项目 3　条件与循环

扫一扫，
看微课

项目任务

项目任务			
工单编号	0003	工单名称	条件与循环
工单类型	基础型	面向专业	计算机类
工单大类	程序设计	能力面向	专业能力
职业岗位	软件工程师		
实施方式	理实一体	考核方式	操作演示
工单难度	中等	前序工单	0002
工单分值	10	完成时限	4　学时
工单来源	教学案例	建议组数	99
组内人数	1	工单属性	院校工单
考核点	正确使用 Python 条件与循环结构		
设备环境	Python 3.10.x 和 PyCharm 2021		
教学方法	实践指导为主		

实施人员信息

实施人员信息			
姓名		班级	
学号		电话	
隶属组		组长	
岗位分工		伙伴成员	

项目日志

项目日志				
日期	工作内容	问题	原因	解决方案
总结反思				

项目介绍

1. 项目背景

程序设计中有 3 种基本控制结构：顺序结构、条件结构、循环结构。使用它们可以组合成任何程序。1966 年，Bohm 与 Jacopini 证明了任何单入口单出口的没有死循环的程序都能由 3 种基本控制结构构造出来。这 3 种基本控制结构就是顺序结构、if—then—else 选择结构、do_while 重复结构或 do until 重复结构。顺序结构是按照指令的顺序依次执行程序，而条件和循环结构则需要相应的控制语句。在 Python 中，用 if 语句实现条件结构，用 for 语句和 while 语句实现循环结构。

2. 项目规划

了解 Python 中基本控制结构的概念及使用方法。

项目目标

1. 知识目标

（1）理解基本控制结构的概念。

（2）掌握条件结构和循环结构的概念。

（3）掌握 break 语句和 continue 语句的概念。

2. 能力目标

（1）正确使用条件结构和循环结构。

（2）正确使用 break 语句和 continue 语句。

（3）独立完成典型应用问题的设计。

项目准备

1. 条件结构

“如果下雨，那么打伞；否则不打伞”，这是一种简单的条件选择逻辑。Python 使用 if 语句、elif 语句、else 语句实现不同的条件选择逻辑。条件结构控制语句根据条件表达式的判断结果为真（非零、非空）还是为假（零、空），选择程序的其中一个分支运行。条件结构主要有单分支、双分支、多分支、嵌套分支、分支结构的三元运算。

1）单分支

if 语句是一种单分支条件结构。if 语句由 4 个部分组成，即关键字 if、条件表达式、冒号、条件表达式的结果为真（非零、非空）时要执行的语句体。其语法格式如下：

```
if 条件表达式：
        语句体
```

图 3-1 所示为单分支条件表达式流程。

if 语句先判断条件表达式结果的真假。如果条件表达式的结果为真（非零、非空），那么执行语句体；如果条件表达式的结果为假（零、空），那么不执行语句体。语句体的缩进形式，代表了其与 if 语句的归属关系。当语句体由多条语句组成时，要有统一的缩进形式，否则可能出现逻辑错误或导致语法错误。if 语句示例代码如下：

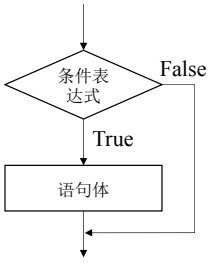

图 3-1 单分支条件表达式流程

```
score = 88
if score >= 60:
    print(' 考试及格！')    #输出“考试及格!”
```

试一试，将 score 的值改为 59，会输出多少？

如果没有确定好要通过语句体实现什么功能，那么可以使用 pass 占位。

```
score = 88
if score >= 60:
    pass  # 没有确定好要通过语句体实现什么功能
```

pass 不代表任何意义，一般只用于占位。使用 pass 是为了保持程序结构的完整性。在程序设计的过程中，可以使用 pass 替代某些代码，在后续过程中再做补充。

2）双分支

单分支条件结构只能处理条件满足的情况，不能处理条件不满足的情况。Python 提供了 if-else 语句。if-else 语句是一种双分支条件结构。其语法格式如下：

```
if 条件表达式：
        语句体 1
else:
        语句体 2
```

图 3-2 所示为双分支条件表达式流程。

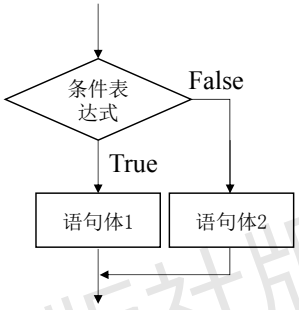

图 3-2　双分支条件表达式流程

if-else 语句先判断条件表达式结果的真假。如果条件表达式的结果为真（非零、非空），那么执行语句体 1；如果条件表达式的结果为假（零、空），那么执行语句体 2。语句体 1 和语句体 2 可以由一条或多条语句组成。if-else 语句示例代码如下：

```
score = 59
if score >= 60:
print(' 考试及格！')
else:
print(' 考试不及格！')   # 输出“考试不及格!”
```

试一试，将 score 的值改为 61，会输出多少？

3）多分支

基于上述示例，由于在实际评价成绩时，会分为优秀、良好、中等、差 4 个等级，因此需要更多的分支来实现。Python 提供了 if-elif-else 语句。if-elif-else 语句是一种多分支条件结构。其语法格式如下：

```
if 条件表达式 1：
        语句体 1
elif 条件表达式 2：
        语句体 2
…
elif 条件表达式 n-1：
```

```
        语句体 n-1
else:
    语句体 n
```

图 3-3 所示为多分支条件表达式流程。

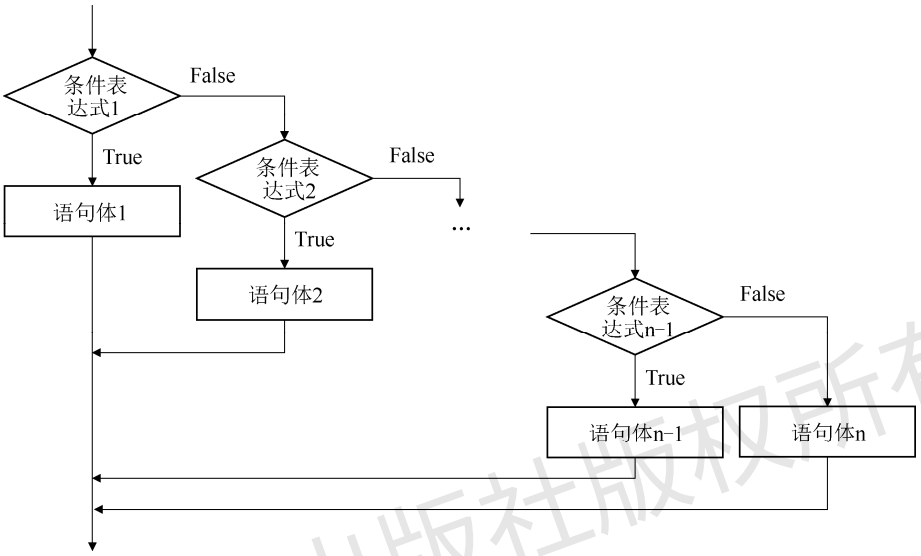

图 3-3　多分支条件表达式流程

语句体 1，语句体 2，…，语句体 n 中，均可以包含一条或多条语句。if-elif-else 语句示例代码如下：


```
score = 86
if score >= 85:
print(' 优秀! ')
elif 75 <= score < 85:
print(' 良好! ')
elif 65 <= score < 75:
print(' 中等! ')
else:
print(' 差! ')   # 输出“优秀!”
```

试一试，将 score 的值改为 72，会输出多少？

4）嵌套分支

“小学毕业才能读初中，初中毕业才能读高中，高中毕业才能读大学”，这是一种典型的嵌套逻辑。外层条件满足，才能去判断内层条件。Python 中通过 if 语句的嵌套可以实现程序中的嵌套逻辑。其语法格式如下：

```
if 条件表达式 1:                # 外层条件结构
语句体 1
if 条件表达式 2:                # 内层条件结构
```

```
    语句体 2
…
```

图 3-4 所示为嵌套分支条件表达式流程。

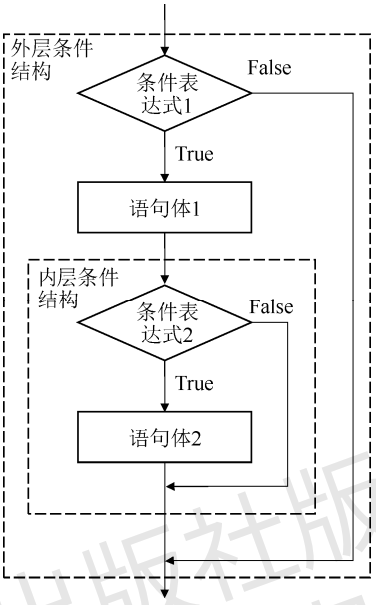

图 3-4　嵌套分支条件表达式流程

一年有 12 个月，1、3、5、7、8、10、12 月有 31 天；4、6、9、11 月有 30 天；闰年 2 月有 29 天，平年 2 月有 28 天，代码如下：

```
year = 2022
month = 5
if month in [1, 3, 5, 7, 8, 10, 12]:
print(' %d 月有 31 天' % month)
elif  month in [4, 6, 9, 11]:
print(' %d 月有 30 天' % month)
elif month == 2:
   if year % 400 == 0 or year % 400 == 0 and year % 100 != 0:
print(' %d 年%d 月有 29 天'%(year, month))
else:
print(' %d 年%d 月有 28 天'%(year, month))  # 输出“2022 年 5 月有 31 天”
```

5）分支结构的三元运算

为了使 if 语句的表达变得简洁，对于简单的 if-else 语句，可以使用三元运算表达式实现。其语法格式如下：

```
变量 = 值 1  if  条件表达式  else  值 2
```

如果条件表达式成立，那么变量取“值 1”，否则变量取“值 2”。

下面两组代码的运行结果是一样的。

```
x = 5
if x > 0:
    y = 1
else:
    y = 0
```

⟺

```
x = 5
y = 1 if x>0 else 0
```

2. 循环结构

“每天早晨出门前，要看一下天气预报”，这是一种简单的循环逻辑。Python 使用 while 语句、for 语句实现不同的循环逻辑。循环结构在给定的条件表达式的结果为真（非零、非空）时，重复执行某些操作；当条件表达式的结果为假（零、空）时，结束循环。Python 中可以使用 break 语句和 continue 语句改变循环结构的执行顺序。

1）while 语句

简单的 while 语句由关键字 while、条件表达式、冒号、条件表达式为真时要执行的循环体组成。其语法格式如下：

```
while 条件表达式:
    循环体
```

图 3-5 所示为 while 语句流程。

while 语句先判断条件表达式结果的真假。如果条件表达式的结果为真（非零、非空），那么执行循环体，否则结束循环。执行完循环体，返回继续判断条件表达式的结果。

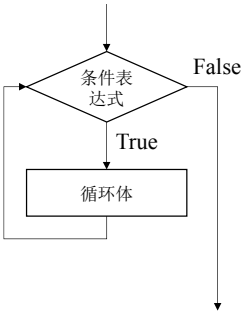

图 3-5 while 语句流程

综上，while 语句的具体执行过程如下。

在循环开始时，如果关键字 while 后面的条件表达式的结果为假（零、空），那么不会执行循环体，直接跳过循环部分。如果关键字 while 后面的条件表达式的结果为真（非零、非空），那么执行循环体。

每执行完一次循环体，重新判断关键字 while 后面的条件表达式的结果，如果为真（非零、非空），那么继续执行循环体，直到条件表达式的结果为假（零、空），结束循环。

使用 while 语句计算 1+2+3+4+5+6+7+8+9+10 的结果，代码如下：

```
i = 1
result = 0
while i < 10:
result += i
i += 1
print(result)  # 输出结果为“55”
```

在使用 while 语句时，要注意以下问题。

（1）循环体既可以由单条语句组成，又可以由多条语句组成。组成循环体的各条语句必须以相同的格式缩进。

（2）如果语句尚未确定，那么可以暂时使用 pass 代替。

（3）循环体中要有语句能够改变某个变量的值，使得条件表达式会因为该变量的值的改变而出现结果为假（零、空）的情况，从而使循环结束，否则会造成死循环。这个变量通常被称为循环控制变量。

（4）while 循环既可以用于解决循环次数确定的问题，又可以用于解决循环次数不确定的问题。此时，要设定好循环退出条件。

2）for 语句

for 语句通过遍历一个序列或迭代器（Iterator）等可迭代对象中的每个元素来建立循环。其语法格式如下：

```
for 变量 in 序列或迭代器等可迭代对象：
    循环体
```

图 3-6 所示为 for 语句流程。

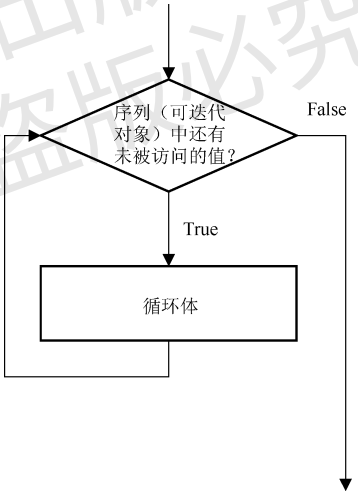

图 3-6　for 语句流程

在循环开始时，关键字 for 后面的循环控制变量从关键字 in 后面的序列或迭代器等可迭代对象中取值，如果无值可取，那么不进入循环；如果有值可取，那么取最前面的值，接着执行循环体。循环体执行完成后，循环控制变量继续取下一个值，如果无值可取，那么结束循环；否则取下一个值后继续执行循环体；重复以上过程，直到无值可取，结束循环。for 语句示例代码如下：

```
for word in 'HelloWorld':
   print(word)
```

程序运行结果如下：

```
H
e
l
l
o
W
o
r
l
d
```

从 Python 3 开始，range()函数用于返回可迭代对象。可以用 for 语句直接遍历 range()函数返回的可迭代对象，示例代码如下：

```
for i in range(1, 5, 1):
    print(i)
```

程序运行结果如下：

```
1
2
3
4
5
```


range()函数的语法格式如下：

```
range(start, end [, step])
```

start：可迭代对象的开始值为 start，默认值从 0 开始。例如，range(6)等价于 range(0, 6)。

end：可迭代对象到 end 结束，但不包括 end。例如，执行 range(0, 6)函数会返回包含 0、1、2、3、4、5 的可迭代对象。

step：步长，返回的可迭代对象元素之间的间隔，默认值为 1。例如，range(0,6)等价于 range(0, 6, 1)。当 step 是负数时，开始值大于结束值。

3）break 语句和 continue 语句

break 语句可以用在 while 语句和 for 语句中。在循环体的执行过程中，如果某个条件被满足（一般通过 if 语句判断是否满足执行 break 语句的条件），那么可以通过 break 语句立即终止本层循环。如果 break 语句在具有两层循环嵌套的内层循环中，那么只终止内层循环，进入外层循环的下一条语句继续执行。

continue 语句可以用在 while 语句和 for 语句中。在循环体的执行过程中，如果遇到 continue 语句，那么程序会跳过本次循环的剩余语句，返回循环开始的地方重新判断是否进入下一次循环。

break 语句与 continue 语句的主要区别有如下两条。

（1）break 语句一旦被执行，整个当前循环将被终止。

（2）执行 continue 语句不会终止整个当前循环，只会跳过本次循环的剩余语句，提前进入下一次循环。

break 语句的代码如下：

```
for word in 'HelloWorld':
  if word == 'W':
      break;
    print(word, end = ' ')
```

程序运行结果如下：

```
H e l l o
```

从程序运行结果中可以看出，程序在遍历字符串时，遇到'W'时终止当前循环，没有输出'W'及其后面的字符。

continue 语句的代码如下：

```
for word in 'HelloWorld':
if word == 'W':
    continue;
    print(word, end = ' ')
```


程序运行结果如下：

```
H e l l o r l d
```

从程序运行结果中可以看出，程序在遍历字符串时，遇到'W'时跳过本次循环的剩余语句，提前进入下一次循环，没有输出'W'。

4）带 else 的循环语句

Python 中的 while 语句和 for 语句后面还可以带 else。

（1）带 else 的 while 语句。

带 else 的 while 语句的语法格式如下：

```
while 条件表达式：
   循环体
else:
    else 语句体
```

如果条件表达式的结果为真（非空、非零），那么反复执行循环体。如果因为条件表达式的结果为假（零、空）而导致循环终止，那么执行一次 else 语句体，结束循环。图 3-7 所示为带 else 的循环语句的正常流程。

如果因为循环体中执行了 break 语句而导致循环终止，那么不会执行 else 语句体，而

会直接结束循环。图 3-8 所示为带 else 的循环语句执行了 break 语句的流程。

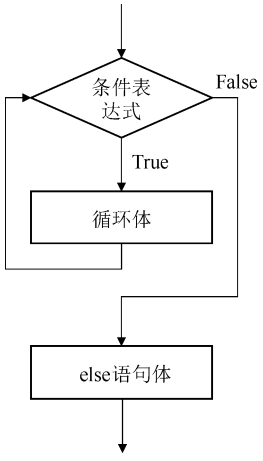

图 3-7　带 else 的循环语句的正常流程

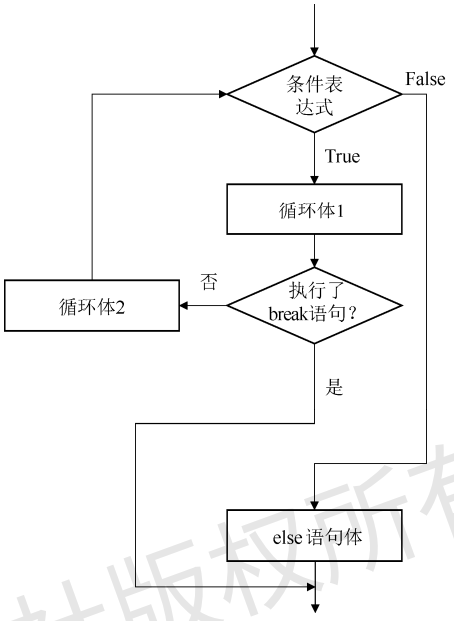

图 3-8　带 else 的循环语句执行了 break 语句的流程

代码如下：

```
i=21
while i>0:
    if i % 23 == 0:
        print("小于或等于",n,"且能被 23 整除的最大正整数是：",i)
        break
    i = i-1
else:
    print("未找到。")  # 输出“未找到。”
```

试一试，将 i 的值改为 47，会输出多少？

（2）带 else 的 for 语句。

带 else 的 for 语句的语法格式如下：

```
for 变量 in 序列或迭代器等可迭代对象：
   循环体
else:
   else 语句体
```

如果变量能够从关键字 in 后面的序列或迭代器等可迭代对象中取到值，那么执行循环体。循环体执行结束后，变量重新从序列或迭代器等可迭代对象中取值。

如果因为变量从关键字 in 后面的序列或迭代器等可迭代对象中取不到值而导致循环终止，那么执行一次 else 语句体，结束循环。图 3-9 所示为带 else 的 for 语句的正常流程。

如果因为循环体中执行了 break 语句而导致循环终止，那么不会执行 else 语句体，而

会直接结束循环。图 3-10 所示为带 else 的 for 语句执行了 break 语句的流程。

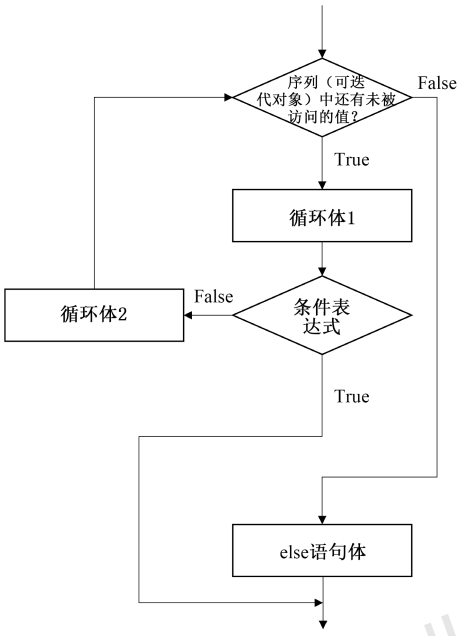

图 3-9　带 else 的 for 语句的正常流程

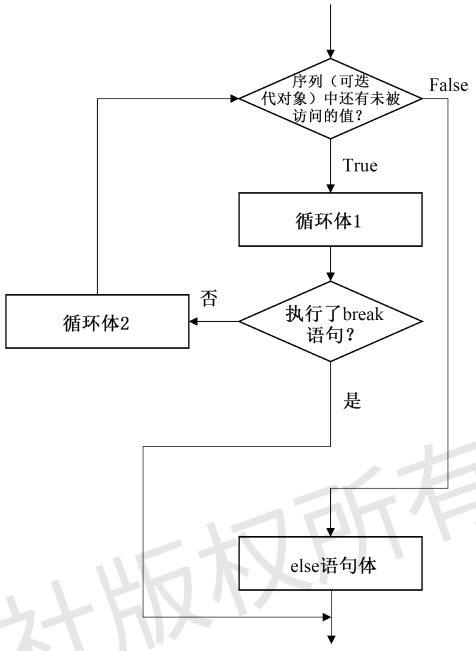

图 3-10　带 else 的 for 语句执行了 break 语句的流程

代码如下：

```
times=[5000, 3000, 8000, 10600, 6000, 5000]
for i in times:
    if i>=10000:
                print("第 1 个大于或等于 10000 的网页访问次数是：",i)
            break
else:
    print('未找到。')  # 输出“第 1 个大于或等于 10000 的网页访问次数是：10600”
```

5）循环的嵌套

循环的嵌套是指在一个循环中包含另外一个循环，即循环体中包含循环语句。循环的嵌套的执行过程为先进入外层循环第 1 轮，执行完所有内层循环，再进入外层循环第 2 轮，再次执行完所有内层循环，以此类推，直到外层循环执行完毕。

while 语句中可以嵌套 while 语句，for 语句中可以嵌套 for 语句。同时 while 语句和 for 语句也可以相互嵌套，如图 3-11 所示。

下面的 while 语句和 for 语句用于实现同一结果：

```
i = 1
while i < 6:
   print("*", end = " ")
   j += 1
```

```
for i in range(1, 6):
   for j in range(i):
      print("*", end = " ")
print()
```

```
print()
i += 1
```

程序运行结果如下：

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

在上述代码中，i 代表图形行数，j 代表图形列数，即每行“*”的数量。由于要打印完一行“*”才换一次行，因此内层循环中修改了 print()函数的结束符。print()函数的默认结束符为\n，通过 end=" "将结束符替换为空格。

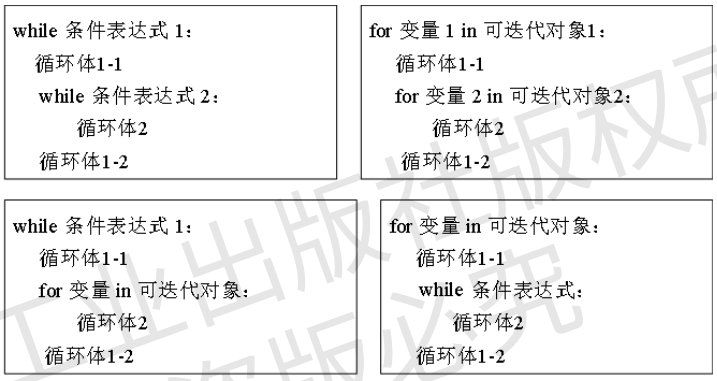

图 3-11 循环的嵌套

项目实施

任务 3.1 打印奇数

扫一扫，看微课

编写程序，要求实现接收一个整数，并打印 0 到该整数的所有奇数（包含负数且只使用一个 for 语句）。

S1 建立“打印奇数.py”文件，用于编写 Python 实验程序。步骤参考项目 1 的任务 1.2 中的“4. 基于 PyCharm”。

S2 编写任务代码。

```
# 接收一个输入的数据，并将该数据转换为整数
number = int(input('Enter a number:'))

# 定义一个 for 语句，使用 range()函数根据输入的数据创建循环区间
for i in range(number) if number > 0 else range(number, 0):
```

```
# 在 for 语句中输入代码，将区间内能够被 2 整除的数据过滤
    if (i%2 == 0):
        pass
    else:
        print(i)
```

PyCharm 中编写的完整代码如图 3-12 所示。

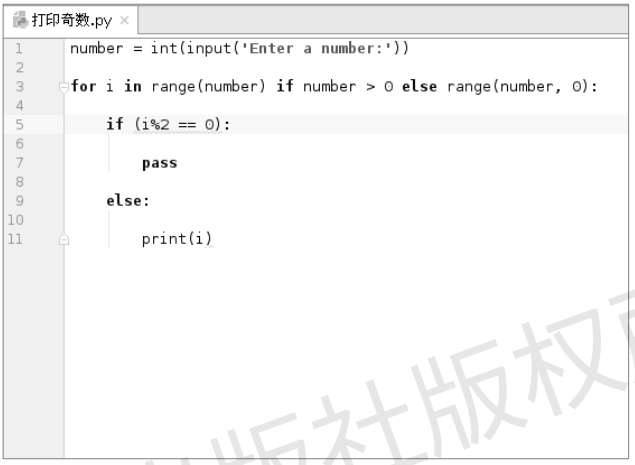

图 3-12　PyCharm 中编写的完整代码

S3 执行程序。

右击“打印奇数.py”文件空白处，在弹出的快捷菜单中选择“Run'Python 主程序所在文件名'”命令，运行程序。在 PyCharm 的用于显示程序运行结果的界面中，提示输入一个数字，如图 3-13 所示。

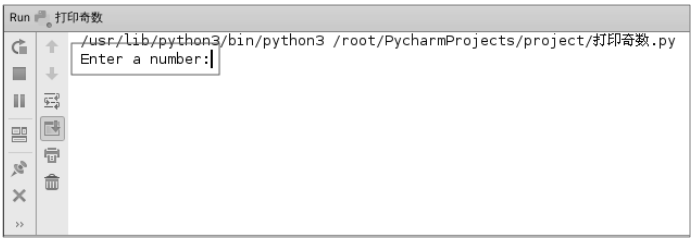

图 3-13　提示输入一个数字

输入数字 9，按回车键，将输出 0～9 的全部奇数，如图 3-14 所示。

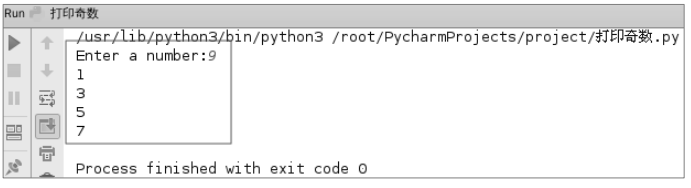

图 3-14　输出 0～9 的全部奇数

任务 3.2 打印 1~100 的素数

扫一扫，看微课

素数是指在大于 1 的自然数中，除了 1 和它本身不再有其他因数的自然数。通过寻找不大于 *n* 的素数，可以练习循环等功能的使用。

S1 建立“打印 1～100 的素数.py”文件，用于编写 Python 实验程序。步骤参考项目 1 的任务 1.2 中的“4. 基于 PyCharm”。

S2 编写任务代码。

```
# 定义一个空列表，用于存储素数
list = []
# 定义一个 for 语句，用来遍历 2～100
for i in range(2, 100):
    # 定义一个变量 j 的初始值为 2，用来存储除数
    j = 2
    # 定义一个 for 语句，用来遍历 2～i 的除数，并依次用 j 保存。计算 i 与 j 取余的结果，结
    # 果为 0，表示 2～i 中存在 j 能够整除 i，i 不为素数，否则 i 为素数。将判断为素数的值追加
    # 至素数列表
    for j in range(2, i):
        if i % j == 0:
            break
    else:
        list.append(i)
print(list)
```

PyCharm 中编写的完整代码如图 3-15 所示。

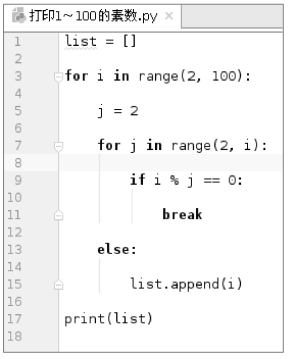

图 3-15 PyCharm 中编写的完整代码

S3 执行程序。

右击“打印 1～100 的素数.py”文件空白处，在弹出的快捷菜单中选择“Run'Python 主程序所在文件名'”命令，运行程序。

查看程序运行结果，如图 3-16 所示。

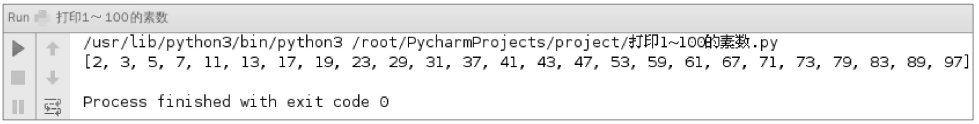

图 3-16　查看程序运行结果

任务 3.3　打印实心菱形和空心菱形


扫一扫，看微课

计算机之所以能做很多自动化的任务，是因为它可以实现条件判断。在 Python 程序中，使用 if 语句实现条件判断。为了让计算机能计算成千上万次的重复运算，需要使用循环语句。在很多情况下，为了完成某项任务，需要把条件判断和循环结合起来。本任务使用 Python 分别打印宽度和高度均为 9cm 的实心菱形。

S1 建立“打印宽度和高度均为 9cm 的实心菱形.py”文件，用于编写 Python 实验程序。步骤参考项目 3 的任务 3.1。

S2 编写任务代码。

```
offset = 1          # 定义一个变量，用来控制“*”的递增与递减
star = 1            # 定义一个变量，用来保存当前行需要打印的“*”的个数
max = 9             # 定义一个变量，用来保存实心菱形的最大宽度和高度
whitespace = int(max / 2)  # 定义一个变量，用来保存每行第一个“*”左侧空格的数量
flag = True         # 定义一个变量，用来控制程序是否打印完成并退出
end = ' '           # 定义一个变量，用来设置 print()函数的参数 end

# 定义一个 while 语句，由于 flag 的初始值为 True，因此该语句为死循环
while(flag):
    # 定义一个 for 语句，用来打印当前行第一个“*”左侧的空格
    for i in range(whitespace):
        print(' ', end=end)
    # 定义一个 for 语句，用来打印当前行的“*”
    for i in range(star):
        print('*', end=end)
    print()         # 打印一个换行，即切换至下一行进行打印
    # 判断当前行打印的“*”是否为 9 个，若为 9 个则将“*”的数量由递增变为递减
    if star == 9:
        offset = -1 * offset
    whitespace -= offset    # 改变下一行输出的空格的数量
    star += offset * 2      # 改变下一行输出的“*”的数量
    # 当下一行输出的“*”的数量小于 0 个时，表示实心菱形已经打印完毕，改变程序开关变量的
    # 值为 False，使得结束 while 语句
```

```
    if star < 0:
        flag = False
```

S3 执行程序。

右击“打印宽度和高度均为 9cm 的实心菱形.py”文件空白处，在弹出的快捷菜单中选择“Run'Python 主程序所在文件名'”命令，运行程序。

查看程序运行结果，如图 3-17 所示。

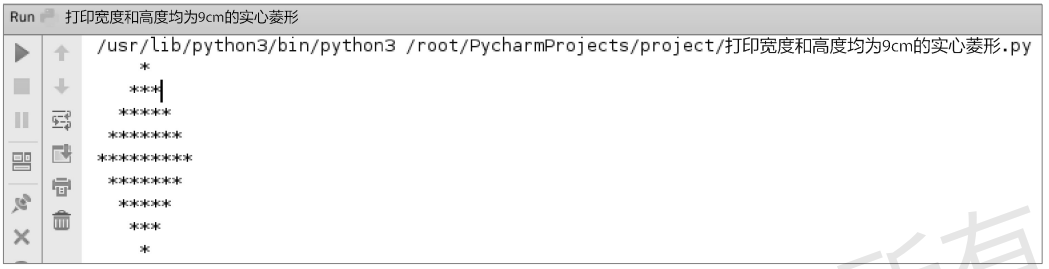

图 3-17 查看程序运行结果

S4 建立“打印宽度和高度均为 9cm 的空心菱形.py”文件，用于编写 Python 实验程序。步骤参考项目 3 的任务 3.1。

S5 编写任务代码。

```
offset = 1                      # 定义一个变量，用来控制“*”的递增与递减
star = 1                        # 定义一个变量，用来保存当前行需要打印的“*”的个数
max = 9                         # 定义一个变量，用来保存空心菱形的最大宽度和高度
whitespace = int(max / 2)       # 定义一个变量，用来保存每行第一个“*”左侧空格的数量
flag = True                     # 定义一个变量，用来控制程序是否打印完成并退出
end = ''                        # 定义一个变量，用来设置 print()函数的参数 end
# 定义一个 while 语句，由于 flag 的初始值为 True，因此该语句为死循环
while(flag):
    # 定义一个 for 语句，用来打印当前行第一个“*”左侧的空格
    for i in range(whitespace):
        print(' ', end=end)
    # 定义一个 for 语句，用来打印当前行的“*”。在 for 语句中判断当前行打印的“*”是否为
    # 当前行的第一个或最后一个。如果为第一个或最后一个，那么打印“*”，否则打印空格
    for i in range(star):
        if (i==0 or i==star-1):
            print('*', end=end)
        else:
            print(' ', end=end)
    print();                    # 打印一个换行，即切换至下一行进行打印
    # 判断当前行打印“*”的数量是否为 9 个，若为 9 个则将“*”的数量由递增变为递减
    if star == 9:
        offset = -1 * offset
    whitespace -= offset # 用于改变下一行输出的空格的数量
```

```
    star += offset * 2 # 用于改变下一行输出的“*”的数量
    # 当下一行输出的“*”的数量小于 0 个时，表示菱形已经打印完毕，改变程序开关变量的值为
    # False，使得结束 while 语句
    if star < 0:
        flag = False
```

PyCharm 中编写的完整代码如图 3-18 所示。

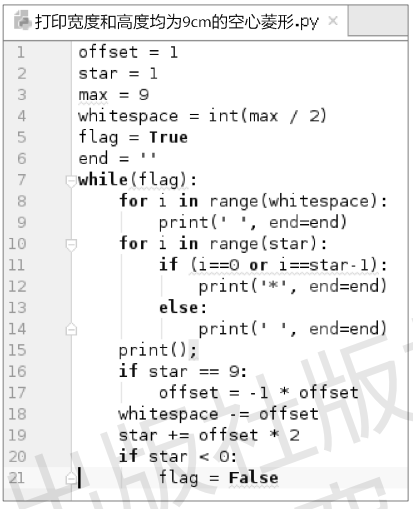

图 3-18　PyCharm 中编写的完整代码

S6 执行程序。

右击“打印宽度和高度均为 9cm 的空心菱形.py”文件，在弹出的快捷菜单中选择“Run'Python 主程序所在文件名'”命令，运行程序。

查看程序运行结果，如图 3-19 所示。

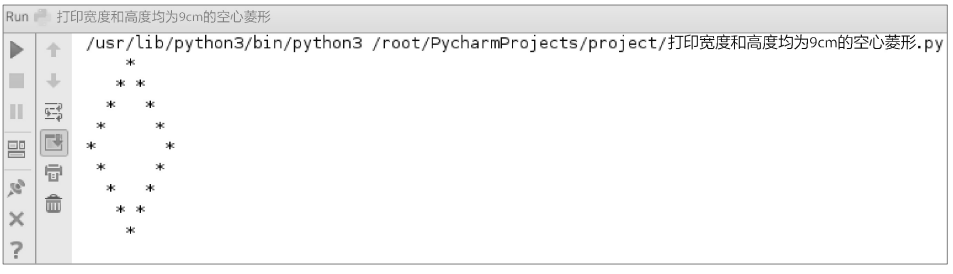

图 3-19　查看程序运行结果

S7 方案对比。

对比上述两段代码可以看出，由于空心菱形需要判断当前行打印的“*”是否为当前行的第一个或最后一个，因此需要在第二个 for 语句中增加一个 if 语句。通过学习，读者可以了解将视觉直观图形转换为代码逻辑的思路和方法。

任务 3.4 冒泡排序之降序

扫一扫，看微课

排序是计算机内经常进行的一种操作，目的是将一组“无序”的记录序列调整为“有序”的记录序列。常见排序算法包括快速排序、希尔排序、堆排序、直接选择排序、基数排序、冒泡排序（Bubble Sort）、直接插入排序、折半插入排序、归并排序等。

冒泡排序是一种计算机科学领域比较简单的排序算法。它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果顺序（从大到小、首字母从 Z 到 A 等）错误，那么把它们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说，该元素列已经排序完成。这个算法的名称是依据越小的元素会经由交换慢慢“浮”到数列的顶端（升序或降序排列），就如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样而取的。

冒泡排序的原理为：比较相邻的元素，如果第一个元素比第二个元素大，那么交换两个元素，对每对相邻的元素进行同样的操作，从开始的第一对相邻的元素到末尾的最后一对相邻的元素，注意最后一个元素应该是最大的数。对除最后一个元素外的所有元素重复以上步骤，直到所有元素比较完成。

S1 建立“冒泡排序之降序.py”文件，用于编写 Python 实验程序。步骤参考项目 3 的任务 3.1。

S2 编写任务代码。

```
numberList = [5, 3, 1, 7, 9, 2, 8, 1, 4, 7, 3] # 定义一个数字列表
for i in range(len(numberList) - 1): # 定义一个循环用于实现数字列表长度减一次匹配
    # 在 for 语句中输入如下代码
    for j in range(len(numberList) - i - 1): # 嵌套一层循环用于遍历匹配项的索引
    # 在内层 for 语句中输入如下代码
    # 对两个相邻的元素进行比较，若前一个元素小于后一个元素，则将两个元素的位置对调
        if numberList[j] < numberList[j+1]:
            numberList[j], numberList[j+1] = numberList[j+1], numberList[j]
print(numberList) # 使用 print()函数打印完成排序的结果列表
```

S3 执行程序。

右击“冒泡排序之降序.py”文件空白处，在弹出的快捷菜单中选择“Run'Python 主程序所在文件名'”命令，运行程序。

查看程序运行结果，如图 3-20 所示。

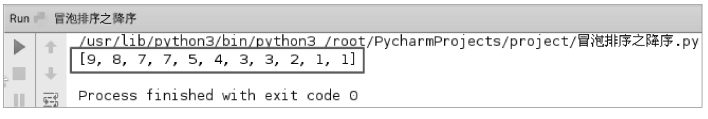

图 3-20 查看程序运行结果

评价与考核

课程名称：		授课地点：		
学习任务：条件与循环		授课教师：		
课程性质：理实一体课程		综合评分：		
知识掌握情况评分（30 分）				
序号	**知识考核点**	**教师评价**	**分数**	**得分**
1	理解基本控制结构的概念		10	
2	掌握条件结构和循环结构的概念		10	
3	掌握 break 语句和 continue 语句的概念		10	
工作任务完成情况评分（50 分）				
序号	**能力操作考核点**	**教师评价**	**分数**	**得分**
1	正确使用条件结构和循环结构		15	
2	正确使用 break 语句和 continue 语句		15	
3	独立完成典型应用问题的设计		20	
违纪扣分（20 分）				
序号	**违纪考核点**	**教师评价**	**分数**	**扣分**
1	迟到早退		5	
2	课上睡觉		5	
3	课上玩手机		5	
4	其他扰乱课堂秩序的行为		5	

程序人生

计算机科学的发展主要涉及硬件和软件的发展，而硬件和软件发展的核心问题是如何保证它们是可靠的、安全的，关键问题是如何保证它们的运行是正确的。如今，硬件的性能变得越来越高、运算速度变得越来越快、体系结构变得越来越复杂，软件的性能也变得越来越强大、复杂，如何开发可靠的硬件系统和软件系统，已经成为计算机科学发展的巨大挑战。特别是现在计算机系统已广泛应用于许多关系国计民生的系统中，如高速列车控制系统、航空航天控制系统、核反应堆控制系统、医疗设备系统等，这些系统中出现任何错误都可能导致灾难性的后果。

现代计算科学和计算机科学技术源自 Church 的 lambda 演算和 Turing 的图灵机等计算模型。这些计算模型是计算的理论基础。计算系统的设计开发需要分析、处理、证明计算机硬件系统和软件系统的性质。形式化方法以形式（逻辑）系统为基础，支持对计算系统进行严格的规约、建模和验证，并且为此设计算法从而建立计算机辅助工具。在现代计算系统的开发过程中，形式化方法在不同的阶段以不同的形式和程度得到应用。例如，在基于模型的软件开发中，建模、模型精化和基于模型的测试都是基于形式化方法的思想和技术发展起来的。程序语言的类型设计、程序分析的算法都是形式化方法中的基本技术。

形式化方法已经成功应用于各种硬件设计，特别是芯片设计中。各大硬件制造商都有一个非常强大的形式化方法团队，为保障系统的可靠性提供技术支持，如 IBM、Intel、AMD 等公司。由于软件系统的复杂性和不确定性远远超出硬件系统，因此形式化方法在软件开发中的应用程度并不高。但是在近些年来，随着形式验证技术和工具的发展，特别是在程序验证中的成功应用，形式化方法在处理软件开发复杂性和提高软件可靠性方面已显示出无可替代的潜力。各个著名的研究机构都已经投入大量的人力和物力从事这方面的研究。例如，美国国家航空航天局（NASA）拥有一支庞大的形式化方法研究团队，其在保证美国航天器控制软件正确性方面发挥了巨大作用。在美国研发“好奇号”火星探测器时，为了提高控制软件的可靠性和生产率，广泛使用了形式化方法。微软、华为、Synopsis、Meta、Amazon 等公司聘用形式化方法的专家从事形式验证技术研究及工具开发工作，以期提高其商业软件的可靠性。目前，国际上已经出现了一批以形式化方法为核心竞争力的高科技公司，如 Galois、Praxis 等。

基本控制结构是软件实现形式化方法的基础。灵活运用条件和循环结构，能够极大地提升使用计算机软件解决问题的能力。

课后练习与技能训练

一、填空题

1．通过设置_____，可以使某些语句在条件满足时执行。

2．通过设置_____，可以使某些语句重复执行多次。

3．下面程序的循环次数是_____，循环结束后 i 的值是_____。

```
i=10
while i>=0:
    i-=1
print(i)
```

4．已知程序：

```
score=eval(input('请输入成绩（0~100 的整数）：'))
if score<60:
      print('不及格')
elif score<70:
      print('及格')
elif score<80:
      print('中等')
elif score<90:
      print('良好')
elif score<=100:
      print('优秀')
```

若输入 77，则输出结果为_____。

5．下面程序的功能是用户输入数字 n，利用 for 循环求 n!。请将程序填写完整。

```
n=eval(input('请输入一个大于 0 的整数：'))
s=
for i in range(1,_________):
   s=
print(s)
```


6．下面程序的功能是求 100 以内能被 7 整除的最大整数。请将程序填写完整。

```
n=100
while_________
   if n%7==0:
      print(n)
      ____________
   n-=1
```

7．下面程序的功能是判断素数。请将程序填写完整。

```
for n in _________(2,101):
   m=int(n**0.5)
   i=2
   while____________
      if n%i==0:
        ____________
      i+=1
   if i>m:
      print(n,end=' ')
```

二、选择题

1．已知程序：

```
score=eval(input('请输入成绩（0~100 的整数）：'))
```

```
if score<60:
    print('你的成绩是%d'%score)
print('不及格')
```

若输入 55，则输出结果为（　　）。

A．你的成绩是 55
　　不及格

B．你的成绩是 55

C．不及格

D．无输出

2．已知程序：

```
score=eval(input('请输入成绩（0~100 的整数）：'))
if score>=60:
    pass
else:
    print('不及格')
```

若输入 55，则输出结果为（　　）。

A．无输出

B．不及格

C．pass

D．报错提示

3．已知程序：

```
n=eval(input('请输入一个整数：'))
if n%2==0:
    print("偶数")
else:
    print("奇数")
```

若输入–5，则输出结果为（　　）。

A．无输出

B．奇数

C．偶数

D．偶数
　　奇数

4．已知程序：

```
d={'Python':1,'C++':2,'Java':3}
for k in d:
    print('%s:%d'%(k,d[k]))
```

输出结果为（　　）。

A．Python
　　C++
　　Java

B．1:Python
　　2:C++
　　3:Java

C．Python:1
　　C++:2
　　Java:3

D．以上都不对

三、简答题

1．已知水仙花数是 3 位整数（100~999），它的各位数的立方和等于该数本身。下面程序的功能是求水仙花数。请写出程序运行结果。

```
for n in range(100,1000):
    bai=n//100
    shi=n//10%10
    ge=n%10
    if bai**3+shi**3+ge**3==n:
        print(n)
```

2．下面程序的功能是输出九九乘法表。请将程序填写完整。

```
for i in range(1,10):
    for j in range(1,_________):
        print(j,"*",i,"=",i*j,end=' ')
    print(_________)
```