描述】

让我们学习用 Python 语言向世界问好。"Hello World"示例程序最早出现于 1972 年，由贝尔实验室成员 Brian Kernighan 撰写的内部技术文件《Introduction to the Language B》之中。不久同作者于 1974 年所撰写的《Programming in C: A Tutorial》，也延用这个示例。

一般来说，这是每一种计算机编程语言中最基本、最简单的程序，亦通常是初学者所编写的第一个程序。它还可以用来确定该语言的编译器、程序开发环境，以及运行环境是否已经安装妥当。将"Hello World"作为第一个示范程序，已经成为编程语言学习的传统。

我们在这里继续传统，这也是一种向前辈们致敬的方式之一吧！

【输入】

没有输入。

【输出】

```
Hello World!
```

【来源】

《Python 程序设计基础》第 1 章编程题 1。

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
print("Hello World!")
```

【描述】

编写程序，输出指定的由'*'组成的倒三角图案。（要求：第一行行首无空格，每行行尾无空格）

【输入】

没有输入。

【输出】

```
* * * *
* * *
* *
*
```

【来源】

《Python 程序设计基础》第 1 章编程题 2。

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
print("* * * *")
print("* * *")
```

```
print("* *")
print("*")
```

【描述】

编写程序，计算下列数学表达式的结果并输出，小数点后保留 3 位。

$$4 \times (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13})$$

【输入】

没有输入。

【输出】

输出结果小数点后保留 3 位。

参考答案：

```
print(format(4 * (1 - 1.0 / 3 + 1.0 / 5 - 1.0 / 7 + 1.0 / 9 -
```

描述】

编写程序，计算下列数学表达式的结果并输出，小数点后保留 3 位。

$$x = \sqrt{\frac{(3^4 + 5 \times 6^7)}{8}}$$

【输入】

没有输入。

【输出】

输出结果小数点后保留 3 位。

【提示】

**为幂运算符，x ** 0.5 为求 x 的平方根。

参考答案：

```
x = ((3 ** 4 + 5 * (6 ** 7)) / 8) ** 0.5
print("%.3f" % (x))
```

【描述】

编写程序，从键盘输入两个整数，计算并输出这两个整数的和、差、积、商。

【输入】

分行输入两个整数。

【输出】

分行输出两个整数的和、差、积、商。

【输入示例】

```
5
3
```

【输出示例】

```
5 + 3 = 8
5 - 3 = 2
5 * 3 = 15
5 / 3 = 1.6666666666666667
```

【来源】

《Python 程序设计基础》第 1 章编程题 3。

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
a = int(input())
b = int(input())
print(a, "+", b, "=", a + b)
print(a, "-", b, "=", a - b)
print(a, "*", b, "=", a * b)
print(a, "/", b, "=", a / b)
```

【描述】

编写程序，给定一个摄氏温度 C，计算对应的华氏温度 F。转换的公式如下：

$$F = \frac{9}{5}C + 32$$

【输入】

在一行中给出一个摄氏温度。

【输出】

在一行中输出对应的华氏温度，精确到小数点后 1 位。

【输入示例】

```
100
```

【输出示例】

```
212.0
```

【来源】

《Python 程序设计基础》第 1 章编程题 4。

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
c = float(input())
f = 9 / 5 * c + 32
print(format(f, ".1f"))
```

【描述】

编写程序，从键盘输入矩形的宽度和高度，计算矩形的面积。结果保留 2 位小数。

【输入】

分行输入矩形宽度和高度。

【输出】

在一行中输出矩形面积，结果保留 2 位小数。

【输入示例】

```
2.5
3.5
```

【输出示例】

```
8.75
```

【来源】

《Python 程序设计基础》第 1 章编程题 5。

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
width = float(input())
height = float(input())
print("%.2f" % (width * height))
```

【描述】

输入一个圆环的内外半径，计算圆环的面积。inside 和 outside 分别表示圆环的内外半径，题目保证外半径大于内半径。

假设π为 3.14159。

【输入】

输入圆环的外半径和内半径。

【输出】

输出对应的圆环面积，结果保留 2 位小数。

【输入示例】

```
3.5
2.5
```

【输出示例】

```
18.85
```

【提示】

可以在下面的编辑框中直接编写程序并保存。也可以在本机编写调试程序，然后将程序复制粘贴至编辑框并保存。

程序中不要有任何用户友好性提示等的输出，只能严格按照题目中所要求的输出格式来输出。可以运行自己的程序，用题目中的输入示例来输入，如果得到的输出和输出示例完全相同，一个字符也不多，一个字符也不少，那么这样的格式就是对的了。

参考答案：

```
outside = float(input())
inside = float(input())
PI = 3.14159
area = PI * (outside * outside - inside * inside)
print("%.2f" % (area))
```

【描述】

慧慧有 5 元钱，她想去买冰棍吃，冰棍的价格各不相同，根据冰棍的价格，计算慧慧最多能买多少根冰棍。

【输入】

一个数，表示一根冰棍的价格，单位是元。

【输出】

一个整数，小明最多能买到的冰棍数。

【输入示例】

```
1.3
```

【输出示例】

```
3
```

参考答案：

```
x = float(input())
print(int(5 / x))
```

【描述】

分行输入三个值 a、b、c，输出如下公式的值。

$$b^2 - 4ac$$

【输入】

分行输入三个整数。

【输出】

在一行中输出公式值。

【输入示例】

```
3
4
5
```

【输出示例】

```
-44
```

参考答案：

```
a, b, c = int(input()), int(input()), int(input())
print(b * b - 4 * a * c)
```

描述】

计算 a+b。a、b 为整数。

【输入】

在一行上输入 a、b，其间以空格间隔。

【输出】

输出一行，a+b 的值。

【输入示例】

```
5 3
```

【输出示例】

```
8
```

参考答案：

```
a, b = map(int, input().split())
print(a + b)
```

描述】

编写程序，从键盘输入两个整数，存放在变量 a 和 b 中，并交换 a 和 b 中的值。

【输入】

一行中给出整数 a 和 b，其间以空格分隔。

【输出】

一行中输出交换后的整数 a 和 b，其间以空格分隔。

【输入示例】

```
5 3
```

【输出示例】

```
3 5
```

【来源】

《Python 程序设计基础》第 2 章编程题 2。

参考答案：

```
a, b = map(int, input().split())
a, b = b, a
print(a, b)
```

【描述】

编写程序，从键盘输入两个整数，计算并输出这两个整数的和、平均值、最小值和最大值。平均值保留 2 位小数。

【输入】

分行输入两个整数。

【输出】

分行输出两个整数的和、平均值、最小值和最大值。平均值保留 2 位小数。

【输入示例】

```
5
4
```

【输出示例】

```
9
4.50
4
5
```

【来源】

《Python 程序设计基础》第 2 章编程题 1。

参考答案：

```
num1 = int(input())
num2 = int(input())
total = num1 + num2
print("%d" % (total))
average = total / 2.0;
print("%.2f" % (average))
print("%d" % min(num1, num2))
```

```
print("%d" % max(num1, num2))
```

【描述】

编写程序，读入一个在 100 和 999 之间的整数，然后输出按位逆序后的数。当输入的整数含有结尾的 0 时，输出不应带有前导的 0。比如输入 100，输出应该是 1。

【输入】

在一行中给出一个在 100 和 999 之间的整数。

【输出】

在一行中输出按位逆序后的数。

【输入示例】

```
123
```

【输出示例】

```
321
```

【来源】

《Python 程序设计基础》第 2 章编程题 3。

【提示】

数的各位分离是指将整数 n 的每一位数取出，在取数的过程中，反复运用'%'和'//'运算符，"n % 10"运算可以取出整数 n 的个位数，而"n //10"运算可以将整数 n 的十位数移至个位数、百位数移至十位数、……，反复运用这两个表达式就可以取出整数 n 的每一位数。

参考答案：

```
n = int(input())
a = n % 10
b = (n // 10) % 10
c = n // 100
n = a * 100 + b * 10 + c
print(n)
```

【描述】

编写程序，从键盘输入 a，计算表达式

$$\frac{\cos(50°)+\sqrt{37.5}}{a+1}$$

的值，a≠-1。结果保留 2 位小数。

【输入】

在一行中输入 a 的值。

【输出】

在一行中输出表达式的值，结果保留 2 位小数。

【输入示例】

```
2
```

【输出示例】

```
2.26
```

【来源】

《Python 程序设计基础》第 2 章编程题 4。

参考答案：

```
import math
a = float(input())
print("%.2f" % ((math.cos(50 * math.pi / 180) + math.sqrt(37.5))/(a+1)))
```

【描述】

编写程序，输入存款（money）、存期（year）和年利率（rate），计算存款到期时的税前利息（interest）。结果保留 2 位小数。公式如下：

$$\text{interest} = \text{money}(1 + rate)^{year} - money$$

【输入】

分行输入存款（money）、存期（year）和年利率（rate）。

【输出】

在一行中输出存款到期时的税前利息（interest），结果保留 2 位小数。

【输入示例】

```
10000
3
0.025
```

【输出示例】

```
768.91
```

【来源】

《Python 程序设计基础》第 2 章编程题 5。

参考答案：

```
money = float(input())
year = int(input())
rate = float(input())
interest = money * (1 + rate) ** year - money
print("%.2f" % (interest))
```

【描述】

将一个大写字母转换为小写字母。

【输入】

输入一个字母。

【输出】

输出对应的小写字母。注意：若输入的是小写字母，则直接输出。

【输入示例】

A

【输出示例】

a

参考答案：

```
ch = input()
print(ch.lower())
```

【描述】

编写程序，顺序读入浮点数 1、整数、字符、浮点数 2，再按照字符、整数、浮点数 1、浮点数 2 的顺序输出。

【输入】

在一行中顺序给出浮点数 1、整数、字符、浮点数 2，其间以空格分隔。

【输出】

在一行中按照字符、整数、浮点数 1、浮点数 2 的顺序输出，其中浮点数保留小数点后 2 位。

【输入示例】

```
2.12 88 c 4.7
```

【输出示例】

```
c 88 2.12 4.70
```

参考答案：

```
line = input().split()
d1 = float(line[0])
i = int(line[1])
ch = line[2]
d2 = float(line[3])
print(ch, i, format(d1, ".2f"), format(d2, ".2f"))
```

【描述】

输入一个整数和进制，转换成十进制输出。

【输入】

在一行中输入整数和进制。注意，整数可能是十六进制的。

【输出】

在一行中十进制输出结果。

【输入示例】

```
45,8
```

【输出示例】

```
37
```

【提示】

可以使用 int(str, base)函数，将 str 字符串中 base 进制的整数转换为十进制整数。

参考答案：

```
line = input().split(',')
print(int(line[0], int(line[1])))
```

【描述】

牛牛最近学习了 Python 语言入门课程，这门课程的总成绩计算方法是：总成绩=作业成绩×20%+小测成绩×30%+期末考试成绩×50%。牛牛想知道，这门课程自己最终能得到多少分？

【输入】

在一行中包含三个非负整数，，其间以空格分隔，分别表示牛牛的作业成绩、小测成绩和期末考试成绩，三项成绩满分都是 100 分。

【输出】

一行一个整数，即牛牛这门课程的总成绩，满分也是 100 分。

【输入示例】

```
60 90 80
```

【输出示例】

```
79
```

参考答案：

```
a, b, c = map(int, input().split())
total = a * 0.2 + b * 0.3 + c * 0.5
print("%d" % (total))
```

【描述】

编写程序，从键盘输入一个整数，检查它是否能同时被 2 和 3 整除，是否被 2 或 3 整除，是否被 2 或 3 整除且只被其一整除。

【输入】

一行中给出一个整数。

【输出】

分行输出检查结果，格式见【输出示例】。

【输入示例】

```
4
```

【输出示例】

```
4 divisible by 2 and 3? False

4 divisible by 2 or 3? True

4 divisible by 2 or 3, but not both? True
```

【来源】

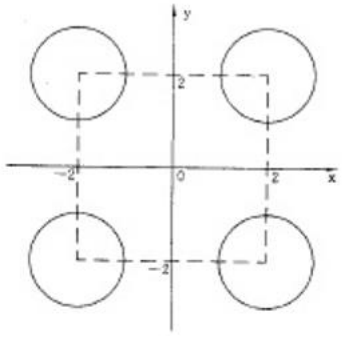
《Python 程序设计基础》第 3 章编程题 1。

参考答案：

```
n = int(input())

print(n, "divisible by 2 and 3?", n % 2 == 0 and n % 3 == 0)

print(n, "divisible by 2 or 3?", n % 2 == 0 or n % 3 == 0)

print(n, "divisible by 2 or 3, but not both?", (n % 2 == 0 or n % 3 == 0)
 and not (n % 2 == 0 and n % 3 == 0))
```

【描述】

有 4 个圆塔，圆心分别为(2，2)、(-2，2)、(2，-2)、(-2，-2)，圆半径为 1。这 4 个塔的高度为 10 米。塔以外无建筑物。请编写程序，输入任一点的坐标，求该点的建筑高度（塔外的高度为零）。



【输入】

输入为浮点数，任一点的坐标，以逗号隔开。

【输出】

输出为整数，该点的建筑高度。

【输入示例】

```
2,2.5
```

【输出示例】

```
10
```

参考答案：

```
line = input().split(',')
x = float(line[0])
y = float(line[1])
if (x - 2) ** 2 + (y - 2) ** 2 <= 1 or (x + 2) ** 2 + (y - 2) ** 2 <= 1 or
 \
   (x + 2) ** 2 + (y + 2) ** 2 <= 1 or (x - 2) ** 2 + (y + 2) ** 2 <= 1:
    print("10")
else:
    print("0")
```

【描述】

编写程序，键盘输入 x，求如下分段函数 y 的值（结果保留 2 位小数）。

$$y=\begin{cases}x^2 & x<0\\ \sqrt{x} & 0\le x<9\\ x-6 & x\ge 9\end{cases}$$

【输入】

在一行中给出给出 x 的值。

【输出】

在一行中输出 y 的值，结果保留 2 位小数。

【输入示例】

```
2.5
```

【输出示例】

```
1.58
```

【来源】

《Python 程序设计基础》第 3 章编程题 2。

参考答案：

```
x = float(input())
if x < 0:
    y = x * x
elif x < 9:
    y = x ** 0.5
else:
    y = x - 6
print("%.2f" % (y))
```

【描述】

某电网执行阶梯电价，安装一户一表的居民用户电价分为两个阶梯：月用电量 50 千瓦时（含 50 千瓦时）以内的，电价为 0.53 元/千瓦时；超过 50 千瓦时的，超出部分的用电量，电价上调 0.05 元/千瓦时。请编写程序计算电费。

【输入】

在一行中给出某用户的月用电量（单位：千瓦时）。

【输出】

在一行中输出该用户应支付的电费（元），结果保留两位小数，格式：cost = 应付电费值；若用电量小于 0，则输出"Invalid Value!"。

【输入示例】

```
10
```

【输出示例】

```
cost = 5.30
```

参考答案：

```
x = float(input())
if x < 0:
    print("Invalid Value!")
elif x <= 50:
    cost = 0.53 * x
    print("cost = %.2f" % (cost))
else:
    cost = 50 * 0.53 + (x - 50) * (0.53 + 0.05)
```

```
    print("cost = %.2f" % (cost))
```

【描述】

编写程序，输入 a、b 和 c，若它们能构成三角形，则输出三角形周长，否则输出"Invalid"。

【输入】

一行中给出 a、b 和 c，其间以空格分隔。

【输出】

一行中输出三角形周长或"Invalid"。

【输入示例】

```
3 3 3
```

【输出示例】

```
9.0
```

【提示】

任意两边之和大于第三边，则三条边构成三角形。

三角形的边长不一定是整数。

参考答案：

```
line = input().split()
a = float(line[0])
```

```
b = float(line[1])
c = float(line[2])
if  a + b > c and  a + c > b and b + c > a:
    print(a + b + c)
else:
    print("Invalid")
```

【描述】

某公司员工的工资计算方法如下：一周内工作时间不超过 40 小时，按正常工作时间计酬；超出 40 小时的工作时间部分，按正常工作时间报酬的 1.5 倍计酬。员工按进公司时间分为新职工和老职工，进公司不少于 5 年的员工为老职工，5 年以下的为新职工。新职工的正常工资为 30 元/小时，老职工的正常工资为 50 元/小时。请按该计酬方式计算员工的工资。

【输入】

在一行中给出 2 个正整数，分别为某员工入职年数和周工作时间，其间以空格分隔。

【输出】

在一行输出该员工的周薪，精确到小数点后 2 位。

【输入示例】

```
5 40
```

【输出实例】

```
2000.00
```

参考答案：

```
year, time = map(int, input().split())
if year < 5:
    if time <= 40:
        wage = time * 30
    else:
        wage = 40 * 30 + (time - 40) * 30 * 1.5
else:
    if time <= 40:
        wage = time * 50
    else:
        wage = 40 * 50 + (time - 40) * 50 * 1.5
print("%.2f" % wage)
```

【描述】

将输入的任意 3 个整数从小到大输出。

【输入】

在一行中给出 3 个整数，其间以空格分隔。

【输出】

在一行中将 3 个整数从小到大输出，其间以“->”相连。

【输入示例】

```
4 2 8
```

【输出示例】

```
2->4->8
```

参考答案：

```
a, b, c = map(int, input().split())
if a > b:
    a, b = b, a
if a > c:
    a, c = c, a
if b > c:
    b, c = c, b
print("%d->%d->%d" % (a, b, c))
```

【描述】

编写程序，根据输入的运算符，对 2 个整数进行加、减、乘、除或求余运算。

【输入】

在一行中依次给出操作数 1、运算符、操作数 2，其间以空格分隔。操作数的数据类型为整型，且保证除法和求余的分母非零。

【输出】

当运算符为+、-、*、/、%时，在一行中输出相应的运算结果。当运算符为/时，实际是做整除（//）运算。若输入是非法符号（即除了加、减、乘、除和求余五种运算符以外的其他符号）则输出"Error"。

【输入示例】

```
-7 / 2
```

【输出示例】

```
-4
```

参考答案：

```
line =input().split()
a = int(line[0])
op = line[1]
b = int(line[2])
if op == '+':
    print(a + b)
elif op == '-':
    print(a - b)
elif op == '*':
    print(a * b)
elif op == '/':
    print(a // b)
elif op == '%':
    print(a % b)
else:
    print("Error")
```

【描述】

股票价格涨跌趋势，常用蜡烛图技术中的 K 线图来表示，分为按日的日 K 线、按周的周 K 线、按月的月 K 线等。以日 K 线为例，每天股票价格从开盘到收盘走完一天，对应一根蜡烛小图，要表示四个价格：开盘价格 Open（早上刚刚开始开盘买卖成交的第 1 笔价格）、收盘价格 Close（下午收盘时最后一笔成交的价格）、中间的最高价 High 和最低价 Low。

如果 Close<Open，表示为"BW-Solid"（即"实心蓝白蜡烛"）；如果 Close>Open，表示为"R-Hollow"（即"空心红蜡烛"）；如果 Open 等于 Close，则为"R-Cross"（即"十字红蜡烛"）。如果 Low 比 Open 和 Close 低，称为"Lower Shadow"（即"有下影线"），如果 High 比 Open 和 Close 高，称为"UpperShadow"（即"有上影线"）。请编程序，根据给定的四个价格组合，判断当日的蜡烛是一根什么样的蜡烛。

【输入】

在一行中给出 4 个正实数，分别对应 Open、High、Low、Close，其间以空格分隔。

【输出】

在一行中输出日 K 蜡烛的类型。如果有上、下影线，则在类型后加上"with 影线类型"。如果两种影线都有，则输出"with Lower Shadow and Upper Shadow"。

【输入示例】

```
5.110 5.250 5.100 5.105
```

【输出示例】

```
BW-Solid with Lower Shadow and Upper Shadow
```

参考答案：

```
import math
EPSILON = 1e-6
line= input().split()
open = float(line[0])
high = float(line[1])
low = float(line[2])
close = float(line[3])
if close < open:
    print("BW-Solid", end = '')
elif close > open:
    print("R-Hollow", end = '')
elif math.fabs(close - open) <= EPSILON:
    print("R-Cross", end = '')
if (low < open and low < close) and (high > open and high > close):
    print(" with Lower Shadow and Upper Shadow")
elif (low < open and low < close):
    print(" with Lower Shadow")
elif (high > open and high > close):
    print(" with Upper Shadow")
```

【描述】

编写程序，输入年、月、日，显示它是一周中的星期几。

蔡勒（ChristianZeller）公式是用于计算某天是星期几的算法，这个公式如下：

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + k + \left\lfloor \frac{k}{4} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor + 5j\right) \%7$$

其中：

h 是一个星期中的某一天（0 为星期六，1 为星期日，2 为星期一，3 为星期二，4 为星期三，5 为星期四，6 为星期五）。

q 是某月的天数。

m 是月份（3 为三月，4 为四月，…，12 为十二月），一月和二月分别记为上一年的十三和十四月。所以需要将输入的月份 1 转换为 13，输入的月份 2 转换为 14，同时将年份改为前一年。

j 是世纪数，即

$$\left\lfloor \frac{\text{year}}{100} \right\rfloor$$

k 是世纪的年数，即 year%100。

$$\lfloor n \rfloor = (int)n$$

【输入】

一行中输入年、月、日，其间以空格分隔。

【输出】

一行中输出星期几（用英文单词表示，首字母大写）。

【输入示例】

```
2002 3 26
```

【输出示例】

```
Tuesday
```

参考答案：

```
line = input().split()
year = int(line[0])
month = int(line[1])
day = int(line[2])
if month == 1 or month == 2:
    month += 12
    year -= 1
j = year // 100
k = year % 100
h = (day + (26 * (month + 1)) // 10 + k + k // 4 + j // 4 + 5 * j) % 7
if h == 0:
    print("Saturday")
elif h == 1:
    print("Sunday")
elif h == 2:
    print("Monday")
elif h == 3:
    print("Tuesday")
elif h == 4:
    print("Wednesday")
elif h == 5:
```

```
    print("Thursday")
elif h == 6:
    print("Friday")
```

【描述】

计算 a+b。a 和 b 为整数。

【输入】

输入有多行。

每一行上有 a 和 b 两个整数，其间以空格间隔。

【输出】

分行输出对应的 a+b 的值。

【输入示例】

```
5 3
10 20
```

【输出示例】

```
8
30
```

参考答案：

```
for line in iter(input, ''):
    a, b = map(int, line.split())
    print(a + b)
```

【描述】

计算 a+b。a、b 为整数。

【输入】

第一行为正整数 n，表示下面有 n 行数据。

接着有 n 行数据。每一行上有 a 和 b 两个整数，其间以空格间隔。

【输出】

输出有 n 行。

分行输出对应的 a+b 的值。

【输入示例】

```
2
5 3
10 20
```

【输出示例】

```
8
30
```

参考答案：

```
n = int(input())
while n > 0:
```

```
    a, b = map(int, input().split())
    print(a + b)
    n -= 1
```

【描述】

计算整数和。

【输入】

输入有多行。

每一行上，第 1 个数是正整数 n，表示后面同一行上有 n 个整数，整数之间均以空格间隔。

最后一行上为 0，表示输入结束。

【输出】

分行输出对应的整数和。

【输入示例】

```
4 1 2 3 4
5 1 2 3 4 5
0
```

【输出示例】

```
10
```

```
15
```

参考答案：

```
while True:
    line = input().split()
    n = int(line[0])
    if n == 0:
        break
    total = 0
    while n > 0:
        total += int(line[n])
        n -= 1
    print(total)
```

【描述】

计算整数和。

【输入】

第一行为正整数 n，表示下面有 n 行数据。

每一行上，第 1 个数是正整数 m，表示后面同一行上有 m 个整数，整数之间均以空格间隔。

【输出】

输出有 n 行，分行输出对应的整数和。

【输入示例】

```
2
4 1 2 3 4
5 1 2 3 4 5
```

【输出示例】

```
10
15
```

参考答案：

```
n = int(input())
while n > 0:
    line = input().split()
    m = int(line[0])
    total = 0
    while m > 0:
        total += int(line[m])
        m -= 1
    print(total)
    n -= 1
```

【描述】

计算如下式子：

$$1+\frac{1}{3}+\frac{1}{5}+\ldots\ldots$$

前 n 项之和，结果保留 3 位小数。

【输入】

输入一个正整数 n。

【输出】

输出数列前 n 项的和，结果保留 3 位小数。

【输入示例】

```
10
```

【输出示例】

```
2.133
```

【来源】

《Python 程序设计基础》第 3 章编程题 5。

参考答案：

```
n = int(input())
total = 0
for i in range(1, n + 1):
    total += 1 / (2 * i - 1)
print("%.3f" % (total))
```

【描述】

读入 2 个正整数 a 和，1≤a≤9, 1≤b≤10，产生整数 aa...a，一共 b 个 a。

【输入】

在一行中输入 a 和 b。

【输出】

在一行中输出整数 aa...a，一共 b 个 a。

【输入示例】

```
1,5
```

【输出示例】

```
11111
```

参考答案：

```
a, b = map(int, input().split(','))
total = 0
for i in range(b):
    total = total * 10 + a
print(total)
```

【描述】

计算如下式子：

$$s_n = a + aa + aaa + \cdots\cdots + \overbrace{aa\cdots\cdots a}^{n个a}$$

的值。

例如，a 为 2，n 为 5，则式子的值为 24690（2+22+222+2222+22222）。

【输入】

输入 a 和 n。

【输出】

输出式子的值。

【输入示例】

```
2 5
```

【输出示例】

```
24690
```

【来源】

《Python 程序设计基础》第 3 章编程题 6。

参考答案：

```
line = input().split()
a = int(line[0])
n = int(line[1])
item = a;
total = a;
for i in range(2, n + 1):
    item = item * 10 + a
    total += item
```

```
print(total)
```

【描述】

编写程序，计算并输出下式的值，计算到最后一项的值小于 0.000001 时为止，结果保留 6 位小数。

$$s = 1 - \frac{1}{4} + \frac{1}{7} - \frac{1}{10} + \frac{1}{13} - \frac{1}{16} + \cdots$$

【输入】

没有输入。

【输出】

显示表达式的值，精确到小数点后 6 位。

【来源】

《Python 程序设计基础》第 3 章编程题 7。

参考答案：

```
EPSILON = 1e-6
n = 1
sign = 1
item = 1
total = 0.0
while abs(item) >= EPSILON:
    total += item
    sign = -sign
    n += 1
```

```
    item = 1.0  * sign / (3 * n - 2)
print("%.6f" % (total))
```

【描述】

给定两个整数 a 和 b，输出从 a 到 b 的所有整数以及这些整数的和。

【输入】

在一行中给出 2 个整数 a 和 b，其中-100≤a≤b≤100，其间以空格分隔。

【输出】

首先顺序输出从 a 到 b 的所有整数，每 5 个数字占一行，每个数字占 5 个字符宽度，向右对齐。最后在一行中按 Sum = x 的格式输出全部数字的和 x。

【输入示例】

```
-3 8
```

【输出示例】

```
   -3   -2   -1    0    1
    2    3    4    5    6
    7    8
Sum = 30
```

参考答案：

```
a, b = map(int, input().split())
count = 0
```

```
total = 0
for i in range(a, b + 1):
    total += i
    print("%5d" % (i), end = '')
    count += 1
    if count % 5 == 0 and i != b:
        print()
print("\nSum = %d" % (total))
```

【描述】

某工地需要搬运砖块，已知男人一人搬 3 块，女人一人搬 2 块，小孩两人搬 1 块。用 45 人正好搬 45 块砖，问有多少种搬法？

【输入】

没有输入。

【输出】

输出搬砖的男人、女人和小孩数。

若有多组答案，则分行输出各组答案。

【输入示例】

没有输入。

【输出示例】

```
A B C
```

【来源】

《Python 程序设计基础》第 3 章编程题 9。

【提示】

输出示例只是格式说明，并非正确答案。A、B、C 分别表示男人、女人、小孩数。

可能有多组答案。

参考答案：

```
for men in range(16):
    for women in range(23):
        child = 45 - men - women
        if men * 3 + women * 2 + child * 0.5 == 45:
            print(men, women, child)
```

【描述】

如果四边形四条边的长度分别为 a、b、c、d，一对对角之和为 2α，则其面积为：

$$\text{area} = \sqrt{(p-a)(p-b)(p-c)(p-d) - abcdcos^2\alpha}$$

$$\text{p} = \frac{1}{2}(a+b+c+d)$$

定义函数：def compute_area(a, b, c, d, alpha)，计算任意四边形的面积。

编写一个 main 函数，设有一个四边形，其四条边边长分别为 3、4、5、5，一对对角之和为 145°，计算它的面积。结果保留 2 位小数。

【输入】

没有输入。

【输出】

输出对应的四边形面积，结果保留 2 位小数。

【来源】

《Python 程序设计基础》第 4 章编程题 1。

参考答案：

```
from math import cos

def compute_area(a, b, c, d, alpha):

    p = (a + b + c + d) * 0.5

    area = ((p - a) * ( p - b) * (p - c) * (p - d) - a * b * c * d * cos(a
lpha) * cos(alpha)) ** 0.5

    return area;


def main():

    PI = 3.14159

    a = 3

    b = 4

    c = d = 5

    alpha = 145 * PI / 360

    print("%.2f" % (compute_area(a, b, c, d, alpha)))


main()
```

【描述】

求一个整数的逆序数。定义函数：def reverse(n)，该函数返回一个整数的逆序数。

当整数含有结尾的 0 时，输出不应带有前导的 0。比如输入 100，输出应该是 1。

编写一个 main 函数，输入一个整数，调用 reverse 函数，显示该整数的逆序数。

【输入】

一行中给出一个整数。

【输出】

一行中输出该整数的逆序数。

【输入示例】

```
-123
```

【输出示例】

```
-321
```

【来源】

《Python 程序设计基础》第 4 章编程题 2。

参考答案：

```
def reverse(n):
    sign = 1
    result = 0
    if n < 0:
        n = -n
        sign = -1
    while n != 0:
        remainder = n % 10
```

```
        result = result * 10 + remainder
        n //= 10
    return sign * result


def main():
    value = int(input())
    print(reverse(value))


main()
```

【描述】

编写程序，计算如下序列的值。结果保留 4 位小数。

$$\mathrm{m(i)} = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots + \frac{(-1)^{i+1}}{2i-1})$$

定义函数：def m(i)，返回序列的值。

编写一个 main 函数，调用 m 函数，并输出序列的值。

【输入】

在一行中给出一个正整数，表示序列的项数。

【输出】

在一行中输出序列的值，结果保留 4 位小数。

【输入示例】

```
901
```

【输出示例】

```
3.1427
```

【来源】

《Python 程序设计基础》第 4 章编程题 3。

参考答案：

```
def m(n):
    total = 0
    flag = 1
    for i in range(1, n + 1):
        item = 1 / (2 * i - 1)
        total += flag * item
        flag = -flag
    return total


def main():
    i = int(input())
    print("%.4f" % (4 * m(i)))


main()
```

【描述】

定义 total 函数，返回若干个整数的和，体会函数默认参数的使用。

【输入】

没有输入。

【输出】

```
100
106
16
36
```

【提示】

根据 total 函数被调用时实参的使用和结果的输出，推测 total 函数的定义。

参考答案：

```
def total(num1 = 0, num2 = 100, num3 = 0):
    return num1 + num2 + num3


def main():
    print(total())
    print(total(6))
    print(total(6, 10))
    print(total(6, 10, 20))


main()
```

【描述】

编写程序，定义和调用函数：def count_digit(number, digit)，number 是整数，digit 为[1, 9]区间内的整数，返回 number 中 digit 出现的次数。

【输入】

一行中输入两个整数，以空格间隔。

【输出】

见【输出示例】

【输入示例】

```
-21252 2
```

【输出示例】

```
Number of digit 2 in -21252: 3
```

参考答案：

```
def count_digit(number, digit):
    t = abs(number)
    cnt = 0;
    while t > 0:
        r = t % 10
        if r == digit:
            cnt += 1
        t //= 10
    return cnt
```

```
def main():
    n, d = map(int, input().split())
    c = count_digit(n, d)
    print("Number of digit {0} in {1}: {2}".format(d, n, c))

main()
```

【描述】

输入一个整数，判断它是否是回文整数。如果一个整数的逆序数和原数一样，这个整数就称为回文整数

定义函数：def is_palindrome(n)，如果 n 是回文数，返回 True，否则返回 False。

编写一个 main 函数，输入一个整数，调用 is_palindrome 函数，判断该整数是否为回文整数。如果该整数是回文数，输出 True，否则输出 False。

【输入】

一行中给出一个整数。

【输出】

如果该整数是回文数，输出 True，否则输出 False。

【输入示例】

```
616
```

【输出示例】

```
True
```

参考答案：

```
def is_palindrome(n):
    result = 0
    t = n
    while t != 0:
        remainder = t % 10
        result = result * 10 + remainder
        t //= 10
    return n == result


def main():
    value = int(input())
    print(is_palindrome(value))


main()
```

【描述】

编写程序，输出前 n（n≤100）个回文素数，每行显示 5 个，并适当对齐。回文素数是这样一种素数：本身为素数且也是回文数。例如，131 是素数，也是回文数。

定义函数：def is_prime(n)，判断 n 是否是素数，如果是素数，返回 True，否则返回 False。

定义函数：def is_palindrome(n)，判断 n 是否是回文数，如果 n 是回文数，返回 True，否则返回 False。

编写一个 main 函数，输入一个正整数 n，调用 is_prime 和 is_palindrome 函数，输出前 n 个回文素数。

【输入】

一行中给出一个正整数 n。

【输出】

输出前 n 个回文素数，每行显示 5 个，每个宽度为 6。

【输入示例】

```
10
```

【输出示例】

```
     2     3     5     7    11
   101   131   151   181   191
```

【提示】

对于整数 i，如果 is_prime(i)与 is_palindrome(i)的返回值都是 True，i 即为回文素数。

参考答案：

```
def is_prime(n):
    flag = True
    for divisor in range(2, n // 2 + 1):
        if n % divisor == 0:
            flag = False
            break
    return flag
```

```
def is_palindrome(n):
    result = 0
    if n < 0:
        n = -n
    t = n
    while t != 0:
        remainder = t % 10
        result = result * 10 + remainder
        t //= 10
    return n == result

def main():
    n = int(input())
    count = 0
    i = 2
    while count < n:
        if is_prime(i) and is_palindrome(i):
            print(format(i, "6d"), end = '')
            count += 1
            if count % 5 == 0:
                  print()
        i += 1

main()
```

【描述】

无暇素数 emirp（英文素数 prime 的逆序）是这样一种素数：本身为素数，且其逆序数也是素数。例如，17 是素数，其逆序数 71 也是，因此 17 和 71 是 emirp。

定义函数：def is_prime(n)，判断 n 是否是素数，如果是素数，函数返回 True，否则返回 False。

定义函数：def reverse(n)，求 n 的逆序数，函数返回 n 的逆序数，如 n 为 17，函数 reverse(n)返回 71。

编写一个 main 函数，调用 is_prime 和 reverse 函数，输出前 n 个 emirp，每行显示 5 个，并适当对齐。

【输入】

输入一个正整数 n。

【输出】

输出前 n 个 emirp，每行显示 5 个，每个宽度为 5。

【输入示例】

```
10
```

【输出示例】

```
    2    3    5    7   11
   13   17   31   37   71
```

【提示】

对于整数 i，如果 is_prime(i)与 is_prime(reverse(i))的返回值都是 True，i 即为无暇素数。

参考答案：

```
def is_prime(n):
```

```
    flag = True
    for divisor in range(2, n // 2 + 1):
        if n % divisor == 0:
            flag = False
            break
    return flag


def reverse(n):
    sign = 1
    result = 0
    if n < 0:
        n = -n
        sign = -1
    while n != 0:
        remainder = n % 10
        result = result * 10 + remainder
        n //= 10
    return sign * result


def main():
    n = int(input())
    count = 0
    i = 2
    while count < n:
        if is_prime(i) and is_prime(reverse(i)):
            print(format(i, "5d"), end = '')
            count += 1
            if count % 5 == 0:
                  print()
        i += 1
```

```
main()
```

【描述】

定义函数：def square_root(value)，返回 value 的平方根。

编写一个 main 函数，读入一个数，调用 square_root 函数，输出该数的平方根。

注意：不能使用数学函数 sqrt 和幂运算符**。

可以通过对下面公式的反复计算近似地得到平方根：

$$\text{nextGuess} = \frac{1}{2}(\text{lastGuess} + \frac{\text{value}}{\text{lastGuess}})$$

当 nextGuess 和 lastGuess 几乎相同时，nextGuess 就是平方根的近似值。nextGuess 最初的猜测值可以是任意一个值（例如 1.0）。这个值就是 lastGuess 的初始值。如果 nextGuess 和 lastGuess 的差小于一个很小的数（例如 0.000001），就可以认为 nextGuess 是 value 平方根的近似值；否则，nextGuess 就赋值给 lastGuess，迭代过程继续进行。

【输入】

一行中给出一个数。

【输出】

输出该数的平方根，结果保留 6 位小数。。

【输入示例】

```
2
```

【输出示例】

```
1.414214
```

参考答案：

```
import math


def square_root(value):
    next_guess = 1.0
    last_guess = 0.0
    while math.fabs(next_guess - last_guess) >= 0.000001:
        last_guess = next_guess
        next_guess = (last_guess + (value / last_guess)) * 0.5
    return next_guess


def main():
    value = float(input())
    print(format(square_root(value), ".6f"))


main()
```

【描述】

定义函数：def f(x, n)，用递归求下列数学式子的值，其中 x 为浮点数，n 为整数。

$$f(x, n) = x - x^2 + x^3 - x^4 + \cdots (-1)^{n-1} x^n, n > 0$$

编写一个 main 函数，读入 x 和 n，输出数学式子的值，结果保留 2 位小数。

【输入】

一行中给出两个数，其间以空格分隔。

【输出】

输出数学式子的值，结果保留 2 位小数。

【输入示例】

```
2 3
```

【输出示例】

```
6.00
```

参考答案：

```
def f(x, n):
    if n == 1:
        return x
    else:
        return x * (1 - f(x, n - 1))


def main():
    line = input().split()
    x = float(line[0])
    n = int(line[1])
    print("%.2f" % (f(x, n)))


main()
```

【描述】

编写程序，统计并输出某给定字符在给定字符串中出现的次数。

【输入】

第一行给出一个以回车结束的字符串；第二行输入一个字符。

【输出】

在一行中输出给定字符在给定字符串中出现的次数。

【输入示例】

```
programming is More fun!
m
```

【输出示例】

```
2
```

参考答案：

```
s = input()
ch = input()
print(s.count(ch))
```

【描述】

编写程序，从给定字符串中查找某指定的字符。

【输入】

第一行是一个待查找的字符。第二行是一个以回车结束的非空字符串。

【输出】

如果找到，在一行内按照格式"index = 下标"输出该字符在字符串中所对应的最大下标（下标从 0 开始）；否则输出"Not Found"。

【输入示例】

```
m
programming
```

【输出示例】

```
index = 7
```

参考答案：

```
ch = input()
s = input()
r = s.rfind(ch)
print(("index = %d" % (r)) if r != -1 else "Not Found")
```

【描述】

编写程序，要求提取一个字符串中的所有数字字符（'0'.....'9'），将其转换为一个整数输出。

【输入】

在一行中给出一个以回车结束的字符串。

【输出】

在一行中输出转换后的整数。

【输入示例】

```
free82jeep5
```

【输出示例】

```
825
```

参考答案：

```
s = input()
t = 0
for ch in s:
    if ch.isdigit():
        t = t * 10 + (ord(ch) - ord('0'))
```

【描述】

脱氧核糖核酸（DNA）由两条互补的碱基链以双螺旋的方式结合而成。而构成 DNA 的碱基共有 4 种，分别为腺瞟呤（A）、鸟嘌呤（G）、胸腺嘧啶（T）和胞嘧啶（C）。在两条互补碱基链的对应位置上，腺瞟呤总是和胸腺嘧啶配对，鸟嘌呤总是和胞嘧啶配对。你的任务就是根据一条单链上的碱基序列，给出对应的互补链上的碱基序列。

【输入】

第一行是一个正整数 n，表明共有 n 条要求解的碱基链。

以下共有 n 行，每行用一个字符串表示一条碱基链。这个字符串只含有大写字母 A、T、G、C，分别表示腺瞟呤、胸腺嘧啶、鸟嘌呤和胞嘧啶。每条碱基链的长度都不超过 255。

【输出】

共有 n 行，每行为一个只含有大写字母 A、T、G、C 的字符串。分别为与输入的各碱基链互补的碱基链。

【输入示例】

```
5
ATATGGATGGTGTTTGGCTCTG
TCTCCGGTTGATT
ATATCTTGCGCTCTTGATTCGCATATTCT
GCGTTTCGTTGCAA
TTAACGCACAACCTAGACTT
```

【输出示例】

```
TATACCTACCACAAACCGAGAC
AGAGGCCAACTAA
TATAGAACGCGAGAACTAAGCGTATAAGA
CGCAAAGCAACGTT
AATTGCGTGTTGGATCTGAA
```

【来源】

《Python 程序设计基础》第 5 章编程题 1。

参考答案：

```
n = int(input())
for i in range(n):
```

```
    line = input()
    for ch in line:
        if ch == 'A':
            print('T', end='')
        elif ch == 'C':
            print('G', end='')
        elif ch == 'G':
            print('C', end='')
        elif ch == 'T':
            print('A', end='')
    print()
```

【描述】

输入一个字符串，统计并输出该字符串中 26 个英文字母（不区分大小写）出现的次数。

【输入】

输入一个字符串。

【输出】

分行输出 26 个英文字母（不区分大小写）出现的次数。

【输入示例】

```
I am a student.
```

【输出示例】

```
a:2
d:1
e:1
i:1
m:1
n:1
s:1
t:2
u:1
```

【来源】

《Python 程序设计基础》第 5 章编程题 2。

参考答案：

```
line = input()
counts = [0 for i in range(26)]
for ch in line:
    if ch.isalpha():
        if ch.isupper():
            ch = ch.lower()
        counts[ord(ch) - ord('a')] += 1
for i in range(26):
    if counts[i] != 0:
        print("%c:%d" % (chr(ord('a') + i), counts[i]))
```

【描述】

输入 5 个字符串，输出其中最大的字符串。

【输入】

输入 5 个字符串。

【输出】

输出 5 个字符串中最大的字符串。

【输入示例】

```
red
blue
yellow
green
purple
```

【输出示例】

```
yellow
```

【来源】

《Python 程序设计基础》第 5 章编程题 3。

参考答案：

```
string_list = []
for i in range(1, 6):
    line = input()
    string_list.append(line)
max_string = max(string_list)
print(max_string)
```

【描述】

两个单词如果包含相同的字母，次序不同，则称为字母易位词（anagram）。例如，"silent"和"listen"是字母易位词。

定义函数：def is_anagram(s1, s2)，检查两个单词是否是字母易位词，如果是，返回 True；否则返回 False。

【输入】

有两行，分别对应两个单词。

【输出】

若两个单词是字母易位词，输出 True，否则输出 False。

【输入示例】

```
silent
listen
```

【输出示例】

```
True
```

【来源】

《Python 程序设计基础》第 5 章编程题 4。

参考答案：

```
def is_anagram(s1, s2):
```

```
    if len(s1) != len(s2):
        return False
    new_s1 = "".join(sorted(s1))
    new_s2 = "".join(sorted(s2))
    if new_s1 != new_s2:
        return False
    return True


def main():
    str1 = input()
    str2 = input()
    print(is_anagram(str1, str2))


main()
```

【描述】

首字母缩略词是由一个短语中每个单词的第一个字母组成，均为大写。例如，CPU 是短语"central processing unit"的缩写。

定义函数：def is_acronym(s)，s 是短语，返回短语的首字母缩略词。

【输入】

一行中输入一个短语，短语中每个单词以空格间隔。

【输出】

该短语的首字母缩略词。

【输入示例】

```
central processing unit
```

【输出示例】

```
CPU
```

参考答案：

```
def is_acronym(s):
    phrase = ''
    s = s.title()
    lst = s.split()
    for i in range(len(lst)):
        phrase += lst[i][0]
    return phrase


def main():
    phrase=input()
    print(is_acronym(phrase))


main()
```

【描述】

依次计算一系列给定字符串的字母值，字母值为字符串中每个字母对应的编号值（A 对应 1，B 对应 2，以此类推，不区分大小写字母，非字母字符对应的值为 0）的总和。例如，Colin 的字母值为 3+15+12+9+14=53

【输入】

多行字符串，每个字符串占一行。

【输出】

计算并输出每行字符串的字母值。

【输入示例】

```
Colin
ABC
```

【输出示例】

```
53
6
```

参考答案：

```
stop = ''
for s in iter(input, stop):
    total = 0
    for i in range(len(s)):
        if s[i].isalpha():
            total += (ord(s[i].upper()) - ord('A') + 1)
    print(total)
```

【描述】

"distance"和"disinfection"的共有前缀是"dis"。

定义函数：def prefix(s1，s2)，检查两个字符串是否有共有前缀，如果有，返回该共有前缀；否则返回 None。

编写一个 main 函数，输入两个字符串，调用 prefix 函数，显示共有前缀或"No common prefix"。

【输入】

有两行，分别对应两个字符串。

【输出】

若两个字符串有共有前缀，输出该共有前缀，否则输出"No common prefix"。

【输入示例】

```
distance
disinfection
```

【输出示例】

```
dis
```

参考答案：

```
def prefix(s1, s2):
    str = ''
    i = 0
```

```
    while i < len(s1) and i < len(s2):
        if s1[i] == s2[i]:
            str += s1[i]
        else:
            break
        i += 1
    if len(str) == 0:
        return None
    else:
        return str


def main():
    str1 = input()
    str2 = input()
    str = prefix(str1, str2)
    print(str if str != None else "No common prefix")


main()
```

【描述】

输入 10 个整数，存放在列表中，找出值最大和最小的元素，输出最大值、最小值及它们所在的元素下标。

【输入】

在一行中输入 10 个整数，其间以空格分隔。

【输出】

第一行输出最大值及其所在的元素下标，最大值和下标以空格间隔。

第二行输出最小值及其所在的元素下标，最小值和下标以空格间隔。

【输入示例】

```
1 3 5 7 9 6 0 8 2 4
```

【输出示例】

```
9 4
0 6
```

【来源】

《Python 程序设计基础》第 5 章编程题 5。

参考答案：

```
def main():
    line = input().split()
    lst = [int(x) for x in line]
    print(max(lst), lst.index(max(lst)))
    print(min(lst), lst.index(min(lst)))

main()
```

【描述】

给定一组整数，要求利用列表把这组数保存起来，实现对列表中的数循环移动。假定共有 n 个整数，则要使前面各数顺序向后移 m 个位置，并使最后 m 个数变为最前面的 m 个数。

一定要保证在输出结果时，输出的顺序和列表中数的顺序是一致的。

【输入】

第一行包含一个正整数 n 和一个正整数 m，n 和 m 以空格间隔。

第二行包含 n 个正整数，整数以空格间隔。

【输出】

依次输出经过循环移动后列表中元素值，元素值以空格间隔。

【输入示例】

```
11 4
15 3 76 67 84 87 13 67 45 34 45
```

【输出示例】

```
67 45 34 45 15 3 76 67 84 87 13
```

【来源】

《Python 程序设计基础》第 5 章编程题 7。

参考答案：

```
n, m = map(int, input().split())
lst = list(map(int, input().split()))
lst1 = lst[0:-m]
lst2 = lst[n - m:len(lst)]
lst3 = lst2 + lst1
```

```
for i in lst3:
    print(i, end = ' ')
```

【描述】

给定一个整数列表，求列表中第 k 大的数。注意，第 k 大的数意味着从大到小排在第 k 位的数。

【输入】

第一行输入 k。第二行输入 n 个整数，整数之间以空格分隔。

【输出】

该列表中第 k 大的数。

【输入示例】

```
2
4 1 3 2
```

【输出示例】

```
3
```

【来源】

《Python 程序设计基础》第 5 章编程题 8。

参考答案：

```
k = int(input())
```

```
line = input().split()
a = [int(x) for x in line]
a.sort(reverse=True)
print(a[k - 1])
```

【描述】

输入 10 个整数，将 10 个整数按升序排列输出，并且奇数在前，偶数在后。

【输入】

输入 10 个整数，以空格间隔。

【输出】

输出升序排列后，奇数在前，偶数在后的数组元素，以空格间隔。最后一个元素后面没有空格。

【输入示例】

```
10 9 8 7 6 5 4 3 2 1
```

【输出示例】

```
1 3 5 7 9 2 4 6 8 10
```

参考答案：

```
#coding=utf-8
line = input().split()
```

```
lst = [int(x) for x in line]
lst.sort()
result = []
# 奇数
for i in range(len(lst)):
    if lst[i] % 2 != 0:
        result.append(lst[i])
# 偶数
for i in range(len(lst)):
    if lst[i] % 2 == 0:
        result.append(lst[i])
for i in range(len(result) - 1):
    print(result[i], end=' ')
print(result[len(lst) - 1])
```

【描述】

定义函数：def mean_median(t)，该函数接受一个正整数元组作为参数，返回元组中正整数的均值和中位数。一组数据按从小到大的顺序依次排列，位于中间位置的一个数或位于中间位置的两个数的平均值被称为中位数。如果这组数的个数为奇数，则中位数是位于中间位置的数；如果这组数的个数为偶数，则中位数是位于中间位置的两个数的平均值。

【输入】

在一行中顺序输入若干个正整数，其间以空格分隔。

【输出】

输出这些正整数的均值和中位数。

【输入示例】

```
3 3 0 1 12 13 15 16
```

【输出示例】

```
(7.875, 7.5)
```

参考答案：

```
def mean_median(t):
    meanValue = sum(t) / len(t)
    new_t = sorted(t)
    if len(new_t) %2 == 1:
        medianValue = new_t[len(new_t) // 2]
    else:
        medianValue = (new_t[len(new_t) // 2 - 1] + new_t[len(new_t) //
2]) /2.0
    return meanValue, medianValue


def main():
    line = input().split()
    lst = [int(x) for x in line]
    print(mean_median(tuple(lst)))


main()
```

【描述】

编写程序，输入 10 个数，计算这 10 个数的均值和标准差。用下面的公式计算均值 mean 和标准差 deviation：

$$mean = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \ldots + x_n}{n} \qquad deviation = \sqrt{\frac{\sum_{i=1}^{n} (x_i - mean)^2}{n-1}}$$

【输入】

一行中给出 10 个数，其间以空格分隔。

【输出】

第一行为均值。

第二行为标准差。

结果保留 2 位小数。

【输入示例】

```
583 566 58 632 244 485 600 432 88 562
```

【输出示例】

```
425.00
216.48
```

参考答案：

```
import math
def mean(lst):
    return sum(lst) / len(lst)
```

```
def deviation(lst):
    mean1 = mean(lst)
    squareSum = 0;
    for i in range(len(lst)):
        squareSum += math.pow(lst[i] - mean1, 2)
    return math.sqrt(squareSum / (len(lst) - 1))

def main():
    line = input().split()
    lst = [float(x) for x in line]
    print("%.2f" % mean(lst))
    print("%.2f" % deviation(lst))

main()
```

【描述】

给定 n 行 m 列的图像各像素点的灰度值，要求用如下方法对其进行模糊化处理：

1. 四周最外侧的像素点灰度值不变；

2. 中间各像素点新灰度值为该像素点及其上下左右相邻四个像素点原灰度值的平均（舍入到最接近的整数）。

【输入】

第一行包含两个整数 n 和 m，表示图像包含像素点的行数和列数。1≤n≤100，1≤m≤100。

接下来 n 行，每行 m 个整数，表示图像的每个像素点灰度。相邻两个整数之间用单个空格隔开，每个元素均在 0~255 之间。

【输出】

n 行，每行 m 个整数，为模糊处理后的图像。相邻两个整数之间用单个空格隔开。

【输入示例】

```
4 5
100 0 100 0 50
50 100 200 0 0
50 50 100 100 200
100 100 50 50 100
```

【输出示例】

```
100 0 100 0 50
50 80 100 60 0
50 80 100 90 200
100 100 50 50 100
```

参考答案：

```
n, m = map(int, input().split())
a = []
b = []
for i in range(n):
    a.append([])
    b.append([])
    line = input().split()
    a[i] = [int(x) for x in line]
    b[i] = [int(x) for x in line]
```

```
for i in range(1, n - 1):
    for j in range(1, m - 1):
        b[i][j] = int((a[i][j] + a[i - 1][j] + a[i + 1][j] + a[i][j - 1] +
 a[i][j + 1]) / 5.0 + 0.5)
for i in range(n):
    for j in range(m):
        print(b[i][j], end=' ')
    print()
```

【描述】

给定 M 行 N 列的整数矩阵 A，如果 A 的非边界元素 A[i][j]大于相邻的上下左右 4 个元素，那么就称元素 A[i][j]是矩阵的局部极大值。求给定矩阵的全部局部极大值及其所在的位置。

【输入】

在第一行中给出矩阵 A 的行数 M 和列数 N（3≤M,N≤20）；下面 M 行，每行给出 A 在该行的 N 个元素的值。数字间以空格分隔。

【输出】

每行按照“元素值 行号 列号”的格式输出一个局部极大值，其中行、列编号从 1 开始。要求按照行号递增输出；若同行有超过 1 个局部极大值，则该行按列号递增输出。若没有局部极大值，则输出“None 总行数 总列数”。

【输入示例】

```
4 5
1 1 1 1 1
1 3 9 3 1
1 5 3 5 1
1 1 1 1 1
```

【输出示例】

```
9 2 3
5 3 2
5 3 4
```

参考答案：

```
def main():
    m, n = map(int, input().split())
    matrix = []
    for row in range(m):
        matrix.append([])
        line = input().split()
        matrix[row] = [eval(x) for x in line]
    flag = False;
    for row in range(1, m - 1):
        for col in range(1, n - 1):
            if matrix[row][col] > matrix[row - 1][col] and \
               matrix[row][col] > matrix[row + 1][col] and \
               matrix[row][col] > matrix[row][col - 1] and \
               matrix[row][col] > matrix[row][col + 1]:
                flag = True
                print("%d %d %d" % (matrix[row][col], row + 1, col + 1))
    if not flag:
        print("None %d %d" % (m, n))
```

```
main()
```

【描述】

读入一个整数 n，范围是[3,100]，这表示井字棋棋盘的边长。比如 n=3 就表示是一个 3x3 的棋盘。然后，要读入 n 行，每行 n 个数字，每个数字是 1 或 0，依次表示[0,0]到[n-1,n-1]位置上的棋子。1 表示 X，0 表示 O（大写字母 O）。

判断其中是否存在某一方获胜，获胜的条件是存在整行或整列或整条对角线或整条反对角线上是相同的棋子。如果存在，则输出代表获胜一方字母：X 或 O（大写字母 X 或 O）；如果没有任何一方获胜，则输出 NIL（三个大写字母，中间是字母 I（India 的 I）。

注意：所给的棋盘上的棋子分布可能出现同一个棋子有多处满足获胜的条件，但是不会出现两种棋子都获胜的情况。

【输入】

一个代表棋盘大小的数字 n，后面跟上 nxn 个 0 或 1 的数字。

【输出】

三种输出之一：

X

O

NIL

均为大写字母。

【输入示例】

```
4
1 0 0 1
0 1 0 0
0 0 1 0
1 0 0 1
```

【输出示例】

```
X
```

参考答案：

```
def main():
    n = int(input())
    tictactoe = []
    for row in range(n):
        tictactoe.append([])
        line = input().split()
        tictactoe[row] = [eval(x) for x in line]
    tag = -1

    for i in range(n):
        num = tictactoe[i][0]
        for j in range(n - 1):
            if tictactoe[i][j] != tictactoe[i][j + 1]:
                break
            if j == n - 2:
                tag = num
```

```
    for j in range(n):
        num = tictactoe[0][j]
        for i in range(n - 1):
            if tictactoe[i][j] != tictactoe[i + 1][j]:
                break
            if i == n - 2:
                tag = num

    for i in range(n - 1):
        if tictactoe[i][i] != tictactoe[i + 1][i + 1]:
            break
        if i == n - 2:
            tag = tictactoe[0][0];

    for i in range(n - 1):
        if tictactoe[n - 1 - i][i] != tictactoe[n - 2 - i][i + 1]:
            break
        if i == n - 2:
            tag = tictactoe[n - 1][0]
    if tag == 1:
        print("X")
    elif tag == 0:
        print("O")
    else:
        print("NIL")

main()
```

【描述】

定义函数：def list_to_tuples(two_dimensional_list)，该函数接受一个二维列表作为参数，内嵌列表的元素是字符串；返回一个二维元组，内嵌元组的内容是内嵌列表的内容的逆序。

例如，如果输入的二维列表是：[['mean', 'really', 'is', 'jean'], ['world', 'my', 'rocks', 'python']]，函数返回元组：(('jean', 'is', 'really', 'mean'), ('python', 'rocks', 'my', 'world'))。

【输入】

每一行输入一个字符串，字符串中单词之间以空格分隔。每行字符串构成二维列表的一个内嵌列表。可能会有多行输入。

【输出】

一个二维元组。

【输入示例】

```
mean really is jean
world my rocks python
```

【输出示例】

```
(('jean', 'is', 'really', 'mean'), ('python', 'rocks', 'my', 'world'))
```

参考答案：

```
def list_to_tuples(two_dimensional_list):
```

```
    lst = []
    for i in range(len(two_dimensional_list)):
        item = two_dimensional_list[i]
        item.reverse()
        lst.append(tuple(item))
    return tuple(lst)


def main():
    lst = []
    sentinel = ''
    for line in iter(input, sentinel):
        lst.append(line.split())
    print(list_to_tuples(lst))


main()
```

【描述】

输入一个 1 到 7 的数字，输出对应的星期名的缩写。

1 Mon

2 Tue

3 Wed

4 Thu

5 Fri

6 Sat

7 Sun

【输入】

输入 1 到 7 之间数字。

【输出】

输出对应的星期名的缩写。

【输入示例】

```
1
```

【输出示例】

```
Mon
```

参考答案：

```
d = {1:"Mon", 2:"Tue", 3:"Wed", 4:"Thu", 5:"Fri", 6:"Sat", 7:"Sun"}
x = int(input())
print(d[x])
```

【描述】

输入一个简单英文句子，统计并依次输出该句子中元音字母 a、e、i、o、u（不区分大小写）出现的次数。

【输入】

一行中输入一个简单英文句子。

【输出】

一行中依次输出 a、e、i、o、u 在句子中出现的次数，整数以空格间隔。

【输入示例】

```
If so,you already have a Google Account. You can sign in on the right.
```

【输出示例】

```
6 4 4 7 3
```

【来源】

《Python 程序设计基础》第 6 章编程题 3。

参考答案：

```
line = input()
d = {}
for ch in line:
    if ch.isalpha():
        if ch.isupper():
            ch = ch.lower()
        if ch in "aeiou":
            if ch in d:
                d[ch] += 1
            else:
                d[ch] = 1
letters = sorted(list(d.items()))
for letter in letters:
```

```
    print(letter[1], end=' ')
```

【描述】

定义函数：def number_to_words(number)，该函数接受一个整数作为参数；返回一个小写英文字符串，字符串的单词描述了该整数。

英文单词：zero、one、two、three、four、five、six、seven、eight、nine。

例如，如果输入的整数是：-4721，函数返回字符串："negative four seven two one"。

【输入】

输入一个整数。

【输出】

整数的英文单词描述（单词之间以一个空格间隔）。

【输入示例】

```
-4721
```

【输出示例】

```
negative four seven two one
```

【来源】

《Python 程序设计基础》第 6 章编程题 4。

参考答案：

```
def number_to_words(number):
    digit_dictionary = {"1" : "one",
                        "2" : "two",
                        "3" : "three",
                        "4" : "four",
                        "5" : "five",
                        "6" : "six",
                        "7" : "seven",
                        "8" : "eight",
                        "9" : "nine",
                        "0" : "zero"}
    output_string = ""
    if number < 0:
        number = -number
        output_string = "negative "
    string_input = str(number)
    splitted = list(string_input)
    for i in splitted:
        output_string += digit_dictionary[i] + " "
    stripped = output_string.rstrip(" ")
    return stripped


def main():
    number = int(input())
    print(number_to_words(number))


main()
```

【描述】

定义函数：def formatted_print(dictionary)，该函数接受一个字典作为参数，字典的键是学生姓名，字典的值是对应学生的平均分数。函数按照如下所指定的格式打印学生姓名及其对应的平均分数。

例如，如果输入的字典是：{'john':34.480,'eva':88.5, 'alex':90.55, 'tim': 65.900}，函数打印出如下信息：

```
alex       90.55
eva        88.50
tim        65.90
john       34.48
```

姓名输出宽度 10 且左对齐；平均分数输出宽度 6，保留 2 位小数，且右对齐。

所有这些信息按照学生的平均分数降序排序输出。

输入数据，调用该函数，输出结果。

【输入】

每一行输入姓名、平均分数，其间以逗号分隔。每行数据构成字典的一个键值对。可能会有多行输入。

【输出】

见【输出示例】。

【输入示例】

```
john,34.480
eva,88.5
alex,90.55
```

```
tim,65.900
```

【输出示例】

```
alex       90.55
eva        88.50
tim        65.90
john       34.48
```

参考答案：

```
def formatted_print(dictionary):
    d = {key:value for key, value in dictionary.items()}
    lst = sorted(d.items(), key=lambda item:item[1], reverse=True)
    for i in range(len(lst)):
        print("{0:10s}{1:>6.2f}".format(lst[i][0], lst[i][1]))


def main():
    d = {}
    sentinel = ''
    for line in iter(input, sentinel):
        line = line.split(',')
        d[line[0]] = float(line[1])
    formatted_print(d)


main()
```

【描述】

给定公司 n 名员工的工龄，要求按工龄增序输出每个工龄段有多少员工。

【输入】

一行中给出 n 个整数，即每个员工的工龄，范围在[0, 50]，整数间以空格间隔。

【输出】

按工龄的递增顺序输出每个工龄的员工个数，格式为：工龄:人数。每项占一行。如果人数为 0 则不输出该项。

【输入示例】

```
10 2 0 5 7 2 5 2
```

【输出示例】

```
0:1
2:3
5:2
7:1
10:1
```

参考答案：

```
lst = [int(x) for x in input().split()]
d = {}
for x in lst:
    if x in d:
        d[x] += 1
    else:
        d[x] = 1
counts = list(sorted(list(d.items())))
for i in range(len(counts)):
```

```
    print(counts[i][0], ':', counts[i][1], sep = '')
```

【描述】

输入若干个整数，输出其中出现了多少个不相同的数。

【输入】

一行中输入若干个整数，整数之间以空格分隔。

【输出】

一个数字，表示多少个不相同的数。

【输入示例】

```
1 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

【输出示例】

```
19
```

【来源】

《Python 程序设计基础》第 6 章编程题 1。

参考答案：

```
numbers = [int(x) for x in input().split()]
new_numbers = list(set(numbers))
print(len(new_numbers))
```

【描述】

小慧最近在数学课上学习了集合。小慧的老师给了小慧这样一个集合：

$s = \{ p / q \mid w \le p \le x, y \le q \le z \}$

根据给定的 w、x、y、z，求出集合中一共有多少个元素。

【输入】

4 个整数，分别是 w（$1 \le w \le x$），x（$1 \le x \le 100$），y（$1 \le y \le z$），z（$1 \le z \le 100$），以空格分隔。

【输出】

集合中元素的个数。

【输入示例】

```
1 10 1 1
```

【输出示例】

```
10
```

【来源】

《Python 程序设计基础》第 6 章编程题 2。

参考答案：

```
w, x, y, z = (int(value) for value in input().split())
p = [value for value in range(w, x + 1)]
q = [value for value in range(y, z + 1)]
result = [p[i] / q[j] for i in range(len(p)) for j in range(len(q))]
print(len(set(result)))
```

【描述】

输入 n 个整数，对这 n 个整数去重之后按原顺序输出。

【输入】

一行中输入 n 个整数。其中 1≤n≤100，每个数的范围 1≤x≤n。整数之间以空格间隔。

【输出】

去重之后按原顺序输出。

【输入示例】

```
1 3 2 1 3
```

【输出示例】

```
1 3 2
```

参考答案：

```
numbers = [int(x) for x in input().split()]
new_numbers = list(set(numbers))
new_numbers.sort(key = numbers.index)
for x in new_numbers:
    print(x, end=' ')
print()
```

【描述】

分析活动投票情况。第一小队有五名队员，序号是 1、2、3、4、5；第二小队也有五名队员，序号 6、7、8、9、10。输入一个得票字符串，求第二小队没有得票的队员序号。

【输入】

在一行中输入得票的队员的序列号，用逗号隔开。

【输出】

在一行中输出第二小队没有得票的队员序号。

【输入示例】

```
1,5,9,3,9,1,1,7,5,7,7,3,3,1,5,7,4,4,5,4,9,5,10,9
```

【输出示例】

```
6 8
```

参考答案：

```
set1 = {6, 7, 8, 9, 10}
lst = [int(x) for x in input().split(',')]
set2 = set(lst)
result = list(set1 - set2)
result.sort()
for x in result:
    print(x, end = ' ')
```

【描述】

给定两组整数，要求找出不是两者共有的元素。

【输入】

分别在两行中给出两组整数，整数间以空格分隔。

【输出】

在一行中按照数字给出的顺序输出不是两者共有的元素，数字间以空格分隔，但行末不得有多余的空格。

题目保证至少存在一个这样的数字。同一数字不重复输出。

【输入示例】

```
3 -5 2 8 0 3 5 -15 9 100
6 4 8 2 6 -5 9 0 100 8 1
```

【输出示例】

```
3 5 -15 6 4 1
```

参考答案：

```
lst1 = [int(x) for x in input().split()]
lst2 = [int(x) for x in input().split()]
lst3 = lst1 + lst2
set1 = set(lst1)
set2 = set(lst2)
result = list(set1 ^ set2)
result.sort(key = lst3.index)
length = len(result)
for i in range(length - 1):
    print(result[i], end = ' ')
print(result[length - 1])
```