yieldについて

るーごん

自己紹介

- 日本一人口が少ない県に住んでいます。
- 一昨年まで、ちょっと福岡に住みました。
- 仕事は主にdbMAGIC。
- プログラミング言語はよく分かりません。
- 好きなもの
  ポケモン、ファイブマン、ジェットマン、ジュウレンジャー、カクレンジャー、はてなスター
- 特技：人見知り　苦手なもの：しゃべること

今日の予定（目標）

- 自己紹介（済）
- イテレータとは・・・
- Rubyのイテレータとyield
- Pythonのイテレータ、ジェネレータとyield
- C#のイテレータとyield
- まとめ

※全ての内容が正しいとは限らないので、鵜呑みにしないでください！

イテレータとは

イテレータとは・・・

言語によって違う・・・？

配列やコレクションなど（複数の要素をもつオブジェクト）の各要素に対し、処理を繰り返す場合に使われる。

[Ruby]イテレータについて

イテレータ
⇒ブロック付きメソッド呼び出し

ブロック・・・
do end または { } で囲った部分を「ブロック」と呼ぶ。

[Ruby]イテレータについて

ブロック付きメソッドの例1

- eachメソッド

  順番に要素を取り出し、その要素を使って処理をする

（例）

```
(1..10).each{|i|
    puts i
}
```

[Ruby]イテレータについて

ブロック付きメソッドの例2

- collectメソッド（mapメソッドと同じ動作）

  ブロックを実行した結果が戻り値として返される。

(例)

```
rugon = (1..10).map{|i| i + 2}
```

[Ruby]イテレータについて

繰り返しでない
ブロック付きメソッド呼び出しもある。

例：sortメソッド
　要素を並び替えた結果を返す

[Ruby]イテレータの定義

主に、次のような方法。。。

1.yieldを使う。

2.procオブジェクトとして渡す。

・・・他

今回は、

yieldを使った場合について・・・・・・！

[Ruby]イテレータの定義

yieldの役割

呼び出し側のブロックを
実行する。

この時、yieldのあとに引数を指定すると、引数をブロックに渡すことができる。

[Python]イテレータとジェネレータ

イテレータ

- iter()メソッドでイテレータオブジェクトを生成する。
- イテレータオブジェクトは、next()で次の要素を返し、要素がなくなったら例外を発生させる。

[Python]イテレータとジェネレータ

イテレータを使うと・・・

- 要素の数を事前に把握したり、インデックスを指定するための変数を用意したりする必要がない。
- インデックスが存在しないようなオブジェクトでも扱える。

[Python]イテレータとジェネレータ

ジェネレータ

- イテレータのような機能を手軽に実装するための機能。
- ジェネレータ関数を呼ぶと、イテレータオブジェクトが返ってくる。
- yieldの入った関数はジェネレータ関数となる

[Python]イテレータとジェネレータ

Pythonでのyield

- ジェネレータ関数を定義するときの、関数の定義の中でのみ使用される。
- yield文を実行すると、関数の制御がいったん止まり、処理が譲られる。
- yieldにより、次に返す値を指定する。

[C#]イテレータ(反復子)

イテレータ(反復子)は・・・

- 列挙子(enumerator)オブジェクトを直感的に作成することができる。
- yield returnを利用して列挙子オブジェクトを返す。
- C#2.0以降。

[C#]イテレータ（反復子）

C#のyieldについて

- ブロックの中でyieldを使うと反復子ブロックになる。

- 反復子ブロックを使うことで、簡単に列挙子オブジェクトを作る事ができる。

[C#]イテレータ（反復子）

yield return

列挙可能型の次の項目を指定する

yield break

次の項目がないことを指定する。

[C#]イテレータ(反復子)

(例)

```
public IEnumerator<string> FiveMan()
{
string[] Member = { "ファイブレッド", "ファイブブルー",
                    "ファイブピンク", "ファイブブラック",
                    "ファイブイエロー" };
for (int i = 0; i < Member.Length; i++)
    yield return Member[i];
}
```

C#とPythonの比較（おまけ）

C#のMoveNextとPythonのnext()の違い

- C#
  MoveNextの戻り値はbool型で、次の要素があれば、Trueを返す。
  MoveNextが実行されるとCurrentの値が変わる。

- Python
  next()メソッドの戻り値が要素の値。
  次の要素がなければ例外を返す。

まとめ

- yieldを使うことで、繰り返しの処理をシンプルに定義できる。

- Rubyと、C#やPython ではyieldの使い方が異なる。
  Ruby⇒呼び出し側のブロックを実行
  C#、Python⇒次に返す要素を指定

参考文献

『たのしいRuby 第2版 Rubyではじめる気軽なプログラミング』

高橋 征義 後藤 裕蔵 著　まつもと ゆきひろ 監修　ソフトバンククリエイティブ(2008)

『Rubyレシピブック 268の技』

青木 峰郎 後藤 裕蔵 高橋 征義 著　まつもと ゆきひろ 監修　ソフトバンククリエイティブ(2006)

『標準講座C#』

Daniel Solis 著　田中正造 監訳　和田隆夫 翻訳　翔泳社(2008)

『みんなのPython』

柴田淳 著　新田光敏 発行　ソフトバンククリエイティブ(2006)