已知 1900.1.1 是星期一，对于一个给定的年份和月份，输出这个月的最后一天是星期几
a） 1900.1.1 是星期一
b） 非闰年的 2 月是 28 天，闰年是 29 天
c） 闰年定义：公元年数能被 4 整除且又不能被 100 整除是闰年；能直接被 400 整除也是闰年
输入格式:
两个整数，分别代表年份和月份
输出格式：
星期数：0 代表星期日，1 代表星期 1，…，6 代表星期 6
1、编写函数实现功能
2、编写测试用例

1、

```
#!/usr/bin/env python
# encoding: utf-8
"""
@author: Hank zhang
@contract:
@file: homework4_ex.py
@time: 2016/10/23 11:34
"""

#非闰年 每月天数
monthdays = {1:31, 2:28, 3:31, 4:30, 5:31, 6:30, 7:31, 8:31, 9:30, 10:31, 11:30,
12:31}
#非闰年与闰年对应的天数
yeardays = {0:365, 1:366}

#计算是否是闰年
def leapYear(year):
    assert year>0, u"输入年份必须大于零的整数"
    if (year%4==0 and year%100!=0) or (year%400==0):
        return 1
    return 0

#计算星期几
def sumWeekday(days):
    # assert type(days) == type(int), u"输入天数必须大于零的整数"

    #星期
    week = {-6:u'星期一', -5:u'星期二', -4:u'星期三', -3:u'星期四', -2:u'星期五', -
    1:u'星期六',0:u'星期日',\
           1:u'星期一', 2:u'星期二', 3:u'星期三', 4:u'星期四', 5:u'星期五', 6:u'星期六
    '}
```

```
        return week[days%7]

    #计算与1990 年 1 月 1 日相距的时间
    def sumDays(year, month):
        assert year>0 and month>0, u"输入年月份必须大于零的整数"
        days = 0
        if year >= 1990:
            for elem in range(1990, year):
                days += yeardays[leapYear(elem)]
            for elem in range(1, month+1):
                if leapYear(year)==1 and elem ==2:
                    days += monthdays[elem] + 1
                else:
                    days += monthdays[elem]
        else:
            for elem in range(year, 1990):
                days -= yeardays[leapYear(elem)]
            for elem in range(1, month+1):
                if leapYear(year)==1 and elem ==2:
                    days += monthdays[elem] + 1
                else:
                    days += monthdays[elem]
        return days

    if __name__ == "__main__":
        year = input("请输入年份：")
        month = input("请输入月份：")
        days = sumDays(year, month)
        print sumWeekday(days)
```

2、

```
import unittest

from src.homework4_ex import sumWeekday


class MyTestCase(unittest.TestCase):
    def test_something(self):
        test_dict = {(2016, 10): u'星期一', (2016, 9): u'星期五', (2016, 8): u'星期三
', (2016, 7): u'星期日',
                     (1988, 10): u'星期一', (1988, 9): u'星期五', (1988, 8): u'星期三',
(1988, 7): u'星期日'}
        keys = test_dict.viewkeys()
        keyslist = list(keys)
```

```
        self.assertEqual(sumWeekday(2016, 10), test_dict[keyslist[0]])
        self.assertEqual(sumWeekday(2016, 9), test_dict[keyslist[1]])
        self.assertEqual(sumWeekday(2016, 8), test_dict[keyslist[2]])
        self.assertEqual(sumWeekday(2016, 7), test_dict[keyslist[3]])
        self.assertEqual(sumWeekday(1988, 10), test_dict[keyslist[4]])
        self.assertEqual(sumWeekday(1988, 9), test_dict[keyslist[5]])
        self.assertEqual(sumWeekday(1988, 8), test_dict[keyslist[6]])
        self.assertEqual(sumWeekday(1988, 7), test_dict[keyslist[7]])

if __name__ == '__main__':
    unittest.main()
```