スクリプト言語 Python の初心者向け教材の作成

山中麻衣子

目　次

——スクリプト言語 Python の初心者向け教材の作成——

1　はじめに

(1) 誰のために作るのか

身近なところで言えば人文情報学科の学生達が Python を学習するにあたって役に立つものをつくるということから。また広く言えばこれから Python を学習するという人達の為に初心者に向けての教材を作成する。C 言語と Python はプログラムが似ているので、C 言語の初心者用テキストを参考にして初心者でも解りやすい Python の入門書を作成する。この題目に決めた理由は今までは Python の初心者用テキストがなかったからである。また従来の Python のテキストは内容が難しく、見た目が分かり図らいものが多い。熟練者なら理解できるかも知れないが Python を勉強し始める人にとっては入門書がないので、ますますとっつきにくさがある。勉強したくても参考にするテキストがない。比較的易しいテキストでも理解するのが、難しい。それが私自身感じてきた事なので、スクリプト言語 Python の初心者向け教材の作成に繋がったのである。初心者の初心者による初心者の為の入門書作成を目指す。

(2) 実際にどういうものを作るか

Python は、教育用の言語に適してるが、その日本語の入門書は存在しない。なので、まず、Python と比較的似ているやさしい C という C 言語の初歩の本を参考に、その本の中に出てくる例文を Python のプログラムに書き換えて実行してみる。C 言語と Python ではプログラムも若干違いがあるし、C 言語で必要であるものが Python では必要がないものもある。例えば C 言語では変数を宣言する必要があるが、Python では変数を宣言する必要がないなど。そういうところに注意しながら書

き換えていき最終的に Python の入門用教材を作成する。その為にはある程度 C 言語の知識も必要になってくる。

(3) それを作ることがどのような役に立つのか。

私自身が体験した事だが、実際に Python を授業で学習していてもう少し初心者にも分かりやすく、見やすく、文章だけでなく実際に自分でプログラムを作る例文や問題文があるテキストがあれば…と思った。そういう私と同じ事を思っている人達の為になるのではないかと思って、学習する効率や意欲を持って Python を学習してくれれば有り難い。今まではあったが、初心者向けではなく難易度が高かったので、理解するのが難しかった。私自身が体験して Python の初心者向け入門書は必要であると感じたから。これから Python を勉強し始める人達の役に立ちたいからである。

2 本論

まず、今まで授業で習ってきた Python のプリントとやさしい C に書かれてあるプログラムとを照らし合わせて C 言語から Python へプログラムを書き換えていく。その時気付いた事や違いなどを書き出していく。基本的に似ているものの C 言語と Python ではプログラムの形が違う。【Python と C 言語の違い】Python と C 言語のプログラムの違いを以下に示す。それぞれ C 言語と Python を記述してその違いを述べる。

Python

```
print "ようこそ Python へ!"
print "Python をはじめましょう!"
```

C 言語

```
#include <stdio.h>
int main(void)
{
  printf("ようこそ C 言語へ!\n");
  printf("C 言語をはじめましょう!\n");
  return 0;
}
```

となる。両者とも同じ動作を行うプログラムだが、Python の方が短く、すっきりとした表記になっている。

まず、表記の違いを見ることにする。

1. ; が文末にない。

   Python は一行に一文を書くことが原則である。しかし、; を分離子として複文にすることも許されている。

2. { }がない。

   ブロック文は:に始まり、同じ深さのインデントで表す。これらの表記法でずいぶんプログラムが簡潔に表記できる。

3. C 言語では`#include <stdio.h>`（画面出力に使う。）や`int main(void)`（main() 関数の開始部分。)return 0;（main() 関数の終了部分。) が必要であるが、Python では入力する必要がない。その点 Python は C 言語より見やすく簡単であることが分かる。

Python は print で画面に表示させるが、C 言語では printf を用いる。また Python では print の後すぐに文字を入力できるが、C 言語では () が必要である。また、Python は自然に改行される為、わざわざ改行コードを入力する必要がない。C 言語では改行する場合、\n というコードを入力する。逆に Python では自然に改行される為、改行しないようにするにはどうすればいいかを知っておく必要がある。Python で改行しないようにするには、文章の後に, (カンマ) を付ける必要がある。

例えば、

Python

```
print "%s は文字です。"  % ("A")
```

というプログラムがあるとする。この場合 print 文で書いた時点で勝手に改行する事になる、そこで文章の最後にカンマをつける。

Python

```
print "%s は文字です。"  % ("A"),
```

これで出力した時は改行されてないのである。

このプログラムの違いは形の違いであって、ごくごく基本的な違いなので、全部の sample に該当する。これだけの違いを見てみると C 言語より Python の方が簡潔で色んなコードを入力する必要がないので、見やすく分かりやすいのが分かる。

C 言語では変数を宣言する必要があるが、Python では変数を宣言する必要はない。その点 C 言語より Python の方が何にでも応用が効いて使いやすいと思われる。

(1) 文字と数値の違い

Python

```
print "%s は文字です。"  % ("A")
print "%d は整数です。"  % (123)
print "%f は小数です。"  % (10.5)
```

C 言語

```
#include <stdio.h>
int main(void)
{
  printf("%c は文字です。\n", 'A');
  printf("%d は整数です。\n", 123);
  printf("%f は小数です。\n", 10.5);
  return 0;
}
```

この例では単に文字列を入力するのではなく、文字中に % ●と記述すると、その部分に後に書き出した文字や数値を埋め込んで出力することができるというものである。そういうしくみ（% ●のこと) を変換仕様と呼ぶ。●の部分は出力したい文字や整数や小数などによって異なる記号を使う。

Python

```
文字を出力させる…%s
整数を出力させる…%d
小数で出力させる…%f
```

C 言語

```
文字を出力させる…%c
整数を出力させる…%d
```

小数を出力させる…%f

Python と C 言語では文字と数値の違いはあまりない。

Python では C 言語と違って出力したい文字列を入力した後、たい文字や数値を入力する。

やり方は例文の通りである。

・文字を埋め込みたい場合は %（""）というようにカンマでくくってその中に文字を入力する。

・整数を埋め込みたい場合や小数を埋め込みたい場合はそのままカッコの中に入力すればよい。

・C 言語では、文字を埋め込みたい場合出力させる文字列を入力した後，’ ’で埋め込みたい文字をカッコ内に入力する。

すると、Python では % 以降に書かれた文字であったり数値であったりが % ●に埋め
込まれて出力されるわけである。
ここでの違いはあまりないように思う。
コードが違うもののやはりプログラムの形はよく似ている事がわかる。

(2) エスケープシーケンスの違い

あまり違いはない。ほとんど同じであるが、ここでは初心者が使わなくてもできるので、省いた。主に使うエスケープシーケンスと言えば、タブと改行くらいである。これさえあればあとは使う機会がない。

(3) 文字コードの違い

まず、プログラムを比較してみよう。

Python

```
print "文字コードが 8 進数の 101 : %s" % ("\101")
print "文字コードが 16 進数の 61 : %s。" % ("\x61")
```

C 言語

```
#include <stdio.h>
int main(void)
{
  printf("文字コードが 8 進数の 101 : %c\n", '\101');
  printf("文字コードが 16 進数の 61 : %c\n", '\x61');
  return 0;
}
```

ここでの違いはほとんどない、Python も C 言語も文字コードを出力する方法は同じということである。

(4) 文字列の違い

C 言語では文字と文字列の違いがある。C 言語では文字 1 字に対して、複数の文字の並びを文字列リテラル（string literal) という。C 言語では、文字列は文字と異なり、' 'ではなく""でくくって記述する。

たとえば、下のような表記が文字列である。

sample4.py

```
"Hello"
"Goodbye"
```

これまで使ってきた printf() の中でも文字列を使っていた。画面に出

力された文字列には、””がつかないことに注意が必要である。C 言語では、文字列は 1 つの文字とは異なった扱いをされることになっている。逆に Python では、C 言語のように文字列と文字の表記の違いはない。文字を記述するときも文字列を記述するときも””でくくって記述する。

(5) 数値の違い

重要

数値は’’や""でくくらない。
整数（10 進数）を出力するには、変数仕様として %d を使う。
浮動小数点数を出力するには、変数仕様として %f を使う。
整数をあらわすときには、8 進数や 16 進数を使うこともできる。

(6) 型の違い

C 言語には、変数の「型」がある。変数には値を記憶させることができる。この値には、いくつかの「種類」がある。値の種類は、データ型(data type)、または型と呼ばれている。

変数を使うためには、どの型の値を記憶させる変数なのかを、あらかじめ決めておく必要がある。たとえば、short int 型という種類の値を記憶できる変数がある。このような変数を使うと、-32768～32767 までの範囲の整数のうちの、どれかひとつの値を記憶させることができるのである。short int 型の変数には、小数点以下のケタをもつ「3.14」などといった値を記憶させることはできない。このような値を格納する場合には、浮動小数点型と呼ばれる double 型などの変数を使わなければならない。

一般的に、サイズが大きいほど、あらわすことができる値の範囲は広

くなる。たとえば、double 型の値を記憶するには int 型の値より多くのメモリを必要とする。しかし、あらわすことができる値の範囲は広くなっている。

なお、C 言語の基本型のサイズは、開発環境によって異なる場合がある。

こういったことが C 言語では必要である。

そこで、Python にも型があるのか？と言えば型はあるが、C 言語のようにいちいち宣言する必要がない。C 言語の変数は型と名前を指定して宣言する。

(7) 変数の利用法

C 言語では変数を宣言する（Python では宣言する必要はないが) と、変数に特定の値を記憶させることができる。

このことを、

値を代入する（assignment)

という。次の図をみてみよう。「値の代入」は、用意した変数のハコに、特定の値を入れる（格納する、記憶する) というイメージである。

代入をするには、次のように=という記号を使って記述する。

図

```
num = 3;(Python の場合はセミコロンはいらない。)
```

少し変わった書き方であるかもしれない。しかし、これで、変数 num に値 3 を記憶させることができる。この=記号は、値を記憶させる機能をもっているのである。

図の説明

(1) 変数 num を宣言する。(Python の場合は必要がない。)

（2）変数 num に 3 という値を代入する。

変数に値を代入するコードのスタイルは、次のようになる。

構文

変数への代入

変数名 = 式;（Python の場合はセミコロンはいらない。）

実際にプログラムを作成して、変数を使ってみる。

Python

```
a = 3
print "変数 a の値は", a, "です。"
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num;
  num = 3;
  printf("変数 num の値は %d です。\n", num);
  return 0;
}
```

ここでの違いは、まず C 言語の方は最初のところで int 型の変数 num を宣言している。そして、次に変数 num に 3 という値を代入している。

Python の場合は事前に変数の宣言をする必要がありません。唯単に値を入れればその時点で変数領域が確保されます。なので、最初のところで num に 3 という値をそのまま代入すればよい。

どちらにも言えることは上部のように、「=」という記号は、数学の式で使われるような、「●と○が等しい」という意味ではない。「値を代

入する」という機能をあらわすものである。そこに注意しておく必要がある。

重要

変数には、= を使って値を代入する。

最後に C 言語のプログラムの printf～部分をみてみよう。ここで、変数 num の値を出
力している。変数の値を出力するためには、変換仕様を使う。整数を出力する場
合に、%d を使ったことを思い出してみよう。
変数に代入されている値が整数の場合も、同じように %d を使う。

カンマのあとには、引用符をつけずに変数名を記述する。すると、変換仕様の部分に出力されるのは

図

```
num
```

という変数名ではなく、変数 num の中に格納されている

図

```
3
```

という値となる。

これで変数の値を画面に出力するコードが記述できる。ここでの Python と C 言語の違いは変数を宣言する必要があるなしということである。わざわざ変数を宣言する必要がない Python は便利であることがわかる。しかし、先程の C 言語のプログラムでは、

図

```
int num;
```

```
num = 3;
```

というように、変数を宣言してから、もう 1 つ文を記述して、変数の中に値を代入した。しかし、C 言語では、変数を宣言したとき、同時に、変数に値を格納するという書きかたもできる。このような処理を、変数を初期化する（initialization) という。変数を初期化するコードは、次のようになる。

図

```
int num = 3;
```

この文は、先程のプログラムで 2 文にわたってあらわした処理を、1 文でまとめて行うものである。この方法によって簡潔にプログラムを記述することができる。先程のプログラムのように長く記述するより簡潔にまとめた方が分かりやすく見た目にもすっきりしている。だから変換の初期化方法をおぼえておくことは大切である。また、初期化方法をおぼえておくと便利なことは、変数の宣言代入を 2 文にわけておくと、値を代入する文を書き忘れてしまうことがよくあるからである。

値を代入しないまま、変数の値を画面に表示しようとすると、意味のない数値が表示されたり、エラーがおこることがある。変数を利用するときには、意味のある値がきちんと代入されているかどうかきちんと画面を確認して注意するようにしなければいけない。Python にはこのような C 言語のようにさまざまな方法はないが、方法が一通りの方が分かりやすい面もある。また、Python では変数の値を代入するのは 2 文にわたってプログラムを記述することがないので、あまり値を代入し忘れるということはないだろう。何の値を代入するかどんな値を表示させたいか、そこをしっかり間違わなければ代入した値がきちんと表示されるはずである。

次に変数の値を変更してみよう。

コード中では、文が 1 つずつ順番に処理されていた。

この性質を使って、いったん代入した変数の値を新しい値に変更することができる。次のコードをみてみよう。

Python

```
a = 3
print "変数 a の値は",a ,"です。"
a = 5
print "変数 a の値を変更しました。"
print "変数 a の新しい値は",a ,"です。"
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num;
  num = 3;
  printf("変数 num の値は %d です。\n", num);  …
  num = 5;  …
  printf("変数 num の値を変更しました。\n");
  printf("変数 num の新しい値は %d です。\n", num);  …
  return 0;
}
```

ここでは、はじめに変数 num に 3 を代入し、　のところで出力している。そして、次に　のところで変数に新しい値として 5 を代入している。このように、変数にもう一度値を代入すると、値を上書きし、変数

の値を変更するという処理ができるのである。

　で変数の値が変更されたので、　で変数 num を出力するときには、新しい値である「5」が出力されている。　と　は同じ処理であるにもかかわらず、変数の値が違うために、画面に出力される値が異なっていることに注意すること。

このように、変数には、いろいろな値を記憶させることができる。変数と呼ばれる理由はこういうことである。

ここでの、Python と C 言語の違いは特にない。基本的には変数の利用のところで記述したプログラムと同じ違いである。

次にほかの変数の値を代入することもできる。

値を代入するときに=の右辺に記述できるのは、3 や 5 といった一定の数値ばかりではない。

次のコードをみてみよう。

Python

```
num1 = 3
print "変数 num1 の値は",num1,"です。"
num2 = num1
print "変数 num1 の値を変数 num2 に代入しました。"
print "変数 num2 の値は",num2,"です。"
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num1, num2;
  num1 = 3;
  printf("変数 num1 の値は %d です。\n", num1);
```

```
  num2 = num1;
  printf("変数 num1 の値を変数 num2 に代入しました。\n");
  printf("変数 num2 の値は %d です。\n", num2);
  return 0;
}
```

ここでは、=記号の右側に数値ではなく、変数の num1 を記述している。すると、変数 num2 には、「変数 num1 の値」が代入されることになる。

画面をみると、たしかに、変数 num2 に変数 num1 の値である 3 が格納されていることがわかるだろう。このように、変数の値をほかの変数に代入するということもできるのである。

ここでの Python と C 言語の違いも特にない。基本的に先程のプログラムと同じような形である。ここでは、変数 num1 の値を変数 num2 に代入することができる、ということをおさえておくこと。

変数に値を代入するときには、注意しておかなければならないことがある。次のコードをみてみよう。

Python

```
num = 3.14
print "変数 num の値は %d です。" % (num)
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num;
  num = 3.14;
```

```
  printf("変数 num の値は %d です。\n", num);
  return 0;
}
```

ここでは、変数 num に「3.14」を代入している。ところが、このプログラムでは、異なる値の「3」が出力されてしまっている。これは、int 型の変数である num には、整数値しか格納できないことになっているからである。

変数は、宣言した型によって、記憶できる値の種類が決まる。整数値を格納する型の変数に小数値を代入すると、自動的に型の変換が行われて、小数点以下が切り捨てられてしまうことになっているのである。

変数の型には注意するようにする。

(8) 式と演算子での違い

Python

```
sum = 1 + 2
print "1+2 は",sum,"です。"
sum1 = 3 * 4
print "3 × 4 は",sum1,"です。"
```

C 言語

```
#include <stdio.h>
int main(void)
{
  printf("1+2 は %d です。\n", 1+2);
  printf("3 × 4 は %d です。\n", 3*4);
  return 0;
```

```
}
```

式と演算子での両者の違いは、まずプログラムのかたちが違う。Python は先に計算式を記述する。その後に print 文で画面に表示させる。C 言語は printf 文の中で計算式を記述してその値を出力させている。

次にいろいろな演算をする。式の中で、演算の対象となるものは、1 や 2 といった一定の数ばかりではない。

Python

```
num1 = 2
num2 = 3
sum = num1 + num2
print "変数 num1 の値は %d です。\n" % (num1)
print "変数 num2 の値は %d です。\n" % (num2)
print "num1+num2 の値は %d です。\n" % (sum)
num1 = num1 + 1
print "変数 num1 の値に 1 をたすと %d です。\n" % (num1)
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num1 = 2;
  int num2 = 3;
  int sum = num1 + num2;
  printf("変数 num1 の値は %d です。\n", num1);
  printf("変数 num2 の値は %d です。\n", num2);
```

```
  printf("num1 + num2 の値は %d です。\n", sum);
  num1 = num1 + 1;
  printf("変数 num1 の値に 1 をたすと %d です。\n", num1);
  return 0;
}
```

このように、特定の値だけでなく変数もオペランドとすることができる。ここでの Python と C 言語の違いは先程の式と演算のプログラムと同じなので、特に違いはない。また、演算子には色々な種類がある。加算、減算、乗算、除算、剰余など。これら色々な種類の演算子を使うプログラムの書き方もこののプログラムと同じなので、特に目立った違いはない。演算子の種類も Python と C 言語は同じである。

(9) キーボードから出力するときの違い

Python

```
sum = 0;
num = 0;
sum = raw_input("1 番目の整数を入力してください")
sum1 = raw_input("2 番目の整数を入力してください")
sum2 = raw_input("3 番目の整数を入力してください")
in_tax = int(sum) + int(sum1) + int(sum2)
print ("3 つの数の合計は", in_tax,"です。")
```

C 言語

```
#include <stdio.h>
int main(void)
{
```

```
  int sum = 0;
  int num = 0;
  printf("1 番目の整数を入力して下さい。\n");
  scanf("%d", &num);
  sum = += num;
  printf("2 番目の整数を入力して下さい。\n");
  scanf("%d", &num);
  sum += num;
  printf("3 番目の整数を入力して下さい。\n");
  scanf("%d", &num);
  sum += num;
  printf("3 つの数の合計は %d です。\n", sum);
  return 0;
}
```

【Python】

このサンプルでは、raw_input というものを使っている。質問を出して、それに対する答をユーザーから受け取るには、raw_input() 関数を使う。raw_input() 関数の返す値を変数に代入することで、ユーザーの返答を保存する事ができるのである。入力された値を数値として計算に使いたいときは、整数に変換するのには int() 関数を、実数に変換するには float() 関数を使う。

【C 言語】

キーボードからの入力を受けつけるには、scanf という言葉を使う。scanf は、キーボードから入力した値を読み込んで、変数に値を格納す

る機能をもっている。

scanf は printf のかたちにとてもよく似ている。整数を入力する場合には、printf と同じように、%d という変換仕様を使う。ただし、scanf で使う変数には、必ず&という記号をつけること。printf では変数名だけを記述すれば十分であるが、scanf では&記号が必要なのである。scanf という文の処理が行われると、実行画面がユーザーからの入力を待つ状態で止まる。そこで、ユーザーは、数値などをキーボードから入力し、エンターキーを押す。すると、入力した値が、scanf の最後に指定した変数に読み込まれるのである。

(10) インクリメント（デクリメント) の違い

C 言語にはインクリメントデクリメントはあるが、そもそも Python にはない。ちなみにインクリメントとは、下図のようなものである。

インクリメント

```
a++
++a
```

簡単に説明すると、前者を後置インクリメント演算子と呼ぶ。後者を前置インクリメント演算子と呼ぶ。変数を 1 増やすという目的だけなら前置と後置どちらでもよい。ただし、前置と後置によってプログラムの実行結果が異なる場合もある。

これは Python にはないので、Python では単純に +1 と記述すればよい。

(11) キャスト演算子

C 言語には、キャスト演算子というものがある。これは、指定した式の型を、() 中で指定した型に変換するという演算を行うものである。

使い方

```
inum = (int)dnum;
```

Python では、キャスト演算子はどうなるのか？

そもそも同じようにあるのか？

Python にもある。

int…整数、float…小数点

使い方

```
inum = int(dnum)
```

(12) 異なる型（同じ型) の演算をするときの違い

Python

```
d = 2;
pi = 3.14;
print "直径が %d センチの円の" % (d)
print "円周は %f センチです。" % (d*pi)
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int d = 2;
  double pi = 3.14;
  printf("直径が %d センチの円の\n", d);
  printf("円周は %f センチです。\n", d*pi);
```

```
  return 0;
}
```

Python

```
num1 = 5
num2 = 4
div = num1 / num2
print "5/4 は %f です。" % (div)
```

C 言語

```
#include <stdio.h>
int main(void)
{
  int num1, num2;
  double div;
  num1 = 5;
  num2 = 4;
  div = num1 / num2;
  printf("5/4 は %f です。\n", div);
  return 0;
}
```

これでは実行したとき、思った通りの答えが表示されない。だから、次のように書き換えてみる。

Python

```
div = float(num1) / float(num2)
```

C 言語

```
div = (double)num1 / (double)num2;
```

このように書き換えることで思った通りの答えが表示される。

このプログラムでの違いは、float と double や () を付ける場所などの違いである。

(13) if 文の違い

Python

```
if 条件判断:
        成り立つ場合の処理
        ...
   elif 次の条件判断:
        成り立つ場合の処理
        ...
   else:
        それ以外の場合の処理
```

C 言語

```
if(条件 1){
   文 1;
   文 2;
   ...
}
else if(条件 2){
   文 3;
   文 4;
   ...
}
else if(条件 3){
```

```
    ...
}
else{
    ...
}
```

この構文の通りである。Python は複数の条件を判断させて処理するとき elif を使うのに対し C 言語では、else if を使って処理をする。

また、C 言語には、if 文と同じように、条件によって処理をコントロールできる switch 文という構文がある。

しかし、switch 文は Python ではない。

ただ単純に if...elif...elif... という風に elif を重ねていく。

switch 文がなくても、問題は if...elif... で解くことができる。

(14) 論理演算子の違い

論理演算子とは、条件をさらに評価して、真または偽の値を得る。という役割をもっているものである。

論理演算子

&&演算子

||演算子（Python では or）

!演算子

以上が Python と C 言語のプログラムの違いである。

3　まとめ

今まで「やさしい C」というテキストを参考にしながら Python との違いを具体的なプログラムを取り入れて列挙してきた。

書き方が微妙に違うところが多々あったが、基本的に構文はあまり変わらないことが実際に問題を解いていくことによって分かった。

「やさしい C」の使い心地としては、初心者には分かりやすい説明書きであると思うが、丁寧に書き過ぎている部分があり、逆にややこしく感じるところがあった。同じような例題が何個か出てきて繰り返しになっていると感じた。同じような問題の繰り返しではなく、もう少し問題に展開があった方がいいと思った。

あとは、Python の復習と tex の復習ができた。

授業で Python と tex は学習してきたが、実際に使わないと忘れてしまうので、制作と論文を通して復習がしっかりできたと思う。

制作をしていく過程で同時に Python の復習ができたので、自分なりにまとめられたと思う。やはり先に Python の復習をしようかとも考えたけど、実際に問題を解いて進めていく方が Python の復習になった。