第十五章：C语言扩展¶

本章着眼于从Python访问C代码的问题。许多Python内置库是用C写的， 访问C是让Python的对现有库进行交互一个重要的组成部分。 这也是一个当你面临从Python 2 到 Python 3扩展代码的问题。 虽然Python提供了一个广泛的编程API，实际上有很多方法来处理C的代码。 相比试图给出对于每一个可能的工具或技术的详细参考， 我么采用的是是集中在一个小片段的C++代码，以及一些有代表性的例子来展示如何与代码交互。 这个目标是提供一系列的编程模板，有经验的程序员可以扩展自己的使用。

这里是我们将在大部分秘籍中工作的代码：

```
/* sample.c */_method
#include <math.h>

/* Compute the greatest common divisor */
int gcd(int x, int y) {
   int g = y;
   while (x > 0) {
      g = x;
      x = y % x;
      y = g;
   }
   return g;
}

/* Test if (x0,y0) is in the Mandelbrot set or not */
int in_mandel(double x0, double y0, int n) {
   double x=0,y=0,xtemp;
   while (n > 0) {
      xtemp = x*x - y*y + x0;
      y = 2*x*y + y0;
      x = xtemp;
      n -= 1;
      if (x*x + y*y > 4) return 0;
   }
   return 1;
}

/* Divide two numbers */
int divide(int a, int b, int *remainder) {
   int quot = a / b;
   *remainder = a % b;
   return quot;
}

/* Average values in an array */
double avg(double *a, int n) {
   int i;
   double total = 0.0;
   for (i = 0; i < n; i++) {
      total += a[i];
   }
   return total / n;
}

/* A C data structure */
typedef struct Point {
```

```
    double x,y;
} Point;

/* Function involving a C data structure */
double distance(Point *p1, Point *p2) {
    return hypot(p1->x - p2->x, p1->y - p2->y);
}
```

这段代码包含了多种不同的C语言编程特性。 首先，这里有很多函数比如 `gcd()` 和 `is_mandel()` 。 `divide()` 函数是一个返回多个值的C函数例子，其中有一个是通过指针参数的方式。 `avg()` 函数通过一个C数组执行数据聚集操作。 `Point` 和 `distance()` 函数涉及到了C结构体。

对于接下来的所有小节，先假定上面的代码已经被写入了一个名叫“sample.c”的文件中， 然后它们的定义被写入一个名叫“sample.h”的头文件中， 并且被编译为一个库叫“libsample”，能被链接到其他C语言代码中。 编译和链接的细节依据系统的不同而不同，但是这个不是我们关注的。 如果你要处理C代码，我们假定这些基础的东西你都掌握了。

Contents: