第 3 章　选 择 结 构

选择结构又称为分支结构，它根据给定的条件是否满足，决定程序的执行路线。在不同的条件下执行不同的操作，这在实际求解问题过程中是大量存在的。例如，输入一个整数，若判断它是否为偶数，则需要使用选择结构来实现。根据程序执行路线或分支的不同，选择结构又分为单分支、双分支和多分支三种类型。例如，输入学生的成绩，需要统计及格学生的人数、统计及格和不及格学生的成绩、统计不同分数段学生的人数，这里就涉及单分支、双分支和多分支的选择结构。

实现选择结构，首先涉及如何表示条件，然后是如何实现选择结构。Python 提供 if 语句来实现选择结构。

本章介绍内容包括条件的描述、选择结构的实现、条件运算、选择结构程序举例等。

3.1 条件的描述

Python 提供了关系运算和逻辑运算来描述程序控制中的条件，这是一般程序设计语言均有的方法。此外，Python 还用成员运算和身份运算来表示条件。

3.1.1 关系运算

Python 的关系运算符有：

```
<（小于）、<=（小于或等于）、>（大于）、>=（大于或等于）、==（等于）、!=（不等于）
```

关系运算符用于两个量的比较判断。由关系运算符将两个表达式连接起来的式子就称为关系表达式，它用来表示条件，其一般格式如下：

```
表达式 1  关系运算符  表达式 2
```

例如，关系表达式 x>=7.8，当 x 的值为 13.14 时，x>=7.8 条件满足；当 x 的值为 0 时，x>=7.8 条件不满足。关系表达式的值是一个逻辑值，即结果为真（True）或假（False）。习惯称表达式值为 True 时，条件满足；表达式值为 False 时，条件不满足。设有 i=1，j=2，k=3，则 i>j 的值为 False，i+j==k 的值为 True。看下面的语句执行结果。

```
>>> i,j,k=1,2,3
>>> i>j
False
>>> i+j==k
True
```

关系运算符的优先级相同，但关系运算符的优先级低于算术运算符的优先级。例如，“a<b+c”等价于“a<(b+c)”。

3.1.2 逻辑运算

1. 逻辑运算符

Python 的逻辑运算符有 and(逻辑与)、or(逻辑或)、not(逻辑非)。

其中，and 和 or 逻辑运算符要求有两个运算量，用于连接两个条件，构成更复杂的条件。not 逻辑运算符只作用于后面的一个逻辑量。逻辑运算产生的结果是一个逻辑量，即 True 或 False。在逻辑运算符中，not 的优先级最高，其次是 and，or 的优先级最低。

2. 逻辑表达式

逻辑表达式是用逻辑运算符将逻辑量连接起来的式子。除 not 以外，and 和 or 构成的逻辑表达式一般形式如下：

```
P 逻辑运算符 Q
```

其中，P 和 Q 是两个逻辑量。

若逻辑运算符为 and，则当连接的两个逻辑量全为 True 时，逻辑表达式取值为 True，只要有一个为 False，便取 False 值。若逻辑运算符为 or，则当连接的两个逻辑量中只要有一个为 True 时，逻辑表达式的值为 True；只有当两个逻辑量同时为 False 时，才产生 False 值。not 对后面的逻辑量取非，如果逻辑量为 False，便产生 True 值；如果逻辑量为 True，便产生 False 值。逻辑运算的功能可用如表 3-1 所示的逻辑运算真值表来表示，表中用 T 表示 True，F 表示 False。

表 3-1 逻辑运算真值表

P	Q	P and Q	P or Q	not P
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

由于在计算机内部以 1 表示 True、0 表示 False，所以参与逻辑运算的分量也可以是其他类型的数据，以非 0 和 0 判定它们是 True 还是 False。例如：

```
>>> not 0
True
>>> not "AA"
False
>>> 3+(not False)
4
```

在程序设计中，常用关系表达式和逻辑表达式表示条件。下面看一些具体例子。

例 3-1 写出下列条件。

（1）判断年份 year 是否为闰年。

（2）判断 ch 是否为小写字母。

（3）判断 m 能否被 n 整除。

（4）判断 ch 既不是字母也不是数字字符。

条件 1：

每 4 年一个闰年，但每 100 年少一个闰年，每 400 年又增加一个闰年。因此，year 年是闰年的条件可用逻辑表达式描述如下：

```
(year%4==0 and year%100!=0) or year%400==0
```

条件 2：

考虑到字母在 ASCII 码表中是连续排列的，ch 中的字符是小写字母的条件可用逻辑表达式描述如下：

```
ch>='a' and ch<='z'
```

条件 3：

m 能被 n 整除，即 m 除以 n 的余数为 0，故表示条件的表达式如下：

```
m%n==0 或 m-m//n*n==0
```

条件 4：

先写 ch 是字母（包括大写或小写字母）或数字字符的条件，然后将该条件取非，故表示条件的表达式如下：

```
not((ch>='A' and ch<='Z') or (ch>='a' and ch<='z') or (ch>='0' and ch<='9'))
```

3. 逻辑运算的重要规则

逻辑与（and）和逻辑或（or）运算分别有如下性质。

（1）a and b：当 a 为 False 时，不管 b 为何值，结果为 False。

（2）a or b：当 a 为 True 时，不管 b 为何值，结果为 True。

Python 利用上述性质，在计算连续的逻辑与运算时，若有运算分量的值为 False，则不再计算后续的逻辑与运算分量，并以 False 作为逻辑与算式的结果；若前面的运算分量的值为 True，则以后续的逻辑与运算分量作为逻辑与算式的结果。在计算连续的逻辑或运算时，若有运算分量的值为 True，则不再计算后续的逻辑或运算分量，并以 True 作为逻辑或算式的结果；若前面的运算分量的值为 False，则以后续的逻辑或运算分量作为逻辑或算式的结果。也就是说，对于 a and b，若 a 的值可解释为 False，则表达式的值为 False，否则表达式的值为 b；对于 a or b，如果 a 的值为 False，则表达式的值为 b，否则表达式的值为 True。

仔细分析，这种规则是很自然的。以 a and b 为例，当 a 的值不能解释为 True（为 False）时，则不管 b 为何值，表达式的值总为 False；当 a 的值能解释为 True 时，则表达式的值就取决于 b 的值，若 b 的值为 True 则表达式的值为 True，若 b 的值为 False 则表达式的值为 False，因此返回 b 的值作为表达式的值。总之，当且仅当 a 和 b 的值都可解释为 True 时，表达式“a and b”的值才为 True。这完全符合逻辑运算的定义。看下面的例子。

```
>>> 2 and "Program"
'Program'
>>> 2>1 and "Program"
'Program'
>>> 2>1 and 2>0 and 10
10
```

```
>>> 2>1 and 2==0 and 10
False
>>> 1==1 and 5>4
True
```

3.1.3 测试运算

1. 成员测试

除以上的关系运算和逻辑运算符外，Python 还支持成员运算符，测试实例中包含了一系列的成员，包括字符串、列表或元组。

in 运算符用于在指定的序列中查找某个值是否存在，若存在则返回 True，否则返回 False。该运算符的使用格式是 x in y，如果 x 在 y 序列中则返回 True，否则返回 False。例如：

```
>>> 3 in(20,15,3,14,5)
True
```

“not in”的含义是，如果在指定的序列中没有找到值，则返回 True，否则返回 False。对于 x not in y，如果 x 不在 y 序列中则返回 True，否则返回 False。例如：

```
>>> 3 not in(20,15,3,14,5)
False
```

2. 身份测试

身份运算符用于测试两个变量是否指向同一个对象。例如：

```
>>> a=20
>>> b=20
>>> a is b
True
>>> a is not b
False
```


3.2 选择结构的实现

选择结构根据给定的条件满足或不满足，分别执行不同的语句。它可分为单分支、双分支和多分支选择结构。Python 提供了实现选择结构的 if 语句。

3.2.1 单分支选择结构

可以用 if 语句实现单分支选择结构，其一般格式如下：

```
if 表达式：
    语句块
```

其中，表达式用来表示条件。语句的执行过程是：计算表达式的值，若值为 True，则执行语句块，然后执行 if 语句的后续语句。若值为 False，则直接执行 if 语句的后续语句。单分支 if 语句的执行过程如图 3-1 所示。

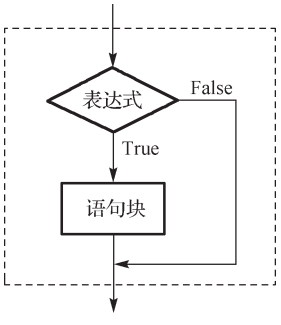

图 3-1 单分支 if 语句的执行过程

注意：

（1）在 if 语句的表达式后面必须加冒号。

（2）因为 Python 把非 0 当成真、0 当成假，所以表示条件的表达式不一定必须是结果为 True 或 False 的关系表达式或逻辑表达式，可以是任意表达式。例如，下列语句是合法的，将输出字符串“BBBBB”。

```
if 'B':
    print('BBBBB')
```

if 语句中条件表示的多样性，可以使得程序的描述灵活多变，但从提高程序可读性的角度考虑，还是直接用逻辑判断为好，因为这样更能表达程序员的思想意图，有利于日后对程序的维护。

（3）if 语句中的语句块必须向右缩进，语句块既可以是单个语句，也可以是多个语句。当包含两个或两个以上的语句时，语句必须缩进一致，即语句块中的语句必须上下对齐。例如：

```
if x>y:
    x=10
    y=20
```

若语句块中的语句缩进不一致，则语句含义就不同了。

（4）如果语句块中只有一条语句，if 语句也可以写在同一行上。例如：

```
var=100
if var==100:print("Value of expression is 100")
print("Good bye!")
```

程序运行结果如下：

```
Value of expression is 100
Good bye!
```

例 3-2　输入两个整数（a 和 b），先输出较大数，然后输出较小数。

分析：输入 a、b，若 a<b，则交换 a 和 b，否则不交换，最后输出 a、b。

程序如下：

```
a,b=eval(input(" 输入 a,b:"))
if a<b:                    # 若 a<b，则交换 a 和 b，否则不交换
    a,b=b,a
print(f"{a},{b}")
```

程序运行结果如下：

```
输入 a,b:234,1251 ↙
1251,234
```

3.2.2　双分支选择结构

可以用 if 语句实现双分支选择结构，其一般格式如下：

```
if 表达式 :
```

```
    语句块 1
else:
    语句块 2
```

语句执行过程是：计算表达式的值，若为 True，则执行语句块 1，否则执行 else 后面的语句块 2，语句块 1 或语句块 2 执行后再执行 if 语句的后续语句。其双分支 if 语句的执行过程如图 3-2 所示。

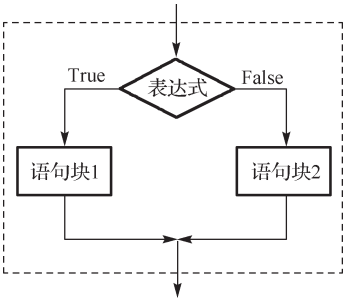

图 3-2　双分支 if 语句的执行过程

注意：与单分支 if 语句一样，对于表达式后面或 else 后面的语句块，应将它们缩进对齐。例如：

```
if i%2==1:
    x=i/2
    y=i*i
else:
    x=i
    y=i*i*i
```

例 3-3　输入三角形的三个边长，求三角形的面积。

分析：设 a、b、c 表示三角形的三个边长，则构成三角形的充分必要条件是任意两边之和大于第三边，即 $a+b>c$，$a+c>b$，$b+c>a$。如果该条件满足，则可按照海伦公式计算三角形的面积：

$$s=\sqrt{p(p-a)(p-b)(p-c)}$$

其中，$p=(a+b+c)/2$。

程序如下：

```
from math import *
a,b,c=eval(input("a,b,c="))
if a+b>c and a+c>b and b+c>a:
    p=(a+b+c)/2
    s=sqrt(p*(p-a)*(p-b)*(p-c))
    print(f"a={a},b={b},c={c}")
    print(f"area={s}")
else:
    print(f"a={a},b={b},c={c}")
    print("input data error")
```

程序的一次运行结果如下：

```
a,b,c=6,7,5 ↙
a=6,b=7,c=5
area=14.696938456699069
```

再次运行程序的结果如下：

```
a,b,c=34,3,21 ↙
a=34,b=3,c=21
input data error
```

选择结构程序运行时，每次只能执行一个分支，所以在检查选择结构程序的正确性时，设计的原始数据应包括每种情况，保证每条分支都检查到。例 3-3 的程序运行时，首先输入的三边能构成一个三角形，求出其面积。再次运行程序时，输入的三边不能构成一个三角形，提示用户输入数据有误。

例 3-4　输入 x，求对应的函数值 y。

$$y=\begin{cases}\ln(-5x)+|x|, & x<0\\ \sin x+\dfrac{\sqrt{x+\mathrm{e}^2}}{2\pi}, & x\geqslant 0\end{cases}$$

分析：这是一个具有两个分支的分段函数，为了求函数值，可以采用双分支结构来实现。程序如下：

```
from math import *
x=eval(input("x="))
if x<0:
    y=log(-5*x)+fabs(x)
else:
    y=sin(x)+sqrt(x+exp(2))/(2*pi)
print(f"x={x},y={y}")
```

还可以采用两个单分支结构来实现，程序如下：

```
from math import *
x=eval(input("x="))
if x<0:
    y=log(-5*x)+fabs(x)
if x>=0:
    y=sin(x)+sqrt(x+exp(2))/(2*pi)
print(f"x={x},y={y}")
```

请思考，第一个 if 语句能否不写，即程序能否改写成如下形式，并分析原因。

```
from math import *
x=eval(input("x="))
y=log(-5*x)+fabs(x)
if x>0:
    y=sin(x)+sqrt(x+exp(2))/(2*pi)
print(f"x={x},y={y}")
```

再思考，第二个 if 语句能否不写，即程序能否改写成如下形式，并分析原因。

```
from math import *
x=eval(input("x="))
if x<0:
    y=log(-5*x)+fabs(x)
y=sin(x)+sqrt(x+exp(2))/(2*pi)
print(f"x={x},y={y}")
```

3.2.3 多分支选择结构

多分支 if 语句的一般格式如下：

```
if 表达式 1:
    语句块 1
elif 表达式 2:
    语句块 2
elif 表达式 3:
    语句块 3
  …
elif 表达式 m:
    语句块 m
[else:
    语句块 n]
```

多分支 if 语句的执行过程如图 3-3 所示。当表达式 1 的值为 True 时，执行语句块 1，否则求表达式 2 的值；为 True 时，执行语句 2，否则处理表达式 3，依次类推。若表达式的值都为 False，则执行 else 后面的语句 n。不管有几个分支，程序执行完一个分支后，其余分支将不再执行。

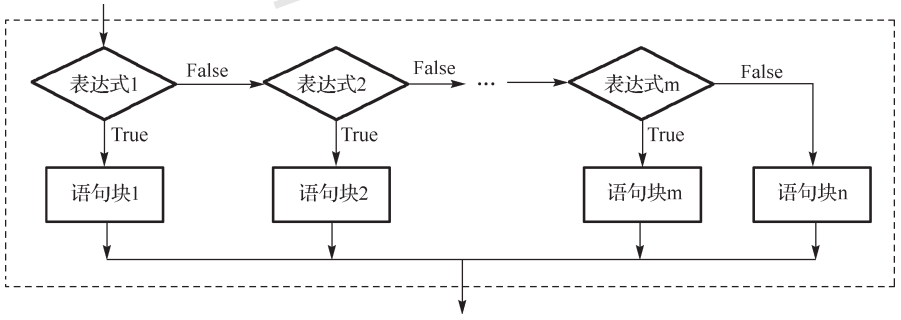

图 3-3　多分支 if 语句的执行过程

请思考：当表达式 1 和表达式 2 都为 True 时，语句的执行路线如何？

例 3-5　输入学生的成绩，根据成绩进行分类，85 分以上为优秀，70~84 分为良好，60~69 分为及格，60 分以下为不及格。

分析：将学生成绩分为四个分数段，然后根据各分数段的成绩，输出不同的等级。程序分为四个分支，既可以用四个单分支结构实现，也可以用多分支 if 语句实现。

程序如下：

```
g=float(input(" 请输入学生成绩 :"))
if g<60:
```

```
    print(" 不及格 ")
elif g<70:
    print(" 及格 ")
elif g<85:
    print(" 良好 ")
else:
    print(" 优秀 ")
```

程序的一次运行结果如下：

```
请输入学生成绩 :45 ↙
不及格
```

再次运行程序的结果如下：

```
请输入学生成绩 :76 ↙
良好
```

3.2.4　选择结构的嵌套

在 if 语句中可以再嵌套 if 语句，例如，有以下不同形式的嵌套结构。

语句一：

```
if 表达式 1:
    if 表达式 2:
        语句块 1
    else:
        语句块 2
```

语句二：

```
if 表达式 1:
    if 表达式 2:
        语句块 1
else:
    语句块 2
```

根据对齐格式来确定 if 语句之间的逻辑关系。在第一个语句中，else 与第二个 if 配对。在第二个语句中，else 与第一个 if 配对。

为了使嵌套层次清晰明了，在程序的书写上常常采用缩进格式，即不同层次的 if-else 出现在不同的缩进级别上。在 Python 语言中，语句的缩进格式代表了 else 和 if 的逻辑配对关系，同时也增强了程序的可读性。

例 3-6　用 if 语句的嵌套实现例 3-5。

程序如下：

```
g=float(input(" 请输入学生成绩 :"))
if g>=60:
    if g>=70:
        if g>=85:
            print(" 优秀 ")
        else:
```

```
            print(" 良好 ")
    else:
        print(" 及格 ")
else:
    print(" 不及格 ")
```

例 3-7　输入三个数，找出其中的最大数。

分析：求三个数中最大数的具体方法是，输入三个数到 x、y、z 后，先假定第一个数是最大数，即将 x 送到 max 变量，然后将 max 分别和 y、z 比较，两次比较后，max 的值为 x、y、z 中的最大数。这里用嵌套的 if 结构来实现，看下面的程序：

```
x,y,z=eval(input("x,y,z=?"))
max=x
if z>y:
    if z>x:
        max=z
else:
    if y>x:
        max=y
print(f"The max is {max}")
```

程序的运行结果如下：

```
x,y,z=?12.3,4.5,34.9 ↙
The max is 34.9
```

程序中嵌套使用了 if 语句，要特别注意 if 和 else 语句的配对关系。

3.3 条件运算

Python 的条件运算有三个运算量，其一般格式如下：

```
表达式 1 if 表达式 else 表达式 2
```

条件运算的运算规则是，先求 if 后面表达式的值，如果其值为 True，则求表达式 1，并以表达式 1 的值为条件运算的结果。如果 if 后面表达式的值为 False，则求表达式 2，并以表达式 2 的值为条件运算的结果。例如：

```
>>> x,y=40,30
>>> z=x if x>y else y
>>> z
40
```

如果条件 x>y 满足，则条件运算取 x 的值，否则取 y 的值，即取 x、y 中较大的值。

注意：条件运算构成一个表达式，它可以作为一个运算量而出现在其他表达式中，它不是一个语句。

使用条件运算表达式可以使程序简洁明了。例如，赋值语句“z=x if x>y else y”中使用了条件运算表达式，很简洁地表示了判断变量 x 与 y 的较大值并赋给变量 z 的功能。因此，使用条件运算表达式可以简化程序。

另外，条件运算的三个运算分量的数据类型具有多样性。例如：

```
>>> i=45
>>> 'a' if i else 'A'
'a'
>>> i=0
>>> 'a' if i else 'A'
'A'
```

其中，i 是整型变量，若 i 的值为非 0，即代表 True，所以条件运算表达式的值为“a”，否则条件运算表达式的值为“A”。

例 3-8　生成三个 2 位随机整数，输出其中最大的数。

这里用条件运算表达式来实现，程序如下：

```
import random
x=random.randint(10,99)
y=random.randint(10,99)
z=random.randint(10,99)
max=x if x>y else y
max=max if max>z else z
print(f"x={x},y={y},z={z}".format(x,y,z))
print(f"max={max}")
```

程序的运行结果如下：

```
x=82,y=46,z=28
max= 82
```

在例 3-7 和例 3-8 中介绍了求三个数中最大数的不同实现方法，这说明了程序实现方法的多样性。在学习过程中，需要不断总结，以选择最简洁、效率最高的实现方法。

3.4 选择结构程序举例

解决在不同条件下执行不同操作的问题需要使用选择结构。选择结构的执行是依据一定的条件选择程序的执行路径，程序设计的关键在于构造合适的分支条件和分析程序流程，根据不同的程序流程选择适当的分支语句。为了加深对选择结构程序设计方法的理解，下面再看几个例子。

例 3-9　输入一个整数，判断它是否为水仙花数。所谓水仙花数，是指这样的一些 3 位整数：各位数字的立方和等于该数本身，例如 $153=1^3+5^3+3^3$，因此 153 是水仙花数。

分析：关键的一步是先分别求 3 位整数个位、十位、百位数字，再根据条件判断该数是否为水仙花数。

程序如下：

```
x=eval(input())
a=x%10                          # 求个位数字
b=(x//10)%10                    # 求十位数字
```

```
c=x//100                                    # 求百位数字
if x==a*a*a+b*b*b+c*c*c:
    print(f"{x} 是水仙花数 ")
else:
    print(f"{x} 不是水仙花数 ")
```

程序的一次运行结果如下：

```
153 ↙
153 是水仙花数
```

再次运行程序的结果如下：

```
223 ↙
223 不是水仙花数
```

例 3-10　输入一个时间（小时：分钟：秒），输出该时间经过 5 分 30 秒后的时间。

程序如下：

```
hour=int(input(' 请输入小时 :'))
minute=int(input(' 请输入分钟 :'))
second=int(input(' 请输入秒 :'))
second+=30
if second>=60:
    second=second-60
    minute+=1
minute+=5
if minute>=60:
    minute=minute-60
    hour+=1
if hour==24:
    hour=0
print(f'{hour}:{minute}:{second}')
```

程序运行结果如下：

```
请输入小时 :8 ↙
请输入分钟 :12 ↙
请输入秒 :56 ↙
8:18:26
```

例 3-11　硅谷公司员工的工资计算方法如下：

（1）工作时数超过 120 小时者，超过部分加发 15%。

（2）工作时数低于 60 小时者，扣发 700 元。

（3）其余按 84 元每小时计发。

输入员工的工号和该员工的工作时数，计算应发工资。

分析：为了计算应发工资，首先分两种情况，即工时数小于或等于 120 小时和大于 120 小时。工时数超过 120 小时时，实发工资有规定的计算方法。而工时数小于或等于 120 小时时，又分为大于 60 和小于或等于 60 两种情况，分别有不同的计算方法。因此，程序分为 3 个分支，即工时数 >120、60< 工时数≤ 120 和工时数≤ 60，既可以用多分支 if 结构实现，也可以用 if

的嵌套实现。

if 嵌套的程序如下：

```
IDNumber,WorkingHours=eval(input(" 请输入工号和工时："))
if WorkingHours>120:
    Payment=WorkingHours*84+(WorkingHours-120)*84*0.15
else:
    if WorkingHours>60:
        Payment=WorkingHours*84
    else:
        Payment=WorkingHours*84-700
print(f"{IDNumber} 号职工应发工资￥{Payment:.2f}.")
```

例 3-12　输入年月，求该月的天数。

分析：用 year、month 分别表示年和月，day 表示每月的天数。考虑到以下两点。

（1）每年的 1、3、5、7、8、10、12 月，每月有 31 天；4、6、9、11 月，每月有 30 天；闰年 2 月有 29 天，平年 2 月有 28 天。

（2）年份能被 4 整除，但不能被 100 整除，或者能被 400 整除的年均是闰年。

程序如下：

```
year=int(input("year="))
month=int(input("month="))
if month in(1,3,5,7,8,10,12):
    day=31
elif month in(4,6,9,11):
    day=30
else:
    logi=(year%4==0 and year%100!=0) or year%400==0
    day=29 if logi else 28
print(f"{year},{month},{day}")
```

程序的一次运行结果如下：

```
year=2022 ↙
month=12 ↙
2022,12,31
```

再次运行程序的结果如下：

```
year=2000 ↙
month=2 ↙
2000,2,29
```

还可以使用 calendar 模块的 isleap() 函数来判断闰年。例如：

```
>>> import calendar
>>> calendar.isleap(2022)
False
>>> calendar.isleap(2000)
True
```

习　题　3

一、选择题

1．以下不合法的表达式是（　）。

A．x in [1,2,3,4,5]　　B．x-6>5　　C．e>5 and 4==f　　D．3=a

2．将数学公式 2<x ≤ 10 表示成正确的 Python 表达式为（　）。

A．2<x ≤ 10　　B．2<x and x<=10　　C．2<x && x<=10　　D．x>2 or x <=10

3．与关系表达式 x==0 等价的表达式是（　）。

A．x=0　　B．not x　　C．x　　D．x!=1

4．下列表达式的值为 True 的是（　）。

A．2!=5 or 0　　B．3>2>2　　C．5+4j>2-3j　　D．1 and 5==0

5．下面 if 语句统计“成绩（mark）优秀的男生和不及格的男生”的人数，正确的语句为（　）。

A．if gender==" 男 " and mark<60 or mark>=90:n+=1

B．if gender==" 男 " and mark<60 and mark>=90:n+=1

C．if gender==" 男 " and (mark<60 or mark>=90):n+=1

D．if gender==" 男 " or mark<60 or mark>=90:n+=1

6．以下 if 语句中，语法错误的是（　）。

A．
```
if a>0:x=20
  else:x=200
```

B．
```
if a>0:x=20
else:
    x=200
```

C．
```
if a>0:
    x=20
  else:x=200
```

D．
```
if a>0
    x=20
else
    x=200
```

7．在 Python 中，实现多分支选择结构的较好方法是（　）。

A．if　　B．if-else　　C．if-elif-else　　D．if 嵌套

8．下列语句执行后的输出结果是（　）。

```
if 2:
    print(5)
else:
    print(6)
```

A．0　　B．2　　C．5　　D．6

9．下面程序段求 x 和 y 中的较大数，不正确的是（　）。

A．
```
maxNum=x if x>y else y
```

B．
```
if x>y:maxNum=x
  else:maxNum=y
```

C．
```
maxNum=y
  if x>y:maxNum=x
```

D．
```
if y>=x:maxNum=y
  maxNum=x
```

10．下列 Python 程序的运行结果是（　）。

```
x=0
y=True
print(x>y and 'A'<'B')
```

A．True　　B．False　　C．true　　D．false

二、填空题

1．表达式 2<=1 and 0 or not 0 的值是_____。

2．若已知 ans='n'，则表达式 ans=='y' or 'Y' 的值为_____。

3．Python 提供了两个对象身份比较运算符_____和_____来测试两个变量是否指向同一个对象。

4．在直角坐标中，*x*、*y* 是坐标系中任意点的位置，用 *x* 和 *y* 表示第一象限或第二象限的 Python 表达式为_____。

5．若已知 a=3、b=5、c=6、d=True，则表达式 not d or a>=0 and a+c>b+3 的值是_____。

6．Python 表达式 16-2*5>7*8/2 or "XYZ"!="xyz" and not(10-6>18/2) 的值为_____。

7．下列 Python 语句的运行结果是_____。

```
x=True
y=False
z=False
print(x or y and z)
```

8．执行下列 Python 语句将产生的结果是_____。

```
m=True
n=False
p=True
b1=m|n^p; b2=n|m^p
print(b1,b2)
```

9．对于 if 语句中的语句块，应将它们_____。

10．当 x=0、y=50 时，语句 z=x if x else y 执行后，z 的值是_____。

三、问答题

1．写出条件“20<x<30 或 x<-100”的 Python 表达式。

2．Python 实现选择结构的语句有哪些？各种语句的格式是十么？

3．下列两个语句各自执行后，x 和 y 的值是多少？它们的作用是什么？

```
x=y=5
x=y==5
```

4．下列 Python 语句的运行结果为_____。

```
x=False
y=True
z=False
if x or y and z:print("yes")
else:print("no")
```

5．下列 Python 语句的运行结果为_____。

```
x=True
y=False
z=True
if not x or y:print(1)
elif not x or not y and z:print(2)
elif not x or y or not y and x:print(3)
else:print(4)
```

6．说明以下三个 if 语句的区别。

语句一：

```
if i>0:
    if j>0:n=1
    else:n=2
```

语句二：

```
if i>0:
    if j>0:n=1
else:n=2
```

语句三：

```
if i>0:n=1
else:
    if j>0:n=2
```