

for beginners

Ruby Programming

+はじめに+

新入生の皆さん、ご入学どうもおめでとうございます。多くの人は今、大学に入って、期待と不安を胸に抱きつつ、日々の生活を送っている事でしょう。

さて、コンピュータクラブの紹介冊子というからには、プログラミング言語の紹介のひとつやふたつ乗ってなければなりません。今年は、Ruby という言語を紹介させて頂きましょう。本当は VisualBasic の紹介を書けー、と言われていたのですが、生憎私は VisualBasic を持っていないもので、ゴネて Ruby の紹介に変えさせてもらいました。(^^;;

それでは、Ruby はまだあまり有名な言語とは言えませんので、まずは簡単にどのような特徴を持つ言語なのかの紹介から始めたいと思います。

+Ruby とは+

Ruby とは国産のスクリプト言語で、オブジェクト指向であるのが大きな特徴のひとつです。スクリプト言語とは、書いたプログラムが実行時に解釈される種類の言語の事です。オブジェクト指向についての説明は後段でいたしましょう。

とにかく、楽しくプログラミングできる、という事を考えて作られた言語です。楽しくプログラミングができます。(たぶん)

Ruby はスクリプト言語ですので、実行時にはそれを解釈するためのインタプリタと呼ばれる種類のプログラムが必要となります。(このインタプリタの事も Ruby と言います。)このインストールからやっているとは紙面が足りないの、ここからは自分でやっていただく事にします。

+オブジェクト指向とは+

オブジェクト指向というのは、なんとなく曖昧模糊として意味の掴みにくい言葉であります。そこで、ちょっと視点を変えてその成立からオブジェクト指

向の説明を試みましょう。

プログラミングというものを実際にしてみるとわかるのですが、プログラムというのは、ほとんどがデータとそれに対する処理（手続き。関数とも言う。）で構成されています。そこで、データとそれに対する処理をまとめたもの、という物を考えてやろう、という考えが出てきます。このデータと処理をまとめたものの事をオブジェクトと呼びます。

オブジェクトを考える事の利点は、プログラムが（オブジェクトを使わない時と比べて）書きやすく読みやすいものになる事、だと私は思っています。（もしかしたらちょっと違うのかもしれませんが。）書きやすく読みやすければ、それだけバグがでにくくなりますし、できてしまったバグを取り除く事も容易になります。

よくオブジェクトは現実のものを表しているのだ、とか、オブジェクトは知性を持ったな物だ、とか言われます。しかし、正直な話、私には今いちその例え話はピンときません。オブジェクトが現実世界を上手く表しているとも思えないし、知性を持っているとも思えません。しかし、それらの例えがよくわからずとも、Ruby でプログラムを書く事はできますので、安心してください。

+まずはHello,World!!から+

あまり紙面ありませんし、それではそろそろ Ruby のプログラムを実際に書くことにしましょう。しかし、その前に一つ二つ、用語の解説をさせていただきます。まずは関数から。

関数とは、よく使われる処理をひとまとめにしたものの事です。引数を受けとってなんらかの処理を行い、戻り値を返します。引数とか戻り値ってなんやねん、と言われそうなので、例をひとつ。まず、 f を関数とします。 $y=f(x)$ の y が戻り値、 x が引数です。数学の関数などと、イメージ的には良く似ていますね。（あくまで私のイメージですが。）

オブジェクト指向の世界では関数の事はメソッドと呼ばれます。（メソッドと関数は違う、という人も居ますが...。）以下の文では用語はメソッドに統一したいと思います。さ、ではそろそろ始めることにしましょう。

さて、では最初のプログラムです。次の一行を保存してコマンドラインから実行してみてください。

```
"Hello,World!!\n".display
```

画面には"Hello,World!!"と表示されるはずですが、されない人は Ruby を使うのをあきらめるなり、もうちょっと粘ってみるなり、私にメールするなりして下さい。メールがくれば時間がある限りはサポートします。

Ruby では全てのデータはオブジェクトとして扱われます。「`"`」(ダブルクォーテーション)で囲まれた文字列は、文字列オブジェクトとして扱われます。そして、ピリオドを用いて「オブジェクト.メソッド」と書くとき、ピリオドの左側のオブジェクトが持つ右側のメソッドを実行する、という意味のプログラムのコードになります。さきほどの"Hello,World!!\n".display を例にとれば、"Hello,World!!\n"というオブジェクトの display というメソッドを実行する(起動するとも言います)という事になりますね。

なんでピリオドを使うんだ、と思われるかもしれませんが、これはこういう言語の仕様になっているのだから、仕方ありません。文句をいってもしようがない。

では、またここでちょっと細かな用語説明を。さきほどのプログラムはオブジェクト.メソッドという形式でしたが、この時、オブジェクトの事をレシーバ、ピリオドの右側に書かれるメソッドのことをメッセージとも呼びます。これは、オブジェクトに対してメッセージを送ると、オブジェクトは受けとったメッセージに対応して適切な処理をしてくれるのである、というオブジェクト指向の考えかたに依る用語です。

オブジェクト指向の捉え方というのは、おそらく人によってそれぞれ少しずつ違うのだと、私は思います。どれもこれも「そういう考えかたもあるんだ」と割切って、必要な用語だけ覚えてしまえば(例えばこの場合、左(のオブジェクト)がレシーバ、右(のメソッド)がメッセージと呼ばれることもある、という事だけ知っていれば問題ないでしょう)、それ以上その考えかたにこだわる必要はないでしょう。

おっと、忘れるところだった。先程なんらかの処理はすべてオブジェクトのメソッドを呼び出す、という形で行われると書きましたが、実はいくつかの例外があります。

いくつかのメソッドについては、レシーバの記述を省略して書く事が許されています。というか、省略した形でしか書いてはいけないメソッドすらあります。ただ、その場合でもレシーバは省略されているだけであって、実際には Ruby の方で適当なオブジェクトのメソッドを呼び出している、という事実は、知っておいて損はないでしょう。

ああ、忘れちゃいけない。display というメソッドがなにをするかは、実行結果を見ればあきらかですよね？レシーバの文字列を画面に表示させます。(正確には、標準出力に文字列を出力する、というのが正しいのですが、そこまでは今は知らなくても問題ありません。)

+変数を使ってみる+

さて、ではここでようやく変数が登場します。Ruby における変数とは、オブジェクトにつけるなふだの様なものです。変数は「代入」を行うことで、オブジェクトを参照できるようになります。参照ってなんだ、と疑問に思われた方、その疑問はとりあえずちょっと横にどけておきましょう。少し後で説明しますから。

では代入の仕方をば。代入は、次の様に「=」を用いて行われます。

```
foo = "Hello,World!!\n"  
foo.display
```

今ここで、foo というのが変数なわけです。C 言語などを使っておられた方は「宣言いらないの?」と思われるかもしれませんが、必要ありません。代入が宣言の代わりです。

これだけでは、代入したあとの変数をどう使うのかわかりませんね。では先程のコードに少しつけたして、以下のようなコードにしてみましょう。

```
foo = "Hello,World!!\n"  
foo.display
```

これで、使い方はわかったでしょう。つまり、変数に代入を行うと、代入したオブジェクトを書くべきところで、代わりに変数を書けばよくなるのです。今挙げた例ではあまり便利だとは言えませんが、これからこの変数というやつが便利に使われる自体が徐々に増えていきます。

さて、さっき参照がどうのという話を少し、しました。これは一体、どういう事でしょうか。

普通の人間の感覚では、さっきの「foo に"Hello,World!!\n"を代入する」という表現を見たら、foo に"Hello,World!!\n"っていう文字列のコピーをするんだな、と思ってしまおうでしょう。しかし、Ruby に於いては、それは違います。

変数への代入は、右辺のオブジェクトへの参照を発生させる、つまり変数でもって別のオブジェクトを指し示せるようにする、という意味になります。コピーを作るには明示的に別のメソッドを用いねばなりません。C言語でいうと、ポインタのようなものです。

話だけではわかりにくいので、また少し例を挙げてみます。ここで、申し訳ないのですが、事ここに至り、使うメソッドが「display」だけでは、話を続けられなくなってしまったので、新しく「chop!」というメソッドを紹介させていただきます。このメソッドの機能は、文字列オブジェクトの末尾の一文字を取り除くことです。

```
foo = "hoge"  
foo.display  
bar = foo  
bar.chop!  
foo.display
```

画面には hogehog と表示されたはずですが、取り除いたのは bar の末尾の一文字なわけですが、bar というのは foo というオブジェクトへの参照ですから、bar の末尾の一文字を取り除くという操作は、foo の末尾の一文字を取り除くのと同じことになる、というわけです。

+メソッドが「値」を返すこと+

まず、foo.bar という記述がスクリプト中にあったとします。これをみて、コンピュータは、foo という変数で参照されるオブジェクトが bar という名前で定義されるメソッド（処理）を持っているかどうか調べます。もし foo が bar という名前で定義されるメソッドを持っているならばそのメソッドを実行し、最後に評価した値（ここでの値とはオブジェクトのことです）を返します。持っていなければ、そのようなメソッドはない、というエラーを出します。

ところで、メソッドは、常に最後に評価した式の値を返します。返すというのはどういうことか、具体的に説明してみましょう。先程でてきた、foo.bar を今回も使ってみましょうか。foo.bar という記述があれば、コンピュータは foo という名前で参照されるオブジェクトが bar というメソッドを持っているかどうか調べて、持っていればそれを実行し、「最後に評価した式の値を返す」のでした。オブジェクトを返すということは、この部分、foo.bar を、別のオブジェ

クトで置き換えてもよいということにならないでしょうか？なるのです。なりませんよ。なるんですってば。たぶん。

というわけで、foo.bar が hoge というオブジェクトを返したとすると、foo.bar という記述は hoge というオブジェクトに置き換えてもよろしいわけです。ここで、さっきの foo.bar という記述をもうちょっと拡張してみましょう。

```
print foo.bar
```

この規則にのっとると、この文は

```
print hoge
```

に置き換えてもよいわけです。グレート。なにがグレートなのかよくわかりませんが、とにかくグレートです。

ちなみに、代入式 (= で書くやつ) は、代入後のオブジェクトを返すようです。例えば、

```
print foo = "Hello,World!!\n"
```

という文は、問題なく Hello,World!!\n を表示します。

+式と文+

Ruby のチュートリアル (Ruby のオフィシャルサイトにて入手できます) によると、ruby では、文とは任意の式をセミコロン (;) または改行で区切った並びである、と定義されています。では、式とはなんでしょう。式とは、なんらかの値を返すもの、です。そして例によって値とはオブジェクトのことです。つまり、式とは、なんらかのオブジェクトで置き換えられるもの、と考えて差しつかえないでしょう。例を挙げると、foo.bar というメソッドも何らかのオブジェクトを返しますから式ですし、hoge なんていうオブジェクトがあったとすると、これはオブジェクトで置き換えられるものにもオブジェクトそのものですから当然の如く式です。

そこで混乱しがちだと思われるのですが、式というのは、別にオブジェクト

を返すだけが能ではありません。もしそうだとしたら、Ruby という言語はおそらく、役立たずな言語になってしまうでしょう。例えばさっき、メソッドもオブジェクトを返すので式である、と書きました。では、具体的なメソッドをひとつ、挙げてみましょう。"Hello,World!!\n".display というメソッドは画面に Hello,World!!\n と表示します。そして、nil という値 (オブジェクト) を返します。

このように、「値 (オブジェクト) を返す」というのは、式であるための必要条件でしかないのです。

さきほど文のことを任意の式をセミコロン (;) または改行で区切った並びである、と書きました。しかし、実は、文も値を返すようにできます。文に値を返させるようにするには、括弧で文を括ります。そして、そのとき文が返す値は、「最後に評価した式の値」になります。最後に評価した式の値とは、平たく書けば、一番後に書いてある式が返す値の事です。

あまり意味のない例ではありますが、一応例を挙げておきましょう。

```
print ( "foo" ; "bar" ; "hoge" )
```

というコードがあったとすると、最後に評価された "hoge" という文字列が画面に出力されます。(print というのはレシーバを省略できるメソッドで、引数としてあたえられたオブジェクトを画面に出力します。そういえば、書き忘れていた気がするのですが、Ruby ではメソッドの引数に括弧をつけてもつけなくても構いません。(そしてしつこい様ですが、画面に出力というのは、正確には標準出力への出力のことです。))

+おわりに

ええと、まだまだ書かねばならない事はたくさんあるのですが、あまり紙面もありませんし、これぐらいでそろそろ終わろうかと思えます。if やら while やらの解説をまったく書かないというのはいくら何でもアレかな、と思わんでもないのですが、締切もだいぶ過ぎちゃってるし...。(注: 締切を既に2週間ちょっとオーバーしてます。ていうかもうこれ書いてる時点で4月に入ってます。)

この文は、私が Ruby を学んだ際にわかりにくかったと思っているところを中心に書いたつもりです。この文を読んだあとに、Ruby の本などを読めば、ちょ

っとはわかりやすいのではないかと思います。

これを読んでくれた人が、Ruby に興味を持って、そして Ruby にチャレンジしてくれれば、幸いです。

+参考文献など+

<http://www.ruby-lang.org/>

Ruby のオフィシャルサイトです。各種情報や Ruby の入手などはこちらから。

Ruby プログラミング入門 (原 信一郎 著、オーム社刊)

Ruby に入門するならこの本が今のところ一番。いきなり「オブジェクト指向スクリプト言語 Ruby」とか読むと挫折しそうになるので、こちらの方がお勧め。

大学図書館にも入れました。注文から入荷まで二ヶ月以上もかかったんで、しびれを切らして注文した本人は、自分で本屋で買っちゃいましたけどね...

文責：徳永拓之 (tkng)