

G2-09 课后练习 —— 短路求值

姓名： 日期：

记忆钩子：

- "and全真才真=严格老师 or有真就真=好说话老师 not取反=反义词"
- "先比大小，再算逻辑；遇到 not，先取反"

1. 布尔运算填空

下面这些布尔运算的结果是什么？在横线上填「True」或「False」。

True and True	=		两个真 and结果是？
True and False	=		一真一假 and结果是？
False and True	=		一假一真 and结果是？
False and False	=		两个假 and结果是？

True or True	=		两个真 or结果是？
True or False	=		一真一假 or结果是？
False or True	=		一假一真 or结果是？
False or False	=		两个假 or结果是？

not True	=		not 取反 True变什么？
not False	=		not 取反 False变什么？

小提示：想想三位老师——严格老师（and）、好说话老师（or）、反义词怪兽（not）！

2. 混合运算挑战——先比大小，再算逻辑

先算出比较结果（True / False），再算出最终结果。

(10 > 5) and (3 < 1)	=	
# 第一步：10 > 5 =	3 < 1 =	
# 第二步： and	=	

(10 > 5) or (3 < 1)	=	
# 第一步：10 > 5 =	3 < 1 =	

```
# 第二步: ____ or ____ = ____
```

```
not (10 > 5) = ____
```

```
# 第一步: 10 > 5 = ____
```

```
# 第二步: not ____ = ____
```

```
(5 >= 5) and (6 != 6) = ____
```

```
(5 >= 5) or (6 != 6) = ____
```

```
not (5 >= 5) = ____
```

```
(20 == 20) and (30 != 30) and (10 > 5) = ____
```

```
# 提示: 三个条件 and 连接, 全部真才真
```

```
(20 == 20) or (30 != 30) or (10 > 5) = ____
```

```
# 提示: 三个条件 or 连接, 有一个真就是真
```

3. 代码补全——安全区检测 (and)

小乌龟的安全区是一个方框: x 从 -100 到 100, y 从 -100 到 100。

在框里 绿色 (安全!), 在框外 红色 (出界!)

请补全下面代码中的四个「__」, 填上正确的边界值。

```
import turtle
t = turtle.Turtle()
t.speed(3)
```

```
# 画安全区边框 (灰色方框)
t.color("gray")
t.penup()
t.goto(-100, -100)
t.pendown()
for _ in range(4):
    t.forward(200)
    t.left(90)
```

```
# 标注"安全区"
t.penup()
t.goto(0, 120)
t.write("安全区", align="center", font=("Arial", 12, "bold"))
```

```
# 测试位置
test_spots = [(0, 0), (-150, 50), (80, -90), (100, 100), (-50, -60)]
```

```
t.penup()
for x, y in test_spots:
    t.goto(x, y)
```

```
# 补全四个边界值
if x > _____ and x < _____ and y > _____ and y < _____:
    t.color("green") # 安全! 四条边都满足
    t.write("安全", font=("Arial", 10, "bold"))
else:
    t.color("red") # 出界! 至少一条边不满足
    t.write("出界", font=("Arial", 10, "bold"))
```

```
t.dot(40)
```

```
turtle.done()
```

想一想:

- 安全区一共有 _____ 道墙 (前、后、左、右) 。
- 四道墙 全部 满足才算安全 必须用 _____ (and / or) 。
- 如果把代码里的 `and` 全部改成 `or`, 框外的点也会变 _____ 色! 因为 or 是"有一个对就行"。

4. 代码补全——围栏反弹 (or)

小乌龟在围栏里随机乱走, 碰到任意一道墙就要弹回来。

围栏范围: x 从 -150 到 150, y 从 -150 到 150。

请补全下面代码中的六个「__」: 四个填边界值, 两个填逻辑运算符 (「and」或「or」)。

```
import turtle
import random
```

```
t = turtle.Turtle()
t.shape("turtle")
t.speed(5)
t.pensize(3)
```

```
# 画围栏边框 (灰色大方框)
t.color("lightgray")
t.penup()
t.goto(-150, -150)
t.pendown()
for _ in range(4):
    t.forward(300)
    t.left(90)
```

```
# === 随机漫步 + 围栏反弹 ===
t.color("purple")
t.penup()
```

```
t.goto(0, 0)
t.pendown()
```

```
for step in range(80):
    # 随机选一个方向走一步
    t.setheading(random.choice([0, 90, 180, 270]))
    t.forward(15)
```

```
# 碰左右墙 反弹! (碰到左墙或右墙, 有一个就弹)
if t.xcor() < _____ t.xcor() > _____:
    t.backward(15) # 填边界 填 and/or 填边界
    t.right(180)
```

```
# 碰上下墙 反弹! (碰到上墙或下墙, 有一个就弹)
if t.ycor() < _____ t.ycor() > _____:
    t.backward(15) # 填边界 填 and/or 填边界
    t.right(180)
```

```
turtle.done()
```

诊断一下:

小明写错了一个地方——他把碰撞检测里的运算符全写成了 `and`:

```
```python
```

```
if t.xcor() < -150 and t.xcor() > 150: # 错误!
```

```
```
```

你觉得小乌龟会发生什么? (圈出正确答案)

>

A. 正常反弹, and 和 or 效果一样

>

B. 小乌龟会跑出围栏! 因为 `and` 要求 x 同时小于 -150 又 大于 150——这不可能!

>

正确答案: _____ 应该改成什么运算符? _____

5. not 专项练习

用 not 改写下面的条件:

| 原始条件 | not 改写 |
|----------------|-----------------------|
| t.isdown() 笔落下 | not t.isdown() 笔_____ |
| x > 0 x 是正数 | not (x > 0) x 是_____ |
| t.xcor() < 200 | not (t.xcor() < 200) |

```
t.xcor() ____ 200
密码正确      密码 ____
灯是亮的      灯是 ____
```

在横线上填空：

```
# not 用法示例
turtle.penup()
# 如果笔是落下的（正在画），先把笔抬起来
if not t.isdown(): # 如果笔没有落下
    _____ # 落下笔
else:
    _____ # 抬起笔
```

6. 弹球游戏分析——四个独立 if

弹球游戏不能只用一句 or，因为需要知道碰了哪道墙来决定反弹方向。

请把下面四个 if 的条件补全：

```
# 弹球游戏——四道墙独立判断
t.speed(3)
t.shape("circle")
t.penup()
```

```
# 小乌龟的运动方向
angle = 45
t.setheading(angle)
```

```
# 边界范围：x从-200到200，y从-200到200
while True:
    t.forward(5)
```

```
# 左墙：x < -200 朝右弹
if t.xcor() < ____:
    t.setheading(0) # 朝右（0度）
```

```
# 右墙：x > 200 朝左弹
if t.xcor() > ____:
    t.setheading(180) # 朝左（180度）
```

```
# 下墙：y < -200 朝上弹
if t.ycor() < ____:
    t.setheading(90) # 朝上（90度）
```

```
# 上墙：y > 200 朝下弹
if t.ycor() > ____:
```

```
t.setheading(270)    # 朝下 (270度)
```

思考题：

- 为什么弹球游戏不能用 or 代替四个独立 if?
 - 如果用 or，我们知道小乌龟_____墙了，但不知道_____碰了哪道墙，也就不知道该往_____方向弹。
 - 填一填：and 适合_____场景，or 适合_____场景，独立 if 适合_____场景。
-

7. 短路求值专项 (and/or的"偷懒"原理)

核心规则：

- and 短路：遇到 False 就停 「False and ???」 后面不执行，结果肯定是 False
- or 短路：遇到 True 就停 「True or ???」 后面不执行，结果肯定是 True

第1题：判断是否会短路

```
False and print("执行了")  会不会执行print? ____
True or print("执行了")    会不会执行print? ____
True and print("执行了")   会不会执行print? ____
False or print("执行了")   会不会执行print? ____
```

第2题：看代码写结果

```
def say_yes():
    print("我说了YES! ")
    return True
```

```
def say_no():
    print("我说了NO! ")
    return False
```

```
# 情况1
result = False and say_yes()
# say_yes() 被执行了吗? ____
# 结果: ____
```

```
# 情况2
result = True or say_no()
# say_no() 被执行了吗? ____
# 结果: ____
```

记忆口诀: "and遇假停, or遇真停"

8. 上机挑战——彩色随机漫步

打开电脑，新建一个 Python 文件。让小乌龟在围栏里随机漫步，碰到墙就弹回来！

基础任务：（每完成一项在前面打 ☐ ）

- ☐ 画一个围栏（建议边界：x 从 -200 到 200，y 从 -200 到 200）
- ☐ 小乌龟随机走（上下左右四个方向，步长 10~20）
- ☐ 碰到任意一道墙就反弹（用 or 判断！）
- ☐ 至少走 100 步

加分挑战：（选做，挑战自己！）

- ☐ 碰到不同的墙变不同的颜色（左墙蓝、右墙红、上墙绿、下墙橙）
- ☐ 把步数改成 200，观察小乌龟会不会跑出去？
- ☐ 把围栏缩小到 -100~100，观察反弹是不是更频繁了？
- ☐ 改写弹球游戏——用四个独立 if，让小乌龟像台球一样弹来弹去

你可以从这里开始：

```
import turtle
import random
```

```
t = turtle.Turtle()
t.shape("turtle")
t.speed(6)
```

```
# 1. 画围栏（灰色方框）
```

```
# 2. 设置小乌龟颜色和位置
```

```
# 3. 随机漫步循环：走一步 判断是否碰墙 碰了就反弹
```

```
turtle.done()
```

提示：如果小乌龟跑出了围栏，检查你的反弹代码——有没有把 `or` 写成了 `and`？

如果是弹球游戏，检查有没有用四个独立 if？每个 if 管一道墙？

自我评价：

全都掌握了！ 大部分会了 还要再练练

今天最有趣的是： _____