第3章　控制结构

【学习目标】

通过本章的学习，应达到如下的学习目标：

1. 了解程序流程的基本概念，掌握程序流程控制的3种结构。

2. 熟练掌握if选择控制语句、for及while循环控制语句，掌握continue与break流程控制语句的使用方法。

3. 理解条件表达式与True/False的等价关系。

【单元导学】

第3章思维导图如图3-1所示。

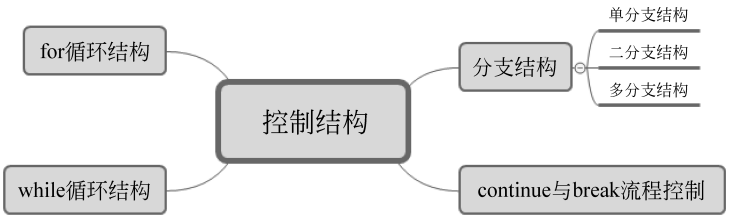

图3-1　第3章思维导图

本章重点包括：

顺序结构、分支结构和循环结构的区别。

本章难点包括：

分支结构条件语句的用法、循环结构控制语句的用法。

【知识回顾】

第2章重点介绍了Python的基本语法和数据结构，请回忆第2章的内容，编写代码，实现录入学生成绩与计算总分及平均分的功能。

代码案例：

```
nat=input("请输入语文成绩:")
math=input("请输入数学成绩:")
eng=input("请输入英语成绩:")
sum=int(nat)+int(math)+int(eng)      #输入值需转换为整型
average=sum/3
print("成绩总分:%d, 平均成绩:%5.2f" %(sum, average))
```

【学前准备】

第 2 章介绍了 Python 的基本语法和数据结构，这些基本语法的代码都是一行一行地由上往下依次执行的，这种形式的代码称为顺序结构。但是，在真正的编码环境中，顺序结构并不能实现多种自动化的任务，而计算机可以通过条件判断或者循环控制完成复杂的任务。

为了更好地学习本章知识，请做到：

1. 复习变量、数据类型、表达式、操作数、运算符等概念。
2. 查阅 range()函数等常用函数。
3. 查阅顺序结构、分支结构和循环结构的特点和相关语句的用法。

实验 3-1　单分支结构

【实验目的】

1. 掌握单分支结构的使用方法。
2. 熟悉 if 语句缩进、冒号等使用要求。

【实验内容】

1. 在 IDLE 中编写程序，实现单分支条件判断语句。
2. 实现判断考试成绩是否合格的程序。
3. 实现布尔操作符在 if 结构中的应用。
4. 实现登录密码验证程序。

【实验步骤】

1. 在 Windows 的“开始”菜单中启动 IDLE 交互环境。
2. 在 IDLE 交互环境中选择 File→New File 菜单项打开代码编辑器。
3. 在代码编辑器中输入以下代码：

```
x=70
if x>=60:       #当 x>=60 时,输出"成绩合格"
print('成绩合格')
```

注意：若条件表达式的值为 True，则执行程序块的操作；若条件表达式的值为 False，则不执行程序块的操作。

4. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-1-1.py”。
5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================RESTART: F: \实验 3-1-1.py======================
成绩合格
>>>
```

Python 条件语句是通过一条或多条语句的执行结果(True 或者 False)来决定执行的代码块的。Python 分支结构中,判断条件以英文冒号(:)结尾,表示接下来是满足条件后要执行的语句块。Python 使用缩进来划分语句块,相同缩进数的语句一起组成一个语句块。分支结构中的逻辑代码块,以相对于条件语句向右 4 个空格或 1 个 tab 为分隔符(建议使用 4 个空格,tab 在不同的系统中表现不一致可能引起混乱,影响代码的跨平台性)。

注意:if 语句后面的冒号(:)不能省略。条件表达式可以是关系表达式,如"x>60",也可以是逻辑表达式,如"x>60 or x<70"。如果程序块内只有一行代码,则可以合并为一行,直接写成如下格式:

```
if(条件表达式):代码
```

6. 继续在代码编辑器中输入以下代码:

```
#判断正确
if 2>1 and not 2>3:
print('判断正确! ')
```

7. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为"实验 3-1-2.py"。

8 .在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
判断正确!
```

以上实例表明,单个 if 语句中的条件表达式可以通过布尔操作符 and、or 和 not 实现多重条件判断。

9. 继续在代码编辑器中输入以下代码:

```
#实现验证输入密码的功能
password=input("请输入密码: ")
if(password=="123456"):
print("登录成功!")
```

注意:代码 password==="123456"中,一个等号与两个等号的区别。因为程序块内只有一行代码,所以还可以将以上代码写成一行:

```
if(password=="123456"):print("登录成功!")
```

10. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为"实验 3-1-3.py"。

11. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,并输入密码"123456",得到输出结果:

```
请输入密码: 123456
登录成功!
```

如果用户输入了错误的密码,则程序不输出任何字符,并结束退出。很显然,这样的程序不够人性化,应该考虑到用户输入不正确密码的可能性,一旦密码不正确,应及时提醒用户出现问题的原因,所以就需要二分支条件语句来完善程序的功能。

实验 3-2　二分支结构

【实验目的】

1. 掌握二分支结构的使用方法。
2. 熟悉 if…else…结构缩进、冒号等使用要求。

【实验内容】

1. 在 IDLE 中编写程序,实现二分支条件判断语句。
2. 实现简单数学逻辑判断的程序。
3. 实现高级验证输入密码的程序。
4. 实现判断一个数字是否为奇数或偶数。

【实验步骤】

1. 在 Windows 的“开始”菜单中,启动 IDLE 交互环境。

2. 二分支结构使用 if…else…结构。Python 提供与 if 搭配使用的 else,如果 if 语句的条件表达式结果为假,那么程序将执行 else 语句后的代码。

3. 在 IDLE 交互环境中选择 File→New File 菜单项打开代码编辑器。

4. 在代码编辑器中输入以下代码:

```
x=-1
#x=-1 条件判断为假,不执行后续缩进内的语句块
if x>0:
    s=x*2
    print(s)
else:
print('x 小于或等于 0 ')
```

5. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-2-1.py”。

6. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
x 小于或等于 0
```

7. 继续在代码编辑器中输入以下代码:

```
#实现验证输入密码的功能
password=input("请输入密码: ")
if(password=="123456"):
    print("登录成功!")
else:
print("密码输入错误!")
```

注意:以上各行代码的缩进位置。

8. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-2-2.py”。

9. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
请输入密码：123
密码输入错误！
>>>
请输入密码：123456
登录成功！
>>>
```

10. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

11. 在代码编辑器中输入以下代码：

```
#判断该数是奇数还是偶数
#如果该数除以 2 余数为 0，则是偶数
#如果该数除以 2 余数为 1，则是奇数
num=int(input("输入一个数字："))
if (num %2)==0:
   print("{0}是偶数".format(num))
else:
   print("{0}是奇数".format(num))
```

12. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-2-3.py”。

13. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
输入一个数字：2021
2021 是奇数
```

现实中遇到的很多案例需求不仅仅是只判断一个变量，只用双分支结构不能完成所有条件的判断。多样化的需求需要更多的条件判断来运行不同的执行代码，这也需要多分支结构来实现更复杂的功能。

实验 3-3　多分支结构

【实验目的】

1. 掌握多分支结构的使用方法。
2. 熟悉多分支条件判断结构缩进、冒号等使用要求。

【实验内容】

1. 在 IDLE 中编写程序，实现多分支条件判断。
2. 编写根据不同的天气情况做出不同决策的程序。
3. 编写判断输入数字是否可被整除的程序。
4. 编写判断用户输入的年份是否为闰年的程序。

【实验步骤】

1. 在 Windows 的“开始”菜单中，启动 IDLE 交互环境。

2. if 语句有一个特点：它是从上往下判断，如果在某个判断上是 True，则执行该判断对应的语句后，就忽略掉剩下的 elif 和 else。elif 是 else if 的缩写，可以有多个 elif。

3. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

4. 在代码编辑器中输入以下代码：

```
weather=input('输入当前天气：')      #输入'sunny'、'cloudy'或其他
if weather=='sunny':                 #注意缩进规则，同时不要少写冒号
    print('shopping')
elif weather=='cloudy':
    print('playing football')
else:
print('do nothing')
```

5. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-3-1.py”。

6. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
输入当前天气：cloudy
playing football
```

7. 在嵌套 if 语句中，可以把 if…elif…else 结构放在另外一个 if…elif…else 结构中。

8. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

9. 在代码编辑器中输入以下代码：

```
num=int(input("输入一个数字："))
if num%2==0:
    if num%3==0:
        print ("你输入的数字可以整除 2 和 3")
    else:
        print ("你输入的数字可以整除 2,但不能整除 3")
else:
    if num%3==0:
        print ("你输入的数字可以整除 3,但不能整除 2")
    else:
        print ("你输入的数字不能整除 2 和 3")
```

10. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-3-2.py”。

11. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
输入一个数字：2021
你输入的数字不能整除 2 和 3
```

12. 闰年是公历中的名词，闰年分为普通闰年和世纪闰年。其中能被 4 整除但不能被 100 整除的年份为普通闰年，能被 400 整除的为世纪闰年。很显然，可以使用多条件结构来分析这一问题。

13. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

14. 在代码编辑器中输入以下代码：

```
year=int(input("输入一个年份: "))
if (year %4)==0:
   if (year %100)==0:
       if (year %400)==0:
           print("{0}是闰年".format(year))       #整百年能被 400 整除的是闰年
       else:
           print("{0}不是闰年".format(year))
   else:
       print("{0}是闰年".format(year))           #非整百年能被 4 整除的为闰年
else:
   print("{0}不是闰年".format(year))
```

15. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-3-3.py”。

16. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码并输入年份，得到输出结果：

```
输入一个年份: 2020
2020 是闰年
输入一个年份: 2021
2021 不是闰年
```

循环是让计算机做重复任务的有效方法。例如，可以使用循环结构依次取出数组中的元素。Python 的循环命令有两个：for 循环和 while 循环，其中 for 循环用于执行固定次数的循环，while 循环用于执行不固定次数的循环。在 while 和 for 循环过程中，为了更加灵活地控制循环次数，Python 还提供了 break 和 continue 循环控制语句。

注意：Python 中没有 switch…case 语句。

实验 3-4　for 循环结构

【实验目的】

1. 掌握 for 循环结构的使用方法。
2. 熟悉 for 循环结构的注意事项。

【实验内容】

1. 在 IDLE 中编写程序，实现 for 循环结构语句。
2. 实现 0～9 的求和。
3. 实现循环操作字符串。
4. 解决猴子吃桃问题。
5. 判断是否为质数问题。
6. 九九乘法表。

【实验步骤】

1. 在 Windows 的“开始”菜单中，启动 IDLE 交互环境。

2. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

3. 在代码编辑器中输入以下代码：

```
#实现 0~9 的求和
num=0
for i in range(10):
    num+=i
print(num)
```

注意：for 使用关键字 in 遍历序列，获取元素，由变量 i 表示，以便在之后的逻辑代码块中使用。for 语句以英文冒号结尾，逻辑代码块以 4 个空格或 tab 分隔。执行语句块中的程序全部需要缩进并对齐。

4. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-4-1.py”。

5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
45
```

6. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

7. 在代码编辑器中输入以下代码：

```
#利用 for 循环取出每个元素进行操作
sList=['I', 'M', 'A', 'U']
s=''
for x in sList:
    s+=x
print(s)
```

8. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-4-2.py”。

9. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
IMAU
```

10. 在代码编辑器中输入以下代码：

```
for letter in 'Python':
  print ('Current Letter :', letter)
fruits=['banana', 'apple', 'mango']
for fruit in fruits:
  print ('Current fruit :', fruit)
```

11. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-4-3.py”。

12. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
```

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

13. 接下来利用 for 循环来解决猴子吃桃问题。

题目：猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃前一天剩下的一半零一个。到第 10 天早上想再吃时，发现只剩下一个桃子了。求第一天共摘了多少个桃子？

解题思路：采取逆向思维的方法，从后往前推断。

14. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

15. 在代码编辑器中输入以下代码：

```
x2=1
for day in range(9,0,-1):
    x1=(x2+1) * 2
    x2=x1
print(x1)
```

16. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-4-4.py”。

17. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
4
10
22
46
94
190
382
766
1534
```

18. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

19. 在代码编辑器中输入以下代码：

```
#检测用户输入的数字是否为质数
#用户输入数字
num=int(input("请输入一个数字：))
#质数大于 1
if num>1:
   #查看因子
   for i in range(2,num):
       if (num %i)==0:
           print(num,"不是质数")
           print(i,"乘以",num/i,"是",num)
           break
   else:
```

```
        print(num,"是质数")

#如果输入的数字小于或等于 1,则不是质数
else:
   print(num,"不是质数")
```

20. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-4-5.py”。

21. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
请输入一个数字: 1
1 不是质数
请输入一个数字: 4
4 不是质数
2 乘以 2 是 4
请输入一个数字: 7
7 是质数
```

22. 在 IDLE 交互环境中选择 File→New File 菜单项,打开代码编辑器。

23. 在代码编辑器中输入以下代码:

```
#九九乘法表
for i in range(1, 10):
    for j in range(1, i+1):
        print('{}x{}={}\t'.format(j, i, i * j), end='')
print()
```

24. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-4-6.py”。

25. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
1×1=1
1×2=2  2×2=4
1×3=3  2×3=6  3×3=9
1×4=4  2×4=8  3×4=12  4×4=16
1×5=5  2×5=10  3×5=15  4×5=20  5×5=25
1×6=6  2×6=12  3×6=18  4×6=24  5×6=30  6×6=36
1×7=7  2×7=14  3×7=21  4×7=28  5×7=35  6×7=42  7×7=49
1×8=8  2×8=16  3×8=24  4×8=32  5×8=40  6×8=48  7×8=56  8×8=64
1×9=9  2×9=18  3×9=27  4×9=36  5×9=45  6×9=54  7×9=63  8×9=72  9×9=81
```

实验 3-5　while 循环结构

【实验目的】

1. 掌握 while 循环结构的使用方法。
2. 熟悉 while 循环结构的注意事项。

【实验内容】

1. 在 IDLE 中编写程序,实现 while 循环结构语句。

2. 实现数列求和运算。

3. 输出斐波那契数列。

【实验步骤】

1. 在 Windows 的“开始”菜单中,启动 IDLE 交互环境。

2. 在 IDLE 交互环境中选择 File→New File 菜单项,打开代码编辑器。

3. 在代码编辑器中输入以下代码:

```
t=10
s=0
while t:
    s=s+t        #也可以写成 s+=t
    t=t-1
print(s)
t=10
s=0
while t>=5 :     #当条件为 True 时,执行循环语句块,否则不执行
    s=s+t        #也可以写成 s+=t
    t=t-1
print(s)
```

注意:执行语句块中的程序全部都要缩进并对齐。

4. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-5-1.py”。

5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
55
45
```

6. 斐波那契数列指的是这样一个数列:0,1,1,2,3,5,8,13,…。特别指出,第 0 项是 0,第 1 项是第一个 1。从第三项开始,每一项都等于前两项之和。

7. 在 IDLE 交互环境中选择 File→New File 菜单项,打开代码编辑器。

8. 在代码编辑器中输入以下代码:

```
#输出斐波那契数列
#获取用户输入的数据
nterms=int(input("你需要几项?"))
#第一项和第二项
n1=0
n2=1
count=2

#判断输入的值是否合法
if nterms<=0:
   print("请输入一个正整数。")
elif nterms==1:
   print("斐波那契数列: ")
```

```
    print(n1)
else:
    print("斐波那契数列: ")
    print(n1,",",n2,end=", ")
    while count<nterms:
        nth=n1+n2
        print(nth,end=", ")
        #更新值
        n1=n2
        n2=nth
        count+=1
```

注意：执行语句块中的程序全部都要缩进并对齐。

9. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-5-2.py”。

10. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
你需要几项？10
斐波那契数列:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

11. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

12. 在代码编辑器中输入以下代码：

```
#使用 while 计算 1 到 100 的总和:
n=100
sum=0
counter=1
while counter<=n:
    sum=sum+counter
    counter+=1
print("1 到 %d 之和为: %d" %(n,sum))
```

13. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-5-3.py”。

14. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
1 到 100 之和为: 5050
```

实验 3-6　continue 与 break 流程控制语句

【实验目的】

1. 掌握 continue 结构的使用方法。
2. 掌握 break 结构的使用方法。
3. 对比 continue 结构和 break 结构。

【实验内容】

1. 在 IDLE 中编写程序，实现 continue 与 break 流程控制语句。

2. 实现奇数序列输出。

3. 实现 continue 与 break 跳出循环语句。

【实验步骤】

1. 在 Windows 的“开始”菜单中,启动 IDLE 交互环境。

2. 在 IDLE 交互环境中选择 File→New File 菜单项,打开代码编辑器。

3. 在代码编辑器中输入以下代码:

```
num=1
while True:          #条件始终为真,执行循环语句,直到出现 break 退出循环
    if num>10:
        break
    elif num%2==0:
        num +=1
        continue
    else:
        print('num 的值为',num)
        num +=1
```

注意:continue 语句用来跳过当前循环语句块中的剩余语句,然后继续进行下一轮循环。break 语句可以跳出 for 和 while 的循环体,所以 continue 可用作过滤条件,break 用于触发跳出循环的条件,如果从 for 或 while 循环中终止,对应的循环 else 语句块将不执行。

4. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-6-1.py”。

5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
num 的值为 1
num 的值为 3
num 的值为 5
num 的值为 7
num 的值为 9
```

6. 在 IDLE 交互环境中选择 File→New File 菜单项,打开代码编辑器。

7. 在代码编辑器中输入以下代码:

```
#continue: 在循环体中,跳过 i==5 的循环,继续执行之后的所有循环
for i in range(0, 10):
    if i==5:
        continue
print (i)
```

8. 在 IDLE 交互环境中选择 File→Save 菜单项,将文件保存为“实验 3-6-2.py”。

9. 在 IDLE 交互环境中选择 Run→Run Module 菜单项,运行代码,得到输出结果:

```
0
1
2
```

```
3
4
6
7
8
9
```

10. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

11. 在代码编辑器中输入以下代码：

```
#break: 直接跳出循环
for i in range(0, 10):
    if i==5:
        break
print (i)
```

12. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-6-3.py”。

13. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
0
1
2
3
4
```

通过以上两个案例，可以分析对比 continue 与 break 跳出循环语句的区别。break 跳出整个循环，而 continue 跳出本次循环。break 和 continue 语句可以用在 while 和 for 循环中。

14. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

15. 在代码编辑器中输入以下代码：

```
n=10
while n>2:
    n=n-1
    if n==5:
        break
    print('n=',n)

print('.........')
n=10
while n>2:
    n=n-1
    if n==5:
        continue
print('n=',n)
```

16. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-6-4.py”。

17. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
n=9
n=8
n=7
n=6
.........
n=9
n=8
n=7
n=6
n=4
n=3
n=2
```

可以看到，break 语句遇到 5 就停止了循环，而 continue 语句遇到 5 仅跳过本次循环，并继续运行程序。

18. for 循环和 while 循环中也会使用到 else 的扩展用法。其中 else 中的程序只在一种条件下执行，即循环正常遍历所有内容或者由于条件不成立而结束循环，且没有因 break 或者 return 而退出循环。若在循环程序块中使用 continue 语句，仍会继续执行 else 中的内容。可以通过以下示例进一步加深理解：

19. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

20. 在代码编辑器中输入以下代码：

```
for i in "python":
    if i=="t":
        continue
    print(i, end="")
else:
    print("程序正常退出")
```

21. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-6-5.py”。

22. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
pyhon 程序正常退出
```

23. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

24. 在代码编辑器中输入以下代码：

```
for i in "python":
    if i=="t":
        break
    print(i, end="")
else:
    print("程序正常退出")
```

25. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-6-6.py”。

26. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
py
```

实验 3-7　分支结构综合案例

【实验目的】

1. 掌握分支结构的使用方法。
2. 熟悉分支条件判断结构缩进、冒号等使用要求。

【实验内容】

1. 在 IDLE 中编写程序，实现未成年人的年龄判断语句。
2. 编写 BMI 指数计算程序。
3. 编写判断输入数字是否可被整除的程序。
4. 编写具有简单数学逻辑运算功能的程序。

【实验步骤】

1. 在 Windows 的“开始”菜单中，启动 IDLE 交互环境。
2. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。
3. 在代码编辑器中输入以下代码：

```
age=input("Please input your age: ")
age=int(age)
if age>=18:
    print('your age is', age)
    print('adult')
else:
    print('your age is', age)
print('teenager')
```

4. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-7-1.py”。

5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行以下代码，并输入年龄 17 和 20，观察输出结果。

```
Please input your age: 17
your age is 17
teenager
>>>
Please input your age: 20
your age is 20
adult
```

6. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，并输入年龄 0.5，观察输出结果：

```
Please input your age: 0.5
Traceback (most recent call last):
  File "F:/ python/实验 3-7-1.py", line 2, in<module>
    age=int(age)
ValueError: invalid literal for int() with base 10: '0.5'
```

运行后程序报错，提示 0.5 是错误的整型变量，进一步修改程序，将类型修改为 float，并加入正数的判断。代码如下：

```
age=input("Please input your age: ")
age=float(age)
if age>=18:
    print('your age is', age)
    print('adult')
elif age>=0:
    print('your age is', age)
    print('teenager')
else:
print('input error! ')
```

7. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，并输入年龄 12.0 和 18.2，观察输出结果：

```
Please input your age: 12.0
your age is 12.0
teenager
>>>
Please input your age: 18.2
your age is 18.2
adult
>>>
Please input your age:imau
Traceback (most recent call last):
  File "F: /python/实验 3-7-1.py", line 2, in<module>
    age=float(age)
ValueError: could not convert string to float: 'imau'
```

运行后，如果用户输入字符串，这个程序将会报错，无法转换字符串类型，而这样的问题往往采用异常处理方法予以解决。请参考异常处理相关章节的介绍。

继续尝试以下练习。

小明身高 1.75m，体重 80.5kg。请根据 BMI 公式$\left(\text{BMI 指数}=\frac{\text{体重}}{\text{身高}^2}\right)$计算小明的 BMI 指数。BMI 指数的取值分为以下 5 种情况。

- 低于 18.5：过轻。
- 18.5～25：正常。

- 25～28：过重。
- 28～32：肥胖。
- 高于 32：严重肥胖。

8. 在 Windows 的“开始”菜单中，启动 IDLE 交互环境。

9. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

10. 在代码编辑器中输入以下代码：

```
height=float(input('请输入你的身高(m)：'))
weight=float(input('请输入你的体重(kg)：'))
BMI=weight/(height**2)
if BMI<18.5:
    print('过轻')
elif BMI<=24.9:
    print('正常')
elif BMI<=27.9:
    print('过重')
elif BMI<=32:
    print('肥胖')
else:
print('严重肥胖')
```

11. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-7-2.py”。

12. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，并输入小明身高 1.75m，体重 80.5kg，观察输出结果：

```
请输入你的身高(m)：1.75
请输入你的体重(kg)：80.5
过重
```

13. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

14. 在代码编辑器中输入以下代码：

```
#写出判断一个数是否能够被 3 或者 7 整除
#但是不能同时被 3 或者 7 整除的条件语句
x=int(input('请输入一个数字：'))
if x %3==0 or x %7==0:
    if x %3==0 and x %7==0:
        print('False')
    else:
        print('True')
else:
print('False')
```

15. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-7-3.py”。

16. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
请输入一个数字：21
False
```

17. 还可以将该案例简写为以下代码，以达到相同的效果：

```
x=int(input('请输入一个数字：'))
print((x %3==0 or x %7==0) and (x %3 !=0 or x %7 !=0))
```

18. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

19. 在代码编辑器中输入以下代码：

```
#输入两个整数，如果两个数相减的结果为奇数，则输出该结果
#否则输出提示信息"结果不是奇数"
x=int(input('请输入一个数字 x:'))
y=int(input('请输入一个数字 y:'))
z=x -y
if z %2 !=0:
    print(z)
else:
print('结果不是奇数')
```

20. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-7-4.py”。

21. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
请输入一个数字 x: 4
请输入一个数字 y: 1
3
```

实验 3-8　循环结构综合案例

【实验目的】

1. 练习分支结构的使用方法。
2. 掌握循环结构的使用方法。

【实验内容】

1. 在 IDLE 中编写程序，利用循环结构实现猜商品价格游戏，要求：随机生成一个 100～200 元的商品价格，由玩家猜商品的价格。计算机根据玩家猜的数字分别给出提示：“价格猜高了”“价格猜低了”或“恭喜猜对了！”。

2. 统计 100 以内个位数是 2 并且能够被 3 整除的数的个数。

3. 输入任意一个正整数，求它是几位数。

4. 实现水仙花数问题。

5. 统计 1～100 中素数的个数，并且输出所有的素数。

6. 解决鸡兔同笼问题。

7. 解决折纸厚度问题。

【实验步骤】

1. 在 Windows 的“开始”菜单中，启动 IDLE 交互环境。

2. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。

3. 在代码编辑器中输入以下代码：

```
import random
answer=random.randint(100,200)
counter=0
while True:
    counter+=1
    number=int(input('请输入商品的价格：'))
    if number>answer:
        print('价格猜高了')
    elif number<answer:
        print('价格猜低了')
    else:
        print('恭喜猜对了！')
        break
print('你总共猜了%d次' %counter)
if counter>7:
print('猜错次数大于7次，请继续努力！')
```

4. 在 IDLE 交互环境中选择 File→Save 菜单项，将文件保存为“实验 3-8-1.py”。

5. 在 IDLE 交互环境中选择 Run→Run Module 菜单项，运行代码，得到输出结果：

```
请输入商品的价格：100
价格猜低了
请输入商品的价格：170
价格猜高了
请输入商品的价格：130
价格猜低了
请输入商品的价格：150
价格猜低了
请输入商品的价格：165
价格猜高了
请输入商品的价格：157
价格猜低了
请输入商品的价格：160
价格猜高了
请输入商品的价格：159
恭喜猜对了！
你总共猜了8次
猜错次数大于7次，请继续努力！
```

6. 在 IDLE 交互环境中选择 File→New File 菜单项，打开代码编辑器。