15. Light Weight Language に関する知識 I

1. 科目の概要

軽量プログラミング言語(Light Weight Language)と呼ばれる言語について解説する。代表的な Light Weight Language である Perl、PHP、Python、Ruby を紹介し、その特徴と基本的な構文を説明する。さらに Ruby を利用したオブジェクト指向プログラミングや Ruby 特有の項目を掘り下げて解説する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説　明	シラバスの対応コマ
I-15-1. Light Weight Languageの歴史や特徴、種類、使い方	Light Weight Languageの概要を紹介し、その歴史、特徴、用途、代表的なLight Weight Languageの種類などについて概説する。さらにLight Weight Languageを利用した開発の流れについて述べ、プログラミングから実行までの手順を示す。	1
I-15-2. Perlの特徴と正規表現	古くから利用されているPerlの概要と特徴について触れ、Perlの動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明し、Perlを特徴付ける「正規表現」について解説する。	2
I-15-3. PHPの特徴と埋め込みプログラミング	Webアプリケーションのロジック記述に利用されることの多いPHPの概要と特徴について触れ、PHPの動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明し、PHPを特徴付ける「埋め込みプログラミング」を解説する。	3
I-15-4. Pythonの特徴と連想配列	設定ファイルや機能拡張言語のベースとしてしばしば利用されるPythonの概要と特徴について触れ、Pythonの動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明、Pythonを特徴付ける「連想配列」を解説する。	4
I-15-5. Rubyの特徴、基本的な構造、型、演算子、制御構文	オブジェクト指向プログラミングを特徴とするRubyの概要と特徴について触れ、Rubyの動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、変数、演算子、制御構文などについて解説する。	5
I-15-6. Rubyによるオブジェクト指向プログラミング	Rubyを題材として、オブジェクト指向プログラミングの基本について解説する。オブジェクト指向の概念を説明し、クラスやメソッドの定義、継承、カプセル化、ポリモルフィズムといった考え方がRubyでどのように表現されるかを示す。	5
I-15-7. Rubyが持つ特徴的なデータクラス(配列、ハッシュ等)	Rubyに組み込まれている配列やハッシュなどの汎用データ型クラスについて説明する。配列、ハッシュ、構造化クラスの概要を解説し、各クラスの構成や使い方を紹介する。	7
I-15-8. Rubyにおけるデータ(数値、文字列、その他)の操作	Rubyに組み込まれている数値や文字列、日付などの固有なデータ操作に特化したクラスについて説明する。数値クラス、文字列クラス、その他特有のクラスについて、その概要を解説し、各クラスの構成や使い方を紹介する。	8
I-15-9. Rubyによるファイル操作のプログラミング	Rubyに組み込まれているファイル入出力やファイル/ディレクトリ操作などのファイル管理に特化したクラスについて説明する。ファイルクラス、IOクラス、その他ファイル操作に関連するクラスについて、その概要を解説し、各クラスの構成や使い方を紹介する。	9
I-15-10. Rubyを用いたGUIアプリケーション開発	GUIアプリケーションのインタフェースを構成するウィジェットやイベント処理について説明し、RubyでGUIアプリケーションを開発する際に利用するライブラリであるRuby/TkやRuby-GNOME2の概要と使い方を説明する。	10

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「15. Light Weight Language に関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）										応用レベル（Ⅱ）				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15. Light Weight Languageに関する知識	<LightWeight Languageの基本>	<Perlの基本構造>	<PHPの基本構造>	<Pythonの基本構造>	<Rubyの基本構造>	<オブジェクト指向プログラミング>	<組み込みクラス[データ構造]>	<組み込みクラス[データ操作]>	<組み込みクラス[ファイル管理]>	<GUIアプリケーション開発>	<Ruby on Rails>	<データベースアプリケーション開発>	<Webアプリケーション開発>	<プラグイン導入と開発>	<オープンソースシステムのカスタマイズ>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_15.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング [15-Ⅰ-6]	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法 [15-Ⅰ-1, 2, 3, 4, 5]	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要 [15-Ⅰ-1]						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計 [15-Ⅰ-6]	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング [15-Ⅰ-2]	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャ	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPUアーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、Ruby に組み込まれたデータ構造と Ruby による GUI アプリケーション開発がある。ここで扱うデータ構造や Ruby のクラスの基本的な考え方は他の言語とも共通している。

科目名	第1回	第2回	第3回	第4回	第5回	第6回	第7回	第8回	第9回	第10回
15.Light Weight Language に関する知識 I	(1) Light Weight Language の説明 (2) LightWeight Language による開発の流	(1) Perl の書き方の特徴 (2) 基本的なプログラム記述の例	(1) PHP の書き方の特徴 (2) 基本的なプログラム記述の例	(1) Python の書き方の特徴 (2) 基本的なプログラム記述の例	(1) Ruby の書き方の特徴 (2) 基本的なプログラム記述の例	(1) オブジェクト指向の説明 (2) オブジェクト指向プログラミングの説明	(1) データ構造の説明 (2) 配列の説明 (3) ハッシュの説明 (4) 構造化クラスの説明	(1) データ操作の説明 (2) 数値の説明 (3) 文字列の説明 (4) その他データ操作のクラスの紹介	(1) ファイル管理の説明 (2) ファイル情報の説明 (3) ファイル入出力の説明 (4) ファイル／ディレクトリ操作の説明	(1) Ruby/Tk によるGUI アプリケーション開発 (2) Ruby-GNOME2 によるGUI アプリケーション (3) GUI ライブラリの特徴と比較

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-1. Light Weight Language の歴史や特徴、種類、使い方	
対応する コースウェア	第 1 回 (Light Weight Language の基本)	

I-15-1. Light Weight Language の歴史や特徴、種類、使い方

Light Weight Language の概要を紹介し、その歴史、特徴、用途、代表的な Light Weight Language の種類などについて概説する。さらに Light Weight Language を利用した開発の流れについて述べ、プログラミングから実行までの手順を示す。

【学習の要点】

＊ Light Weight Language は比較的小規模なプログラミングに適している。

＊ Web アプリケーション開発の増大に伴って、Light Weight Language が脚光を浴びている。

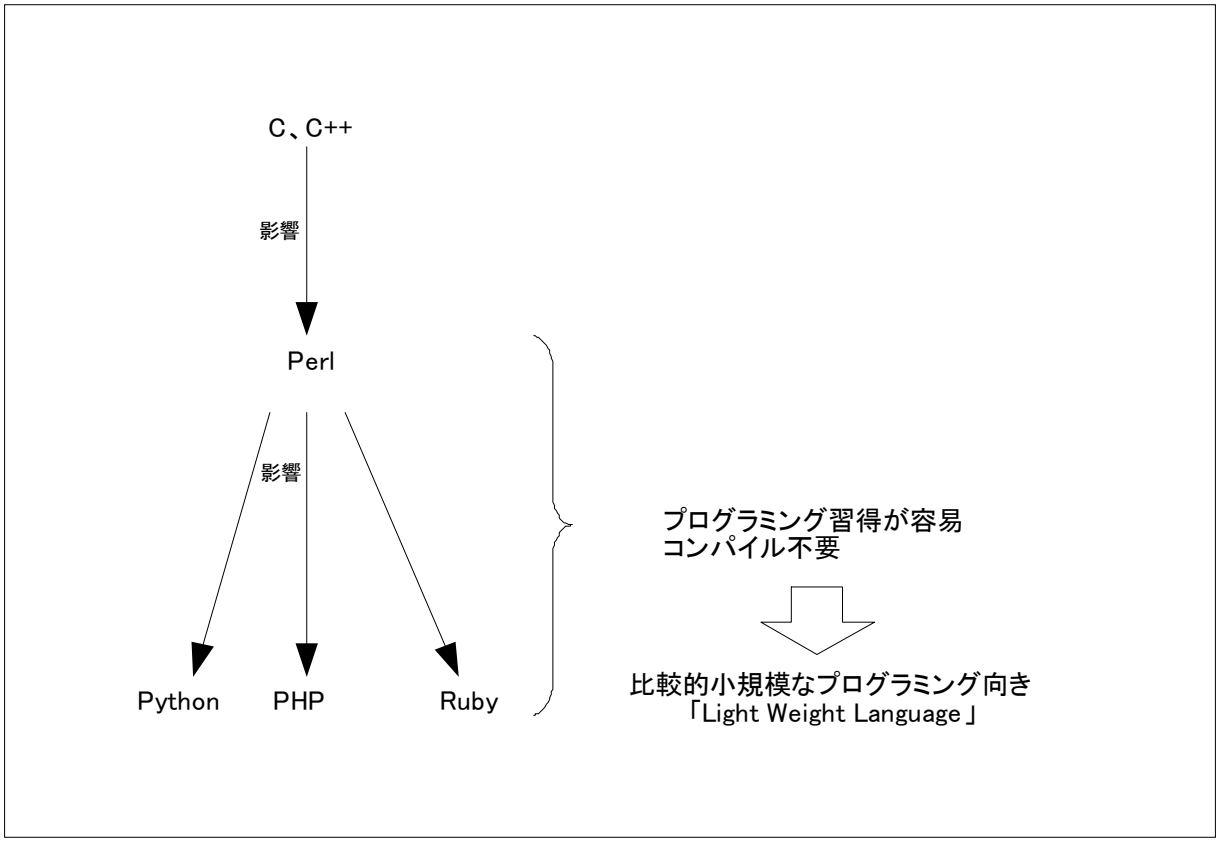

図 I-15-1.　Light Weight Language の誕生

【解説】

1) Light Weight Language の概要, 歴史, 特徴、用途

＊ 概要

C や C++、Java などに比べ学習が容易で手軽に扱える言語を総称して、「Light Weight Language」と呼ぶ。LL と省略される場合も多い。

＊ 歴史

CGI や Web アプリケーションの開発が増え、スクリプト言語が開発で多く用いられるようになった。C や C++に比べてプログラム作成や修正、実行が容易であることからプログラマへの負担が「軽い」ため「軽量」という言葉が用いられる。1987 年に Perl が作成されたのが始まりとされ、以降 Perl の影響を受けたいくつかの Light Weight Language が開発されている。近年ではその開発効率の高さゆえに Ruby が注目を集めている。

＊ 特徴

各言語それぞれの特徴があるが、全般的には、事前のコンパイルが不要であること、動的型付けができること、正規表現の利用が容易であること、といった特徴が挙げられる。

＊ 用途

CGI や Web アプリケーション開発、テキスト処理などが例として挙げられ、その手軽さと開発速度から様々な場面での利用が増えている。

2) 代表的な Light Weight Language

代表的なものとしては以下の言語が挙げられる。

＊ Perl

＊ PHP

＊ Python

＊ Ruby

3) Light Weight Language による開発の流れ

開発の流れは以下のようになる。

＊ 開発環境のインストール

＊ プログラミング

テキストエディタによりプログラムコードを記述する。各言語用の統合開発ツール等も用意されている。

＊ 実行

インタプリタとして動作するので、コンパイルの作業を必要とせず、プログラムコードを記述したファイルを作成すれば、すぐに実行が可能である。コマンドラインからの直接実行、CGI による実行、Web サーバのモジュールとして実行などの実行方法がある。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-2. Perl の特徴と正規表現	
対応する コースウェア	第 2 回 (Perl の基本構造)	

I-15-2. Perl の特徴と正規表現

古くから利用されている Perl の概要と特徴について触れ、Perl の動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明し、Perl を特徴付ける「正規表現」について解説する。

【学習の要点】

* Perl は Light Weight Language 登場初期に誕生し、他の Light Weight Language に影響を与えている。
* Perl は強力な正規表現機能を備えており、テキスト処理プログラムを記述しやすい。

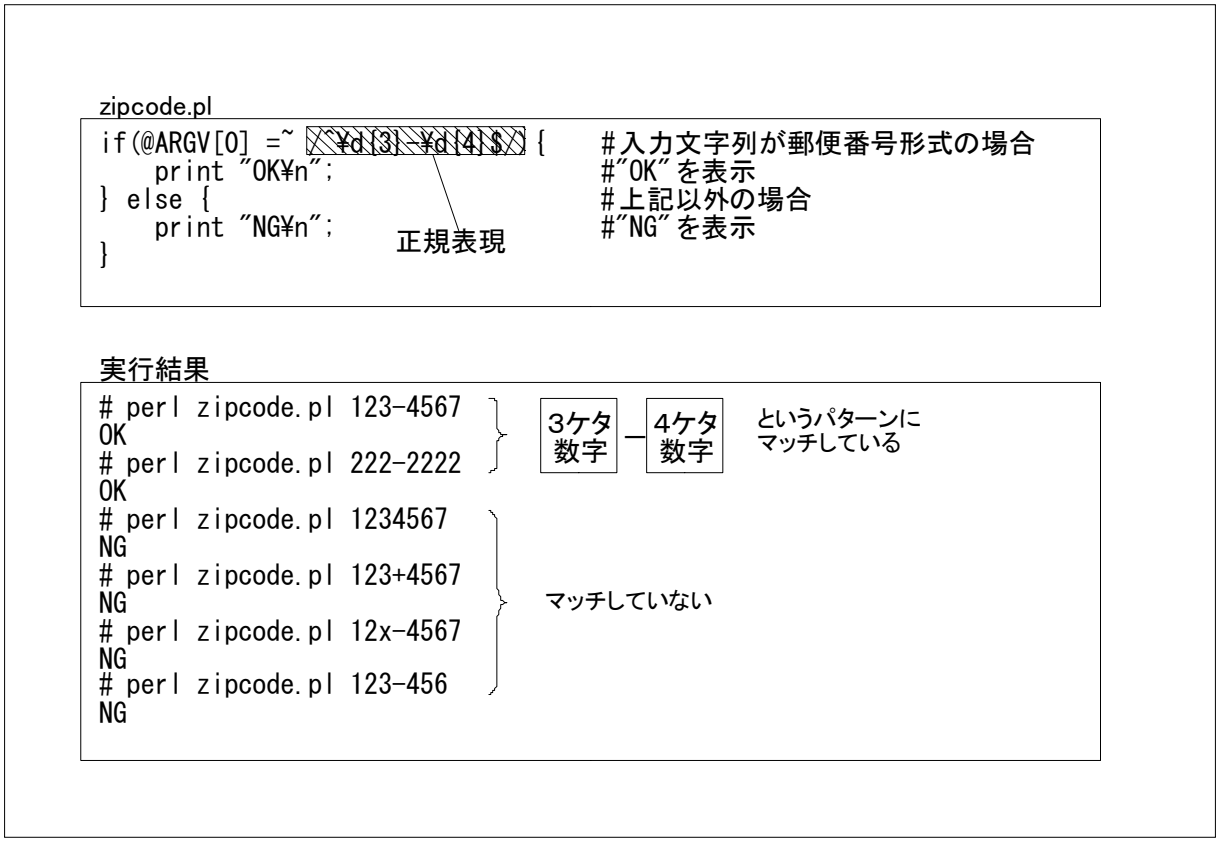

図 I-15-2.　Perl による正規表現プログラミングの例

【解説】

1) 概要と特徴、動作、ライセンス形態、代表的な利用法

＊ 概要

Perl は 1987 年に初期リリースされた、コンパイル操作を必要としないスクリプト言語である。Perl は Python、PHP、Ruby に影響を与えたと言われ、Light Weight Language の初期の代表的な言語であり、現在でも広く利用されている。多くの Linux ディストリビューションでは、OS をインストールした時点で Perl の処理系が利用できるようになっている。

＊ 特徴

Perl の大きな特徴として強力な正規表現(文字列のパターンマッチング機能)を利用できる点が挙げられる。正規表現以外にも、テキスト操作のための関数が豊富であり、このため Perl はテキスト処理プログラムを記述しやすい言語であるといえる。正規表現の機能は特に後発言語に大きな影響を与えている。

＊ 動作

コマンドラインからプログラムを呼び出して実行する方法、CGI(Common Gateway Interface)という方式で Web サーバから呼び出して実行する方法などがある。

＊ ライセンス形態

GPL、または、GPL の条件が緩和された Artistic License のいずれかから選択できる。

＊ 代表的な利用法

主にコマンドラインレベルでの軽量プログラミングや、CGI による Web アプリケーションなどにおいて利用されている。

2) 主な型、演算子、制御構文

＊ 型

全てのデータは、スカラー、配列、ハッシュの 3 つの型として扱い、変数名にそれぞれ「$」「@」「%」を付与して取り扱う。

＊ 演算子

- 算術演算子 (+ − ＊ /)
- 代入演算子 (= += −= ＊= /=)
- ビット演算子 (| & << >>)
- 比較演算子 (== < >)
- 論理演算子 (&& || !)
- 条件演算子 (?:)

＊ 制御構文

- ループ: for, while, until, foreach, last, next, redo
- 条件分岐: if, elsif, else, unless

3) 正規表現

使用している文字や文字の並びなど、ある条件を持ったいろいろな文字列をパターン化し、一つの文字列で示す手法、およびそのパターン化した文字列。文字列のパターンマッチングに利用され、テキスト処理をする際に非常に有用である。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-3. PHP の特徴と埋め込みプログラミング	
対応する コースウェア	第 3 回 (PHP の基本構造)	

I-15-3. PHP の特徴と埋め込みプログラミング

Web アプリケーションのロジック記述に利用されることの多い PHP の概要と特徴について触れ、PHP の動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明し、PHP を特徴付ける「埋め込みプログラミング」を解説する。

【学習の要点】

* PHP は HTML 中への埋め込み言語として誕生したが、文法の簡易性やドキュメントの充実などにより Web アプリケーションで広く利用されるようになった。
* PHP は、モジュールとして Web サーバに組み込んだ状態で動作させるのが一般的である。

now.php

```
<?
//現在日時の取得
$now = strftime("%Y/%m/%d %H:%M:%S");
?>
<!-- HTML中に現在日時を埋め込み -->
<html>
  <head>
    <title>What time is it now?</title>
  </head>
  <body>
    <p><?= $now ?></p>
  </body>
</html>
```

ウェブブラウザで now.php にアクセスした結果

```
2008/02/14 21:53:52
```

図 I-15-3.　PHP による埋め込みプログラミングの例

【解説】

1) 概要と特徴、動作、ライセンス形態、代表的な利用法

＊ 概要

PHP は 1995 年に誕生したスクリプト言語である。動的な Web ページを簡単に作成するために開発された。多くの Linux ディストリビューションでは、PHP が標準でバンドルされている。

＊ 特徴

PHP の大きな特徴として埋め込みプログラミングが挙げられる。HTML 中にタグ<?php～?>で PHP コードを埋め込むことが出来る。

＊ 動作

コマンドラインからプログラムを呼び出して実行する方法、CGI として動作させる方法などがある。また、多くの Web サーバ上でモジュールとして動作させることができ、Web サイトで利用されている PHP は、多くの場合 Apache HTTP Server 上のモジュールとして動作している。

＊ ライセンス形態

GPL に比べてソースコードの扱いが緩い PHP ライセンスに基づいている。

＊ 代表的な利用法

Web アプリケーション開発に特化した言語であるため、サーバサイドで動作する Web アプリケーションの構築に特に利用される。

2) 主な型、演算子、制御構文

＊ 型

- スカラー型 boolean、integer、float または double、string
- 複合型 array、object
- 特殊型 resource、NULL

＊ 演算子

- 算術演算子 (+ - * /)
- 比較演算子 (== < >)
- 代入演算子 (= += -= *= /=)
- 論理演算子 (&& || !)
- 条件演算子 (?:)

＊ 制御構文

- ループ： for、while、foreach
- 条件分岐： if、else、switch

3) 埋め込みプログラミング

HTML などのテキスト中に、部分的にプログラムを挿入することを埋め込みプログラミングと呼ぶ。静的な HTML 中の一部で、アクセスカウンタなど動的な表示をしたい場合に適している。PHP は元来埋め込みプログラミングを主目的に開発されたが、普及にともない機能拡張され、大規模な Web アプリケーションでも利用されるようになっている。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-4. Python の特徴と連想配列	
対応する コースウェア	第 4 回 (Python の基本構造)	

I-15-4. Python の特徴と連想配列

設定ファイルや機能拡張言語のベースとしてしばしば利用される Python の概要と特徴について触れ、Python の動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、型、演算子、制御構文などについて説明し、Python を特徴付ける「連想配列」を解説する。

【学習の要点】

* Python は特に欧米で広く利用されている。
* Python には辞書型があり、連想配列が使用できる。

wordcount.py

```
dic = {}                                    #辞書型変数の初期化
f = open("wordlist")                        #"wordlist"ファイルのオープン
for line in f:                              #ファイルの各行でループ
    dic[line] = dic.get(line, 0) + 1        #ワードのカウントを1増加
for word, num in dic.items():               #ワード毎にループ
    print word + str(num)                   #ワードとカウントを表示
```

入力ファイル
(wordlist)

```
dog
cat
dog
ant
duck
dog
cat
```

実行結果

```
# python wordcount.py
cat
2
dog
3
duck
1
ant
1
```

図 I-15-4.　Python による連想配列プログラミングの例

【解説】

1) 概要と特徴、動作、ライセンス形態、代表的な利用法

＊ 概要

＊ Python は 1990 年代初頭に誕生したスクリプト言語である。特に欧米で広く利用されており、多くの Linux ディストリビューションでは、OS をインストールした時点で Pyhon の処理系が利用できるようになっている。

＊ 特徴

Python の特徴として「連想配列」が挙げられる。Python では辞書型と呼ばれ，配列の要素の識別子として、数値や文字列以外の値も利用できる。

＊ ライセンス形態

GPL に比べてソースコードの扱いが緩い PSF(Python Software Foundation)ライセンスに基づいている。

＊ 代表的な利用法

主にコマンドラインレベルでの軽量プログラミングや、CGI による Web アプリケーションなどにおいて利用されている。

2) 主な型、演算子、制御構文

＊ 型

基本的な型は以下のようなものがある。

整数型、浮動小数点型、複素数型、文字列型、タプル型、リスト型、辞書型

＊ 演算子

- 算術演算子 (+ − ＊ /)
- 比較演算子 (== < >)
- 代入演算子 (= += −= *= /=)
- 論理演算子 (&& || !)
- 条件演算子 (?:)

＊ 制御構文

- ループ：for, while
- 条件分岐：if

3) 連想配列

通常の配列は要素の識別子として利用できる値は整数のみであるが、整数以外の数値や文字列を利用できる配列もあり、このような配列を連想配列という。Python の場合、数値や文字列だけでなく、様々な型の値を利用できる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-5. Ruby の特徴、基本的な構造、型、演算子、制御構文	
対応する コースウェア	第 5 回（Ruby の基本構造）	

I-15-5. Ruby の特徴、基本的な構造、型、演算子、制御構文

オブジェクト指向プログラミングを特徴とするRubyの概要と特徴について触れ、Rubyの動作、ライセンス形態、代表的な利用方法などを紹介する。また基本的なプログラミングができるように、変数、演算子、制御構文などについて解説する。

【学習の要点】

* Ruby は純粋なオブジェクト指向言語であり、変数はすべてオブジェクトであり、データ型がない。
* Ruby は、現在最も注目されている Light Weight Language の一つである。

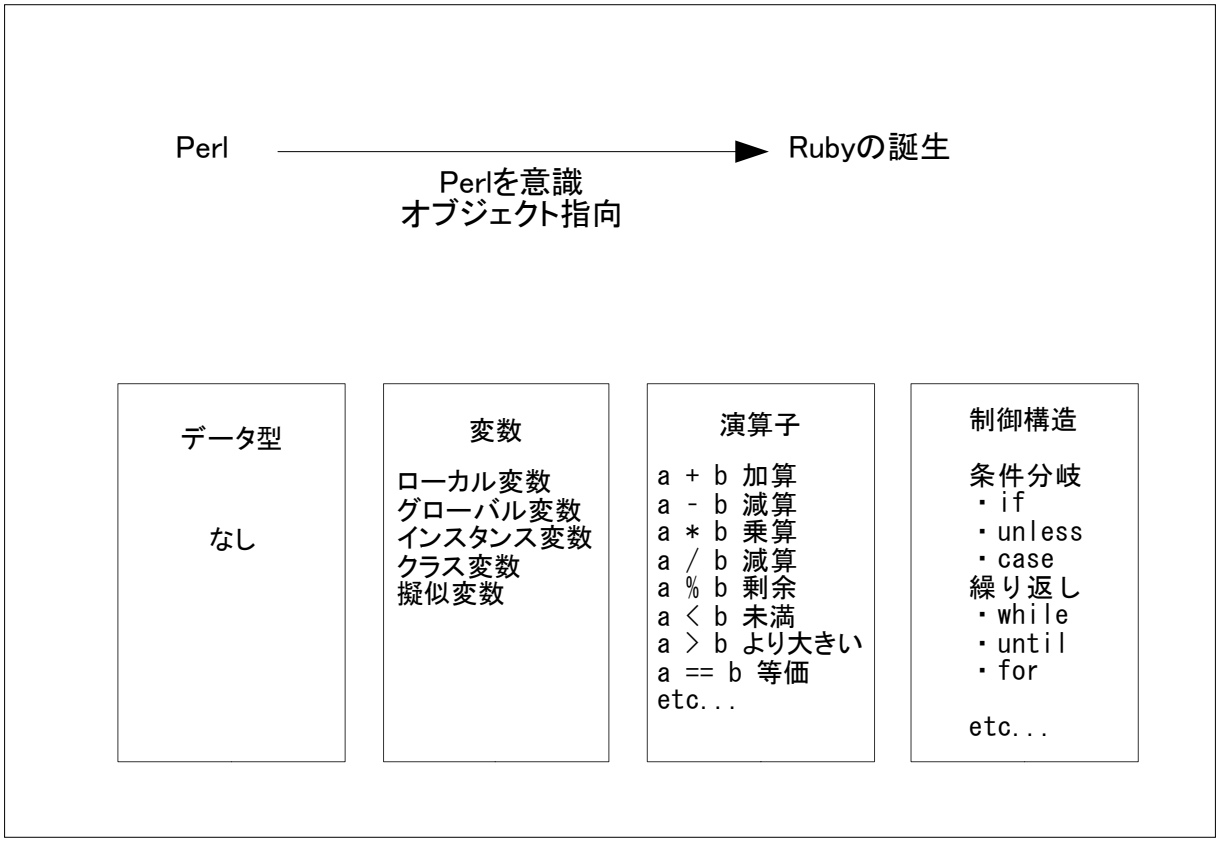

図 I-15-5.　Ruby の誕生、Ruby の変数/演算子/制御構造

【解説】

1）Ruby の概要と特徴、動作、ライセンス形態、代表的な利用法

＊ 概要

Ruby は 1993 年に日本で誕生したスクリプト言語である。Perl を意識して開発されている。多くの Linux ディストリビューションで、Ruby の処理系が標準バンドルされるようになってきている。

＊ 特徴

オブジェクト指向言語である。数値や文字列などもオブジェクトとして扱う。

＊ 動作

コマンドラインからプログラムを呼び出して実行する方法、CGI として動作させる方法などがある。

＊ ライセンス形態

Perl と同様、GPL または Artistic License のいずれかから選択できる。

＊ 代表的な利用法

主に Web アプリケーションに利用される。開発フレームワーク「Ruby on Rails」を載せた形で利用されることも多い。

2）プログラミングに必要な変数、演算子、制御構文

＊ 変数

Ruby ではローカル変数、インスタンス変数、クラス変数、グローバル変数という変数がある。

- ローカル変数

  英小文字と__で記述。ローカルスコープからアクセスできる。

- インスタンス変数

  @で始まり__と英小文字で記述。そのクラスのメソッドからアクセスできる。

- クラス変数

  @@で始まり、__と英小文字で記述。そのクラスとサブクラスと各クラスのインスタンスのメソッドからアクセスでき、外部からはアクセスできない。

- グローバル変数

  $ で始まり、__と英小文字で記述。スクリプト中のどこからでもアクセスできる。

＊ 演算子

- 算術演算子 (+ - * /)
- 比較演算子 (== < >)
- 代入演算子 (= += -= *= /=)
- 論理演算子 (&& || !)
- 条件演算子 (?:)

＊ 制御構造

- ループ ： for、while、until、next、redo、retry
- 条件分岐 ： if、unless、case

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-6. Ruby によるオブジェクト指向プログラミング	
対応する コースウェア	第 5 回（Ruby の基本構造）	

I-15-6. Ruby によるオブジェクト指向プログラミング

Ruby を題材として、オブジェクト指向プログラミングの基本について解説する。オブジェクト指向の概念を説明し、クラスやメソッドの定義、継承、カプセル化、ポリモルフィズムといった考え方が Ruby でどのように表現されるかを示す。

【学習の要点】

* 純粋なオブジェクト指向言語である Ruby では、オブジェクト指向プログラミングを行うための基本機能は完備されている。
* Ruby ではインスタンス変数は、自クラスのメソッドの中からしか参照できない。

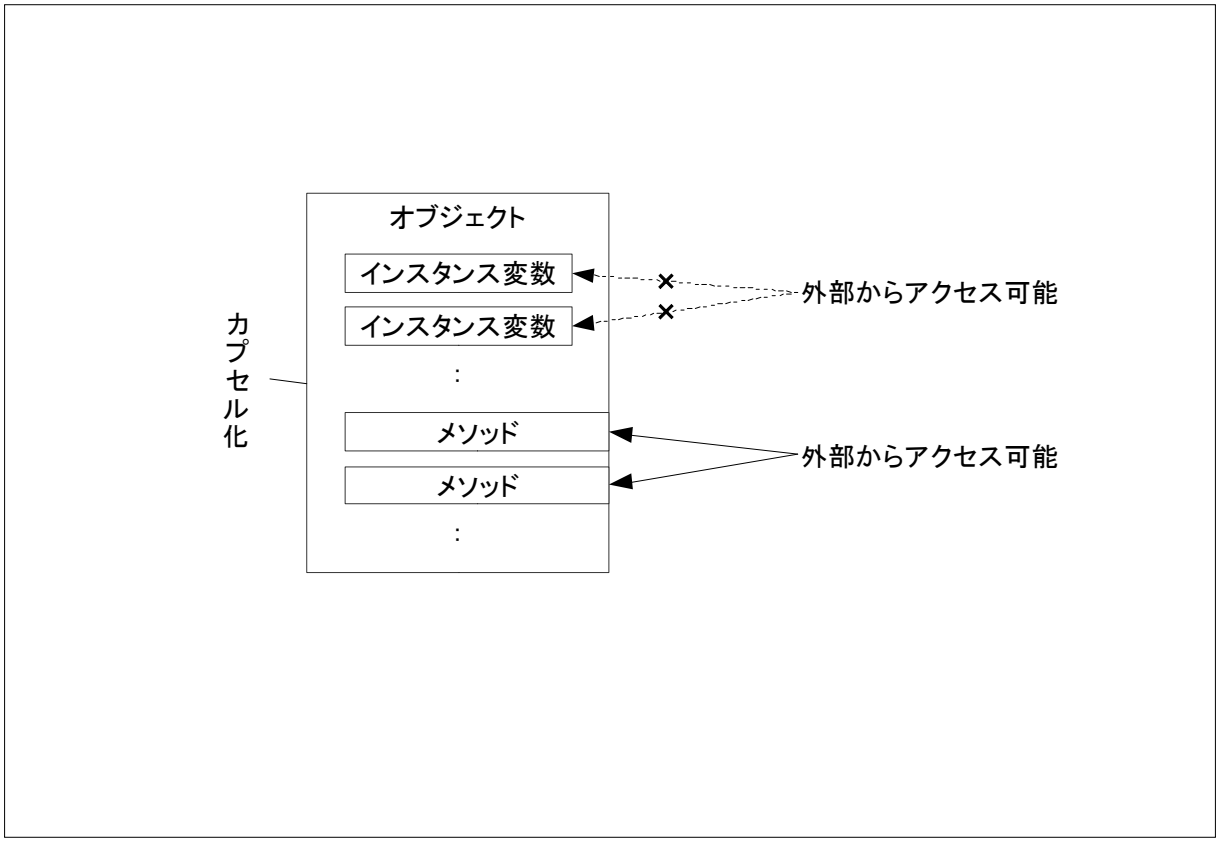

図 I-15-6.　Ruby でのカプセル化

【解説】

1) Ruby のオブジェクト、クラス、メソッド

Ruby では、全ての値をオブジェクトとして扱う。クラスは生成するオブジェクト(インスタンス)が行う一連の処理(メソッド)を定義するもので、1 つのクラスに複数のメソッドを定義できる。クラスやメソッドを利用する際の表現は以下のようになる。

* クラスの定義　　class クラス名
* メソッドの定義　　def メソッド名(引数)
* メソッドの呼び出し　　クラス名.メソッド名

2) インスタンス変数

オブジェクトに付随する変数をインスタンス変数と呼ぶ。インスタンス変数の定義はクラスで行うが、生成されたオブジェクト毎に異なる変数の値を持つことができる。

3) 継承

あるクラスで定義されているインスタンス変数やメソッドをそのまま引き継いで別のクラスを定義することを継承という。このとき元のクラスを新しいクラスのスーパークラス、新しいクラスを元のクラスのサブクラスという。スーパークラスで定義されているインスタンス変数やメソッドは、自クラスでもそのまま定義されていることになる。Ruby のクラス定義でスーパークラスを指定するには、「class クラス名 < スーパークラス名」のように記述する。

4) カプセル化

Ruby では、あるオブジェクトのメソッドを、他のオブジェクトからは呼び出せないようにすることができる。また、インスタンス変数はそのオブジェクト内からしか参照できないようになっている。このように、オブジェクトの状態や機能(インスタンス変数やメソッド)を、他のオブジェクトからアクセスできないようにすることをカプセル化と呼ぶ。Ruby でインスタンス変数の値を他のオブジェクトから参照するには、インスタンス変数の値を返すメソッドを定義する必要がある。

5) ポリモルフィズム

Ruby では、例えば、異なるクラスのオブジェクトを同一の変数として扱い、それぞれのクラスで定義された同じ名前のメソッドを呼び出すことができる。このように、異なるオブジェクトに対して、同じ名前のメソッドを呼び出し、それぞれのオブジェクトに適した処理が実行されることをポリモルフィズムと呼ぶ。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-7. Ruby が持つ特徴的なデータクラス(配列、ハッシュ等)	
対応する コースウェア	第 7 回 (組み込みクラス[データ構造])	

I-15-7. Ruby が持つ特徴的なデータクラス(配列、ハッシュ等)

Ruby に組み込まれている配列やハッシュなどの汎用データ型クラスについて説明する。配列、ハッシュ、構造化クラスの概要を解説し、各クラスの構成や使い方を紹介する。

【学習の要点】

* Ruby では、手続き型言語で頻繁に利用される、配列、構造体に相当するクラスが組み込まれている。
* ハッシュは、他言語の連想配列や辞書に相当するものである。

```
name = Array.new                          #Arrayクラスのオブジェクトを生成
name[0] = "Taro"
name[1] = "Hanako"
name[2] = "Jiro"

hometown = Hash.new                       #Hashクラスのオブジェクトを生成
hometown["Taro"] = "Tokyo"
hometown["Hanako"] = "Nagoya"
hometown["Jiro"] = "Osaka"

Person = Struct.new(:name, :hometown)     #StructクラスからPersonクラスを生成
person1 = Person.new                      #Personクラスのオブジェクトを生成
person1.name = "Taro"
person1.hometown = "Tokyo"
```

図 I-15-7.　Ruby での Array、Hash、Struct クラスの使用例

【解説】

1) Ruby に組み込まれた汎用データクラスについて

Ruby には頻繁に使われるクラスは初期状態で使えるように用意されている。初期状態で利用できるデータクラスの代表的なものとして、Array、Hash、Struct のようなものがある。

2) 配列の概要と構成、使い方

＊ 配列の概要

通常の配列は、任意のオブジェクトに識別子(要素番号)を付け、ひとまとまりとして扱えるようにしたものである。要素番号は 0 以上の整数となる。

＊ Array クラスの構成、使い方

配列は Array.new で初期化する。以下のような形式で要素を指定できる。

- 配列名[要素番号]
- 配列名[開始要素番号..終了要素番号]
- 配列名[開始要素番号,要素数]

メソッドには、配列結合の concat、要素をソートする sort、reverse などがある。他にも要素の削除、追加といったメソッドがある。

3) ハッシュの概要と構成、使い方

＊ ハッシュの概要

通常の配列と異なり、識別子(キー)としていろいろなオブジェクトを利用できる連想配列。ただし、Array や Hash などはキーとしては適切ではない。

＊ Hash クラスの構成、使い方

ハッシュは Hash.new で初期化する。以下のような形式で要素(キーに関連付けられた値)を指定できる。

- ハッシュ名[キー]

メソッドには、要素数取得の length、size などがある。他にもキーと値の削除、ハッシュ結合といったメソッドがある。

4) 構造化クラスの概要と構成、使い方

＊ 構造化クラスの概要

構造化クラスとは、構造体を扱うクラスである。Struct クラスのサブクラスで、個々の構造体を定義する。

＊ Struct クラスの構成、使い方

Struct.new で個々の構造体を定義するサブクラスを生成する。以下のような形式で構造体のメンバにアクセスできる。

- 構造体名.メンバ名
- 構造体名[メンバ番号]

メソッドには、メンバ数取得の length、size などがある。他にもメンバ名を配列として返すメソッドなどがある。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-8. Ruby におけるデータ(数値、文字列、その他)の操作	
対応する コースウェア	第 8 回 (組み込みクラス[データ操作])	

I-15-8. Ruby におけるデータ(数値、文字列、その他)の操作

Ruby に組み込まれている数値や文字列、日付などの固有なデータ操作に特化したクラスについて説明する。数値クラス、文字列クラス、その他特有のクラスについて、その概要を解説し、各クラスの構成や使い方を紹介する。

【学習の要点】

* Ruby では、データ型を持たない代わりに、データ操作に特化したクラスが組み込まれており、数値や文字列といったデータの処理も容易に実装できる
* Ruby では、正規表現もデータ操作のクラスとして利用できるようになっている。

```
# "1","3"はStringクラスのオブジェクト
p "1" + "3"                 #文字列が連結されるので"13"と表示される
p "1".to_i + "3".to_i       #整数が足されるので 4 と表示される
                            #to_iメソッドで整数に変換

# 100,50はIntegerクラスのオブジェクト
p 100 + 50                  #数値が足されるので 150 と表示される
p 100.to_s + 50.to_s        #文字列が連結されるので"10050"と表示される
                            #to_sメソッドで文字列に変換
```

```
$ ruby -e 'p “1” + “3”'
“13”
$ruby -3 'p 1 + 3'
4
$
```

図 I-15-8.　Ruby でのデータ操作に特化したクラスの使用例

【解説】

1) データ操作に特化したクラスについて

Numeric、String、Time クラスに代表されるようなデータ操作に特化したクラスが Ruby には組み込まれている。これらのクラスにより Ruby ではユーザにとって各データが扱いやすく、処理しやすい。

2) 数値クラスの概要、構成、使い方

＊ 数値クラスの概要

Ruby ではあらゆるデータがオブジェクトであり、数値も例外ではない。数値全般を表す抽象クラスとして Numeric クラスがあり、これを用いることでオブジェクトとして数値を扱える。演算や比較を行うメソッドはサブクラスのほうで定義されている。

＊ Numeric クラスの構成

サブクラスとして Integer クラスと Float クラスを持つ。メソッドとしては算術演算、数値型変換などのメソッドがある。

＊ Numeric クラスの使い方

- 算術演算：div、quo、modulo、remainder
- 数値型変換：coerce

3) 文字列クラスの概要と構成、使い方

＊ 文字列クラスの概要

任意の長さの文字列を扱うことができる String クラスがある。文字列操作に関するメソッドがあり、初期状態で利用可能である。

＊ String クラスの構成

String クラスがもつメソッドとして文字分割、文字比較、文字検索／置換などがある。他にも文字列連結，正規表現とのマッチ、文字数カウント、暗号化などのメソッドがある。

＊ String クラスの使い方

- 文字分割：split
- 文字比較：==, >, >=, <, <=
- 文字検索／置換：downcase、gsub

4) その他データ操作クラスの概要

＊ 日付操作クラス(Time クラス)

時刻を扱うクラスであり、現在の時刻の取得を始めとして、時刻演算、時刻比較、タイムゾーンの設定、時刻要素取得などの時刻に関する多くの操作がメソッドを通して利用できる。

＊ 正規表現による文字のパターンマッチング(Regexp クラス)

指定文字列とのパターンマッチなどに利用される。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-9. Ruby によるファイル操作のプログラミング	
対応する コースウェア	第 9 回（組み込みクラス[ファイル管理]）	

I-15-9. Ruby によるファイル操作のプログラミング

Ruby に組み込まれているファイル入出力やファイル/ディレクトリ操作などのファイル管理に特化したクラスについて説明する。ファイルクラス、IO クラス、その他ファイル操作に関連するクラスについて、その概要を解説し、各クラスの構成や使い方を紹介する。

【学習の要点】

* Ruby にはファイル操作に特化したクラスが組み込まれており、他の言語と同様、ファイルやディレクトリの操作を行うことができる。
* Ruby のファイル入出力についてはテキスト入出力/バイナリ入出力が用意されている。

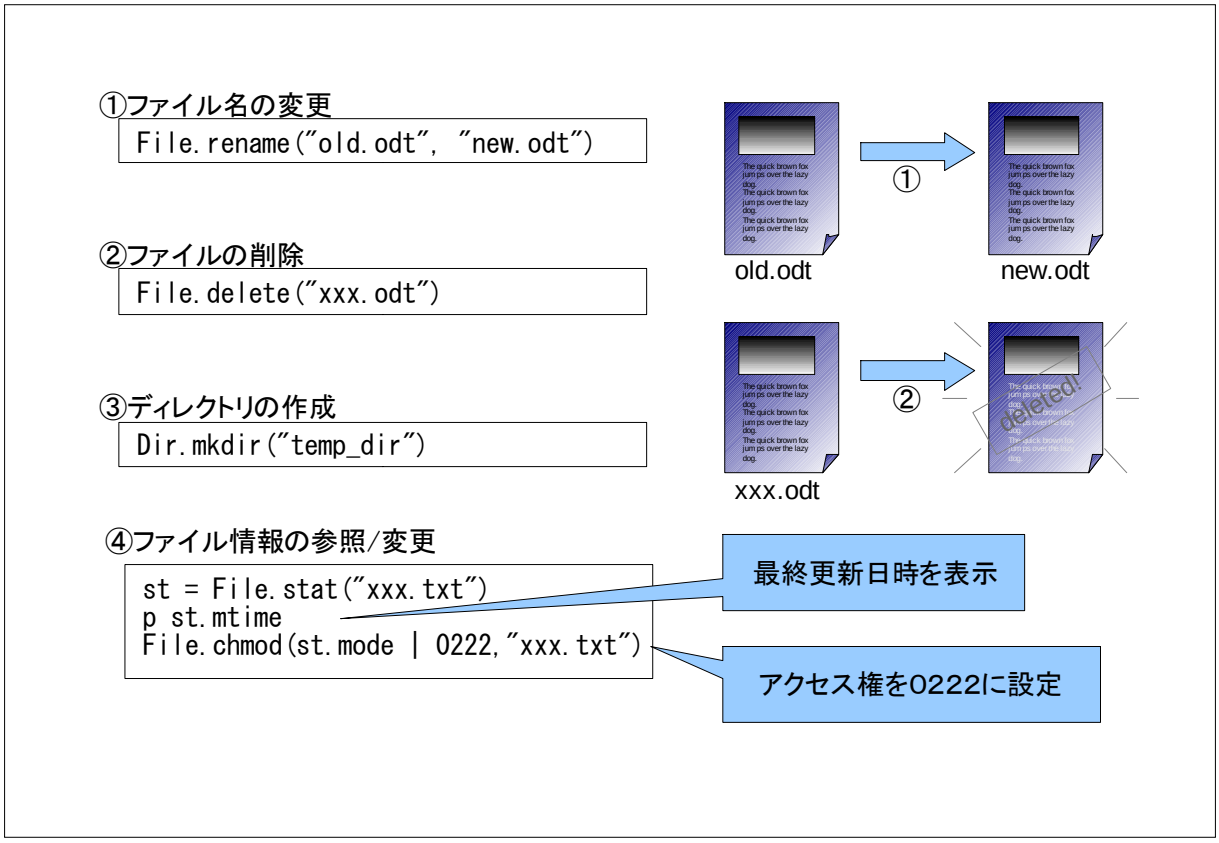

図 I-15-9.　Ruby でのファイル操作プログラミング例

【解説】

1) ファイル管理に特化したクラスについて

File::Stat、File、Dir、IO クラスに代表されるようなファイル管理に特化したクラスが Ruby には組み込まれている。これらのクラスにより Ruby ではユーザがプログラムからファイル管理を行いやすいようになっている。

2) ファイルクラスの概要、構成、使い方

＊ ファイルクラスの概要
このクラスではファイルの更新日時、ファイルタイプ、アクセス権、ファイルのユーザやファイルのモードなどのファイルそのものに関する情報を扱っている。

＊ File::Stat クラスの構成
ファイルに関する各種の情報を取得するメソッドが用意されている。

＊ File::Stat クラスの使い方
- 更新時刻関連:atime、mtime、ctime
- 所有者関連:uid、gid
- サイズ関連:size、blksize、blocks

3) IO クラスの概要と構成、使い方

＊ IO クラスの概要
基本的な入出力を提供するクラス。テキスト入出力、バイナリ入出力など一般的に利用される入出力がカバーされる。

＊ IO クラスの構成
IO クラスがもつメソッドとしてバイナリ入出力、テキスト入出力などがある。

＊ IO クラスの使い方
- バイナリ入出力:binmode
- テキスト入出力:getc、gets、printf、putc、puts、read

4) その他ファイル操作クラスの概要

＊ ファイル/ディレクトリ操作クラス
ファイルやディレクトリを操作するクラス。File クラスはファイル操作のためのクラスであり、File::Stat クラスと似通った点も多い。Dir クラスはディレクトリ操作を行うためのディレクトリストリーム操作のためのクラスである。

＊ File/Dir クラスの構成
File クラスではファイルのモードやオーナー変更、ファイルのロックなどのメソッドがある。Dir クラスではディレクトリストリームの読み込み，ディレクトリストリームの現在位置取得などのメソッドがある。

＊ File/Dir クラスの使い方
- ファイル操作関連:chmod、chown、flock、rename、open
- ディレクトリ操作関連:mkdir、pwd、open、chdir、read

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	15　Light Weight Language に関する知識 I	基本
習得ポイント	I-15-10. Ruby を用いた GUI アプリケーション開発	
対応する コースウェア	第 10 回（GUI アプリケーション開発）	

I-15-10. Ruby を用いた GUI アプリケーション開発

GUI アプリケーションのインタフェースを構成するウィジェットやイベント処理について説明し、Ruby で GUI アプリケーションを開発する際に利用するライブラリである Ruby/Tk や Ruby-GNOME2 の概要と使い方を説明する。

【学習の要点】

* Ruby には、GUI を扱うためのライブラリである Ruby/Tk が標準添付されている。
* Ruby-GNOME2 とは、GNOME(GNU Network Object Model Environment)の提供する C のライブラリを Ruby からアクセスできるようにしたものである。

① Ruby/Tk での Hello World

```
require 'tk'
TkLabel.new {
    text "Hello World!"
    pack
}
Tk.mainloop
```

② Ruby-GNOME2でのウィンドウ生成

```
require 'gtk2'
Gtk.init
window = Gtk::Window.new
window.show
Gtk.main
```

図 I-15-10.　Ruby での GUI アプリケーションプログラミング例

【解説】

1) GUI アプリケーションのインタフェースを構成するウィジェットやイベント処理について

＊ ウィジェット

GUI(Graphical User Interface)の画面を構成する個々の要素をウィジェットと呼ぶ。例として、ウィンドウ、テキストボックス、ボタンなどが挙げられる。

＊ イベント処理

ウィジェットに何らかの操作をした場合やプログラムに対して何か操作を行うとイベントが発生する. イベントの例としては、キーボードの操作、マウスクリックなどが挙げられる。GUI プログラミングはそれに対するプログラムの動作を記述するイベントドリブンでのプログラミングである。GUI プログラムはユーザとのインタラクションを通じて処理を行っていく場合が多く、GUI プログラミングにおいては重要な処理である。

2) Ruby/Tk の概要と使い方

＊ Ruby/Tk の概要

Ruby/Tk は Ruby で GUI を扱うためのライブラリで、Ruby に標準添付されている。強力なスクリプト言語である Ruby から GUI を扱えるようになっている。

＊ Ruby/Tk の使い方

Ruby/Tk プログラムは一般に require ‘tk’ から始まり、 Tk.mainloop で終わる。HelloWorld は図の①のように書ける。

3) Ruby-GNOME2 の概要と使い方

＊ Ruby-GNOME2 の概要

Ruby-GNOME2 は、Linux の GUI 環境である GNOME 上での GUI アプリケーションを Ruby で開発するためのライブラリである。GUI アプリケーションにとって重要な基本ライブラリの Ruby/Glib2、Ruby/GTK2、Ruby/ATK、Ruby/Pango、Ruby/GdkPixbuf2 と、印刷、マルチメディア処理といった拡張機能を提供するその他のライブラリとで構成される。

＊ Ruby-GNOME2 の使い方

Ruby/GTK2 を例に取ると、プログラムは通常 require ‘gtk2’ から始まり、Gtkl.main で終わる。単純なウィンドウを生成するだけのプログラムは図の②のように書ける。