【正誤表一覧】

Ruby技術者認定試験合格教本 Silver/Gold対応 Ruby公式資格教科書

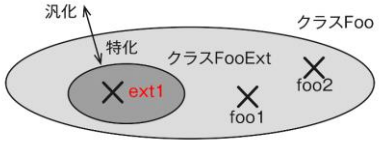
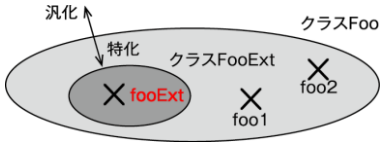
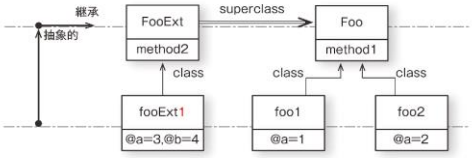
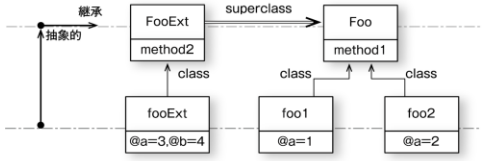
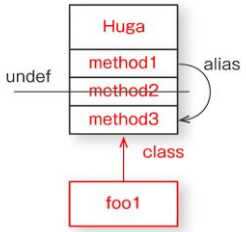
ISBN:978-4-7741-7567-6
2015年9月20日 初版 第1刷発行

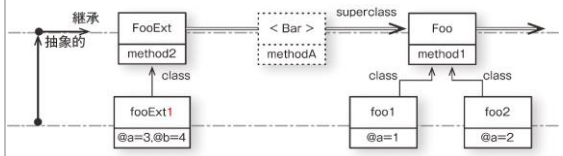
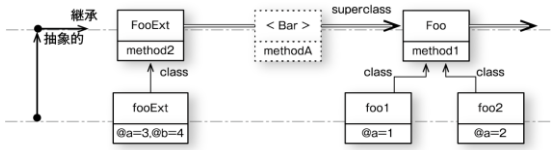
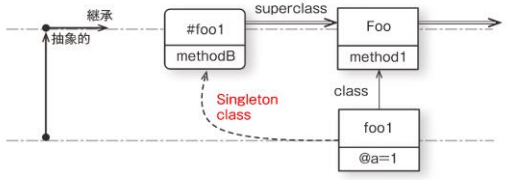
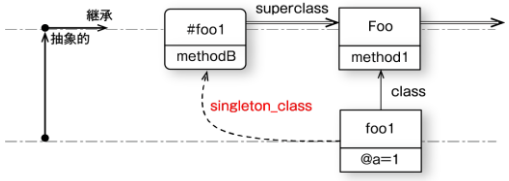
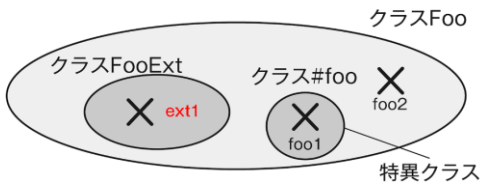
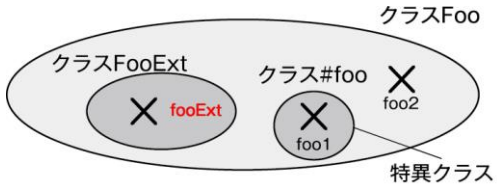
本書の以下の部分に誤りがありました。ここに訂正するとともに、ご迷惑をおかけしたことを深くお詫び申し上げます。

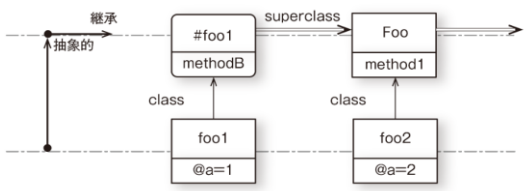
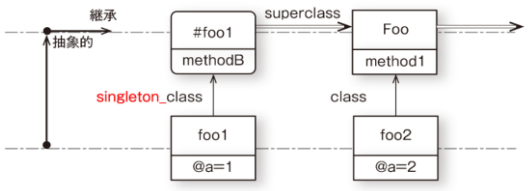
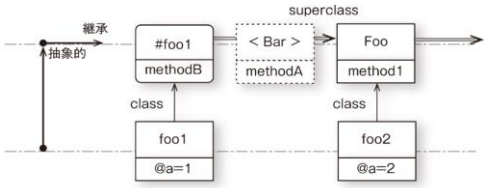
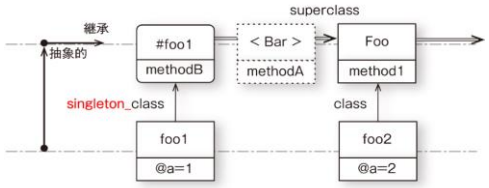
頁	該当箇所	誤	正	備考
p.8	「コード2-2 RVMのインストールをする」	gem install rvm	**curl -L https://get.rvm.io | bash -s stable --autolibs=enabled**	
p.8〜9	コード2-2〜コード2-3の間の本文	なお、社内LAN環境など、外部との接続にProxyを経由する場合は、コード2-3のように~/.curlrcにProxyの設定を行ってからインストールしてください。 ~/.curlrcは以下のように設定します。 proxy-user = "<username>:<password>" proxy = "http://<hostname>:<port>" また、プロキシ環境下の場合はgemコマンドなどを実行する際には、http_proxyとhttps_proxyを設定しておくと良いでしょう。	**なお、社内LAN環境など、外部との接続にプロキシを経由する場合は、コード2-3のように~/.curlrcにプロキシの設定を行ってから実行してください。** **proxy-user = "<username>:<password>"** **proxy = "http://<hostname>:<port>"** **また、プロキシ環境下の場合は上記の設定以外にもgemコマンドなどを実行する際にプロキシの設定が必要になるため、コード2-3のように環境変数に、http_proxyとhttps_proxyを設定しておくと良いでしょう。**	
p.21	1行目からコード2-19まで	クラス名やメソッド名が分からない場合は、コード2-19のように、searchの後にキーワードを指定して検索することができます。また、-lオプションを付けて実行すると、表示可能なクラスやメソッドを全てリスト表示します。 ▼コード2-19　riでキーワード検索 ri search String	**本章で紹介しているインストール方法ではriのドキュメントはインストールされないため、別** **途インストールする必要があります。** **RVMを利用している場合には、ドキュメントを作るための専用コマンドが用意されています。** **$ rvm docs generate**	Windowsではri用のドキュメントを用意する手段はありません（Web上のリファレンスマニュアル等を参照してください）。
p.33	「コード3-13 数値クラス」2行目	1000000000000000000.class #=> Fixnum※	**削除**	
p.33	側注	※ 64bit版での実行結果です。	**削除**	
p.43	「コード3-32 andの優先度」2行目	p (1) and 2と同義	**(p 1)** and 2と同義	
p.43	下から2行目	文字列をスペースで区切って記述すると、それらを連結した文字列を生成します。	**文字列リテラルを連続して記述する（間に空白があってもかまいません）と、**それらを連結した文字列を生成します。	
p.46	「表3-5 バックスラッシュ記法」の「指定内容」9行目	＼b	**¥**b	半角の「¥」と「＼」はフォントによって字体が異なるだけで、文字コードは
p.46	「表3-5 バックスラッシュ記法」の「指定内容」10行目	＼t	**¥**t	同上
p.46	「表3-5 バックスラッシュ記法」の「指定内容」11行目	＼v	**¥**v	同上
p.46	「表3-5 バックスラッシュ記法」の「指定内容」17行目	＼unnnn	**¥**unnnn	同上
p.46	「表3-5 バックスラッシュ記法」の「指定内容」18行目	＼u{nnnn}	**¥**u{nnnn}	同上
p.47	「コード3-37 pとprintとputsの違い（irbでの実行結果）」3行目	=> nil	=> **"a¥nb"**	同上
p.47	「コード3-38 pのバックスラッシュ記法」3行目	p "¥xff" #=> "¥377"	**削除**	

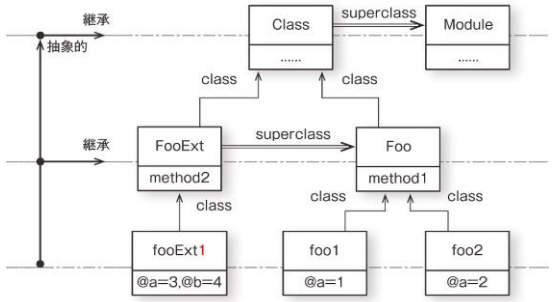
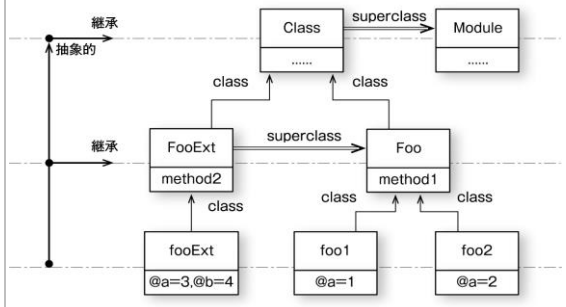
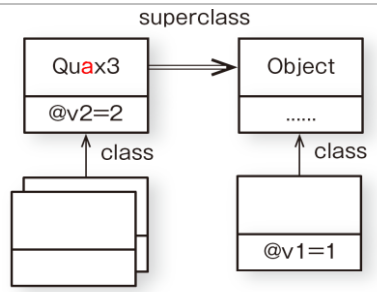
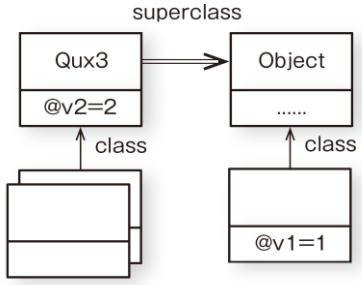
p.47	下から3行目	バックスラッシュ記法をシングルクォートで囲んだ場合は、シングルクォートのエスケープのみ適用され	バックスラッシュ記法をシングルクォートで囲んだ場合は、シングルクォート**とバックスラッシュ**のエスケープのみ適用され	
p.52	側注	※Ruby1.9では、これらのメソッドは文字数を返すように変更されています。	**削除**	
p.54	「コード3-56 文字列とシンボルの変換」差し替え	v1 = "foo1" v2 = v1.to_sym #:foo1 v3 = v2.to_s　#"foo1"	**v1 = "foo1" #=> "foo1"** **v2 = v1.to_sym #=> :foo1** **v3 = v2.to_s #=> "foo1"**	
p.60	「コード3-64 破壊的メソッドの例」6行目	"foo""	"foo"	
p.60	最下行	注意して使用してほしいメソッドに「！」を付ける場合もあるようです。	注意して使用してほしいメソッドに「！」を付ける場合も**あります。**	
p.61	「コード3-65 文字列とシンボルの違い」5行目～7行目	p v1.object_id p v2.object_id p v3.object_id	p v1.object_id #=> **23298400** p v2.object_id #=> **20179580** p v3.object_id #=> **22993260**	
p.61	「コード3-65 文字列とシンボルの違い」 13行目～15行目	p v1.object_id p v2.object_id p v3.object_id	p v1.object_id **# 718728** p v2.object_id **# 718728** p v3.object_id **# 718728**	同上
p.65	「コード3-72 インデックスに負の整数を指定」1行目	p v1　　#=> [10, nil, nil, "aa"]	p v1 = **[10, nil, nil, "aa"]**	
p.71	「コード3-88 ＋演算子と－演算子」4行目	#=> [1, 1, 2, 2, 2, 2, 3, 3]	#=> [1, 1, 2, 2, 2, 2, 3, **4**]	
p.73	本文、上から8行目	なお、後述する eachメソッドはスコープを作成します。eachメソッドのコードブロック中で初期化したローカル変数は、ブロックの外からは参照できません。	なお、**コード3-94のように**eachメソッドはスコープを作成します。**そのため、**eachメソッドのコードブロック中で初期化したローカル変数は、ブロックの外からは参照できません。 **for式も内部でeachメソッドを呼び出しますが、eachメソッドを直接呼び出した場合はブロックによってスコープが作成されるという違いがあります。**	
p.74	「コード3-95 ハッシュリテラルの例」4行目	#=> {"foo3"=>3, "foo1"=>1, "foo2"=>20}	#=> {"foo**1**"=>**1**, "foo**2**"=>**20**, "foo**3**"=>**3**}	
p.75	本文、上から5行目	文字列を使用する場合も同様に	文字列を使用する場合*****も同様に	側注のアスタリスク追記。
p.75	本文、上から8行目	配列のように…注意してください※。	**削除**	
p.76	「コード3-99 [ ] によるハッシュの生成」2行目	a = Hash[:hoofoo1, 1, :hoofoo2, 2, :hoofoo3, 3]	a = Hash[**:foo**1, 1, **:foo**2, 2, **:foo**3, 3]	
p.76	「コード3-99 [ ] によるハッシュの生成」5行目	a[:hoofoo1] #=> 1	a[**:foo**1] #=> 1	
p.78	「コード3-104 for式に範囲オブジェクトを適用」3行目	# aからzまでの文字が順に出力される	# **"a"**から**"z"**までの文字が順に出力される	
p.78	「コード3-105 配列の添字演算子に範囲オブジェクトを適用」4行目	["c"]"]:	["c"]**:**	
p.78	本文、下から3行目	その位置の文字の文字コードが返ります。	その位置の**文字が**返ります。	
p.78	「コード3-106 文字列の添字演算子に範囲オブジェクトを適用」2行目	#=> 98	**#=> "b"**	

p.79	「コード3-107 case式の例」14行目	1　#こちらは実行されない	1 #こちらは実行されない	「#こちらは…」の前に全角スペースがあるので半角スペースに修正。
p.81	「コード3-111 until式の例」2行目	#	#	全角から半角へ修正。
p.85	上から5行目	「/^bbb$/」と	「/^**bb**$/」と	
p.89	「コード3-123 正規表現グループ」3行目	wwwの次の任意の一文字	wwwの次の**ピリオド**	
p.89	「コード3-123 正規表現グループ」4行目	www(¥.)の次の任意の文字列の0個以上の繰り返し	**www.**の次の任意の文字の0個以上の繰り返し	
p.90	「コード3-125 コマンド出力」差し替え	puts `date` # 現在の時刻が表示される	**puts `date +%Y/%m/%d` # 今日の日付が表示される（Mac OS、Linux）** **puts `date /T`　　# Windowsのコマンドプロントの場合はこちら**	
p.97	本文、上から2行目	このように作成したProcインスタンスはlambdaとも呼ばれ、Procやブロックよりもメソッドに近い動きをします。	このように作成したProcインスタンスはlambdaとも呼ばれ、**オブジェクト化されていないブロックやProc.newで作成したProcインスタンスよりも**メソッドに近い動きをします。	
p.97	本文、上から6行目	Procでは生成元のスコープを脱出します。	**proc中のreturn**では生成元のスコープを脱出します。	
p.98	本文、上から2行目	Procとブロックでは無視するか、nilを代入します。	**procやオブジェクト化されていないブロックでは、余分な実引数を無視したり、実引数が足りない場合にnilを代入します。**	
p.98	「コード3-140 「->」を使用したlambda記法」3行目	p p1.call(x, y)	**p1.call(1, 2) #=> 3**	
p.99	上から3行目～5行目	またRuby 1.9以降からlambdaを「->」の形式で書くことが出来るようになりました。「->」を使った形式では引数を->(x){ … }のようにブロックの外で定義します。	**削除**	
p.99	「コード3-141 「->」を使用したlambda記法」	コード3-141　「 ->」を使用したlambda記法 p1 = ->(x, y){ p x + y } p p1.call(x, y)	**削除**	
p.100	「コード3-146 範囲オブジェクトのeachメソッド」3行目	#aからzまで順に出力される	#**"a"から"z"**まで順に出力される	
p.102	「コード3-150 start」5行目	end	end **sleep 1**	5行目の後に「sleep 1」を追加。
p.102	見出し「3-10-7　ファイバ」から下に2行目	並列処理するための機能です。スレッドと異なる点としては、スレッドは並列処理している～	**並行**処理するための機能です。スレッドと異なる点としては、スレッドは処理している～	
p.103	本文、下から6～7行目	また、Fiberの細かい解説については、第5章でも触れていますので、合わせて読んでください。	また、Fiber**については**第5章でも触れていますので、**あわせてお読み**ください。	
p.108	側注の3行目	Standard Errorを発生します。	**RuntimeError**を発生します。	「Standard」と「Error」の間のスペースを削除。
p.109	「コード3-162 例外の再発生」5行目	#=> ZeroDivisionError	#　ZeroDivisionError**の再発生**	
p.119	本文、最下行	別レイヤに配置にします。	別レイヤに配置します。	

p.121	「図4-4 継承クラス」内			図内の「ext1」を「fooExt」に修正。
p.121	本文、4行目	foo1とext1は	foo1と**fooExt**は	
p.121	「図4-5 継承したクラスオブジェクト」内			図内の「fooExt1」を「fooExt」に修正。
p.122	下から16行目	同様に、fooExt1の	同様に、fooExtの	
p.122	下から14行目	fooExt1のmethod2も、	fooExtのmethod2も、	
p.122	下から2行目	一方、fooExt1の	一方、fooExtの	
p.123	コード2行目	p foo1.method2 #=>NameError	p foo1.method2 #=> **NoMethodError**	
p.123	コード3行目	fooExt1 = FooExt.new(3,4)	fooExt = FooExt.new(3,4)	
p.123	コード4行目	p fooExt1.method1 #=>3	p fooExt.method1 #=>3	
p.125	「コード4-9 instance_methodsとinstance_variables」4行目	#=> [:@a, :@b]	#=> [:@**b**, :@**a**]	
p.127	「図4-7 undef式とalias式」内			・図内上部の文字を修正。 Huga → Hoge method1 → huga1 method2 → huga2 method3 → huga3 ・図内下部の「foo1」「class」を削除。
p.127	8行目	厳密にはObjectクラスの	厳密には**BasicObject**クラスの	
p.132	「コード4-17 includeメソッド」4行目	fooExt**1** = FooExt.new(3, 4) p fooExt**1**.methodA #=> 3	fooExt = FooExt.new(3, 4) p fooExt.methodA #=> 3	
p.133	「コード4-18 インクルードしたクラスの継承チェーン」1行目	#=> [FooExt, Bar, Foo, Object, Kernel]	#=> [FooExt, Bar, Foo, Object, Kernel, **BasicObject**]	

p.134	「図4-11 無名クラスが挿入された継承チェーン」内			図内の「fooExt1」を「fooExt」に修正。
p.135	「4-4 特異クラス」の3行目	特異クラスとは、指定したインスタンスだけに適用される特別なクラスです。	特異クラスとは、指定したインスタンスだけに適用される特別なクラスです。**シングルトンクラス（Singleton Class）とも呼ばれます。**	
p.137	「図4-13 特異クラスオブジェクト」内			図内の文字を修正。 Singleton class ↓ singleton_class
p.137	上から6行目	呼んでている	呼んでいる	
p.137	下から5行目	この考え方に基づいて本当のsingleton_classメソッドの参照先を描くと、	この考え方に基づいて、**クラスと特異クラスとインスタンスの関係を整理すると、**	
p.137	「図4-14 特異クラスの表現」内			図内の「ext1」を「fooExt」に修正。

p.137	下から5行目	この考え方に基づいて本当のclassメソッドの参照先を描くと、	この考え方に基づいて本当の**singleton_classメソッド**の参照先を描くと、	
p.138	「図4-15 特異クラスとインスタンスの関係」			図中左の「class」を「singleton_class」に修正。
p.138	「コード4-22 特異クラスの取得」5行目	#=> <Class:#<Foo:0xnnnnnnnn>>	# <Class:#<Foo:0xnnnnnnnn>>	
p.138	項見出し	4-4-2　特異クラスと参照と再オープン	4-4-2　特異クラスと参照と**定義**	
p.138	本文下から5行目とコードのキャプション（2ヵ所）	特異クラスの再オープン式	特異クラスの**定義**	
p.139	コード4-23のキャプション	コード4-23　特異クラスの再オープンによるメソッド定義	コード4-23　特異クラス**定義内での**メソッド定義	
p.139	下から4行目	そのメソッドが定義したクラスに	そのメソッドが定義**された**クラスに	
p.140	「コード4-25 def式中におけるdef式の呼び出し（メソッドのネスト）」9行目	#=> ["method1"]	#=> [**:method1**]	
p.140	「コード4-25 def式中におけるdef式の呼び出し（メソッドのネスト）」11行目	#=> ["method2", "method1"]	#=> [**:method2, :method1**]	
p.140	ページ下から14行目、5行目、4行目、2行目（計4ヵ所）	～特異クラスを再オープンして～ ～特異クラスの参照と再オープン方法を学びました。 　特異クラスの再オープンに関連して～ ～特異クラスを再オープンして～	～特異クラスを**定義**して～ ～特異クラスの参照と**定義**方法を学びました。 　特異クラスの**定義**に関連して～ ～特異クラスを**定義**して～	
p.141	「図4-16 特異クラスにMix-inした継承チェーン」			図中左の「class」を「singleton_class」に修正。
p.141	「コード4-27 extendメソッド」3行目	p foo1.methodA #1	p foo1.methodA #=> **1**	
p.142	「コード4-28 prependとincludeの違い」15行目	# => m1が出力される	# => **"m1"**が出力される	
p.142	「コード4-28 prependとincludeの違い」16行目	# => c2が出力される	# => **"c2"**が出力される	
p.142	下から8行目	同名のメソッドをモジュールを	同名のメソッド**を持つ**モジュールを	
p.144	コード4-31のキャプション	refinements	**R**efinements	

p.148	「図4-18 Classクラスオブジェクトの位置」内			図内の「fooExt1」を「fooExt」に修正。
p.151	本文、5行目	一般的なインスタンスの特異クラスへ特異メソッドを定義する考え方と～	一般的なインスタンスの特異クラスへメソッドを定義する考え方と～	
p.154	「コード4-41 引数を指定した呼び出し」5行目	def protected_method2; 2;	def protected_method2; 2; **end**	
p.156	3行目	putやprocなど	**puts**やprocなど	
p.156	コード4-44のキャプション	トップクラスにおけるselfの参照先	トップ**レベル**におけるselfの参照先	
p.156	本文、下から9行目	また、Objectクラスは全クラスのスーパークラスなので、Kernelモジュールで定義されたメソッドはプログラム中のどこででも呼び出すことができます。	また、Objectクラスは**BasicObject、およびBasicObjectのサブクラス以外の**全クラスのスーパークラスなので、Kernelモジュールで定義されたメソッドはプログラム中の**ほぼ**どこででも呼び出すことができます。	
p.157	「コード4-46 独自の組み込み関数の定義（Kernelを拡張）」見出し	独自の組み込み関数	独自の関数	
p.157	「コード4-47 独自の組み込み関数の定義（Objectを拡張）」見出し	独自の組み込み関数	独自の関数	
p.160	本文、上から3行目	@V2	@v2	
p.160	本文、上から4行目	Qux3クラスに	Qux3クラスに	
p.160	「図4-22 class中で宣言したインスタンス変数の格納先」内			図内の「Quax3」を「Qux3」に修正。
p.162	上から5行目	アクセッサを経由して	**メソッド**を経由して	
p.165	「コード4-59 メソッド定義の中の定数宣言」2～3行目	B = 1 end #=> SyntaxError	B = **2 # 文法エラー** end	

p.167	「コード4-64 constantsメソッド」差し替え	M.constants #=> ["B", "A", "C"] M::B.constants #=> ["A"] M::C.constants #=> []	M.constants #=> [**:A, :B, :C**] M::B.constants #=> [**:A**] M::C.constants #=> []	
p.172	「5-2.Objectクラス」から1行目	スーパークラスです*。	スーパークラスです。	側注のアスタリスク削除。
p.173	「コード5-1 オブジェクトのID（1）」1行目の後に1行追加	> a = "foo" > a.object_id	> a = "foo" **=> "foo"** > a.object_id	
p.174	本文、上から3行目～4行目	eql?メソッドはハッシュ（Hash）のキーが同じかどうか～	eql?メソッドは**オブジェクトのハッシュ値**が同じかどうか～	
p.175	「コード5-6 オブジェクトのメソッド一覧」の上1行目	配列の要素はメソッド名の文字列です。	配列の要素はメソッド名の**シンボル**です。	
p.175	「コード5-6 オブジェクトのメソッド一覧」4行目	> [:<=>,	=> [:<=>,	
p.176	「コード5-8 インスタンス変数の取得と設定」6行目	=> nil	**=> :initialize**	
p.176	「コード5-8 インスタンス変数の取得と設定」10行目	=> ["@hoge"]	=> [**:@hoge**]	
p.177	本文、上から7行目	ただしレシーバが指定されていないときはNameErrorになる場合があります。	**削除**	
p.177	「コード5-9 未定義メソッドの呼び出し」上から6行目	=> nil	**=> :method_missing**	
p.177	「コード5-9 未定義メソッドの呼び出し」下から2行	> hoge NameError: undefined local variable or method 'hoge' for main:Object	**> “string”.hoge** **NoMethodError: undefined method `hoge' for "string":String**	
p.178	上から3行目	inspectメソッドは主にデバッグ用途で使われます。	**例えばto_sメソッドはオブジェクトのクラス名を表示するのに対して、inspectメソッドはインスタンス変数とその値まで表示します。このため、inspectメソッドは主にデバッグ用途で使われます。**	
p.178	コード5-10	> a = 1.2 => 1.2 > a.to_s => "1.2" > Object.new.inspect => "#<Object:0x1079a4c98>"	> a = 1.2 => 1.2 > a.to_s => "1.2" **> class Hoge** **> def initialize** **> @foo = "bar"** **> end** **> end** **=> :initialize** **> hoge = Hoge.new** **=> #<Hoge:0x007fd39ed0c540 @foo="bar">** **> hoge.to_s** **=> "#<Hoge:0x007fd39ed0c540>"** **> hoge.inspect** **=> "#<Hoge:0x007fd39ed0c540 @foo=¥"bar¥">"**	
p.178	本文、最下行	Numeric、Integer、Fixnum、Bignum、Floatの5つあります。	Numeric、Integer、Fixnum、Bixnum、**Complex、Rational、**Float**の7つあります。**	
p.181	「コード5-16 整数の除算」4行目	=> 3.33333333333333	=> 3.3333333333333335	

p.181	最下行	ただしアスキーコードの範囲外の整数に対しては	ただし**対応する文字が存在しない場合は**	
p.182	「コード5-19 整数を使った繰り返し」3行目	> 10.times do {|i| sum += i}	> 10.times {|i| sum += i}	
p.184	「コード5-21 ビット演算」8行目	# 0b10101	# 0b**..**10101	
p.185	「図5-3 複素数の四則演算」の乗算、除算	$(a+bi)+(c+di)=(a+c)+(b+d)i$ $(a+bi)-(c+di)=(a+c)+(b-d)i$ $(a+bi)\times(c+di)=(ac+bc)+(bc+ad)i$ $\frac{a+bi}{c+di}=\frac{ac+bd}{c^2+d^2}+\frac{bc+ad}{c^2+d^2}i$	$(a+bi)+(c+di)=(a+c)+(b+d)i$ $(a+bi)-(c+di)=(a+c)+(b-d)i$ $(a+bi)\times(c+di)=(ac-bd)+(bc+ad)i$ $\frac{a+bi}{c+di}=\frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}i$	
p.187	「コード5-26 複素数から極座標への変換」差し替え	#=> 3.162277660168338 #=> 1.24904577239825 #=> [3.162277660168338, 1.24904577239825]	#=> 3.162277660168**3795** #=> 1.249045772398**2544** #=> [3.162277660168**3795**, 1.249045772398**2544**]	
p.189	「表5-1 floor、ceil、truncate、roundメソッドの意味」1行目	**floor** | **小さい方の整数に丸める**	floor | 小さい方の整数に丸める	表1行目セルのフォント修正。
p.189	「表5-1 floor、ceil、truncate、roundメソッドの意味」3～4行目	truncate　切り捨てをする round　四捨五入する	truncate　**0に近い方の整数に丸める** round　**絶対値を四捨五入する**	
p.193	「表5-4 主な文字コード」1行目	**UTF-8** | **主に利用されている文字コード**	UTF-8 | 主に利用されている文字コード	表1行目セルのフォント修正。
p.196	「コード5-40 数値を指定した場合」1行目～4行目	> 'abcdefg'[2] => "c" > 'abcdefg'.slice(2) => "c"	**削除**	
p.197	「コード5-41 範囲指定の場合（1）」10行目	aefg	aefg **=> nil**	10行目の後に「=> nil」を追加。
p.198	「コード5-42 範囲指定の場合（2）」12行目	aefg	aefg **=> nil**	12行目の後に「=> nil」を追加。
p.198	「コード5-43 文字列で指定する」12行目	aefg	aefg **=> nil**	12行目の後に「=> nil」を追加。
p.205	「コード5-54 文字列の末尾や先頭にある空白や改行を削除する」2行目	"abcdef¥n"	=> "abcdef¥n"	
p.205	「コード5-55 文字列を逆順にする」2行目	"abcdef"	=> "abcdef"	
p.206	「コード5-56 文字列の長さ」2行目	"abcdef"	=> "abcdef"	
p.206	「コード5-57 文字列のバイト数」2行目	"るびー"	=> "るびー"	
p.207	コード5-58のキャプション	文字列の長さ	文字列の**割り付け**	
p.207	「コード5-58」2行目	"abc"	=> "abc"	
p.207	「コード5-59 非表示文字列を変換する」2行目	"abc¥tdef¥tghi¥n"	=> "abc¥tdef¥tghi¥n"	
p.210	本文、上から2行目	eachメソッドとeach_lineメソッド、linesメソッドは	**each_lineメソッドと**linesメソッドは	

p.210	「コード5-63 文字列に対する繰り返し」1行目	.each {|c|	.each_**line** {|c|	
p.213	本文、下から1〜2行目	空文字やヌル文字「¥0」を含む文字列の場合は、ArgumentErrorが発生します。	**削除**	
p.214	「コード5-70 配列の生成（2）」1行目	> Array [1, 2, 3]	> Array[1, 2, 3]	「Array」の後のスペースを削除。
p.219	「コード5-79 配列の要素を参照する（2）」4行目	IndexError: index 4 out of array	IndexError: index 4 **outside of array bounds: -3...3**	
p.223	「コード5-90 配列の要素を削除する（6）」8行目	> [4, 5]	=> [4, 5]	
p.226	本文、下から1〜3行目	nitemsメソッドはnilではない要素の数を返します。ブロックが与えられた場合には、ブロックの評価結果がnilではない要素の数を返します。	**削除**	
p.226	「コード5-99 配列の長さを求める（2）」	> [1, nil, nil, 3, nil, 4].length => 6 > [1, nil, nil, 3, nil, 4].nitems => 3	**削除**	
p.227	上から1〜3行目	バージョン1.9では、nitemsメソッドは廃止されています。代わりにcountメソッドを使います。詳細はリファレンスマニュアルを参照してください。	**削除**	
p.231	「コード5-112 ハッシュの生成(1)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.231	「コード5-113 ハッシュの生成(2)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.233	上段の網掛け内の最下行	select	select **find_all**	1行追加。
p.233	「コード5-117 ハッシュのキーや値を取得する(1)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.233	「コード5-118 ハッシュのキーや値を取得する(2)」差し替え	> a = {"apple" => "fruit", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"fruit"} > a.keys => ["coffee", "apple"] > a.values => ["drink", "fruit"]	> a = {"apple" => "fruit", "coffee" => "drink"} **=> {"apple"=>"fruit", "coffee"=>"drink"}** > a.keys **=> ["apple", "coffee"]** > a.values **=> ["fruit", "drink"]**	
p.234	「コード5-121 ハッシュのキーや値を取得する（5）」直前の行	返り値がハッシュではなくキーと値の配列になることに注意しましょう。	**selectメソッドはキーと値の組み合わせについてブロックを評価して、結果が真となる組み合わせのみを含むハッシュを返します。find_allメソッドはselectメソッドと同じようにキーと値の組み合わせについてブロックを評価しますが、返り値はハッシュではなくキーと値の配列になります。**	
p.234	「コード5-121 ハッシュのキーや値を取得する（5）」4行目	=> [[2, "b"], [4, "d"]]	=> {2=>"b", 4=>"d"} **> a.find_all{|key, value| key % 2 == 0}** **=> [[2, "b"], [4, "d"]]**	4行目の修正と、その後に2行分を追加。
p.235	「コード5-122 ハッシュを変更する(1)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.235	「コード5-122 ハッシュを変更する(1)」6行目	=> {"coffee"=>"drink", "apple"=>"red"}	=> {**"apple"=>"red", "coffee"=>"drink"**}	

p.235	「コード5-122 ハッシュを変更する(1)」10行目	=> {"orange"=>"orange", "coffee"=>"drink", "apple"=>"red"}	=> {**"apple"=>"red"**, "coffee"=>"drink", **"orange"=>"orange"**}	
p.235	「コード5-123 ハッシュを変更する(2)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.236	「コード5-124 ハッシュを変更する(3)」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.236	「コード5-124 ハッシュを変更する(3)」6行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.236	「コード5-125 ハッシュを変更する（4）」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.236	「コード5-126 ハッシュを変更する（5）」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.236	「コード5-126 ハッシュを変更する（5）」5行目、6行目（2ヵ所）	"tee" => "drink"	**"tea"** => "drink"	
p.237	上から1行目～3行目	shiftメソッドはハッシュからキーと値の組み合わせを1つ取り除き、その組み合わせを配列として返します。どの組み合わせが取り除かれるのかは不定です。	shift メソッドはハッシュから**先頭のキーと**値の組み合わせを1つ取り除き、その組み合わせを配列として返します。	
p.237	「コード5-127 ハッシュを変更する(6)」差し替え	> a = {"apple" => "fruit", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"fruit"} > a.shift => ["coffee", "drink"] > a => {"apple"=>"fruit"}	> a = {"apple" => "fruit", "coffee" => "drink"} **=> {"apple"=>"fruit", "coffee"=>"drink"}** > a.shift **=> ["apple", "fruit"]** > a **=> {"coffee"=>"drink"}**	
p.237	「コード5-128 ハッシュを変更する(7)」差し替え	> a = {"apple" => "foods", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"foods"} > a.merge({"orange" => "fruit", "tee" => "drink", "apple" => "fruit"}) => {"orange"=>"fruit", "coffee"=>"drink", "apple"=>"fruit", "tee"=>"drink"} > a => {"coffee"=>"drink", "apple"=>"foods"} > a.merge({"orange" => "fruit", "tee" => "drink"}){|key, self_val, other_val| > self_val > } => {"orange"=>"fruit", "coffee"=>"drink", "apple"=>"foods", "tee"=>"drink"}	> a = {"apple" => "foods", "coffee" => "drink"} **=> {"apple"=>"foods", "coffee"=>"drink"}** > a.merge({"orange" => "fruit", **"tea"** => "drink", "apple" => "fruit"}) **=> {"apple"=>"foods", "coffee"=>"drink", "orange"=>"fruit", "tea"=>"drink"}** > a **=> {"apple"=>"foods", "coffee"=>"drink"}** > a.merge({"orange" => "fruit", **"tea"** => "drink"}){|key, self_val, other_val| > self_val > } **=> {"apple"=>"foods", "coffee"=>"drink", "orange"=>"fruit", "tea"=>"drink"}**	
p.238	「コード5-129 ハッシュを変更する（8）」差し替え	> a = {"apple" => "foods", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"foods"} > a.merge!({"orange" => "fruit", "tee" => "drink", "apple" => "fruit"}) => {"orange"=>"fruit", "coffee"=>"drink", "apple"=>"fruit", "tee"=>"drink"} > a => {"orange"=>"fruit", "coffee"=>"drink", "apple"=>"fruit", "tee"=>"drink"}	> a = {"apple" => "foods", "coffee" => "drink"} **=> {"apple"=>"foods", "coffee"=>"drink"}** **> a.merge!({"orange" => "fruit", "tea" => "drink", "apple" => "fruit"})** **=> {"apple"=>"fruit", "coffee"=>"drink", "orange"=>"fruit", "tea"=>"drink"}** **> a** **=> {"apple"=>"fruit", "coffee"=>"drink", "orange"=>"fruit", "tea"=>"drink"}**	

p.238	「コード5-130 ハッシュを変更する（9）」2行目	=> {"coffee"=>"drink", "apple"=>"foods"}	=> {**"apple"=>"foods", "coffee"=>"drink"**}	
p.238	「コード5-130 ハッシュを変更する（9）」5行目	"tee" => "drink"	"**tea**" => "drink"	
p.238	「コード5-130 ハッシュを変更する（9）」6行目	=> {"drink"=>"tee", "fruit"=>"apple"}	=> {**"fruit"=>"apple", "drink"=>"tea"**}	
p.238	「コード5-131 ハッシュを変更する（10）」2行目	=> {"coffee"=>"drink", "apple"=>"foods"}	=> {**"apple"=>"foods", "coffee"=>"drink"**}	
p.239	本文、上から1行目	配列の	**ハッシュ**の	
p.239	本文、上から2行目	配列が	**ハッシュ**が	
p.239	「コード5-132 ハッシュを調べる（1）」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.239	「コード5-133 ハッシュを調べる（2）」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.240	「コード5-134 ハッシュを調べる（3）」2行目	=> {"coffee"=>"drink", "apple"=>"fruit"}	=> {**"apple"=>"fruit", "coffee"=>"drink"**}	
p.240	本文、上から1行目～2行目	ここでも繰り返しの順序は不定となることに注意しましょう。	**削除**	
p.240	「コード5-135 ハッシュを使った繰り返し（1）」差し替え	> a = {"apple" => "fruit", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"fruit"} > a.each{|key, value| puts "#{key} => #{value}¥n"} coffee => drink apple => fruit => {"coffee"=>"drink", "apple"=>"fruit"}	> a = {"apple" => "fruit", "coffee" => "drink"} **=> {"apple"=>"fruit", "coffee"=>"drink"}** > a.each{|key, value| puts "#{key} => #{value}¥n"} **apple => fruit** **coffee => drink** **=> {"apple"=>"fruit", "coffee"=>"drink"}**	
p.240	「コード5-136 ハッシュを使った繰り返し（2）」差し替え	> a = {"apple" => "fruit", "coffee" => "drink"} => {"coffee"=>"drink", "apple"=>"fruit"} > a.each_key{|key| puts "key: #{key}¥n"} key: coffee key: apple > a.each_value{|value| puts "value: #{key}¥n"} value: drink value: fruit	> a = {"apple" => "fruit", "coffee" => "drink"} **=> {"apple"=>"fruit", "coffee"=>"drink"}** **> a.each_key{|key| puts "key: #{key}¥n"}** **key: apple** **key: coffee** **=> {"apple"=>"fruit", "coffee"=>"drink"}** **> a.each_value{|value| puts "value: #{value}¥n"}** **value: fruit** **value: drink** **=> {"apple"=>"fruit", "coffee"=>"drink"}**	
p.241	「コード5-137 ハッシュをソートする」2行目	=> {1=>"d", 2=>"c", 3=>"b", 4=>"a"}	=> {**4=>"a", 3=>"b", 2=>"c", 1=>"d"**}	
p.241	「コード5-138 ハッシュを変換する」2行目	=> {1=>"d", 2=>"c", 3=>"b", 4=>"a"}	=> {**4=>"a", 3=>"b", 2=>"c", 1=>"d"**}	
p.241	「コード5-138 ハッシュを変換する」4行目	=> [[1, "d"], [2, "c"], [3, "b"], [4, "a"]]	=> [**[4, "a"], [3, "b"], [2, "c"], [1, "d"]**]	
p.241	本文、下から2行目～1行目	順序は不定となることに注意しましょう。	**削除**	
p.243	「コード5-140 定義済みのSymbolオブジェクト一覧」4行目	=> [:_deprecated_default_system_source_cache_dir, ...（省略）	=> [**:freeze, :inspect, :intern,** ...（省略）	
p.243	コード5-142のキャプション	ディレクトリを開く	ディレクトリを開く**、閉じる**	

p.244	コード5-142の末尾	.....（省略）	.....（省略） **> dir.close** **=> nil**	
p.244	コード5-143のキャプション	ディレクトリを閉じる	ディレクトリ**が自動的に閉じられる場合**	
p.246	本文、下から4行目	「5-7-3.IOクラス」の項で	「5-**8**-3.IOクラス」の項で	
p.247	本文、上から3行目	「5-7-3.IOクラス」で	「5-**8**-3.IOクラス」で	
p.247	本文、下から3行目	「5-7-3.IOクラス」を	「5-**8**-3.IOクラス」を	
p.249	「コード5-153 ファイルの最終更新時刻を取得する」2行目	=> Thu Aug 18 22:35:49 0900 2011	=> **2011-08-18** 22:35:49 **+0900**	
p.249	「コード5-153 ファイルの最終更新時刻を取得する」4行目	=> Thu Aug 18 22:35:49 0900 2011	=> **2011-08-18** 22:35:49 **+0900**	
p.250	「表5-8 ファイルをテストするメソッド」1行目	File.exits?	File.**exist**?	
p.253	「コード5-164 コマンドの出力結果を得る」2行目	=> #<IO:0x100d42af0>	=> #<IO:**fd 11**>	
p.254	「コード5-166 標準出力に書きこむ」1行目	technology.'	technology.'	「'」を全角から半角に修正。
p.254	「コード5-167 ファイルを閉じる」2行目	"This is new README"	This is new README	
p.254	「コード5-168 パイプを開く」1行目	> IO.popen('grep -i ruby') do |io|	> IO.popen('grep -i ruby', **'r+'**) do |io|	
p.254	「コード5-168 パイプを開く」6行目	'This is Ruby program'	This is Ruby program	
p.255	上から10行目～14行目	それ以外は原則として、IOオブジェクトに内部エンコーディングが指定されている場合は外部エンコーディングから内部エンコーディングへの変換が行われ、内部エンコーディングの指定がない場合は外部エンコーディングで指定されたエンコーディングとなります。	**削除**	
p.257	下から1行目	整数を返します。	**文字列**を返します。	
p.258	「コード5-176 ファイルを読み取る（8）」4行目	=> 84　　# "T"	=> "T"	
p.258	「コード5-176 ファイルを読み取る（8）」6行目	=> 104　　# "h"	=> "h"	
p.259	「コード5-177 文字列を標準出力に出力する（1）」1行目	>STDOUT.write('There is new technology.')	> STDOUT.write('There is new technology.')	「>」と「STDOUT」間にスペースを挿入して、「technology.'」の「'」を全角から半角に修正。
p.260	本文、上から4行目	引数が0～255の整数の場合にはそれに対応する文字を、文字列の場合には先頭の1バイトを出力します。	**引数が整数の場合にはその最下位バイトを文字コードとする文字を、文字列の場合には先頭の1文字を出力します。**	
p.260	「コード5-181 文字列を標準出力に出力する（5）」末尾に追加	a => "a"	a => "a" **> STDOUT.putc(0x3042)** **B => 12354**	
p.261	「コード5-183 内部バッファをフラッシュする」1行目	, 'w+')	, 'w+')	「'」を全角から半角に修正。

p.264	「コード5-190 ファイルポインタを移動する（2）」10行目	=> "8¥n19¥n20¥n"	=> "**¥n18**¥n19¥n20¥n"	
p.265	本文、上から7行目	日本であれば0900と	日本であれば**+0900**と	
p.265	「コード5-191 Timeオブジェクトの生成（1）」2行目	=> Sun Aug 14 12:50:42 0900 2011	=> **2011-05-30** 12:50:42 **+0900**	
p.265	「コード5-192 Timeオブジェクトの生成（2）」2行目	=> Sat Feb 14 08:31:30 0900 2009	=> **2009-02-14** 08:31:30 **+0900**	
p.265	「コード5-192 Timeオブジェクトの生成（2）」4行目	=> Sat Feb 14 08:52:04 0900 2009	=> **2009-02-14** 08:52:04 **+0900**	
p.266	本文6～7行目	年に関しては、0から38の場合には2000を追加して、69以上138以下の場合には1900を追加した年と解釈されます※。	**削除**	
p.266	側注	※ ただしバージョン1.9以降ではこのような処理は行われずに、入力した年となることに注意してください。	**削除**	
p.266	「コード5-193 Timeオブジェクトの生成（3）」2行目	=> Sat Jan 01 00:00:00 0900 2011	=> **2011-01-01** 00:00:00 **+0900**	
p.266	「コード5-193 Timeオブジェクトの生成（3）」4行目	=> Thu Jul 07 00:00:00 0900 2011	=> **2011-07-07** 00:00:00 **+0900**	
p.266	「コード5-194 Timeオブジェクトの生成（4）」2行目	=> Thu Jul 07 00:00:00 0900 2011	=> **2011-07-07** 00:00:00 **+0900**	
p.266	「コード5-195 Timeオブジェクトの生成（5）」2行目	=> Sat Jan 01 00:00:00 UTC 2011	=> **2011-01-01** 00:00:00 UTC	
p.266	「コード5-195 Timeオブジェクトの生成（5）」4行目	=> Thu Jul 07 00:00:00 UTC 2011	=> **2011-07-07** 00:00:00 UTC	
p.267	「コード5-196 Timeオブジェクトの属性（1）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.268	「コード5-197 Timeオブジェクトの属性（2）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.268	「コード5-198 Timeオブジェクトの属性（3）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.268	「コード5-199 Timeオブジェクトの属性（4）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.269	「コード5-200 タイムゾーンを変更する（1）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.269	「コード5-200 タイムゾーンを変更する（1）」4行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.269	「コード5-200 タイムゾーンを変更する（1）」6行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-01** 18:04:05 **UTC**	
p.269	「コード5-201 タイムゾーンを変更する（2）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.269	「コード5-201 タイムゾーンを変更する（2）」6行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.270	「コード5-202 Timeオブジェクトの演算（1）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	

p.270	「コード5-202 Timeオブジェクトの演算（1）」4行目	=> Sun Jan 02 06:04:05 0900 2011	=> **2011-01-02** 06:04:05 **+0900**	
p.270	「コード5-202 Timeオブジェクトの演算（1）」6行目	=> Sun Jan 02 00:04:05 0900 2011	=> **2011-01-02** 00:04:05 **+0900**	
p.270	「コード5-203 Timeオブジェクトの演算（2）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.270	「コード5-203 Timeオブジェクトの演算（2）」4行目	=> Thu Feb 03 04:05:06 0900 2011	=> **2011-02-03** 04:05:06 **+0900**	
p.270	「コード5-204 Timeオブジェクトの変換（1）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.271	「コード5-204 Timeオブジェクトの変換（1）」6行目	1293905045.00001	1293905045.00000**6**	
p.271	「コード5-205 Timeオブジェクトの変換（2）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.271	「コード5-206 Timeオブジェクトの変換（3）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.271	「コード5-206 Timeオブジェクトの変換（3）」4行目	=> "Sun Jan 02 03:04:05 +0900 2011"	=> "**2011-01-02** 03:04:05 +0900"	
p.272	「コード5-207 Timeオブジェクトの変換（4）」2行目	=> Sun Jan 02 03:04:05 0900 2011	=> **2011-01-02** 03:04:05 **+0900**	
p.273	「コード5-209 正規表現オブジェクトを生成する（2）」2行目	=> abcdefgmi	=> **/abcdefg/mi**	
p.273	「コード5-210 正規表現オブジェクトでマッチングする（1）」2行目	=> abc	=> **/abc/**	
p.274	「コード5-211 正規表現オブジェクトでマッチングする（2）」2行目	=> abc	=> **/abc/**	
p.274	「コード5-211 正規表現オブジェクトでマッチングする（2）」4行目	=> #<MatchData "abc">	=> **0**	
p.274	「コード5-211 正規表現オブジェクトでマッチングする（2）」6行目	=> #<MatchData "abc">	=> **0**	
p.274	「コード5-212 正規表現オブジェクトでマッチングする（3）」2行目	=> abc	=> **/abc/**	
p.274	「コード5-213 正規表現オブジェクトでマッチングする（4）」2行目～3行目	> a = Regexp.new("abc") => abc	=> "abcdefg" > a = Regexp.new("abc") => **/abc/**	1行目と2行目の間に1行追加。2行目だった「=> abc」を「=> /abc/」に修正。
p.276	「コード5-217 正規表現の論理和」2行目	=> abc	=> **/abc/**	

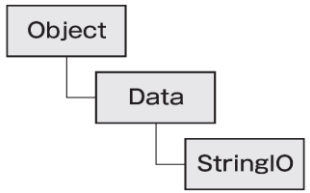
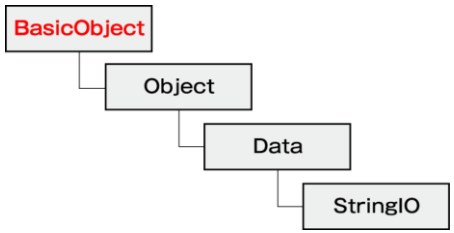
p.276	「コード5-217 正規表現の論理和」4行目	=> ABC	=> **/ABC/**	
p.276	「コード5-217 正規表現の論理和」6行目	=> -mix:abc-mix:ABC	=> **/(?**-mix:abc)|(**?**-mix:ABC)/	
p.276	中段の網掛け内の3行目	kcode	**encoding**	
p.276	「コード5-218 正規表現オブジェクトのオプションや属性を取得する（1）」2行目	=> abcdefgmi	=> /abcdefg/mi	
p.277	本文、上から1行目～2行目	casefold? メソッドはオプションRegexp::IGNORECASEが設定されているかどうかを返します。	casefold? メソッドはオプションRegexp::IGNORECASEが設定されているかどうかを返します。**Regexpオブジェクトのエンコーディングはencodingメソッドで取得します。**	
p.277	「コード5-219 正規表現オブジェクトのオプションや属性を取得する（2）」2行目	=> abcdefg	=> **/abcdefg/**	
p.277	「コード5-219 正規表現オブジェクトのオプションや属性を取得する（2）」6行目	=> abcdefgmi	=> /abcdefg/mi	
p.277	本文、上から3行目～6行目	kcodeは正規表現オブジェクトがコンパイルされている文字コードを、$KCODEと同じ形式で返します。マッチ時点での$KCODEを用いる場合など、特定の文字コードに対してコンパイルされていない場合にはnilが返ります。	**encodingは正規表現オブジェクトがコンパイルされている文字コードをEncodingオブジェクトとして返します。**	
p.277	「コード5-220 正規表現オブジェクトのオプションや属性を取得する（3）」差し替え	> a = Regexp.new("abcdefg") => abcdefg > a.kcode => nil > a = Regexp.new("abcdefg", nil, 'u') => abcdefgu > a.kcode => "utf8"	**> a = Regexp.new("ルビー")** **=> /ルビー/** **> a.encoding** **=> #<Encoding:UTF-8>** **> a = Regexp.new("ルビー".encode("EUC-JP"))** **=> /¥x{A5EB}¥x{A5D3}¥x{A1BC}/** **> a.encoding** **=> #<Encoding:EUC-JP>**	
p.278	「コード5-221 正規表現オブジェクトのオプションや属性を取得する（4）」1行目	Regexp:IGNORECASE, 'u')	Regexp**::IGNORECASE)**	
p.278	「コード5-221 正規表現オブジェクトのオプションや属性を取得する（4）」2行目	=> abcdefgmiu	=> **/abcdefg/mi**	
p.278	「コード5-221 正規表現オブジェクトのオプションや属性を取得する（4）」8行目	=> "/abcdefg/miu"	=> "/abcdefg/mi"	
p.278	「コード5-222 例外クラスを自作する」9行目	=> nil	=> **#<MyError: MyError>**	
p.279	「コード5-223 エラーメッセージを指定する」9行目	=> nil	=> **#<MyError: エラーが発生しました。>**	
p.279	「コード5-224 エラーメッセージを取得する」9行目	=> nil	=> **"エラーが発生しました。"**	

p.280	「コード5-225 バックトレースを取得する」9行目	=> nil	=> **["(irb):xxx:in `irb_binding'",…**	
p.280	「コード5-226 バックトレースを取得する」10行目	=> nil	=> **["This is new backtrace."]**	
p.281	本文、上から2行目	引数の値は、	引数の**数**は、	
p.282	「5-12-3. Procクラス以外の手続きオブジェクト」見出し	Procクラス以外の手続きオブジェクト	Proc.**new**以外の手続きオブジェクト**生成**	
p.282	下から4行目	Procクラス以外に	Proc.**new**以外に	
p.283	「コード5-232 手続きオブジェクトにおける引数の数」7行目	=> nil	**[1, 9, nil]**	
p.283	本文、下から1行目～4行目	まずブロック構文中やlambdaメソッド、procメソッドで生成した手続きオブジェクトでは、 breakを使うとその手続きオブジェクトを抜けますが、Proc.newで生成した手続きオブジェクトではLocalJumpErrorが発生します。	まず**lambdaメソッドで**生成した手続きオブジェクトでは、breakを使うとその手続きオブジェクトを抜けますが、**Proc.newやprocメソッドで生成した**手続きオブジェクトではLocalJumpErrorが発生します。	
p.284	上から1行目	ただしブロックに渡した場合には、どの手続きオブジェクトでもLocalJumpErrorが発生します。	**削除**	
p.284	「コード5-234 手続きオブジェクトの中でのジャンプ構文（2）」全体	> f = Proc.new { break } => #<Proc:0x0000000103796590@(irb):19> > 10.times(&f) LocalJumpError: break from proc-closure > g = lambda { break } => #<Proc:0x0000000103796590@(irb):19> > 10.times(&g) LocalJumpError: break from proc-closure	**削除**	
p.284	本文、上から3行目～5行目	次にreturnでは、lambdaメソッドとprocメソッドでの手続きオブジェクトの場合、その手続きオブジェクト自身を抜けます。一方、Procクラスの手続きオブジェクトでは、	次にreturnでは、**lambdaメソッドで生成した**手続きオブジェクトの場合、その手続きオブジェクト自身を抜けます。一方、**Proc.newやprocメソッドで生成した**手続きオブジェクトでは、	
p.284	「コード5-235 手続きオブジェクトの中でのジャンプ構文（3）」差し替え	> f = Proc.new { return } => #<Proc:0x00000001038161c8@(irb):27> > f.call LocalJumpError: unexpected return > g = lambda { return } => #<Proc:0x00000001037fc0e8@(irb):29> > g.call => nil	> f = Proc.new { return } **=> #<Proc:0x007fd907fce558@5-235.rb:1>** > f.call LocalJumpError: unexpected return **> def foo** **> Proc.new {** **> return 1** **> }.call** **> return 2** **> end** **=> :foo** **> foo** **=> 1** > g = lambda { return } **=> #<Proc:0x007fd907fa8740@5-235.rb:10 (lambda)>** > g.call => nil	

p.285	「コード5-236 モジュールを使った機能の追加」6行目	=> nil	=> **:foo**	
p.285	「コード5-237 定数の一覧を取得する（1）」2行目	=> ["String", "Dir", "Math", "RUBY_RELEASE_DATE", ...(省略)	=> [**:Object, :Module, :Class, :BasicObject, :Kernel,** ...(省略)	
p.286	「コード5-238 定数の一覧を取得する（2）」全体	> Array.constants => ["Enumerator"]	**> class MyClass** **> FOO = 1** **> end** **=> 1** **> MyClass.constants** **=> [:FOO]**	
p.286	「コード5-240 定数の値を取り出す」2行目	=> "2.1.0"	=> "**2.1.9**"	
p.287	本文、上から1行目～2行目	ancestorsメソッドは、そのクラスやモジュールの祖先クラスとインクルードしているモジュールの一覧を返します。	**削除**	
p.287	「コード5-243 祖先クラスを取得する」全体	> Array.ancestors => [Array, Enumerable, Object, Kernel]	**削除**	
p.287	「コード5-244 インスタンスに設定されているメソッドの一覧」2行目	=> ["zip", "last", "find_index", "fill", "sort!", ...(省略)	=> [**:inspect, :to_s, :to_a, :to_h, :to_ary,** ...(省略)	
p.288	「コード5-246 インスタンス属性として設定」4行目	=> MyClass	=> **nil**	
p.289	本文、上から3～4行目	文字列で指定できる	文字列**かシンボル**で指定できる	
p.289	「コード5-247 メソッドの別名を定義する」10行目	=> nil	=> **:foo**	
p.289	「コード5-247 メソッドの別名を定義する」14行目	=> "barfoo"	=> "**bar foo**"	
p.290	「コード5-248 モジュールのコンテキストで評価する（1）」6行目	=> nil	=> **:foo**	
p.290	下から2行目	class_variableメソッド	class_variable**s**メソッド	
p.291	「コード5-250 クラス変数の一覧」6行目	=> ["@@foo"]	=> [**:@@foo**]	
p.291	「コード5-252 クラス変数の取得や設定」8行目	=> nil	=> **:get**	
p.291	「コード5-252 クラス変数の取得や設定」9行目	> def MyClass.set=(var)	> def MyClass.set(var)	
p.291	「コード5-252 クラス変数の取得や設定」12行目	=> nil	=> **:set**	
p.292	「コード5-252 クラス変数の取得や設定」16行目	=> nil	=> **:clear**	
p.292	「コード5-252 クラス変数の取得や設定」23行目	> MyClass.set = 'newvar'	> MyClass.set**('newvar')**	

p.293	「コード5-253 クラスやインスタンスにモジュールの機能を取り込む」6行目	=> nil	=> **:foo**	
p.293	「コード5-254 インクルードしたときに実行するメソッド」6行目	=> nil	=> **:included**	
p.295	本文、最下行	Module.ancestorsメソッドを	Module#ancestorsメソッドを	
p.295	「コード5-258 祖先クラスを取得する」2行目	=> [Array, Enumerable, Object, Kernel]	=> [Array, Enumerable, Object, Kernel, **BasicObject**]	
p.300	本文、1行目～2行目	firstメソッドは先頭の要素か、引数があれば先頭からその数だけ要素を返します。	**削除**	
p.300	コード5-270のキャプション	countメソッドとfirstメソッド	countメソッド	
p.300	「コード5-270」3行目から6行目	> [1, 2, 3, 4, 5].first => 1 > [1, 2, 3, 4, 5].first(3) => [1, 2, 3]	**削除**	
p.301	「コード5-272 group_byメソッド」2行目	=> {0=>[2, 4, 6, 8, 10], 1=>[1, 3, 5, 7, 9]}	=> {**1=>[1, 3, 5, 7, 9], 0=>[2, 4, 6, 8, 10]**}	
p.301	「コード5-275 take_whileメソッド」1行目	> [:a, :b, :c, :d, :e].take { |e| e != :d }	> [:a, :b, :c, :d, :e].**take_while** { |e| e != :d }	
p.303	本文、最下行	例えば以下の様なSampleクラスを考えます。	例えば以下の様なSampleクラスを考えます。 **このSampleクラスは通常の大小関係とは逆の挙動をするように実装されています。**	
p.303	「コード5-280 Sampleクラスの定義」6行目	* def value	> def value	
p.304	「コード5-280 Sampleクラスの定義」10行目	* def <=>(other)	> def <=>(other)	
p.304	「コード5-280 Sampleクラスでの比較の例」11行目	> self.value <=> other.value	> **other.value** <=> **self.value**	
p.304	本文、1行目	この場合のそれぞれの～	**このクラスにComparableモジュールをインクルードした場合、**それぞれの～	
p.304	「コード5-281 Sampleクラスでの比較の例」4行目	* def initialize(value)	> def initialize(value)	
p.304	「コード5-281 Sampleクラスでの比較の例」8行目	* def value	> def value	
p.304	「コード5-281 Sampleクラスでの比較の例」12行目	* def <=>(other)	> def <=>(other)	
p.305	箇条書きの下から4行目	・クラスメソッドを定義したオブジェクト	・**特異メソッド**を定義したオブジェクト	
p.305	「コード5-282 Rubyオブジェクトを文字列化する（1）」2行目	=> "¥004¥b{¥b:¥006ci¥n:¥006ai¥006:¥006bi¥b"	=> **"¥x04¥b{¥b:¥x06ai¥x06:¥x06bi¥b:¥x06ci¥n"**	
p.306	「コード5-284 Rubyオブジェクトを復元する（1）」2行目	=> "¥004¥b{¥b:¥006ci¥n:¥006ai¥006:¥006bi¥b"	=> **"¥x04¥b{¥b:¥x06ai¥x06:¥x06bi¥b:¥x06ci¥n"**	
p.306	「コード5-284 Rubyオブジェクトを復元する（1）」4行目	=> {:c=>5, :a=>1, :b=>3}	=> {**:a=>1, :b=>3, :c=>5**}	
p.306	「コード5-285 Rubyオブジェクトを復元する（2）」8行目	=> {:c=>5, :a=>1, :b=>3}	=> {**:a=>1, :b=>3, :c=>5**}	
p.306	下から5行目	ネイティブスレッド	**Rubyコード**	

p.306 〜307	p.306 最下行〜p.307 上から1行目	優先順位付きのラウンドロビン・スケジューリングという方式で管理されています*。	**管理方法はプラットフォームに依存しています。**	
p.307 〜314	コード5-286〜5-300にかけて	※コード中のスレッドの状態出力全般 => #<Thread:0x..... sleep>	=> #<Thread:0x..... **run**>　※補足をお読みください	実行するタイミングに依存。sleepとなることもあるため厳密には誤りではありませんが、runとなるケースが多いでしょう。
p.309	上段の網掛け内の2行目	kill! / exit!	**削除**	
p.309	本文、上から2行目〜4行目	killメソッドと exitメソッド、 kill!メソッドと exit!メソッドはともに、スレッドを終了します。メソッドの末尾に「!」がある場合は、次で説明するようにensure節の実行するかどうかの違いです。	killメソッドと exitメソッドはともに、スレッドを終了します。	
p.309	「コード5-290 スレッドの終了」の4行目	=> #<Thread:0x10c084e38 dead>	=> #<Thread:0x10c084e38 **run**> **> t** **=> #<Thread:0x10c084e38 dead>**	
p.309	本文、下から3行目〜1行目	ただしkill!メソッドが呼ばれたときには実行されません。 メインスレッドのみkill!メソッドが呼ばれても実行されます。	**削除**	
P.309 〜310	「コード5-291 ensure節の実行」の8行目以降	=> #<Thread:0x10c227678 sleep> > t.kill Killed => #<Thread:0x10c227678 dead>	=> #<Thread:0x10c227678 **run**> > t.kill Killed => #<Thread:0x10c227678 **sleep**> **> t** **=> #<Thread:0x10c227678 dead>**	
p.310	「コード5-292 スレッド中の例外」10行目	Raise exception => #<Thread:0x107925d08 sleep>	**=> #<Thread:0x107925d08 run> Raise exception**	
p.311	「コード5-294 スレッドのデッドロック」4〜6行目	deadlock 0x1038c9868: sleep:- - (irb):xxx deadlock 0x1035a0268: sleep:- (main) - (irb):xxx fatal: Thread(0x1035a0268): deadlock	**fatal: No live threads left. Deadlock?**	
p.312	「コード5-296 スレッドのリスト」最下行	=> [#<Thread:0x107f838d0 sleep>, #<Thread:0x107c71278 run>]	**=> [#<Thread:0x107c71278 run>, #<Thread:0x107f838d0 sleep>]**	
p.313	「コード5-299 スレッドの終了を待つ」1〜2行目	> Thread.critical => false	**削除**	
p.315	「コード5-303 データ名の一覧」最下行	=> [:foo, :__recursive_key__]	**=> [:__recursive_key__, :foo]**	
p.316	「コード5-304 ファイバーにおけるコンテキストの切り替え」8行目	=> #<Fiber:0x007fe6d5b7f0e8>	=> #<Fiber:0x007f**8972fd02e8**>	先頭に半角スペース挿入。

p.316	「コード5-304 ファイバーにおけるコンテキストの切り替え」13行目以降～差し替え	hello' child -> parent parent -> child 'hello' child -> parent parent -> child 'hello' child -> parent parent -> child => 3	**parent -> child** **hello** **child -> parent** **parent -> child** **hello** **child -> parent** **parent -> child** **hello** **child -> parent** **=> 3**	
p.322	「図6-1 StringIOの継承ツリー」内			「Object」の左上に「BasicObject」を追加。
p.323	「図6-2 StringIOのバッファ（string）とポインタ（pos）」内下から2行目	ポイントが移動し、	ポイン**タ**が移動し、	
p.324	「表6-2 インスタンス作成時に利用できるモード一覧」行「w」の「説明」	空文字となる。	空文字**列**となる。	
p.324	「コード6-3 openメソッドによるStringIOインスタンスの作成」6行目	p sio => #<StringIO:0x100609ae0>	p sio **#**=> #<StringIO:0x100609ae0>	
p.324	「コード6-3 openメソッドによるStringIOインスタンスの作成」9行目	p io.read #=> Hello StringIO.	p io.read #=> **"**Hello StringIO.**"**	先頭のスペースが全角なので半角に修正。
p.325	「コード6-4 putcメソッドによるバッファへの書き込み」4行目	sio.putc "a" #=> a	sio.putc "a" #=> **"a"**	
p.325	「コード6-4 putcメソッドによるバッファへの書き込み」5行目	sio.string #=> a	sio.string #=> **"a"**	
p.325	「コード6-4 putcメソッドによるバッファへの書き込み」10行目	sio.string #=> b	sio.string #=> **"b"**	
p.326	本文、上から8行目～11行目	ただし、nilに対しては文字列"nil"を出力します。 printメソッドは末尾に改行を付加しない点を除き、putsメソッドと同じ動きをします。	**printメソッドはputsメソッドと異なり、末尾に改行を付加しません。**	
p.326	「コード6-5 putsメソッド、printメソッドによるバッファへの書き込み」2行目	require 'stringio' sio.puts "abc" #=> nil	require 'stringio' **sio = StringIO.new** sio.puts "abc" #=> nil	2行目の空行に追加。

p.328	「コード6-6 readメソッドによるバッファからの文字の取り出し」2行目	require 'stringio' sio.string = "Hello World."	require 'stringio' **sio = StringIO.new** sio.string = "Hello World."	2行目の空行に追加。
p.328	本文、上から1行目～2行目	文字に対応するFixnumインスタンスを返します。	**Stringオブジェクトで**返します。	
p.328	本文、上から4行目～5行目	Fixnumインスタンスを返しますが、	**Stringオブジェクトで**返しますが、	
p.328	本文、最下行	例外EOFErrorが	EOFErrorが	
p.328	「コード6-7 getcメソッド、readcharメソッドによる文字の取り出しと動きの違い」2行目	require 'stringio' sio.string = "Hello World."	require 'stringio' **sio = StringIO.new** sio.string = "Hello World."	2行目の空行に追加。
p.328	「コード6-7 getcメソッド、readcharメソッドによる文字の取り出しと動きの違い」4行目	sio.getc #=> 72	sio.getc #=> **"H"**	
p.329	「コード6-8getsメソッド、readlineメソッドによる文字の取り出しと挙動の違い」2行目	require 'stringio' sio.string = "Hello World.¥nHello IOString World."	require 'stringio' **sio = StringIO.new** sio.string = "Hello World.¥nHello IOString World."	2行目の空行に追加。
p.329	「コード6-8getsメソッド、readlineメソッドによる文字の取り出しと挙動の違い」4行目	sio.gets #=> Hello World.¥n	sio.gets #=> **"Hello World.¥n"**	
p.329	「コード6-8getsメソッド、readlineメソッドによる文字の取り出しと挙動の違い」5行目	sio.readline #=> Hello IOString World.	sio.readline #=> **"Hello IOString World."**	
p.330	「コード6-9 pos=メソッド、seekメソッドによるポインタの移動」3行目	require 'stringio' sio.string = "ABCDEFHIJKLMNOPQRSTUVWXYZ"	require 'stringio' **sio = StringIO.new** sio.string = "ABCDEF**G**HIJKLMNOPQRSTUVWXYZ"	2行目の空行に追加。
p.333	「コード6-13 YAML.load_streamメソッドによるYAMLデータの読み込み」5行目	p yaml_stream.documents	p yaml_stream	
p.334	「コード6-15 YAML.dumpメソッドによるYAMLデータの書き出し」4行目	YAML.dump colors #=> "--- ¥n- Red¥n- Green¥n- Blue¥n"	YAML.dump colors #=> "**---¥n-** Red¥n- Green¥n- Blue¥n"	半角スペースを削除。
p.334	「コード6-15 YAML.dumpメソッドによるYAMLデータの書き出し」7行目	# 引数ioを指定した場合	# 引数ioを指定した場合（**ファイル出力のため書き込み可能オプションでファイルをオープン**）	
p.334	「コード6-15 YAML.dumpメソッドによるYAMLデータの書き出し」8行目	YAML.dump colors, File.open("sample.yml", "w+") # ファイル出力のため書き込み可能オプションで作成	**File.open("sample.yml", "w+") do |f|** **YAML.dump(colors, f)** **end**	
p.335	「コード6-16 YAML.dump_streamによるYAMLデータの書き出し」ページ上から2行目	#=> "--- ¥n- Red¥n- Green¥n- Blue¥n--- ¥n- Yellow¥n- Pink¥n- White¥n"	#=> "**---¥n-** Red¥n- Green¥n- Blue**¥n---¥n-** Yellow¥n- Pink¥n- White¥n"	半角スペースを削除。
p.337	「コード6-22 オブジェクトをJSON形式に変換し、ファイルに出力する」4行目	f = File.open("dump.json", "w")	File.open("dump.json", "w") **do |f|**	

p.337	「コード6-22 オブジェクトをJSON形式に変換し、ファイルに出力する」5行目	JSON.dump(array, f) # => <File:dump.json>	JSON.dump(array, f) **end**	「JSON.dump(array, f)」の先頭にスペースを追加。
p.339	「読み込みの可能なメソッド」の1つ目のコード 1行目	foreach	**削除**	
p.340	「読み込みの可能なメソッド」の2つ目のコード 1行目	read/readlines	**削除**	
p.340	「コード6-25 CSV.readメソッド,、CSV.readlinesメソッドによるCSVデータの読み込み」見出し	CSV.readメソッド,、	CSV.readメソッド、	
p.340	「コード6-25 CSV.readメソッド,、CSV.readlinesメソッドによるCSVデータの読み込み」13行目	p csv_table	csv_table	
p.343	「mkdir / mkfir_p」見出し	mkdir / mkfir_p	mkdir / mk**dir**_p	
p.345	側注	に-enable-socksの	に**--**enable-socksの	
p.345	「図6-3 socketライブラリのクラス」内	SOCKSSocket	SOCKSSocket*****	側注のアスタリスク追記。
p.348	「コード6-28 UDPSocketを使って10000ポートでUDP通信を待ち受ける」6行目	socket.recv (MAX_PACKET)	socket.recv(MAX_PACKET)	「socket.recv」と「(MAX_PACKET)」の間のスペースを削除。
p.349	上段ボックス内の1行目	1. connectメソッドを使用して、UDPサーバに接続してからsendメソッドを使用する方法	1. **connectメソッドで接続先ホストとポートを指定して、**sendメソッドを使用する方法	
p.349	「コード6-29 UDPSocketを使って10000ポートに対してUDPでデータを送信する」7行目	# サーバー側コンソールににHello UDP.と表示されます	# サーバー側コンソール**に**Hello UDP.と表示されます	
p.349	下から2行目（見出し内）	UnixSocket	**UNIX**Socket	
p.353	「コード6-32 URI.parseメソッドでURLをパースする」7行目	#=> "80"	#=> **80**	
p.353	「コード6-33 URI.parseで正しくパースできない例」5行目	URI::InvalidURIError:	#=> URI::InvalidURIError:	
p.354	「escape / encode」の見出し	escape / encode	**encode_www_form_component**	
p.354	「escape / encode」のコード	URI.escape(str, [, unsafe]) URI.encode(str, [, unsafe])	**URI.encode_www_form_component(str)**	
p.354	本文、上から5行目～7行目	URI.escapeメソッドは、引数strで指定したURI文字列をエンコードした文字列で返します。encodeメソッドは、escapeメソッドのエイリアスメソッドです。	URI.**encode_www_form_component**メソッドは、引数strで指定した文字列をエンコードした**文字列をURLエンコードした文字列で**返します。	
p.354	本文、上から10行目	文字列をNKFなどで変換する必要があります。	文字列を**変換しておく**必要があります。	
p.354	本文、下から3行目～1行目	引数のunsafeには、URIに使用できない文字を正規表現か文字列で指定します。デフォルトでは、URI : : UNSAFEに定義されたものが使用されるので、特別な用途以外では指定する必要はありません。	**削除**	
p.354	「コード6-35 URI.escapeでURLをエンコードする」見出し	URI.escapeでURLをエンコードする	URLをエンコードする	
p.354	「コード6-35 URI.escapeでURLをエンコードする」4行目	p URI.escape "http://www.example.com/Ruby技術者試験対策教科書"	p **"http://www.example.com/" + URI.encode_www_form_component("Ruby技術者試験対策教科書")**	

p.355	「コード6-35 URI.escapeでURLをエンコードする」9行目	p URI.escape "http://www.example.com/Ruby技術者試験対策教科書".encode("EUC-JP")	p **"http://www.example.com/" + URI.encode_www_form_component("Ruby技術者試験対策教科書".encode("EUC-JP"))**	
p.355	「unescape / decode」の見出し	unescape / decode	**decode_www_form_component**	
p.355	「unescape / decode」のコード	URI.unescape(str) URI.decode(str)	**URI.decode_www_form_component(str)**	
p.355	本文、上から1行目～2行目	unescapeメソッドは、URLエンコードされた文字列を元の文字列に戻します。 decodeメソッドはunescapeのエイリアスメソッドです。	**decode_www_form_component**メソッドは、URLエンコードされた文字列を元の文字列に戻します。	
p.355	「コード6-36 URI.decodeでURLをデコードする」見出し	URI.decodeでURLをデコードする	URLをデコードする	
p.355	「コード6-36 URI.decodeでURLをデコードする」3行目	escaped_uri =	uri =	
p.355	「コード6-36 URI.decodeでURLをデコードする」6行目	p URI.decode(escaped_uri)	**parsed_uri = URI.parse(uri)** **p URI.decode_www_form_component(parsed_uri.path[1..-1])**	
p.355	「コード6-36 URI.decodeでURLをデコードする」7行目	#=> http://www.example.com/Ruby技術者試験対策教科書	#=> Ruby技術者試験対策教科書	
p.357	下から2行目	戻り値にはHTTPRequestクラスが返ります。	**Net::HTTPResponseのサブクラス（Net::HTTPOKクラス、Net::HTTPNotFoundクラスなど）のインスタンスが**返ります。	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」3行目～4行目	$KCODE = "UTF8" res = nil	**削除**	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」7行目	net = Net::HTTP.new("www.ruby-lang.org")	net = Net::HTTP.new("**docs.**ruby-lang.org")	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」9行目	res = net.get("/ja/documentation/")	res = net.get("/ja**/2.1.0/doc/index.html**")	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」15行目	net = Net::HTTP.new("www.ruby-lang.org")	net = Net::HTTP.new("**docs.**ruby-lang.org")	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」17行目	res = http.get("/ja/documentation/")	res = http.get("/ja**/2.1.0/doc/index.html**")	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」19行目	print res.body	print res	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」22行目	res = Net::HTTP.get("www.ruby-lang.org", "/ja/documentation/")	**body** = Net::HTTP.get("**docs.**ruby-lang.org", "/ja**/2.1.0/doc/index.html**")	
p.358	「コード6-37 Net::HTTPを使用した、HTMLの取得サンプル」23行目	print res.body	print **body.force_encoding("UTF-8")**	
p.360	「コード6-39 BASIC認証用のヘッダを作成する」5行目	#=> ["Basic cnVieXVzZXI6cGFzc3dvcmQ="]	#=> "Basic cnVieXVzZXI6cGFzc3dvcmQ="	

p.360	本文、下から7行目	Net::HTTPResponseのインスタンスが返ります。	Net::HTTPResponseの**サブクラスの**インスタンスが返ります。	
p.360	本文、下から6行目	HTTPResponseインスタンスには、	**戻り値**には、	
p.361	「コード6-40 ステータスコード302のHTTPResponseインスタンスら、正しいURIを取得する」7行目	#=> "302"	#=> **302**	
p.362	「コード6-41 Dateクラスの減算」1行目	require 'datetime'	require '**date**'	
p.362	「コード6-41 Dateクラスの減算」5行目	#=> Rational(326, 1)	#=> **(326/1)**	
p.364	「コード6-42 parseを使用した文字列からDateインスタンスへの変換」の7行目	#=> "0011-11-01"	#=> "0011-**01**-01"	
p.365	「today」の1行目	Date.today([start])	Date.today(**start = Date::ITALY**)	
p.366	「コード6-44 stepを使って7月1日から7月31日までを7日毎に出力する」2行目	start_from = Date.new(2011, 07, 01)	start_from = Date.new(2011, **7**, **1**)	
p.366	「コード6-44 stepを使って7月1日から7月31日までを7日毎に出力する」3行目	end_to = Date.new(2011, 07, 31)	end_to = Date.new(2011, **7**, 31)	
p.367	「civil / new」コード差し替え	DateTime.civil([year[, mon[, mday[, hour[, min[, sec[, offset[, start]]]]]]]]) DateTime.new([year[, mon[, mday[, hour[, min[, sec[, offset[, start]]]]]]]])	DateTime.civil(**year=-4712, mon=1, mday=1, hour=0, min=0, sec=0, offset=0, start=Date::ITALY**) DateTime.new(**year=-4712, mon=1, mday=1, hour=0, min=0, sec=0, offset=0, start=Date::ITALY**)	
p.367	「コード6-45 DateTimeインスタンスの作成」4行目	DateTime.new(2011, 7, 1, 12, 0, 0, Rational(9/24)).to_s	DateTime.new(2011, 7, 1, 12, 0, 0, Rational(**9, 24**)).to_s **#=> "2011-07-01T12:00:00+09:00"** **# offsetを文字列で渡す** **DateTime.new(2011, 7, 1, 12, 0, 0, "+0900").to_s**	4行目の修正して、その後に3行分を追加。
p.368	「コード6-46 DateTime#zoneとDateTime#offsetの動きの違い」3行目	time = DateTime.new(2011, 7, 1, 0, 0, 0, Rational(9, 24))	time = DateTime.new(2011, 7, 1, 0, 0, 0, **"+0900"**)	
p.368	「コード6-46 DateTime#zoneとDateTime#offsetの動きの違い」8行目	#=> Rational(3, 8)	#=> **(3/8)**	
p.369	「コード6-47 Thread.exclusiveメソッドの有無による動きの比較」22行目	5.times{|num| print "t"; sleep 1}	5.times{ print "t"; sleep 1}	
p.370	上から2行目	Mutal Excluison	Mutex；MUTual EXclusion	
p.372	本文、下から7行目	nilが返ります。	**ThreadErrorが発生します。**	
p.373	「コード6-49 テスト対象のプログラム」7行目	return "Hello #{name}"	return "Hello #{name}**.**"	
p.375	「コード6-51 コード6-49のFooメソッドを書き換える」見出し	Fooメソッドを	Foo**クラス**を	

p.375	「コード6-51 コード6-49のFooメソッドを書き換える」7行目	return "Hello #{name}"	return "Hello #{name}."	
p.378	本文、上から1行目	検証対処の	検証**対象**の	
p.379	「flunk」の1行目	flunk(message=""Flunked"")	flunk(message="Flunked")	
p.381	「見出し」の図			2行目以降の見出し「Head」の番号を1ずつ増やすよう訂正。
p.384	「実行結果」2行目	{"output"=>true, "input"=>true}	{**"input"**=>true, **"output"**=>true}	
p.385	「banner、width、indentを指定したときの--helpオプション呼び出し時の動作」2行目	Option Parser Test	Option Parser Test **Program**	
p.387	「コード6-57 onメソッド・ロングオプションのサンプルコード」3行目	with Args	with Arg	
p.387	「コード6-57 onメソッド・ロングオプションのサンプルコード」の「実行結果」	# ruby optparse_test.rb -xvalue=10	# ruby optparse_test.rb --xvalue=10 **"10"**	1行目の後に追加。
p.404	「問題2」コード3行目	3: print x. + " "	3: print x.**to_s** + " "	
p.404	「問題2」選択肢1〜2行目	1.「1 2 3 3」 2.「1 2 3 0」	1.「1 2 3 3」**と表示される。** 2.「1 2 3 0」**と表示される。**	
p.405	「問題4」コード2行目	y [x] raise "failed"	y [x] (raise "failed")	
p.419	「問題38」問題文	数字のみで構成される行に〜	**一つ以上の数字のみで**構成される行に〜	
p.426	「問題2」解説下から3行目	値を代入する前のxを参照することになります。値を代入する前の変数を参照すると、例外が発生します。	**定義されていない変数**xを参照することになります。**定義されていない**変数を参照すると、例外が発生します。	
p.428	「問題11」解説下から2行目	他に、自己代入演算子（+=など）、**!=、!~**もオーバーライドできません。	他に、自己代入演算子（+=など）もオーバーライドできません。	
p.431	「問題23」解説2行目	**配列に**アクセスする位置と〜	アクセスする位置と〜	
p.441	「問題36」解説	正規表現/o../ は、oで始まる任意の3文字にマッチします。sliceは、引数で指定した正規表現にマッチした文字列を返します。出題コードの/o../ は、ogeにマッチします。	**sliceの引数に正規表現を指定すると、そのパターンに最初にマッチした文字列を返します。正規表現/o../は、oで始まる任意の3文字にマッチしますので、ogeが返ります。**	どこがマッチしたのかをより厳密に示すよう訂正。
p.441	「問題38」解説1行目	「数字のみで構成される行」とは	**「一つ以上の数字のみで構成される行」**とは	
p.448	「問題6」選択肢	x ※選択肢1〜4のすべてについて（計4ヵ所）	|x|	ボックスのxではなく、実際のコードとしての|x|に
p.448	「問題7」実行結果	Hello, World.	Hello, World.</**p**>	半角スペースを削除。
p.449	「問題8」実行結果	[1,2,3]	**[1, 2, 3]**	カンマのあとに半角スペースが入る。

p.450	「問題10」コード2行目	p hi.call("World")	hi.call("World")	
p.454	「問題17」コード3行目	class Cls1	class Cls1	先頭のスペースを削除。
p.458	「問題25」実行結果	"foo"	foo	
p.459	「問題28」実行結果	2000/11/10	2000-11-10	
p.469	「問題16」コード1行目	require "Date"	require **'date'**	
p.471	「問題20」コード1行目	tag = ->(|t, msg|	tag = ->(t, msg){	
p.472	「問題20」実行結果	Hello, World	<**p**>Hello, World</**p**>	
p.477	「問題29」コード2行目~4行目	puts a puts b puts c	**p** a **p** b **p** c	
p.490	「問題5」解説	コンパイルエラー	**文法エラー**	
p.491	「問題10」の解答	解答：1	解答：**3**	
p.497	「問題5」解説	attrはattr_readerと同義で、属性を変更するメソッドは定義できません。	**attrはattr_readerと同じで、属性の値を参照するメソッドを作成します。**	

以上