# 流程控制语句

> 流程控制的三种基本结构：顺序结构、分支结构、循环结构。

# 一、分支语句

## 1. 简单分支：if 结构

> **适用场景**：当满足某个条件时执行特定代码，不满足则不执行（仅单分支逻辑）。

```
if 条件表达式:
    代码块  # 条件为 True 时执行，False 则跳过
```

- 条件表达式：结果为布尔值（`True` / `False`）的语句（如比较运算、逻辑运算等）。
- 代码块：必须缩进（4 个空格），代表属于该条件的执行内容。

> 案例 1：判断是否成年

```
age = int(input('请输入你的年龄'))
if age >= 18:
    print("你已成年，可独立购票")
# 输出：你已成年，可独立购票
```

> 案例 2：检查数值是否为正数

```
num = 5
if num > 0:
    print(f"{num} 是正数")
# 输出：5 是正数
```

> 练习 1：输入一个整数，若能被 5 整除，则打印 “该数是 5 的倍数”。

## 2. if-else分支：if...else 结构

> **适用场景**：满足条件执行 A，不满足则执行 B（非此即彼的逻辑）。

```
if 条件表达式:
    代码块1  # 条件为 True 时执行
else:
    代码块2  # 条件为 False 时执行
```

> 案例 1：判断奇偶数

```
num = int(input('请输入一个数字'))
if num % 2 == 0:
    print(f"{num} 是偶数")
else:
    print(f"{num} 是奇数")
# 输出：7 是奇数
```

> 案例 2：登录验证（简化版）

```
username = input('Enter your name: ')
password = input('Enter your password: ')
if username == 'admin' and password == '123456':
    print('登录成功')
else:
    print('登录失败')
```

> **练习 2**：输入两个数，比较大小并打印 “前者大” 或 “后者大”（假设两数不相等）。

### 补充：三目运算符

```
true_expression if condition else false_expression
```

- condition是判断条件，true_expression 和 false_expression 是两个表达式，用 if…else… 连接。
- 如果 condition （结果为真），就执行 true_expression，并把 true_expression 的结果作为整个表达式的结果。
- 如果 condition （结果为假），就执行 false_expression，并把 false_expression 的结果作为整个表达式的结果。

> 案例1：判断一个数是奇数还是偶数

```
num = int(input('请输入一个数字'))
result = '奇数' if num % 2 != 0 else '偶数'
print(result)
```

## 3.多重分支：if...elif...else 结构

> **适用场景**：存在多个条件，需依次判断，满足其中一个则执行对应代码（多选一逻辑）。

```
if 条件1:
    代码块1
elif 条件2:
    代码块2
elif 条件3:
    代码块3
...
else:
    代码块N  # 所有条件都不满足时执行（可选）
```

- 执行逻辑：从上到下依次判断条件，一旦某条件为 `True`，执行对应代码块后跳出整个分支结构。
- `elif` 可多个，`else` 可选（若省略，所有条件不满足时不执行任何操作）。

> 案例 1：成绩评级

```
score = float(input('请输入你的成绩'))
if score >= 90:
    print("等级：A")
elif score >= 80:
    print("等级：B")
elif score >= 60:
    print("等级：C")
else:
    print("等级：D（不及格）")
# 输出：等级：C
```

> 注意：条件需按逻辑顺序排列（如先判断高分段，再判断低分段）。

> 案例 2：判断月份季节

```
month = int(input('请输入月份'))
if month in [3,4,5]:
    print("春季")
elif month in [6,7,8]:
    print("夏季")
elif month in [9,10,11]:
    print("秋季")
elif month in [12,1,2]:
    print("冬季")
else:
    print("输入的月份无效")
# 输出：春季
```

> **练习 3**：根据输入的星期数（1-7），打印对应星期几（如 1→“星期一”，7→“星期日”，无效则提示错误）。

## 4. 多层条件分支：嵌套 if 结构

> **适用场景**：在一个条件判断的内部，还需要进一步判断（多层逻辑）。

```
if 外层条件:
    外层代码块
    if 内层条件:
        内层代码块1
    else:
        内层代码块2
else:
    外层else代码块
```

- 缩进决定层级：内层 `if` 需缩进在外部 `if` 的代码块中。
- 执行逻辑：先判断外层条件，若满足，再判断内层条件。

> 案例 1：判断学生是否能参加竞赛
>
> 条件：年龄≥10 且 成绩≥90 可参加。

```
age = int(input('请输入你的年龄'))
if age >= 10:
    score = float(input('请输入你的竞赛成绩'))
    if score >= 90:
        print('可以参加竞赛')
    else:
        print('成绩不合格,无法参加')
else:
    print('年龄不合格,无法参加')
```

> 案例 2：嵌套分支判断三角形类型

```
a = int(input('请输入第一条边'))
b = int(input('请输入第二条边'))
c = int(input('请输入第三条边'))

if a + b > c and a + c > b and b + c > a:
    print('可以构成三角形')
    if a == b == c:
        print('等边三角形')
    elif a == b or a == c:
        print('等腰三角形')
    else:
        print('其他三角形')
else:
    print('不能构成三角形')
```

> **练习 4**：某公园收费规则：1.2 米以下儿童免费；1.2-1.5 米儿童半价（50 元）；1.5 米以上成人全价（100 元）。若周末则成人票价上浮 20%。输入身高、是否周末，计算应付金额。

### 5. 常见错误与注意事项

1. **缩进错误**：忘记缩进或缩进不一致（建议用 4 个空格，避免混合使用空格和 Tab）。
2. **条件顺序错误**：多重分支中，范围大的条件放在前面会导致后面的条件失效（如先判断 `score >= 60` 再判断 `score >= 90`，会导致 90 分以上也被归为 60 分档）。
3. **嵌套层级混乱**：嵌套超过 3 层时逻辑易混乱，建议简化或拆分代码。
4. **遗漏冒号**：`if` / `elif` / `else` 语句末尾必须加冒号 `:`。

## 二、循环语句

### 1. for 循环

> **适用场景**：已知循环次数，或需要遍历序列（列表、字符串、元组等）中的元素。

```
for 变量 in 序列:
    循环体  # 依次取出序列中的元素赋值给变量，执行循环体
```

- 序列：可以是列表、字符串、`range()` 生成的数字序列等。
- 循环次数：由序列的长度决定（遍历完所有元素后自动结束）。

> 案例 1：遍历字符串

```
message = "Python"
for char in message:
    print(char)
# 输出：
# P
# y
# t
# h
# o
# n
```

> 案例 2：使用 `range()` 生成数字序列
>
> `range(start, stop, step)` 说明：
>
> - `start`：起始值（可选，默认 0）
> - `stop`：结束值（必选，不包含该值）
> - `step`：步长（可选，默认 1，可为负数）

```
# 生成 0-4 的数字（不包含5）
for i in range(5):
    print(i)  # 输出：0 1 2 3 4

# 生成 2-8 的偶数
for i in range(2, 9, 2):
    print(i)  # 输出：2 4 6 8
```

> 案例 3：计算列表元素总和

```
nums = [10, 20, 30, 40]
total = 0
for num in nums:
    total += num
print(f"列表总和：{total}")  # 输出：100
```

> 练习 1：用 `for` 循环遍历列表 `["苹果", "香蕉", "哈密瓜","水晶葡萄"]`，并打印每个元素的长度（提示：用 `len()` 函数）。

## 2. while 循环

> **适用场景**：未知循环次数，但已知循环终止条件（满足条件时继续循环）。

```
while 条件表达式：
    循环体  # 条件为 True 时执行，直到条件为 False 时终止
```

- 注意：循环体中必须包含改变条件的语句，否则会导致**死循环**（无限执行）。

> 案例 1：累加 1-10 的和

```
total = 0
i = 1  # 初始值
while i <= 10:  # 条件：i 不超过10
    total += i
    i += 1  # 改变条件（i 自增）
print(f"1-10的和：{total}")  # 输出：55
```

> 案例 2：猜数字游戏（基础版）

```
secret = random.randint(1, 10)  # 1-10的随机数
guess = 0  # 初始猜测值

while guess != secret:
    guess = int(input("猜一个1-10的数字："))
    if guess > secret:
        print("太大了！")
    elif guess < secret:
        print("太小了！")
print("恭喜猜对了！")
```

> **练习 2**：用 `while` 循环打印 10-1 的数字（从 10 递减到 1）。

## 3. 循环控制关键字

> 用于在循环中改变默认执行流程。

### 3.1 break 关键字

> 作用：立即终止当前循环，跳出循环体，执行循环外的后续代码。

> 案例 1：查找列表中是否有大于10的数字，找到就结束

```
for number in [1, 5, 12, 18, 22]:
    if number > 10:
        print(f"Found: {number}")
        break  # 一旦找到第一个大于10的数字，就跳出循环
```

### 3.2 continue 关键字

> 作用：跳过本次循环的剩余代码，直接进入下一次循环。

> 案例 2：输出列表中所有的偶数

```
for num in [2,5,7,9,12,13,28]:
    if num % 2 != 0:
        continue  # 跳过奇数，进入下一次循环
    print(f"偶数：{num}")
```

> 案例 3： `break` 与 `continue` 对比

```
# 使用 break
print("使用 break：")
for i in range(10):
    if i == 3:
        break
    print(i)  # 输出：0 1 2

# 使用 continue
print("使用 continue：")
for i in range(10):
    if i == 3:
        continue
    print(i)  # 输出：0 1 2 4 5 6 7 8 9
```

> **练习 3**：将输入的内容，进行输出，如果输入 `exit` ，结束聊天。如果输入敏感词汇，不输出显示

## 4. 循环嵌套

> 在一个循环内部包含另一个循环，用于处理多维或多层重复逻辑。

> 案例 9：打印 5 行 5 列的星号方阵

```
for i in range(5):  # 外层循环控制行数
    for j in range(5):  # 内层循环控制每行的星号数
        print("*", end=" ")
    print()  # 换行
# 输出：
# * * * * *
# * * * * *
# * * * * *
# * * * * *
# * * * * *
```

> 案例 10：打印乘法表（九九表）

```
for i in range(1, 10):  # 行数（1-9）
    for j in range(1, i+1):  # 列数（每行打印 i 个）
        print(f"{j}×{i}={i*j}", end="\t")
    print()  # 换行
```

> **练习4**：今有鸡兔同笼，上有35个头，下有94足，问鸡兔各几只？

## 5、常见错误与注意事项

1. **死循环**： `while` 循环中忘记修改条件（如 `i` 未自增），导致循环无限执行（可按 `Ctrl+C` 强制终止）。
2. **`range()` 参数顺序错误**： `range(stop, start)` 会导致无输出（如 `range(5, 1)` 因起始值大于结束值且步长为正，生成空序列）。

3. **`break` 与 `continue` 混淆**：`break` 是终止整个循环，`continue` 是跳过本次循环，需根据逻辑选择。
4. **循环嵌套层级过多**：超过 3 层的嵌套会降低代码可读性，建议拆分逻辑。