

# 暨南大学本科实验报告专用纸

课程名称 python程序设计 成绩评定           

实验项目名称 python作业第三章

姓名 崔嘉容 学号 2020100069 学院 网络空间安全 专业 网络空间安全

实验时间 2023 年 10 月 9 日 ~ 10 月 16 日 实验地点: 516

## 一、实验目的

掌握三种基本控制结构：顺序结构、选择结构、循环结构。

## 二、实验环境和设备

实验环境：操作系统-Windows10，python版本-3.11.3，开发环境-pycharm

实验设备：华为MateBook14-2020，处理器-i7-10510U，内存-16GB

## 三、实验内容和结果

**题目一：**编写程序，使用不同的实现方法，输出2000~4000之间的所有闰年，每行显示10个数字。

### 实验代码：

方法一：使用循环和条件语句

```
def find_leap_years(start, end):
    leap_years = []
    for year in range(start, end+1):
        if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
            leap_years.append(year)
    return leap_years

start_year = 2000
end_year = 4000
leap_years = find_leap_years(start_year, end_year)

for i, year in enumerate(leap_years):
    print(year, end=' ')
```

# 暨南大学本科实验报告专用纸(附页)

```
if (i + 1) % 10 == 0:  
    print()
```

方法二：使用列表推导式

```
start_year = 2000  
end_year = 4000
```

```
leap_years = [year for year in range(start_year, end_year+1) if (year %  
4 == 0 and year % 100 != 0) or (year % 400 == 0)]
```

```
for i, year in enumerate(leap_years):  
    print(year, end=' ')  
    if (i + 1) % 10 == 0:  
        print()
```

方法三：使用 datetime 模块

```
import datetime
```

```
def find_leap_years(start, end):  
    leap_years = []  
    for year in range(start, end+1):  
        if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):  
            if datetime.date(year, 2, 29).year == year:  
                leap_years.append(year)  
    return leap_years
```

```
start_year = 2000  
end_year = 4000  
leap_years = find_leap_years(start_year, end_year)
```

```
for i, year in enumerate(leap_years):  
    print(year, end=' ')  
    if (i + 1) % 10 == 0:  
        print()
```

**实验结果：**

# 暨南大学本科实验报告专用纸(附页)

```
运行: 1_LeapYear_loop (1) ×
E:\Python3.11\python.exe D:\学习\大四上\python_learn\homework_chapter3\1_LeapYear_loop.py
2000 2004 2008 2012 2016 2020 2024 2028 2032 2036
2040 2044 2048 2052 2056 2060 2064 2068 2072 2076
2080 2084 2088 2092 2096 2104 2108 2112 2116 2120
2124 2128 2132 2136 2140 2144 2148 2152 2156 2160
2164 2168 2172 2176 2180 2184 2188 2192 2196 2204
2208 2212 2216 2220 2224 2228 2232 2236 2240 2244
2248 2252 2256 2260 2264 2268 2272 2276 2280 2284
2288 2292 2296 2304 2308 2312 2316 2320 2324 2328
2332 2336 2340 2344 2348 2352 2356 2360 2364 2368
2372 2376 2380 2384 2388 2392 2396 2400 2404 2408
```

```
运行: 1_LeapYear_loop (1) ×
3648 3652 3656 3660 3664 3668 3672 3676 3680 3684
3688 3692 3696 3704 3708 3712 3716 3720 3724 3728
3732 3736 3740 3744 3748 3752 3756 3760 3764 3768
3772 3776 3780 3784 3788 3792 3796 3804 3808 3812
3816 3820 3824 3828 3832 3836 3840 3844 3848 3852
3856 3860 3864 3868 3872 3876 3880 3884 3888 3892
3896 3904 3908 3912 3916 3920 3924 3928 3932 3936
3940 3944 3948 3952 3956 3960 3964 3968 3972 3976
3980 3984 3988 3992 3996 4000
进程已结束,退出代码0
```

## 实验分析和总结:

这个实验的目的是编写一个程序来输出2000到4000年之间的所有闰年，并且每行显示10个数字。使用三种不同的实现方法，分别使用了循环和条件语句、列表推导式以及`datetime`模块。

### 1. 使用循环和条件语句

这是最基本的实现方法，通过一个循环遍历2000到4000年的所有年份，然后使用条件语句来判断是否为闰年。如果是闰年则将其添加到一个列表中，最后按照要求进行输出。

### 2. 使用列表推导式

这种方法是一种更为简洁的实现方式，利用了Python中的列表推导式来过滤出满足条件的年份。这种方式相对于第一种方法更为紧凑。

### 3. 使用`datetime`模块

这种方法利用了Python的`datetime`模块，它提供了日期处理的功能。通过尝试创建2月29日的日期对象来检查是否是闰年，从而确定是否将其添加到闰年列表中。

# 暨南大学本科实验报告专用纸(附页)

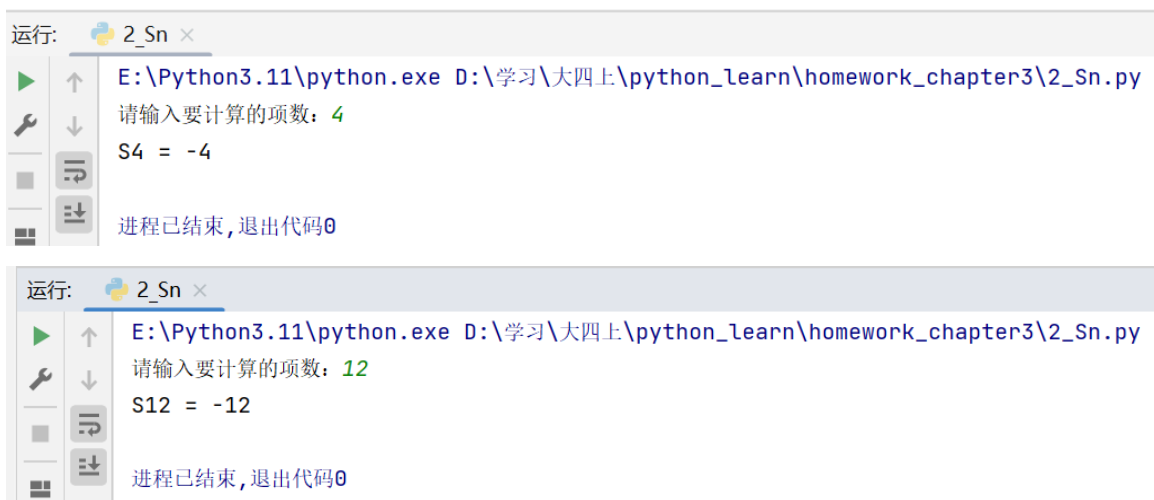
**题目二:** 编写程序, 计算 $S_n=1-3+5-7+9-11+\dots$ 。

**实验代码:**

```
def calculate_Sn(n):  
    Sn = 0  
    for i in range(1, n+1):  
        if i % 2 == 1:  
            Sn += 2*i - 1  
        else:  
            Sn -= 2*i - 1  
    return Sn
```

```
n = int(input("请输入要计算的项数: "))  
result = calculate_Sn(n)  
print(f"S{n} = {result}")
```

**实验结果:**



**实验分析和总结:**

通过定义一个函数 `calculate_Sn(n)` 来计算前  $n$  项的和, 其中  $n$  为用户输入的项数。在 `calculate_Sn` 函数中, 使用循环来遍历每一项, 根据奇偶性来决定是加上还是减去该项。最终计算得到和, 并输出结果。

在使用 `input` 函数来接收用户的输入, 并将其转换为整数类型。如果用户输入的不是一个合法的整数, 可能会引发异常, 需要进行相应的错误处理。

直接使用 `input` 会出现 “`TypeError: can only concatenate str (not "int") to str`” 的报错, 需要使用强制类型转换将输入的字符串转为 `int` 类型的值。

# 暨南大学本科实验报告专用纸(附页)

**题目三：**编写程序，打印九九乘法表。要求输出的九九乘法表的各种显示效果（上三角、下三角、矩形块等方式）。

**实验代码：**

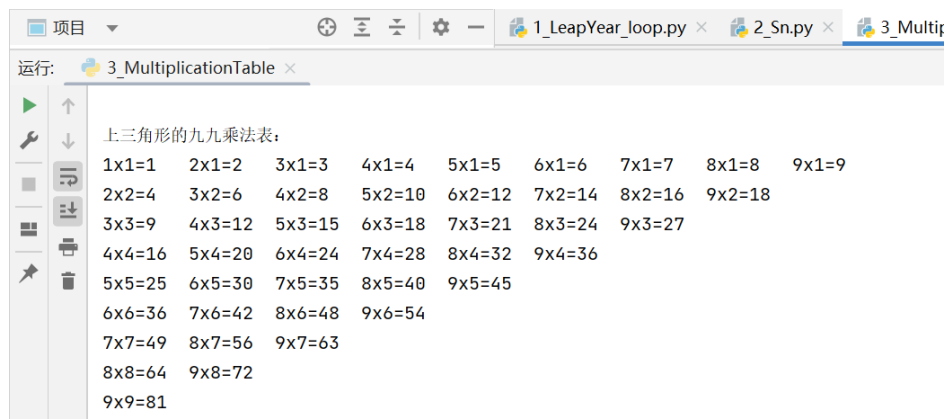
```
def print_upper_triangle():
    print("\n 上三角形的九九乘法表：")
    for i in range(1, 10):
        for j in range(i, 10):
            print(f' {j}x{i}={i*j}', end=' \t')
        print()

def print_lower_triangle():
    print("\n 下三角形的九九乘法表：")
    for i in range(1, 10):
        for j in range(1, i+1):
            print(f' {j}x{i}={i*j}', end=' \t')
        print()

def print_rectangle():
    print("\n 矩形块的九九乘法表：")
    for i in range(1, 10):
        for j in range(1, 10):
            print(f' {j}x{i}={i*j}', end=' \t')
        print()

# 调用各种打印函数
print_upper_triangle()
print_lower_triangle()
print_rectangle()
```

**实验结果：**



```
运行: 3_MultiplicationTable x
上三角形的九九乘法表:
1x1=1  2x1=2  3x1=3  4x1=4  5x1=5  6x1=6  7x1=7  8x1=8  9x1=9
2x2=4  3x2=6  4x2=8  5x2=10 6x2=12 7x2=14 8x2=16 9x2=18
3x3=9  4x3=12 5x3=15 6x3=18 7x3=21 8x3=24 9x3=27
4x4=16 5x4=20 6x4=24 7x4=28 8x4=32 9x4=36
5x5=25 6x5=30 7x5=35 8x5=40 9x5=45
6x6=36 7x6=42 8x6=48 9x6=54
7x7=49 8x7=56 9x7=63
8x8=64 9x8=72
9x9=81
```

# 暨南大学本科实验报告专用纸(附页)

```
下三角形的九九乘法表：
1x1=1
1x2=2  2x2=4
1x3=3  2x3=6  3x3=9
1x4=4  2x4=8  3x4=12  4x4=16
1x5=5  2x5=10  3x5=15  4x5=20  5x5=25
1x6=6  2x6=12  3x6=18  4x6=24  5x6=30  6x6=36
1x7=7  2x7=14  3x7=21  4x7=28  5x7=35  6x7=42  7x7=49
1x8=8  2x8=16  3x8=24  4x8=32  5x8=40  6x8=48  7x8=56  8x8=64
1x9=9  2x9=18  3x9=27  4x9=36  5x9=45  6x9=54  7x9=63  8x9=72  9x9=81

矩形块的九九乘法表：
1x1=1  2x1=2  3x1=3  4x1=4  5x1=5  6x1=6  7x1=7  8x1=8  9x1=9
1x2=2  2x2=4  3x2=6  4x2=8  5x2=10  6x2=12  7x2=14  8x2=16  9x2=18
1x3=3  2x3=6  3x3=9  4x3=12  5x3=15  6x3=18  7x3=21  8x3=24  9x3=27
1x4=4  2x4=8  3x4=12  4x4=16  5x4=20  6x4=24  7x4=28  8x4=32  9x4=36
1x5=5  2x5=10  3x5=15  4x5=20  5x5=25  6x5=30  7x5=35  8x5=40  9x5=45
1x6=6  2x6=12  3x6=18  4x6=24  5x6=30  6x6=36  7x6=42  8x6=48  9x6=54
1x7=7  2x7=14  3x7=21  4x7=28  5x7=35  6x7=42  7x7=49  8x7=56  9x7=63
1x8=8  2x8=16  3x8=24  4x8=32  5x8=40  6x8=48  7x8=56  8x8=64  9x8=72
1x9=9  2x9=18  3x9=27  4x9=36  5x9=45  6x9=54  7x9=63  8x9=72  9x9=81

进程已结束 退出代码0
Python 软件包 Python 控制台 服务
```

## 实验分析和总结：

打印上三角、下三角和矩阵块形式的九九乘法表的思路：

1. 上三角形的九九乘法表：只需要输出乘法表的上半部分（即  $i \geq j$  的部分）。使用两层循环，外层循环  $i$  控制行数，内层循环  $j$  控制列数。在内层循环中，只打印满足  $j \leq i$  的部分。
2. 下三角形的九九乘法表：只需要输出乘法表的下半部分（即  $i \leq j$  的部分）。同样使用两层循环，外层循环  $i$  控制行数，内层循环  $j$  控制列数。在内层循环中，我们打印满足  $i \leq j$  的部分。
3. 矩阵块形式的九九乘法表：包括所有乘积，使用两层循环遍历所有行和列。

格式控制和对齐：

问题：打印乘法表时，会出现格式不对齐的情况，导致表格显示混乱。

解决方法：在打印每个乘积时，可以使用制表符 `\t` 或者手动添加空格来保持格式的整齐。可以根据实际情况进行调整

# 暨南大学本科实验报告专用纸(附页)

**题目四:** 编写程序, 产生两个0~100之间(包括0和100)的随机整数a和b。求这两个整数的最大公约数和最小公倍数。

**实验代码:**

```
import random

# 生成随机整数 a 和 b
a = random.randint(0, 100)
b = random.randint(0, 100)

# 辗转相除法求最大公约数
def gcd(x, y):
    while y:
        x, y = y, x % y
    return x

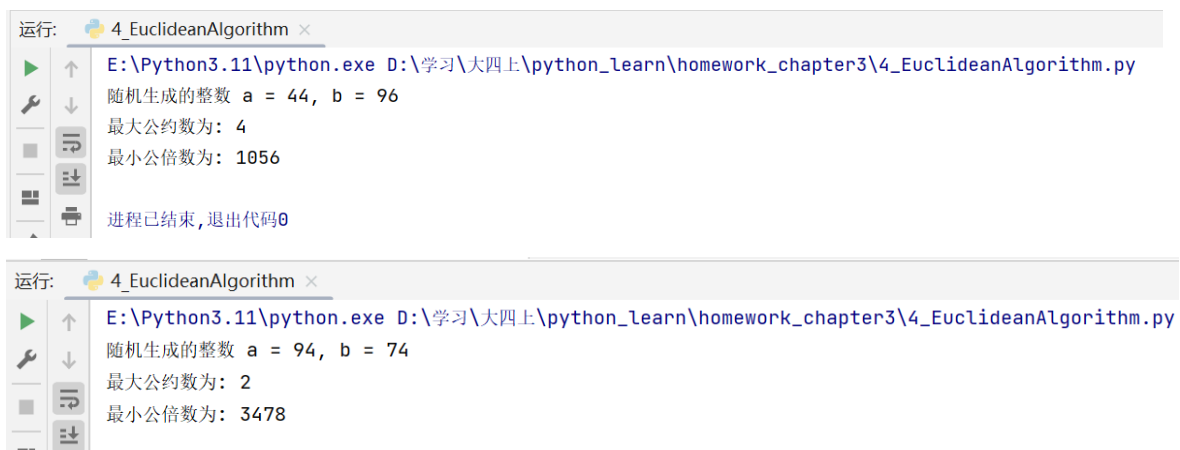
# 最小公倍数通过最大公约数求得
def lcm(x, y):
    return x * y // gcd(x, y)

# 输出随机生成的两个整数
print(f"随机生成的整数 a = {a}, b = {b}")

# 计算最大公约数和最小公倍数
gcd_result = gcd(a, b)
lcm_result = lcm(a, b)

print(f"最大公约数为: {gcd_result}")
print(f"最小公倍数为: {lcm_result}")
```

**实验结果:**



```
运行: 4_EuclideanAlgorithm x
E:\Python3.11\python.exe D:\学习\大四上\python_learn\homework_chapter3\4_EuclideanAlgorithm.py
随机生成的整数 a = 44, b = 96
最大公约数为: 4
最小公倍数为: 1056
进程已结束,退出代码0

运行: 4_EuclideanAlgorithm x
E:\Python3.11\python.exe D:\学习\大四上\python_learn\homework_chapter3\4_EuclideanAlgorithm.py
随机生成的整数 a = 94, b = 74
最大公约数为: 2
最小公倍数为: 3478
```

# 暨南大学本科实验报告专用纸(附页)

## 实验分析和总结:

编写这个程序需要分为以下几个步骤:

生成随机整数a和b: 使用Python的random模块中的randint()函数生成两个0~100之间的随机整数, 分别赋值给变量a和b。

求最大公约数: 编写一个函数gcd(x, y), 实现辗转相除法来求解最大公约数。辗转相除法是一种有效的求解最大公约数的算法, 它通过反复地用除数去除余数来求解。

求最小公倍数: 编写一个函数lcm(x, y), 利用最大公约数来求得最小公倍数。最小公倍数可以通过两数相乘除以最大公约数来得到。

如何使用Python生成指定范围内的随机整数? 使用random.randint(a, b)函数可以生成指定范围内的随机整数, 其中a和b分别是随机数的最小值和最大值。

如何使用辗转相除法求解两个数的最大公约数? 辗转相除法是一种常用的求解最大公约数的算法, 通过反复地用除数去除余数来求解, 直到余数为0。这样得到的除数就是最大公约数。

如何通过最大公约数求得两个数的最小公倍数? 最小公倍数可以通过两数的乘积除以最大公约数来得到。

如何使用格式化字符串输出结果? 使用f-string或者其他格式化字符串的方法可以清晰地输出结果。