第 3 章

选 择 结 构

选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。本章主要介绍了程序设计的一般方法、结构化程序设计的思想以及 Python 语言选择结构的实现方法。

本章要求了解结构化程序设计的思想，并且可以按照程序设计的一般方法完成简单程序的设计过程，熟练掌握顺序结构和选择结构的使用。

学习目标：

- 程序设计方法
- 结构化程序设计
- 简单 if 语句
- 复杂 if 语句

3.1 程序设计方法

程序设计的一般方法可以概括为以下四个步骤。

1. 设计算法

找出解决问题的规律，选择解题的算法，完成实际问题。

2. 画流程图

根据算法思想，画出程序流程图。

3. 编写程序

将算法翻译成计算机程序设计语言，本书采用 Python 语言进行编码。

4. 运行调试

能得到运行结果并不意味着程序正确，要对结果进行分析，看它是否合理。若不合理

则要对程序进行调试，即通过上机发现和排除程序中的 Bug。

【实例 3.1】计算梯形的面积。

第一步：根据梯形面积的计算公式，定义 top、bottom、height、area 为浮点型变量，分别用于存储梯形的上底、下底、高和面积，即得到：

$$\text{area} = \frac{(\text{top} + \text{bottom}) \times \text{height}}{2}$$

第二步：用流程图表示算法直观形象，可以比较清楚地显示出各个框之间的逻辑关系。流程图用来表示各种操作的图框，图 3.1 所示为几种常见的程序流程图符号。

图 3.1　流程图常见符号

本题的程序流程图如图 3.2 所示。

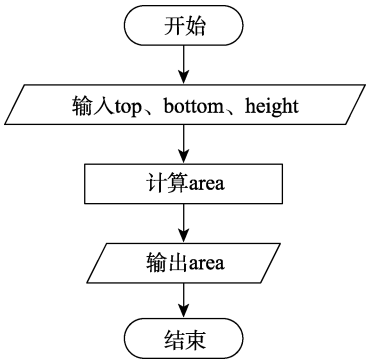

图 3.2　求梯形面积的算法流程图

第三步：根据程序流程图，编写代码。

```
#Example3.1.py

top = float( input('top = ') )
bottom = float( input('bottom = ') )
height = float( input('height = ') )
area = ( top + bottom ) * height / 2
print('梯形的面积为：{:.2f}'.format(area))
```

第四步：对已经编写好的源程序进行上机调试，并且核对结果。如果不正确，则修改程序再调试，直至得到期望的结果值。

3.2　结构化程序设计

结构化程序设计（Structured Programming）是以模块功能和处理过程设计为主的详细设计的基本原则。结构化程序设计是过程式程序设计的一个子集，它对写入的程序使用逻辑结构，使得理解和修改更有效更容易。结构化程序设计的三种基本结构是：顺序结构、

选择结构和循环结构，流程图如图 3.3 所示。

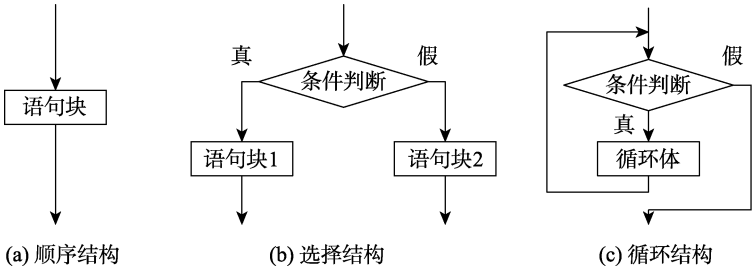

图 3.3　结构化程序设计的三种基本结构

1. 顺序结构

顺序结构表示程序中的各操作是按照它们出现的先后顺序执行的。实例 3.1 便是顺序结构。

2. 选择结构

选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。选择结构有单选择、双选择和多选择三种形式。

3. 循环结构

循环结构表示程序反复执行某个或某些操作，直到某条件为假（或为真）时才可终止循环。在循环结构中最主要的是：什么情况下执行循环？哪些操作需要循环执行？

3.3　简单 if 语句

3.3.1　单分支 if 语句

单分支 if 语句的语法格式如下：

```
if 条件判断:
    语句块
```

单分支 if 语句的流程图如图 3.4 所示。语句块是 if 语句的条件判断成立时执行的一条或多条语句序列，语句块中语句通过与 if 所在行形成缩进表达包含关系。当条件判断为真（True）时，执行语句块；当条件判断为假（False）时，则跳过 if 语句块。

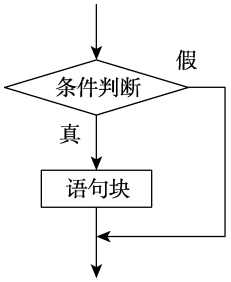

图 3.4　单分支 if 语句流程图

【实例 3.2】计算圆的周长和面积。

```
#Example3.2.py

import math                                                    #①

radius = float(input('周长 = '))                               #②
if radius > 0:                                                 #③
    perimeter = 2 * math.pi * radius                           #④
    area = math.pi * radius ** 2                               #⑤
    print('半径为{:.2f}圆的周长为{:.2f}，面积为{:.2f}'.format(radius,
perimeter,area))

print('计算完毕')                                              #⑥
```

【运行结果一】

```
周长 = 3↙
半径为 3.00 圆的周长为 18.85，面积为 28.27
计算完毕
```

【运行结果二】

```
周长 = -1↙
计算完毕
```

说明：

☑ 为了使用圆周率 pi 的值，语句①使用 import 语句导入 math 模块。

☑ 语句②使用 float()函数将用户输入的表示数值的字符串转换为浮点类型数值。

☑ 语句③当 radius 的值大于 0 成立时，执行 if 语句包含的语句块；否则跳过 if 语句块。通过代码的缩进包含关系，使得 if 语句块由 3 条语句组成。

☑ 语句④⑤通过 math.pi 使用 3.141592653589793，参与圆的周长和面积的计算。

☑ 语句⑥不属于 if 语句，所以不论 if 条件判断是否为真，都将执行。

【实例 3.3】某公园门票正常价格是 40 元，老人（>=60 岁）或儿童（<=12 岁）门票半价。输入游客的年龄，输出游客的年龄和门票价格。

```
#Example3.3.py

price = 40
age = int(input('age = '))
if age >= 60 or age <= 12:                                     #①
     price //= 2                                               #②

print('您的年龄为{}岁,票价为￥{}元.'.format(age,price))
```

【运行结果一】

```
age = 20↙
您的年龄为 20 岁,票价为￥40 元.
```

【运行结果二】

```
age = 10↙
```

```
您的年龄为 10 岁,票价为￥20 元.
```

【运行结果三】

```
age = 60↙
您的年龄为 60 岁,票价为￥40 元.
```

说明：

☑ or 为逻辑或运算，两侧有一个表达式的值为 True 时，逻辑运算表达式结果为 True，见语句①。

☑ 因为 price 的原价为 40，所以计算半价票时用到了“//”整数除法运算符，见语句②。

3.3.2 双分支 if-else 语句

双分支 if-else 语句的语法格式如下：

```
if 条件判断:
     语句块 1
else:
     语句块 2
```

双分支 if-else 语句的流程图如图 3.5 所示。if 语句通过行缩进关系包含语句块 1，else 语句通过行缩进关系包含语句块 2。当条件判断为 True 时，执行语句块 1；当条件判断为 False 时，则执行 else 包含的语句块 2；if 和 else 语句块具有互斥关系。

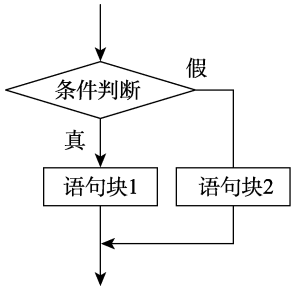

图 3.5 双分支 if-else 语句流程图

【实例 3.4】使用 if-else 语句计算圆的周长和面积。

```
#Example3.4.py

import math                                                    #①导入 math 模块

radius = float(input('周长 = '))
if radius > 0:                                                 #②
    perimeter = 2 * math.pi * radius
    area = math.pi * radius ** 2
    print('半径为{:.2f}圆的周长为{:.2f}，面积为{:.2f}'.format(radius,
perimeter,area))
```

```
else:
    print('半径值必须大于 0! ')

print('计算完毕')                                    #③
```

【运行结果一】

```
周长 = 3↙
半径为 3.00 圆的周长为 18.85，面积为 28.27
计算完毕
```

【运行结果二】

```
周长 = -1↙
半径值必须大于 0!
计算完毕
```

说明：

☑ 在 Python 语言中，使用“#”表示单行注释，“#”后面的文字不是代码，见语句①。

☑ 代码注释的作用就是对代码进行解释说明，为日后的阅读或者他人阅读源程序提供方便。

☑ 虽然没有强行规定程序中一定要写注释，但是为程序代码写注释是一个良好的习惯，这会为以后查看代码带来很大方便。并且如果程序交给别人看，他人便可以快速掌握程序的思想与代码的作用。所以养成编写良好的代码格式规范和添加详细的注释习惯，是一个优秀程序员应该具备的素质。

☑ 当 radius 的值大于 0 成立时，见语句②，执行 if 语句包含的语句块；否则执行 else 语句包含的语句块。

☑ 根据代码行之间的包含关系，语句③既不属于 if 语句，也不属于 else 语句，所以不论 if-else 语句如何执行，它都将执行。

【实例 3.5】计算某年的天数。

```
#Example3.5.py

year = int(input('请输入年份：'))
days = 365
if ( year % 4 == 0 and year %100 != 0 ) or year % 400 == 0: #①
    days += 1
    print('{}年为闰年,共{}天.'.format(year,days))
else:
    print('{}年共{}天.'.format(year,days))
```

【运行结果一】

```
请输入年份：2019↙
2019 年共 365 天.
```

【运行结果二】

```
请输入年份：2000↙
2000 年为闰年,共 366 天.
```

说明：

☑ 判断任意年份是否为闰年，需要满足以下条件中的任意一个：该年份能被 4 整除同时不能被 100 整除；或者该年份能被 400 整除。

☑ if 语句的条件判断表达式使用了逻辑与（and）、逻辑或（or）运算符，因为 and 运算符的优先级高于 or 运算符，所以可以不使用小括号，见语句①。

3.4 复杂 if 语句

3.4.1 if-elif-else 语句

当程序的分支数量大于 2 时，可以使用 if-elif-else 多分支语句，语法格式如下：

```
if 条件判断 1:
     语句块 1
elif 条件判断 2:
     语句块 2
…
else:
     语句块 N
```

多分支 if-elif-else 语句的流程图如图 3.6 所示。elif 语句具有同上一条 if 或 elif 或 else 语句的互斥性，还具有条件判断的功能。当 if 语句条件判断为 True 时，执行其包含的语句块，跳过剩余的 elif 和 else 语句块；当 if 语句条件判断为 False 时，如果某个 elif 语句条件判断为 True 时，执行其包含的语句块，跳过剩余的 elif 和 else 语句块；如果 if 和所有 elif 语句条件判断都为 False 时，则执行 else 包含的语句块。在 if-elif 语句后面并不要求必须有 else 语句块。

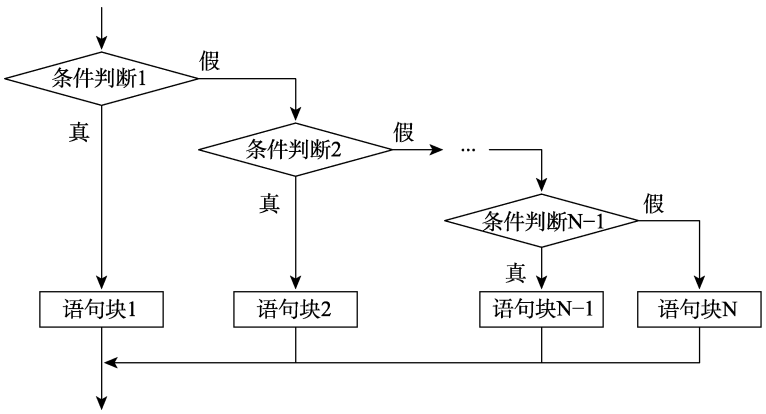

图 3.6　多分支 if-elif-else 语句流程图

【实例 3.6】将输入的百分制成绩转换为 A、B、C、D、E 五个等级。

```
#Example3.6.py

grade = int(input('grade = '))
```

```
if grade > 100:
    print('成绩有误! ')
elif grade >= 90:
    print('Level A.')
elif grade >= 80:
    print('Level B.')
elif grade >= 70:
    print('Level C.')
elif grade >= 60:
    print('Level D.')
elif grade >= 0:
    print('Level E.')
else:
    print('成绩有误! ')
```

【运行结果一】

```
grade = 85↙
Level B.
```

【运行结果二】

```
grade = -78↙
成绩有误!
```

说明：

☑ 根据题干要求进行分析，本题成绩等级的计算共分为七种情况，所以使用 if-elif-else 语句实现。

☑ 将输入的百分制成绩从上向下，依次进行if语句的条件判断或elif语句的条件判断，如果某个条件判断语句为 True 时，则执行其包含的语句块，跳过剩余的 elif 和 else 语句块。

☑ 如果 if 语句和所有的 elif 语句的条件判断都为 False 时，则执行 else 语句块。

☑ 本题的程序流程图如图 3.7 所示。

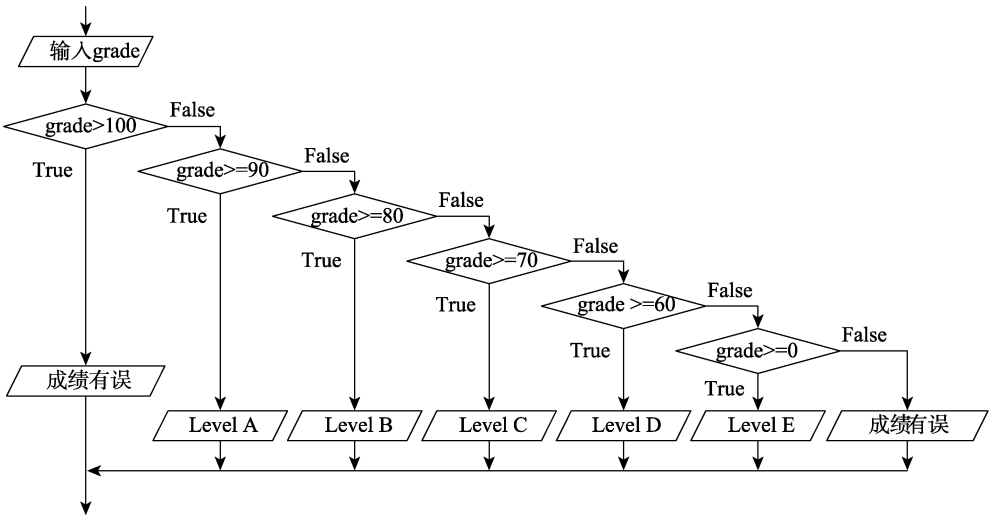

图 3.7　实例 3.6 程序流程图

【实例 3.7】企业发放的奖金根据利润提成。利润低于等于 10 万元时，奖金可提 10%；利润高于 10 万元，低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分，可提成 7.5%；20 万到 40 万之间时，高于 20 万元的部分，可提成 5%；40 万到 60 万之间时高于 40 万元的部分，可提成 3%；60 万到 100 万之间时，高于 60 万元的部分，可提成 1.5%，高于 100 万元时，超过 100 万元的部分按 1%提成，从键盘输入当月利润，求应发放奖金总数。

```
#Example3.7.py

profit = float(input('请输入利润:'))
benefit = 0
if profit <= 100000:
    benefit = profit * 0.1
elif profit <= 200000:
    benefit = (profit - 100000) * 0.075 + 10000
elif profit <= 400000:
    benefit = (profit - 200000) * 0.05 + 10000 + 7500
elif profit <= 600000:
    benefit = (profit - 400000) * 0.03 + 10000 + 7500 + 10000
elif profit <= 1000000:
    benefit = (profit - 600000) * 0.015 + 10000 + 7500 + 10000 + 6000
else:
    benefit = (profit - 1000000) * 0.01 + 10000 + 7500 + 10000 + 6000 +
6000

print('该员工获得的奖金为￥{:.2f}元.'.format(benefit))
```

【运行结果】

```
请输入利润:750000
该员工获得的奖金为￥35750.00元.
```

3.4.2 if-else 嵌套语句

当有多个分支语句时，也可以使用 if-else 嵌套语句实现，语法格式如下：

```
if 条件判断 1:
    if 条件判断 2:
        语句块 1
    else:
        语句块 2
else:
    if 条件判断 3:
        语句块 3
    else:
        语句块 4
```

if-else 嵌套语句的流程图如图 3.8 所示。if 语句的嵌套中，else 与 if 的配对的原则是：else 总是与同一语法层次中离它最近的尚未配对的 if 配对。

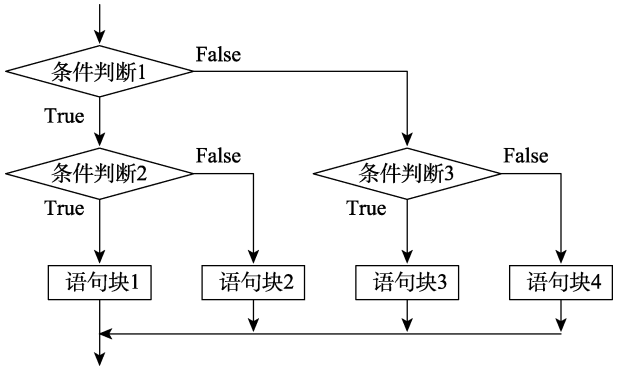

图 3.8　if-else 嵌套语句流程图

【实例 3.8】从键盘输入三个整数值，输出最大值。

```
#Example3.8.py

num1 = int(input('num1 = '))
num2 = int(input('num2 = '))
num3 = int(input('num3 = '))
if num1 >= num2:
    if num1 >= num3:
        max = num1
    else:
        max = num3
else:
    if num2 >= num3:
        max = num2
    else:
        max = num3
print('{},{},{}的最大值为{}.'.format(num1,num2,num3,max))
```

【运行结果】

```
num1 = 3↙
num2 = 5↙
num3 = 4↙
3,5,4 的最大值为 5.
```

说明：

- ☑ 从 else 去找它前面的 if 配对，不能用 if 来找 else 配对。
- ☑ 配对的 if-else 语句必须是同一语法层次的，不能出现在不同的层次里。
- ☑ 必须是离 else 最近的那个 if，不要越级。
- ☑ 找到的这个 if 必须是没有和其他的 else 配对的，不能抢其他 else 的。

【实例 3.9】从键盘输入三个整数，请把这三个数由小到大输出。

```
#Example3.9.py

x,y,z = eval(input('请输入三个整数：'))                    #①
if x > y:
```

```
    if y > z:
        print('从小到大排序结果为{},{},{}'.format(z,y,x))
    elif x > z:
        print('从小到大排序结果为{},{},{}'.format(y,z,x))
    else:
        print('从小到大排序结果为{},{},{}'.format(y,x,z))
else:
    if x > z:
        print('从小到大排序结果为{},{},{}'.format(z,x,y))
    elif y > z:
        print('从小到大排序结果为{},{},{}'.format(x,z,y))
    else:
        print('从小到大排序结果为{},{},{}'.format(x,y,z))
```

【运行结果】

```
请输入三个整数：4,5,3↙
从小到大排序结果为 3,4,5
```

说明：

☑ 本题中从键盘通过执行 input()函数输入“4,5,3”字符串。eval()函数将字符串“4,5,3”的值，依次赋值给“x,y,z”，使得 x=4，y=5，z=3，而且 x、y、z 变量均为整数类型。见语句①。

☑ 本题中 if-else 语句块中分别嵌套了 if-elif-else 语句。

3.5 本 章 小 结

结构化程序设计的三种基本结构是：顺序结构、选择结构和循环结构。顺序结构的程序是一条语句接一条语句顺序地往下执行。选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。循环结构表示程序反复执行某个或某些操作，直到某条件为假（或为真）时才可终止循环。

在 Python 语言中分为单分支、双分支和多分支选择结构。只使用 if 语句可以实现单分支结构，使用 if-else 语句可以实现双分支结构，使用 if-elif-else 语句或 if-else 嵌套语句可以实现多分支结构。

3.6 习 题

一、单项选择题

1. 以下不属于结构化程序设计的三种基本结构是（　　）。

   A. 顺序结构　　B. 选择结构

   C. 逻辑结构　　D. 循环结构

2. 在 Python 语言中，逻辑“真”值等价于（　　）。

   A. 大于零的数　　B. 大于零的整数

   C. 非零的数　　D. 非零的整数

3. 在 if 嵌套语句中，为避免 else 匹配错误，规定 else 总是与（　　）组成配对关系。

A. 最近的 if　　B. 在其之前未配对的 if

C. 在其之前尚未配对的最近的 if　　D. 同一行的 if

4. 关于分支结构，以下选项中描述不正确的是（　　）。

A. if 语句中语句块执行与否依赖于条件判断

B. if 语句中条件部分可以使用任何能够产生 True 和 False 的语句和函数

C. 二分支结构有一种紧凑形式，使用保留字 if 和 elif 实现

D. 多分支结构用于设置多个判断条件以及对应的多条执行路径

5. 以下关于 Python 的控制结构，错误的是（　　）。

A. 每个 if 条件后要使用冒号“:”

B. 在 Python 中，没有 switch-case 语句

C. Python 中的 pass 是空语句，一般用作占位语句

D. elif 可以单独使用

6. 以下关于程序控制结构描述错误的是（　　）。

A. 单分支结构是用 if 保留字判断满足一个条件，就执行相应的处理代码

B. 二分支结构是用 if-else 根据条件的真假，执行两种处理代码

C. 多分支结构是用 if-elif-else 处理多种可能的情况

D. 在 Python 的程序流程图中可以用处理框表示计算的输出结果

7. 关于 Python 的分支结构，以下选项中描述错误的是（　　）。

A. 分支结构使用 if 保留字

B. Python 中 if-else 语句用来形成二分支结构

C. Python 中 if-elif-else 语句描述多分支结构

D. 分支结构可以向已经执行过的语句部分跳转

8. 下面程序执行后，a、b、c 值是（　　）。

```
a = 1
b = 2
c = 3
if a>b:
    c=a
    a=b
b=c
```

A. a=1, b=3, c=3　　B. a=1, b=1, c=3

C. a=2, b=3, c=3　　D. a=2, b=2, c=3

9. Python 语言中，if 语句的条件判断表达式的值叙述正确的是（　　）。

A. 必须是逻辑值　　B. 必须是整数值

C. 必须是正数　　D. 可以是任意合法的数值

10. 以下程序的输出结果是（　　）。

```
t = 'Python'
print(t if t>='python' else 'None')
```

A. Python　　B. python

C. t　　D. None

11. 以下程序的输出结果是（　　）。

```
a = 30
b = 1
if a >=10:
    a = 20
elif a>=20:
    a = 30
elif a>=30:
    b = a
else:
    b = 0
print('a={}, b={}'.format(a,b))
```

A. a=30, b=1　　B. a=30, b=30
C. a=20, b=20　　D. a=20, b=1

二、判断题

1. if 和 else 语句必须成对出现。（　　）
2. Python 里每一行语句后必须用分号来结束。（　　）
3. Python 使用“#”号来标示单行注释。（　　）
4. 变量名在引用前必须赋值。（　　）
5. Python 中在语句块周围采用缩进形式将语句分组。（　　）

三、程序填空题

1. 以下程序的功能是：输入一个整数，如果输入的整数大于等于 100，输出显示该数大于等于 100。否则输出显示该数小于 100。请填空。

```
num = int(input('num = '))
if _________________:
    print('{}的值大于等于100'.format(num))
else:
    print('{}的值小于100'.format(num))
```

2. 以下程序的功能是：已知计算三角形面积的公式为：area = sqrt(s*(s-a)*(s-b)*(s-c))，其中 s=(a+b+c)/2。公式中 a、b 和 c 分别为三角形的三条边。编写程序判断三边能否构成三角形。如果能构成三角形求该三角形的面积，否则输出“无法构成三角形”。

```
import math
a,b,c = eval(input('请输入三角形三条边的值：'))
if ”_________________”
    s = (a + b + c) / 2
    area = math.sqrt( s * (s - a) * (s - b) * (s - c) )
    print(area)
else:
    print('无法构成三角形')
```

四、编程题

1. 输入两个整数，输出其中较大的数。
2. 输入一个正整数，判断该数是否既是 5 又是 7 的倍数。若是，输出“yes”；若否，

输出“no”。

3. 输入整型变量 x 的值：当 x<1 时，y=x；当 1<=x<10 时，y=2x-1；当 x>=10 时，y=3x+11；最后输出 y 的值。

4. 用 if 语句实现：输入一个字符，判断该字符是数字、英文字母还是其他字符。

5. 输入两个数和一个符号。如果该符号为‘+’，则输出两个数的和。如果该符号为‘－’，则输出两个数的差。如果该符号为‘*’，则输出两个数的积。如果该符号为‘/’，则输出两个数的除。

6. 人体指数 BMI 的计算为：BMI = 体重（kg）/（身高*身高）（m）。下面是 BMI 对应的国际标准指数与国内标准指数，请实现用户输入身高、体重，输出二者对应的指数值。

分类	国际 BMI 值（kg/m^2）	国内 BMI 值（kg/m^2）
偏瘦	<18.5	<18.5
正常	18.5~25	18.5~24
偏胖	25.1~29.9	24.1~27.9
肥胖	≥30	≥28

7. 要求输入某年某月某日，求判断输入日期是当年中的第几天？