} を変数 h2 に格納する ③ 変数 h1 と h2 の内容をそれぞれ pp メソッドで表示する ④ merge メソッドで h1 と h2 を組みあわせた結果を変数 profile に格納する ⑤ profile の内容を pp メソッドで表示する 実行結果は以下のようになる **{:name=>"ルビィ太郎", :age=>30}** **{:address=>"東京都千代田区", :job=>"プログラマ"}** **{:name=>"ルビィ太郎", :age=>30, :address=>"東京都千代田区", :job=>"プログラマ"}**

No	対応頁	演習
18-1	これまでに習った範囲	次の内容を実行するプログラムを作成しなさい。 ① ランダムに 1 桁の数字をプログラム内部で選ぶ ② 外部入力を受け付ける ③ 外部入力された数字と①で選んだ数字を比較、判定する ④ 判定結果を表示して終了する ⑤ 余裕がある場合は次の内容の機能を追加しなさい。 ⑥ 判定結果が NG の場合、続けて次の入力を受け付ける ⑦ 外部入力で 2 桁以上の数字や文字列などが入力された場合、メッセージを表示して入力しなおしさせる
18-2	なし	ペアプログラミングの利点や、実施する際の注意点について述べなさい。

第 7 週　演習

学習ポイント

第 19 コマ：簡単なゲームを作る(2)

・復習、チーム開発の入門
  a) Hit & Blow ゲーム
  b) ペアで開発

第 20 コマ：クラスを作る(1)

（『たのしい Ruby』対応範囲: 第 8 章）
・クラスとインスタンス
・class 文, initialize メソッド, インスタンス変数
・アクセスメソッド
・self

第 21 コマ：クラスを作る(2)

（『たのしい Ruby』対応範囲: 第 8 章）
・定数
・クラス変数
・public, private, protected

演習課題

No	対応頁	演習
19-1	これまでに習った範囲	ヒット・アンド・ブロウ（Hit&Blow）の出題をするプログラムを作成しなさい。ゲームのルールを以下に説明する。 ① 出題者は回答者に分からないように 4 桁の数字を選ぶ。（0〜9 から重複しない 4 つを選ぶ。） ② 回答者は数字を予想して回答する。 ③ 出題者は回答を判定する。 ・位置も数字も正しければ「ヒット」とする。 ・数字は正しいが位置が違う場合は「ブロウ」とする。 ・ヒットとブロウの数を回答者に伝える。 ④ ②と③を繰り返してすべてヒットになれば終了する。 **例：出題が「0123」のとき、** **回答が「1234」であれば「0H 3B」となる。** **回答が「0234」であれば「1H 2B」となる。**
19-2	なし	ペアプログラミングの利点や、実施する際の注意点について述べなさい。
20-1	p143	Array.new を使って Array のインスタンスを作成しなさい。 リテラル表記を使って Array のインスタンスを作成しなさい。 リテラル表現を使って String のインスタンスを作成しなさい。
20-2	p143	Array と String のインスタンスを生成し、各インスタンスの class メソッドを使ってクラス名を表示させなさい。
20-3	p143	Array と String のインスタンスを生成し、各インスタンスの instance_of?メソッドを使って、そのインスタンスのクラスを判定しなさい。

No	対応頁	演習
20-4	p145	is_a?メソッドを使って、継承関係を考慮したクラスを判定しなさい。 **String のインスタンスを生成し、String のインスタンスであるか** **String のインスタンスを生成し、Object のインスタンスであるか** **String のインスタンスを生成し、Array のインスタンスであるか** そのほか、図 8.3 を参考にして Array、Hash、Integer などについて同様に判定しなさい。
20-5	p148	ファイルを作成し、List8.1 の receipt.rb を、class 文を定義して実行しなさい。 最初に **class 文**を作成しなさい。
20-6	p149	receipt.rb に initialize メソッドを追加してインスタンスを生成しなさい。
20-7	p150	output メソッドを定義して実行しなさい。
20-8	p152	receipt.rb にインスタンス変数@name に対するアクセスメソッドを追加して実行しなさい。
20-9	p153	receipt.rb に replace_name メソッドを追加して self の使い方を確認しなさい。
20-10	p153	self、nil、true、false への代入がエラーになることを確認しなさい。
21-1	p154	receipt.rb にクラスメソッド create_receipt_xyz を追加して実行しなさい。
21-2	p154	「class << self; end」を使ってクラスメソッドを定義しなさい。
21-3	p155	「def クラス名.メソッド名」を使ってクラスメソッドを定義しなさい。
21-4	p155	receipt.rb に定数 VERSION を追加して Receipt::VERSION として参照できることを確認しなさい。

No	対応頁	演習
21-5	p156	List8.4 receipt_count.rb を作成し、クラス変数の動作を確認しなさい。
21-6	p158	List8.5 access_test.rb を作成し、パブリックメソッドとプライベートメソッドの動作を確認しなさい。
21-7	p159	access_test.rb の public と private の呼び出しの方法を変えて動作を確認しなさい。
21-8	p159	List8.6 point.rb を作成し、protected の動作を確認しなさい。

第８週　演習

学習ポイント

第 22 コマ：クラスを拡張する

（『たのしい Ruby』対応範囲: 第 8 章）
・継承, 組み込みクラスの継承関係, instance_of?, is_a?
・オープンクラス
・alias, undef
・特異クラス

第 23 コマ：モジュールを作る

（『たのしい Ruby』対応範囲: 第 8 章）
・モジュールの主な使い方を紹介: Mix-in, 名前空間の提供
・モジュール: 定数, module_function
・Math モジュールの紹介

第 24 コマ：モジュールでクラスを拡張

（『たのしい Ruby』対応範囲: 第 8 章）
・Mix-in: include, prepend, extend, メソッド検索のルール

演習課題

No	対応頁	演習
22-1	p160	List8.7 ext_string.rb を作成し、既存のクラス(String)に新しいインスタンスメソッドを追加して実行しなさい ※class 文によって既存のクラスにもメソッド定義などを追加できる性質のことを「オープンクラス」と言います

No	対応頁	演習
22-2	p161	List8.7 ring_array.rb を作成し、Array クラスを継承した RingArray クラスを定義して実行しなさい
22-3	p162	instance_methods メソッドにより、以下のクラスに定義されているインスタンスメソッドを調べなさい **Object クラス** **String クラス** **Array クラス**
22-4	p162	instance_methods メソッドの戻り値はシンボルの配列になります String クラスと Object クラスのインスタンスメソッドの差分を調べなさい ※「String.instance_methods - Object.instance_methods」を実行すると、左側の配列の要素から右側の配列の要素を削除した配列を得ることができます。String クラスのインスタンスメソッドから、Object クラスのインスタンスメソッドを取り除くことにより、String クラスで独自に定義されたメソッドを調べることができます。
22-5	p164	List8.9 alisa_sample.rb を作成し、実行しなさい
22-6	p165	String オブジェクトの特異クラス定義を使って、そのオブジェクト専用のメソッドを定義しなさい
22-7		クラスに対して特異クラス定義を用いることで、クラスオブジェクトにメソッドを追加できます。p154 のクラスメソッドの定義を見直して、特異メソッドとの共通点を確認しなさい。
23-1	p167	p167 のサンプルコードを使って Math モジュールを試しなさい **Math.sqrt メソッド** **Math::PI 定数**

No	対応頁	演習
23-2	p168	List8.11 hello_module.rb を作成し、以下の動作を確認しなさい。 **定数の定義** **メソッドの定義** **モジュール関数の定義**
24-1	p170	List8.12 mixin_test.rb を作成し、実行しなさい。 **クラス定義内で include メソッドによってモジュールを mix-in できること** **クラスメソッド include?を使って mix-in されていることを確認できること**
24-2	p171	ancesotrs メソッドにより、継承関係を確認しなさい。 サンプルプログラムの C クラスだけでなく、String クラスや Array クラスについても同様に試しなさい。
24-3	p171	List8.12 prepend_test.rb を作成し、実行しなさい。 **クラス定義内で prepend メソッドによってモジュールを mix-in できること** ancestors メソッドによって include との違いを確認しなさい。
24-4	p174	inlcude や extend を使って、クラスに複数のモジュールを mix-in したのちに ancestors メソッドによって継承の関係を確認しなさい。
24-5	p177	extend メソッドによって、個々のインスタンスにモジュールを mix-in できることを確認しなさい。
24-6	p178	extend メソッドによってクラスに対して mix-in を行うことによって、クラスメソッドを使出来ることを確認しなさい。 extend メソッドは特異メソッドを追加すること **クラスは Class クラスのインスタンス(Class オブジェクト)であること** **Class オブジェクトのインスタンスメソッドがクラスメソッドであること**

第９週　演習

学習ポイント

第 25 コマ：オブジェクト指向プログラミング

（『たのしい Ruby』対応範囲: 第 8 章）
・オブジェクト = データ + 手続き
・カプセル化, ポリモーフィズム, ダックタイピング

第 26 コマ：オブジェクト指向プログラミングをやってみる

（『たのしい Ruby』対応範囲: 第 8 章 ）
・オブジェクト指向プログラミング」で学んだ内容を実践する

第 27 コマ：テストフレームワーク

・フレームワークを使ってテストコードを効率よく実装する方法を学ぶ
・test-unit の基本的な使い方（テストケースの定義とアサーション）
・スタブの定義

演習課題

No	対応頁	演習
25-1	なし	object_id メソッドを使って、変数、文字列、Array、クラスなどのオブジェクト番号がどうなるか調べなさい。また、その結果がどうしてそうなるか考えなさい。

No	対応頁	演習
25-2	なし	下記のコードに protected と private を設定し、メソッドの呼び出しができなくなる範囲を確認しなさい。 **class Greeting** **def morning** **puts “Good morning”** **end** **def afternoon** **puts “Good afternoon”** **end** **def evening** **puts “Good evening!”** **end** **def night** **puts “Good night”** **end** **end**
25-3	なし	下記を実行したとき、出力が異なる理由を考えなさい。 **puts nil.nil? # true** **puts "awesome_string".nil? # false**

No	対応頁	演習
25-4	なし	**下記のコードを case 文を使わず、同じ出力が得られるようにクラスおよびメソッドを定義して書き直しなさい。そのとき、クラス・メソッド定義で実装する場合の利点を考えなさい。** **def sound(animals)** **animals.each do |animal|** **case animal[:type]** **when "duck"** **puts "quack"** **when "cat"** **puts "myaa"** **end** **end** **animals = [{type: "duck"}, {type: "cat"}]** **sound(animals)**
26-1	なし	下記にあるコードをリファクタリングしなさい。 practice/hit_and_blow/hit_and_blow_low.rb
27-1	なし	テストケースの定義の仕方にどんなものがあるか、test-unit と minitest のマニュアルで確認しなさい。
27-2	なし	アサーションのメソッドにどんなものがあるか、test-unit と minitest のマニュアルで確認しなさい。

第１０週　演習

学習ポイント

第 28 コマ：自動テスト

・require, require_relative
・これまでの演習で作成したプログラムのテストコードを実装する

第 29 コマ：演算子を使いこなす

（『たのしい Ruby』対応範囲: 第 9 章）
・各種演算子（比較、代入、範囲演算子など）
・三項演算子
・演算子の優先順
・演算子メソッドを定義する

第 30 コマ：Range クラス, Comparable モジュール

（『たのしい Ruby』対応範囲: 第 9 章, 第 12 章のコラム）
・ダックタイピングと Mix-in の実例を学び、クラス定義の復習を行う
・自分の定義したクラスのインスタンスで Range オブジェクトを動作させる（succ メソッド）
・自分の定義したクラスのインスタンスで Comparable モジュールを動作させる（<=>メソッド）

演習課題

No	対応頁	演習
28-1	なし	18,19 コマで作成したゲームのテストを書きなさい。

No	対応頁	演習
29-1	p189	代入によって変数を作成しなさい。 再び代入することによって変数の値を変更しなさい。
29-2	p189	代入演算子によって以下の計算の結果を確認しなさい。 **sum = 0** **1.upto(10) do |i|** **sum += i** **end** **p sum** 足し算以外にも掛け算や割り算による結果も確認しなさい。その際、適切な初期値を検討しなさい。
29-3	p190	以下のプログラムを実行して結果を確認しなさい。 **def jouken1** **p "条件 1"** **false** **end** **def jouken2** **p "条件 2"** **false** **end** **p jouken1 && jouken2** **p jouken1 || jouken2**
29-4	p193	p193 の例を参考にして if 文と条件演算子の動きを比べなさい。
29-5	p194	範囲演算子を使って整数の 0 から 99 を含む Range オブジェクトを作成しなさい。「..」と「...」を使う方法をそれぞれ試してください。

No	対応頁	演習
29-6	p194	("a00".."a99").to_a の結果を確認しなさい。 順序を逆にした("a99".."a00").to_a の結果を確認しなさい。 ("a0".."z9").to_a の結果を確認しなさい。
29-7	p195	表 9.2 を参考にして演算子の優先順位を確認しなさい。
29-8	p198	List9.1 point.rb を作成して動作を確認しなさい。
29-9	p201	List9.2 を point.rb に追加して動作を確認しなさい。
30-1	p201	List9.3 を point.rb に追加して動作を確認しなさい。

第１１週　演習

学習ポイント

第 31 コマ：IO クラス

（『たのしい Ruby』対応範囲: 第 17 章）
・IO クラス
・StringIO クラス
・open-uri ライブラリ
・エンコーディング（Encoding クラス）

第 32 コマ：File クラス, Dir クラス

（『たのしい Ruby』対応範囲: 第 18 章）
・File クラス
・FileTest モジュール
・tempfile, fileutils ライブラリ
・Dir クラス
・find ライブラリ

第 33 コマ：データのシリアライズ
・JSON, YAML の簡単な説明
・JSON クラス, YAML クラス, Marshal クラス

演習課題

No	対応頁	演習
31-1	p366	標準入力と標準出力にメッセージを出力するプログラムを作成しなさい。 シェルのリダイレクトを使用して出力を分離しなさい。

No	対応頁	演習
31-2	p369	テキストファイルからデータを 1 行ずつ読み込んでそのまま標準出力に出力するプログラムを作成しなさい。先頭に行番号を付与するなど、テキストを加工するようにアレンジしなさい。
31-3	p371	File.read を使ってファイルの内容をすべて一度に読み込むプログラムを作成しなさい。読み込んだデータを加工して標準出力に出力しなさい。
31-4	p372	ファイルから 1 行ずつデータを読み込んで行数を数えるプログラムを作成しなさい。ファイルから 1 文字ずつデータを読み込んで文字数を数えるプログラムを作成しなさい。ファイルから 1 バイトずつデータを読み込んでバイト数を数えるプログラムを作成しなさい。
31-5	p376	ファイルを開いてテキストデータを書き込むプログラムを作成しなさい。ファイルを open する際にモードの指定を"w"や"a"を使い、かつさまざまなサイズのメッセージを出力してファイルの内容を確認しなさい。
31-6	p379	（Windows 環境のみ） テキストモードとバイナリモードの違いを確認しなさい。
31-7	p381	IO.popen を使って ls コマンド（Windows 環境の場合は dir コマンド）の出力をプログラムに取り込むプログラムを作成しなさい。
31-8	p382	open-uri ライブラリを使って、Web サイトからデータをダウンロードするプログラムを作成しなさい。
31-9	p383	stringio ライブラリはどのような場合に有用であるか考えなさい。
32-1	p388	File.rename メソッドを使ってファイル名を変更しなさい。
32-2	p389	FileUtils.cp メソッドを使ってファイルをコピーしなさい。
32-3	p390	File.unlink メソッドを使ってファイルを削除しなさい。
32-4	p392	Dir.open メソッド、Dir.each メソッドを使ってディレクトリ配下のファイルの一覧を取得しなさい。
32-5	p395	Dir.glob メソッドを使ってディレクトリ配下のファイルの一覧を取得しなさい。

No	対応頁	演習
32-6	p397	Dir.mkdir メソッドと Dir.rmdir メソッドを使ってディレクトリの作成と削除をしなさい。
32-7	p397	File.stat メソッドを使ってファイルやディレクトリの属性を取得しなさい。Dir.each メソッドで取得したファイルの一覧をファイルの更新時間順でソートしなさい。
32-8	p403	File.basename メソッド、File.dirname メソッド、File.extname メソッドの動作をそれぞれ確認しなさい。File.expand_path メソッドの動作を確認しなさい。
32-9	p404	__FILE__擬似変数と__dir__メソッドの動作を確認しなさい。
32-10	p405	find ライブラリを使ってディレクトリ配下のファイルのバイト数の合計を求めるプログラムを作成しなさい。出力は du コマンドを参考にしなさい。
33-1	p405	json ライブラリを読み込むと、JSON.dump メソッドによってオブジェクトを JSON データとしてシリアライズできるようになります。文字列や数値を含むハッシュや配列を作成し、JSON データに変換するプログラムを作成しなさい。
33-2	p405	JSON.load を使って JSON データを読み込むプログラムを作成しなさい。
33-3	p405	yaml ライブラリを読み込むと、YAML.dump メソッドによってオブジェクトを YAML データとしてシリアライズできるようになります。文字列や数値を含むハッシュや配列を作成し、YAML データに変換するプログラムを作成しなさい。
33-4	p405	YAML.load を使って YAML データを読み込むプログラムを作成しなさい。
33-5	p405	Marshal は Ruby に組み込まれたシリアライズ形式です。Marshal.dump メソッドを使ってオブジェクトをシリアライズするプログラムを作成しなさい。
33-6	p405	Marshal.load メソッドを使ってオブジェクトをシリアライズするプログラムを作成しなさい。

No	対応頁	演習
33-7	p405	Marshal を使用すると自分を要素に含む配列のように参照が循環するオブジェクトもシリアライズ/でシリアライズできます。 **ary = []** **ary[0] = ary** のようなオブジェクトを JSON、YAML、Marshal でシリアライズ/でシリアライズして結果を確認しなさい。

第１２週　演習

学習ポイント

第 34 コマ：エラー処理と例外

（『たのしい Ruby』対応範囲: 第 10 章）
・begin - rescue - ensure - end
・自分で例外を投げる
・自分で例外クラスを定義する

第 35 コマ：ブロックつきメソッド

（『たのしい Ruby』対応範囲: 第 11 章）
・block_given?
・yield を使ってブロックと値をやり取りする
・ブロック内での break, next の挙動の復習

第 36 コマ：Proc クラス

（『たのしい Ruby』対応範囲: 第 21 章）
・ブロック引数を proc クラスのインスタンスとして受け取る方法
・Proc クラス、クロージャ
・Lambda クラス

演習課題

No	対応頁	演習
34-1	p205	例外になる処理を実行してエラーメッセージを確認しなさい。

No	対応頁	演習
34-2	p207	例外になる処理に例外処理を追加して例外オブジェクトを捕捉するプログラムを作成しなさい。
34-3	p210	例外処理に後処理を追加して必ず実行されることを確認しなさい。
34-4	p211	例外処理に retry を追加して繰り返し実行するようすを確認しなさい。
34-5	p213	例外処理で捕捉する例外を指定して動作を確認しなさい。指定されていない例外が捕捉されないことを確認しなさい。
34-6	p214	オリジナルの例外クラスを定義しなさい。
34-7	p217	オリジナルの例外クラスを定義し、例外を発生させるプログラムを作成しなさい。
35-1	p223	Array#each Hash#each String#each_line のいずれかを使ったプログラムを作成しなさい。
35-2	p225	File.open にブロックを使わない書き方と、ブロックを使う書き方を比較しなさい。
35-3	p226	文字列を要素とする配列を作成し様々な順序でソートしなさい（アルファベット順、長さ順、これらの逆順など）。sort メソッドと sort_by メソッドの動作を比較しなさい。
35-4	p229	メソッドを定義し、呼び出しの際に指定されたブロックを yield によって実行したときの動きを確認しなさい。
35-5	p230	ブロック変数を受け取り、この値を加工してブロックの結果として返すメソッドを作成しなさい。
35-6	p233	Array#each Hash#each String#each_line のいずれかを使ったプログラムにおいてブロックの中で next や break を実行したときの動作を確認しなさい。
35-7	p234	ブロックを引数として受け取るプログラムを作成し、call メソッドによって実行するプログラムの動きを確認しなさい。

No	対応頁	演習
35-8	p236	ブロック変数とブロックの外側のローカル変数として同じ名前を使用した場合の動作を確認しなさい。
36-1	p441	Proc オブジェクトを作成して実行するプログラムを作成しなさい。
36-2	p443	lambda オブジェクトを作成して実行するプログラムを作成しなさい。
36-3	p446	ブロックを引数として受け取るプログラムを作成し、call メソッドによって実行するプログラムの動きを確認しなさい。
36-4	p446	p446 のサンプルコードを見て Symbol#to_proc の動作を確認しなさい。
36-5	p447	ブロック変数を受け取る Proc オブジェクトを作成し、その引数の仕様を確認するプログラムを作成しなさい。
36-6	p449	Proc#call Proc#[] Proc#yield Proc#() Proc#=== を使って Proc オブジェクトを実行しなさい。
36-7	p449	Proc オブジェクトを実行するために複数のメソッドが用意されている理由を考えなさい。（ヒント：ダックタイピング）

第１３週　演習

学習ポイント

第 37 コマ：CSV クラス

（『たのしい Ruby』対応範囲: 第 23 章）
・CSV ファイルの読み書き
・Ruby リファレンスマニュアルの使い方の復習

第 38 コマ：Ruby Gem を使ってみる

（『たのしい Ruby』対応範囲: 第 23 章）
・gem のインストール方法
・gem のドキュメントの読み方

第 39 コマ：Rack を使ってみる

（『たのしい Ruby』対応範囲: 第 23 章）
・Web サービスの開発を始める前段階として、Rails や Sinatra などのフレームワークの基盤となっている Rack の基本的な使い方を学ぶ

演習課題

No	対応頁	演習
37-1	p239	Ruby のリファレンスマニュアルで csv ライブラリのページを参照しなさい。CSV は「標準添付ライブラリ」のセクションにあります。
37-2	p471	CSV ファイルを作成し、CSV データを Ruby スクリプトで読み込みなさい。p471 のサンプルコードだけでなく、リファレンスマニュアルを参考にして 1 行ずつ読んで処理する方法と、全体を一度に読む方法の両方を確認しなさい。

No	対応頁	演習
37-3	p471	前の演習で読み込んだ CSV データを加工して出力するプログラムを作成しなさい。先頭にカラムを追加して行番号を追加しなさい。
37-4	p471	エンコーディング「Windows-31J」で日本語を含む CSV ファイルを作成し、Microsoft Excel を使って内容を確認しなさい。また「Windows-31J」の CSV ファイルを csv ライブラリで正しく読み込めることを確認しなさい。
38-1	p472	p472 の手順を参考にして sqlite3 ライブラリをインストールしなさい
38-2	p481	p481 の手順を参考にして sqlite3 ライブラリをインストールしなさい
38-3	p476	第 23 章の郵便番号データの検索プログラムを作成しなさい
39-1		sinatra ライブラリをインストールする
39-2		ルーティングと動作を記述する
39-3		静的ファイルを返す
39-4		テンプレートを使って HTML を返す
39-5		セッションにデータを保存する

第１４週　演習

学習ポイント

第 40 コマ：Ruby Gem を使いこなす

（『たのしい Ruby』対応範囲: 付録 B）

・rubygem コマンドの使い方
・よく知られた gem の紹介
・Bundler の紹介

第 41 コマ：ゲーム開発(1)

・総合演習として、ペアでアドベンチャーゲームを開発する
- どんなシナリオにするか計画を立てる

第 42 コマ：ゲーム開発(2)

・総合演習として、ペアでアドベンチャーゲームを開発する
- プレイヤーからの入力を文字列と数値で受け取る
- 条件分岐と繰り返しを使い、シナリオに沿って進行する

演習課題

No	対応頁	演習
40-1	p503	RubyGems の管理に用いられる基本的なコマンド（gem list / gem search / gem install / gem fetch / gem uninstall / gem update）を実行しなさい

第１５週　演習

学習ポイント

第 43 コマ：ゲーム開発(3)

・総合演習として、ペアでアドベンチャーゲームを開発する
- Array や Hash クラスを使ってゲーム内の状態を管理する

第 44 コマ：ゲーム開発(4)

・総合演習として、ペアでアドベンチャーゲームを開発する
- クラスを定義してプレイヤーの情報を管理する

第 45 コマ：ゲーム開発(5)

・総合演習として、ペアでアドベンチャーゲームを開発する
- セーブ機能を実装する

2019 年度「専修学校による地域産業中核的人材養成事業」
札幌（北海道）をモデルとした地域創生のための IT 人材育成と企業連携推進事業

2019 年度「専修学校による地域産業中核的人材養成事業」
札幌（北海道）をモデルとした地域創生のための IT 人材育成と企業連携推進事業

たのしい Ruby　演習と解説（教員用）

令和 2 年 2 月

学校法人吉田学園（吉田学園情報ビジネス専門学校）
〒060-0063　北海道札幌市中央区南 3 条西 1 丁目
TEL 011-272-6070　　FAX 011-272-6075