Python To C

担当 : 大野

概要

PythonとC言語の違いを概説して、C言語のプログラミングの基礎を学ぶ

- コンパイル
- main関数
- インデントと中括弧（{ }）
- 変数・型
- 演算
- 条件分岐
- 繰り返し(for, while)
- 関数

コンパイル

- C言語はコンパイラ言語
- プログラムの実行には、コンパイルという作業が必要
- コンパイルするには、C言語用のコンパイラが必要

- Pythonはインタプリタ言語
- コンパイルすることなく、実行可能

- C言語の方が高速！

コンパイル

C言語のプログラムの実行手順

1. geditなどのエディタで、C言語のプログラム（ソースコード）を作る
2. Cコンパイラでコンパイラする→実行ファイルができる
3. 実行ファイルを実行する

$ gcc test.c ←ソースコード(test.c)をコンパイル、a.outができる

$./a.out ←a.outを実行

main関数

- C言語のプログラムでは、必ずmain関数を記述しなければならない
- プログラムを実行すると、main関数が実行される
- 関数の外に処理を書いてはいけない！

```
int main(int argc, char **argv)
{
  処理
  return 0;
}
```

main関数

```
#include <stdio.h>


int main(int argc, char **argv)
{
  printf(“hello, world¥n”);
  return 0;
}
```

printf関数を使用するために必要。
Pythonのimportのようなもの。
他にもstdlib.hやmath.hなどがある。
全角文字は使わないこと!

¥nは改行を表す
文字列は“を使う。’はダメ

インデントと中括弧 { }

- Pythonでは、インデントでブロックを表現するが、C言語では{ }で表現する
- 関数やif文、for文などは、{ } で囲まれる
- 改行やインデントは、比較的自由にできる
- 「；」を忘れずに

Python
```
if a == b:
    print(“a equal b”)
```

C言語
```
if(a == b){
printf(“a equal b¥n”);
}
```

変数・型

- C言語では、変数を使用する前に宣言が必要
- 変数の宣言は、ブロックの上の方（最初は関数のはじめに宣言すると覚えておけば良い）
- 宣言と同時に値を代入することもできる
- 変数の宣言には、変数の型(int、floatなど)も指定する必要がある
- 一度宣言したら、その型の変数としてしか使用できない
- 変数名に全角文字は使えない

変数・型

- PythonとC言語の対応
- C言語の下記の型は、単なる箱なので、メソッドは持っていない

	Python	C言語	Numpy
整数	int	int long	int32 int64
浮動小数点	float	float double	float32 float64
文字列	str	char	

変数・型

int型, long型

- Pythonのintでは、とてつもなく大きな数を扱うことができる
- C言語では、int型は32bit、long型は64bit
- int型
  - -2147483648 ～ 2147483647
- long型
  - -9223372036854775808 ～ 9223372036854775807

扱える整数の大きさに注意！

変数・型

float 型(単精度)、double型(倍精度)

- C言語では、float型(32bit)、double型(64bit)
- 有効数字：float型は7桁程度、double型は15桁程度

扱える範囲（絶対値）

- float型
  - 1.175494e-38～3.402823e+38
- double型
  - $2.225074\ 10^{-308} \sim 1.797693\ 10^{+308}$
- Pythonのfloat型は、C言語のdouble型相当

変数・型

char型

- char型は8bit
- 英数字一文字を記憶できる
- -128～127の整数を記憶できる
- 文字列を扱いたいときは、配列とする

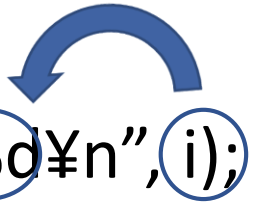
※ unsinged修飾子: 非負

- unsigned char : 0～255
- unsigned int : 0～4294967295
- unsigned long : 0～18446744073709551615

変数・型

printf関数で、変数の値を表示

整数型の変数iの表示



printf(“i=%d¥n”, i);

%dのところに、iの中身が表示される

long型なら%ld, 浮動小数点型なら、%fを使う。

char型%c (文字列、つまりchar型の配列なら%s)

複数の変数を同時に表示：int iとfloat fの場合

printf(“i=%d, f=%f¥n”, i, f);

変数・型

宣言方法

型名　変数名;

- 変数の命名方法は、Pythonと同じと考えて良い（ただし全角は使えない）

例

int a; ←int型のaを宣言

a=1; ←aに1を代入

printf(“a=%d¥n”,a);でaの中身を表示できる

float b, c; ←float型のbとcを宣言

int d =1; ←int型のdを宣言すると同時に1を代入

変数・型

キャスト

int a; ←int型のaを宣言

a=1.1; ←aに1,1を代入

printf(“a=%d¥n”,a); ← a=1と表示される（小数点以下は切り捨てられる）

int a;

float b;

b=1.25;

a=(int)b; 明示的にキャスト

変数・型

```
#include <stdio.h>

int main(int argc, char **argv)
{
  int a;
  float b;
  a = 1;
  printf(“a = %d¥n”, a);
  b =1.25;
  a = (int)b;
  printf(“a = %d, b=%f¥n”, a, b);
  return 0;
}
```

- a=1.1; としてみよう

変数・型

配列の宣言方法

型名　変数名[要素数];

（スライスなどの操作はできない）

例

int a[10]; ←int型で要素数10の変数aを宣言

a[0] = 1; a[1]=2;

↑Pythonのリストと同じくa[0]～a[9]まで。ただし、a[-1]などはできない。

printf(“a[0]=%d, a[1]=%d¥n”,a[0],a[1]);

変数・型

```
#include <stdio.h>

int main(int argc, char **argv)
{
int a[10];
a[0] = 1; a[1]=2;
printf(“a[0] = %d, a[1]=%d¥n”, a[0],a[1]);
return 0;
}
```

- a[2]～a[9]に数値を代入して、表示してみよう

演算

	Python	C言語
足し算	+	+
引き算	-	-
掛け算	*	*
割り算	/ // (整数徐算)	/
余り	%	%
冪乗	** x**y	pow関数 pow(x,y)

演算

Python とC言語の割り算の違いに注意
C言語には // はない

Python
>>>5/2
2.5
>>>5//2
2
>>>5.0//2
2

C言語
5/2 -> 2
5.0/2 -> 2.5

確認
```
double a, b;
a = 5/2; b=5.0/2;
printf(“a=%f b=%f¥n”, a, b);
```

確認
```
int a, b;
a = 5*2; b=5%2;
printf(“a=%d b=%d¥n”, a, b);
```

条件分岐

- C言語のif文は、ほぼPythonと同じ
- 条件は、( )で囲む
- インデントではなく{ }でブロックを表現する

Python

```
if a == b:
    print(“a equal b”)
```

C言語

```
if(a == b){
    printf(“a equal b¥n”);
}
```

条件分岐

Python

```
if a == b:
    print(“a equal b”)
elif a > b:
    print(“a > b”)
else:
    print(“a < b”)
```

C言語

```
if(a == b){
  printf(“a equal b¥n”);
} else if (a > b) {
 printf(“a > b¥n”);
} else {
 printf(“a < b¥n”);
}
```

elif → else if

else → else

条件分岐

```
#include <stdio.h>

int main(int argc, char **argv)
{
int a, b;
a=1; b=1;
if(a == b){
  printf(“a equal b¥n”);
}
return 0;
}
```

```
#include <stdio.h>

int main(int argc, char **argv)
{
int a, b;
a=1; b=1;
if(a == b){
  printf(“a equal b¥n”);
} else if (a > b) {
 printf(“a > b¥n”);
} else {
 printf(“a < b¥n”);
}
return 0;
}
```

条件分岐

比較演算子

Python	C言語
==	==
>	>
>=	>=
<	<
<=	<=
!=	!=

論理演算子

Python	C言語
and	&&
or	||
not	!

繰り返し

for文：C言語では、一般に、整数の制御変数で繰り返しを制御する

Python
```
for i in range(10):
    print(i)
```

C言語
```
int i;
for(i=0;i<10;i++){
  printf(“%d¥n”,i);
}
```

iの初期値を0とし、
10より小さい間繰り返す
i++はi=i+1の意

繰り返し

while文：Pythonと同様に、自分でループから抜ける工夫をしないといけない

Python

```
i=0
while i<10:
    print(i)
    i=i+1
```

C言語

```
int i;
i=0;
while(i<10){
  printf(“%d¥n”,i);
  i=i+1;
}
```

if文と同様、条件の部分は( )で囲む

繰り返し

繰り返し分で使用するcontinue, breakは、 Pythonもc言語も使い方は同じ

Python

```
i=0
while i<10:
    if i%2==0:
        continue
    print(i)
    i=i+1
```

C言語

```
int i;
i=0;
while(i<10){
  if(i%2==0){
    continue;
  }
  printf(“%d¥n”,i);
  i=i+1;
}
```

関数

- C言語の場合、戻り値の型と引数の型も指定しなければならない。ない場合は、voidとする
- 関数の呼び出し方はPythonと同じ（名前付き引数はできない）
- 呼び出す前に、定義を書くか、プロトタイプ宣言が必要
- 無名関数はできない

Python
```
def func(num):
  num=num+1
  print(num)
```

C言語
```
void func(int num){
  num=num+1;
  printf(“%d¥n”, num);
}
```

関数

Python
```
def plusone(num):
  num=num+1
  return num
```

C言語
```
int plusone(int num){
  num=num+1;
  return num;
}
```

int型の戻り値なので、voidではなく、int plusoneとなっている。

関数

Python

```
def plusone(num):
  num=num+1
  return num

a=1
b=plusone(a)
print(b)
```

C言語

```
#include <stdio.h>

int plusone(int num){
 num=num+1;
 return num;
}

int main(int argc, char **argv){
int a,b;
a=1;
b=plusone(a);
printf(“%d¥n”,b);
return 0;
}
```

関数

C言語

```
#include <stdio.h>
int plusone(int);

int main(int argc, char **argv){
int a,b;
a=1;
b=plusone(a);
printf(“%d¥n”,b);
return 0;
}

int plusone(int num){
  num=num+1;
  return num;
}
```

関数

C言語のグローバル変数

- 関数内からアクセスできる
- 関数内で、グローバル変数と同じ名前のローカル変数が定義された場合には、ローカル変数が優先される

関数

C言語

```
#include <stdio.h>
int a,b;

void setval(void){
 a=1; b=2;
 return;
}

int main(int argc, char **argv){
int b;
a=3; b=4;
setval();
printf("a=%d b=%d¥n",a, b);
return 0;
}
```

グローバル変数
どの関数からもアクセスできる

実行結果は
a=1, b=4

関数

引数

- C言語の引数によって値をわたす場合、値渡し→関数内から値を変更できない
- 配列の場合は、ポインタが渡される→配列の内容は変更可能
- 参照渡しと値渡しの違いについて、各自調べた方が良い

関数

C言語

```
#include <stdio.h>

void setval(int a, int *b){
 a=1;
 printf(“%d %d %d¥n”,b[0],b[1],b[2]);
 b[0]=1;
 return;
}

int main(int argc, char **argv){
int n, m[3];
n=10; m[0]=0;m[1]=1;m[2]=2;
setval(n, m);
printf(“n=%d m[0]=%d¥n”,n, m[0]);
return 0;
}
```

実行結果は

0 1 2

n=10, m[0]=1