Python基础2：选择和循环语句

一、选择

1.【了解】选择应用场景介绍

2.【理解】 比较和逻辑运算符

2.1 【理解】比较运算符

- 比较2个数据的结果，判断正确，返回True, 否则，返回False

```
"""
比较运算符：==、>、<、!=, >=, <=
"""
a = 3
b = 3
print(a == b)  # True
print(a != b)  # False

a = 20
b = 10
print(a > b)  # True
print(a < b)  # False

print(a >= b)  # True
print(a <= b)  # False
```

2.2 【理解】逻辑运算符

```
"""
and：与，并且，  左右2边都为True，结果才为True
or：  或，或者，  只有有1个为True，结果为True
not：非，取反，  原来是True就变成False
"""
```

```python
# and: 并且， 左右2边都为True，结果才为True
print(True and True)  # True
print(5 > 3 and 4 > 3)  # True
print(5 > 3 and 4 < 3)  # False
print(True and False)  # False

# or: 或者， 只有有1个为True，结果为True
print(True or False)  # True
print(True or True)  # True
print(False or True)  # True
print(False or False)  # False

print(1 == 1 or 5 < 3)  # True

# not: 非，取反，原来是True就变成False
print(True)  # True
print(not True)  # False
print(not (4 > 2))  # False
```

3.【重点】if语句的基本使用

3.1 【重点】if语句的使用

```
'''
if 条件判断:
    条件满足，执行if代码块..
'''
# 1. 定义一个变量age，保存年龄
# 2. if 判断条件(判断是否满 18 岁):
# 3. 满足条件，打印一句话：允许进网吧嗨皮


print("if之前")

# 1. 定义一个变量age，保存年龄
age = 19
# 2. if 判断条件(判断是否满 18 岁):
if age >= 18:
    # 3. 满足条件，打印一句话：允许进网吧嗨皮
    print("允许进网吧嗨皮")

# 其他代码是无条件执行
print("if之后")
```

3.2【重点】if-else语句的使用

```
'''
if 条件判断:
    条件满足，执行if的代码块
else:
    条件不满足，执行else的代码块
'''

"""需求:
# 1. 输入用户年龄，input返回值是字符串，类型转换为int
# 2. if 判断是否满 18 岁:
    # 2.1 如果满 18 岁，打印一句话：允许进网吧嗨皮
# 3. 否则，未满 18 岁
    # 3.1 打印一句话：提示回家写作业
"""


# 1. 输入用户年龄，input返回值是字符串，类型转换为int
# age = input("请输入您的年龄: ")
# age = int(age)
```

```
age = int(input("请输入您的年龄： "))

# 2. if 判断是否满 18 岁：
if age >= 18:
    # 2.1 如果满 18 岁，打印一句话：允许进网吧嗨皮
    print("允许进网吧嗨皮")
# 3. 否则，未满 18 岁
else:
    # 3.1 打印一句话：提示回家写作业
    print("回家写作业")
```

3.3【记忆】通过if实现的三目运算符

- 简洁版的if-else
- `a if a > b else b`: 如果 a > b为True, 结果为a, 否则， 为b

```
"""
if 实现三目运算操作，就是简洁版的if else

需求：通过if-else求2个数的最大值
# 1. 定义2个变量，赋值
# 2. if 用大于比较2个变量大小：
```

```
    #   2.1 如果满足条件，把大的那个数保存
# 3. 否则，不满足条件：
    #   3.1 把小的那个数保存
# 4. if 语句的外面，大于最大值
"""

# 需求：通过if-else求2个数的最大值
# 1. 定义2个变量，赋值
a = 10
b = 20

# 2. if 用大于比较2个变量大小：
if a > b:
    #   2.1 如果满足条件，把大的那个数保存
    _max = a
# 3. 否则，不满足条件：
else:
    #   3.1 把另一个大的那个数保存
    _max = b
# 4. if 语句的外面，打印最大值
print(_max)


# 重新给a，b赋值
a = 100
b = 200
```

```
# 通过 三目运算符实现 同样效果
# 如果 a > b的条件成立，三目运算的结果是a，否则就是b
_max = a if a > b else b

print(_max)
```

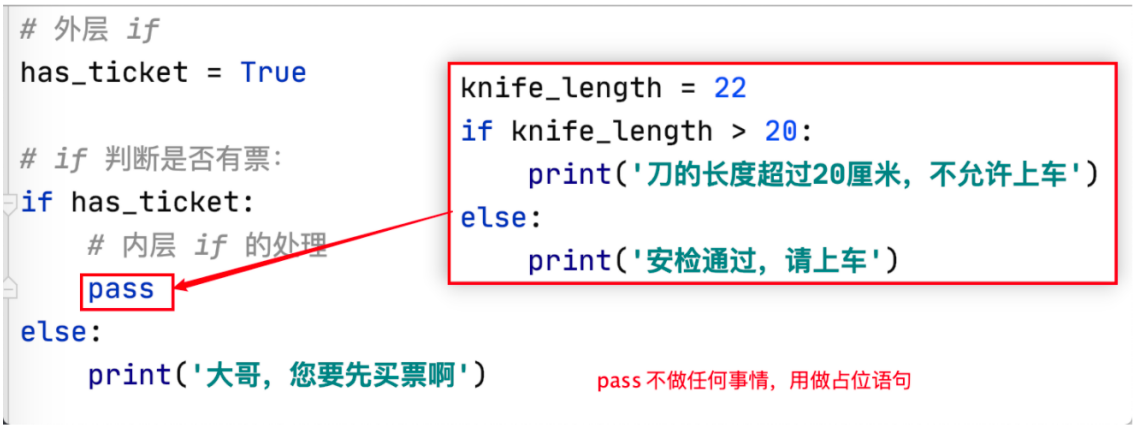
3.4【重点】if-elif-else语句的使用

```
'''
if 条件1判断:
    条件1满足，执行条件1的if代码块
elif 条件2判断:
    条件2满足，执行条件2的elif代码块
elif 条件3判断:
    条件3满足，执行条件3的elif代码块
...
else:
    以上条件都不满足，执行else代码块

特点:
    条件从上到下判断
    如果满足了其中一个条件，执行满足条件的代码块，解决整个if语句，不再判断下面的其他条件。
'''
```

```python
# 1. 定义 holiday_name 字符串变量记录节日名称
holiday_name = "情人节"

# 2. if 判断是否为情人节:
#    是两个=，==，判断是否相等
if holiday_name == "情人节":
    # 2.1 如果是 情人节，打印:  买玫瑰 / 看电影
    print("买玫瑰 / 看电影")
# 3. elif 判断是否为平安夜:
elif holiday_name == "平安夜":
    # 3.1 如果是 平安夜，打印:  买苹果 / 吃大餐
    print("买苹果 / 吃大餐")
# 4. elif 判断是否为生日 :
elif holiday_name == "生日":
    # 4.1 如果是 生日，打印:  买蛋糕
    print("买蛋糕")
# 5. else 其它情况:
else:
    # 5.1 打印: 每天都是节日啊……
    print("每天都是节日啊……")
```

4.【难点】if的嵌套

- 方法：先写外层if, 再写内层if




```
# 外层 if
has_ticket = True

# if 判断是否有票:
if has_ticket:
    # 内层 if 的处理
    pass
else:
    print('大哥，您要先买票啊')
```

```
knife_length = 22
if knife_length > 20:
    print('刀的长度超过20厘米，不允许上车')
else:
    print('安检通过，请上车')
```

```
"""
步骤流程：先写外层的if，再写内层的if

外层 if 步骤流程:
# 1. 定义布尔型变量 has_ticket，赋值为True表示有票
# 2. if 判断是否有票:
    # 2.1 如果有票，打印：有车票，即将进行下一步的刀具安检
    # 2.2 内层if的处理(先空着，不处理)
# 3. 否则，没有票
    # 3.1 打印：大哥，您要先买票啊

内层 if 步骤流程:
# a. 定义整型变量 knife_length 表示刀的长度，单位：厘米
```

```
# b. if 判断刀具长度是否超过20:
    # b.1 如果超过 20 厘米，打印：刀的长度超过20厘米，不允许上车
# c. 否则，不超过20厘米:
    # c.1 打印：安检通过，请上车
"""

# 外层 if 步骤流程:
# 1. 定义布尔型变量 has_ticket，赋值为True表示有票
has_ticket = True
# 2. if 判断是否有票:
# if has_ticket == True:
if has_ticket:
    # 2.1 如果有票，打印：有车票，即将进行下一步的刀具安检
    print("有车票，即将进行下一步的刀具安检")
    # 2.2 内层if的处理(先空着，不处理)
    # 内层 if 步骤流程:
    # a. 定义整型变量 knife_length 表示刀的长度，单位：厘米
    knife_length = 300
    # b. if 判断刀具长度是否超过20:
    if knife_length > 20:
        # b.1 如果超过 20 厘米，打印：刀的长度超过20厘米，不允许上车
```

```
            print("刀的长度超过20厘米，不允许上车")
        # c. 否则，不超过20厘米：
        else:
            # c.1 打印：安检通过，请上车
            print("安检通过，请上车")

# 3. 否则，没有票
else:
    # 3.1 打印：大哥，您要先买票啊
    print("大哥，您要先买票啊")
```

5. 【应用】猜拳游戏

5.1【应用】基础代码实现

```
"""
规则：石头 1，  剪刀 2，  布 3
角色：
    电脑 computer
    用户 user
用户赢的情况：
    user == 1 and computer == 2
    user == 2 and computer == 3
    user == 3 and computer == 1
```

```
步骤流程：
# 1. 用户输入数字：请输入（石头 1， 剪刀 2， 布 3）
# 2. 电脑固定一个数字，假定只会出石头
# 3. if 用户赢电脑的判断：
    # 3.1 如果用户赢，打印：电脑弱爆了
# 4. elif 平局：
    # 4.1 打印：心有灵犀，再来一盘！
# 5. 否则，电脑赢
    # 5.1 打印：不行，我要和你决战到天亮！
"""


# 1. 用户输入数字：请输入（石头 1， 剪刀 2， 布 3）
user = int(input("请输入（石头 1， 剪刀 2， 布 3）：  "))
# 2. 电脑固定一个数字，假定只会出石头
computer = 1

# 3. if 用户赢电脑的判断：
if (user == 1 and computer == 2) or (user == 2 and computer == 3)\
        or (user == 3 and computer == 1):
    # 3.1 如果用户赢，打印：电脑弱爆了
```

```
    print("电脑弱爆了")
# 4. elif 平局:
elif user == computer:
    # 4.1 打印: 心有灵犀，再来一盘!
    print("心有灵犀，再来一盘！")
# 5. 否则，电脑赢
else:
    # 5.1 打印: 不行，我要和你决战到天亮!
    print("不行，我要和你决战到天亮！")
```

5.2【记忆】随机数的处理

```
"""
功能: 产生某个范围的随机数

步骤流程:
# 1. 导入模块，工具包，后面会学习
# 2. 调用工具里面的函数，返回一个随机值
"""
# 1. 导入模块，工具包，后面会学习
import random

# 2. 调用工具里面的函数，返回一个随机值
```

```
num = random.randint(1, 3)  # 产出1, 2, 3中随机的一个数
print(num)
```

5.3【应用】完整代码

```
"""
规则: 石头1 剪刀2 布3
角色: user用户, computer电脑
用户赢的情况:
    user 1 -> computer 2
    user 2 -> computer 3
    user 3 -> computer 1


步骤流程:
# 1. 用户输入数字: 请输入(石头 1, 剪刀 2, 布 3)
# 2. 电脑 随机 出拳, 即随机产生一个[1,3]范围的数
# 3. if 用户赢电脑的判断:
    # 3.1 如果用户赢, 打印: 电脑弱爆了
# 4. elif 平局:
    # 4.1 打印: 心有灵犀, 再来一盘!
# 5. 否则, 电脑赢
```

```
    # 5.1 打印：不行，我要和你决战到天亮！
"""
import random


# 1. 用户输入数字：请输入（石头 1， 剪刀 2， 布 3）
# user_str = input("请输入（石头 1， 剪刀 2， 布 3）：")
# user = int(user_str)
user = int(input("请输入（石头 1， 剪刀 2， 布 3）："))

# 2. 电脑 随机 出拳，即随机产生一个[1,3]范围的数
computer = random.randint(1, 3)

# 3. if 用户赢电脑的判断：
# user 1 -> computer 2
# user 2 -> computer 3
# user 3 -> computer 1
if (user == 1 and computer == 2) or (user == 2 and computer == 3) or (user == 3 and computer == 1):
    # 3.1 如果用户赢，打印：电脑弱爆了
    print("电脑弱爆了")
# 4. elif 平局：
```

```
elif user == computer:
    # 4.1 打印：心有灵犀，再来一盘！
    print("心有灵犀，再来一盘！")
# 5. 否则，电脑赢
else:
    # 5.1 打印：不行，我要和你决战到天亮！
    print("不行，我要和你决战到天亮！")
```

二、循环

1.【了解】循环应用场景介绍

- 作用：让 **指定的代码** 重复执行

2.【重点】while语句的基本使用

2.1【重点】while语句的使用

```
10
11  # 1. 定义一个条件变量，一般赋值为0
12  i = 0
13
14  # 2. while 判断条件：
15  while i < 10:
16      # 2.1 满足条件的代码块
17      print('跑 %d 圈' % (i+1))
18      # 2.2 条件变量的改变【非常重要】
19      i += 1
```

1）i=0，判断i<10条件为True，进入循环体
print()
i+=1 ，i变成1
2）i=1，判断i<10条件为True，进入循环体
print()
i+=1 ，i变成2
重复操作，直到i为10，不满足循环条件，跳出循环

```
'''
定义条件变量，赋予初始值

while 判断条件(使用条件变量判断):
    满足条件执行的代码块
    ...
    条件变量的改变
'''

# 1. 定义一个条件变量，一般赋值为0
num = 0
# 2. while 判断条件:
while num < 10:
    # 2.1 满足条件的代码块
    print("跑了 %d 圈" % (num+1))
    # 2.2 条件变量的改变【非常重要】
    # num = num + 1
    num += 1
```

2.2【知道】死循环

```
# 条件永远为真的(True)循环即为死循环，也叫无限循环
while True:
    print("根本停不下来...")
```

2.3【应用】计算1~100之间的累加和

```
# 1. 定义条件变量 i，赋值为1
i = 1
# 2. 定义一个辅助变量，用于保存累计的结果
_sum = 0

# 3. while 条件(i <= 100):
while i <= 100:
    # 3.1 完成累加
    _sum = _sum + i  # _sum += i
    # 3.2 条件变量改变
    i += 1

# 4. 在循环的外面，打印最终的累加结果
print(_sum)
```

1）i=1, _sum=0　判断i<=100条件为True, 进入循环体
　　_sum=0+1
　　i+=1, i变成2
2）i=2, _sum=1　判断i<=100条件为True, 进入循环体
　　_sum=_sum+i ====> _sum=1+2
　　i+=1, i变成3
3）i=3, _sum=3　判断i<=100条件为True, 进入循环体
　　_sum=_sum+i ====> _sum=3+3
　　i+=1, i变成4
重复操作，直到为101，不满足条件，跳出循环

```
"""
需求：计算1~100之间的累加和，1 + 2 + 3 ..... 100

核心代码分析：
    _sum = 0

    i = 1
    _sum = _sum + i      --> 0 + 1
```

```
    i = 2
    _sum = _sum + i      --> 0 + 1 + 2

    i = 3
    _sum = _sum + i      --> 0 + 1 + 2 + 3
    ....

    i = 100
    _sum = _sum + i      --> 0 + 1 + ... +
99 + 100

步骤流程:
# 1. 定义条件变量 i, 赋值为1
# 2. 定义一个辅助变量, 用于保存累计的结果
# 3. while 条件(i <= 100):
    # 3.1 完成累加
    # 3.2 条件变量改变
# 4. 在循环的外面, 打印最终的累加结果
"""

# 1. 定义条件变量 i, 赋值为1
i = 1
# 2. 定义一个辅助变量, 用于保存累计的结果
_sum = 0
# 3. while 条件(i <= 100):
while i <= 100:
```

```
    # 3.1 完成累加
    # print(i)
    _sum += i
    # 3.2 条件变量改变
    i += 1
# 4. 在循环的外面，打印最终的累加结果
print("_sum: %d " % _sum)
```

2.4【应用】计算1~100之间偶数的累加和

```
# 1. 定义条件变量 i，赋值为1
i = 1
# 2. 定义一个辅助变量，用于保存累计的结果
_sum = 0

# 3. while 条件(i <= 100)：
while i <= 100:
    # 3.1 if 判断i是否为偶数
    if i % 2 == 0:
        # 3.3 完成累加
        _sum = _sum + i  # _sum += i

    # 3.3 条件变量改变
    i += 1

# 4. 在循环的外面，打印最终的累加结果
print(_sum)
```

1）i = 1, _sum = 0, 判断 i <= 100 条件为True，进入循环体
判断 i % 2 == 0 条件为False，不会进入if代码块
i += 1, i变成了2
2）i = 2, _sum = 0, 判断 i <= 100 条件为True，进入循环体
判断 i % 2 == 0 条件为True，会进入if代码块
_sum = sum+i ==> _sum = 0+2 = 2
i += 1, i变成了3
3）i = 3, _sum = 2, 判断 i <= 100 条件为True，进入循环体
判断 i % 2 == 0 条件为False，不会进入if代码块
i += 1, i变成了4

重复操作，直到i为101

```
"""
需求：计算1~100之间偶数的累计和

判断一个变量是否为偶数：对2求余，余数为0即为偶数
i = 4
```

```
if i % 2 == 0:
    print("偶数")

步骤流程:
# 1. 定义条件变量
# 2. 设置辅助变量
# 3. while 条件:
    # 3.1 if 判断i是否为偶数
        # 3.1.1 累加
    # 3.2 条件变量的改变
# 4. 循环外面，打印累加结果
"""


# 1. 定义条件变量
i = 1
# 2. 设置辅助变量
_sum = 0
# 3. while 条件:
while i <= 100:
    # 3.1 if 判断i是否为偶数
    if i % 2 == 0:
        # 3.1.1 累加
        _sum += i
    # 3.2 条件变量的改变
    i += 1
```

```
# 4. 循环外面，打印累加结果
print(_sum)
```

2.5【操作】通过debug调试代码

1. 设置断点
2. 调试运行
3. 单步执行代码
4. 观察变量变化
    - 重复步骤3、4，观察变量和程序执行流程
    - 重复步骤3、4，观察变量和程序执行流程
    - 重复步骤3、4，观察变量和程序执行流程

流程图片参考讲义

3.【难点】while循环嵌套

3.1【理解】while循环嵌套

```python
# 外循环
i = 0
while i < 5:
    print('跑步第 %d 圈' % (i + 1))

    # 内循环
    pass

    i += 1
```

```python
j = 0
while j < 10:
    print('第%d俯卧撑' % (j + 1))
    j += 1
```

```
"""
while循环嵌套：while里面包含while
步骤流程：先写外层的while，再写内层的while
需求：跑步 5 圈，每跑步1圈，做10个俯卧撑

外层 while 步骤流程：
# 1. 设置外层循环的条件变量 i， 赋值为0
# 2. 外层while 条件(i < 5)：
    # 3. 打印跑了第几圈
    # 4. 做10个俯卧撑内循环处理(先空着，不处理)
    # 5. 外层循环的条件变量改变

内层 while 步骤流程：
# 4.1 定义内层循环条件变量 j = 0
# 4.2 内层循环 while j < 10：
    # 4.3 打印第几个俯卧撑
    # 4.4 内层循环的条件变量改变
"""
```

```
# 1. 设置外层循环的条件变量 i,  赋值为0
i = 0
# 2. 外层while 条件(i < 5):
while i < 5:
    # 3. 打印跑了第几圈
    print("跑了 %d 圈" % (i+1))
    # 4. 做10个俯卧撑内循环处理(先空着, 不处理)
    # 4.1 定义内层循环条件变量 j = 0
    j = 0
    # 4.2 内层循环 while j < 10:
    while j < 10:
        # 4.3 打印第几个俯卧撑
        print("第%d个俯卧撑" % (j + 1))
        # 4.4 内层循环的条件变量改变
        j += 1
    # 5. 外层循环的条件变量改变
    # i = i + 1
    i += 1
```

3.2【应用】通过循环打印1行星星

```
"""
要求: 每一次只能打印 1 个 *
```

```
print()默认是换行的
print('* ', end=''), end=''让打印不换行
"""
print("* ", end='')
print("* ", end='')
print("* ", end='')
print("* ", end='')
print("* ", end='')

print()
print('===============华丽分割线
=============')

"""
要求：通过循环实现打印1行多个星星
步骤流程：
# 1 定义循环条件变量 j = 0
# 2 循环 while j < 5:
    # 3 打印一个星星，end=''作用指定不换行打印
    # 4 循环的条件变量改变
"""
# 1 定义循环条件变量 j = 0
j = 0
# 2 循环 while j < 5:
while j < 5:
    # 3 打印一个星星，end=''作用指定不换行打印
```

```
    print("* ", end='')
    # 4 循环的条件变量改变
    j += 1
```

3.3【应用】通过循环嵌套打印正方形

```
"""
外层 while 步骤流程:
# 1. 定义外层循环的条件变量 i = 0
# 2. 外层while 条件(i < 5):
    # 3. 内循环处理(先空着, 不处理)
    # 4. 单独打印空行
    # 5. 外层循环的条件变量改变

内层 while 步骤流程:
# 3.1 定义内层循环条件变量 j = 0
# 3.2 内层循环 while j < 5:
    # 3.3 打印一个星星, end=''作用指定不换行打印
    # 3.4 内层循环的条件变量改变
"""

# 1. 定义外层循环的条件变量 i = 0
i = 0
# 2. 外层while 条件(i < 5):
```

```
while i < 5:
    # 3. 内循环处理(先空着，不处理)
    # 3.1 定义内层循环条件变量 j = 0
    j = 0
    # 3.2 内层循环 while j < 5:
    while j < 5:
        # 3.3 打印一个星星，end=''作用指定不换行打印
        print("* ", end='')
        # 3.4 内层循环的条件变量改变
        j += 1
    # 4. 单独打印空行
    print()
    # 5. 外层循环的条件变量改变
    i += 1
```

3.4【应用】通过循环嵌套打印三角形

```
"""
外层 while 步骤流程:
# 1. 定义外层循环的条件变量 i = 0
# 2. 外层while 条件(i < 5):
    # 3. 内循环处理(先空着，不处理)
    # 4. 单独打印空行
```

```
    # 5. 外层循环的条件变量改变

内层 while 步骤流程：第1次打印1个星，第2次打印2个星，如此内推
# 3.1 定义内层循环条件变量 j = 0
# 3.2 内层循环 while j < i+1:
    # 3.3 打印一个星星，end=''作用指定不换号打印
    # 3.4 内层循环的条件变量改变
"""

# 1. 定义外层循环的条件变量 i = 0
i = 0
# 2. 外层while 条件(i < 5):
while i < 5:
    # 3. 内循环处理(先空着，不处理)
    # 3.1 定义内层循环条件变量 j = 0
    j = 0
    # 3.2 内层循环 while j < i+1:
    while j < i + 1:
        # 3.3 打印一个星星，end=''作用指定不换号打印
        print("* ", end='')
        # 3.4 内层循环的条件变量改变
        j += 1

    # 4. 单独打印空行
```

```
    print()
    # 5. 外层循环的条件变量改变
    i += 1
```

4. break 、 continue和循环的else

4.1【重点】break的作用

- 跳出循环

```
# break:  跳出循环，不再执行循环体中后续重复的代码

"""
需求：跑步10圈，跑5圈后，不再跑了

步骤流程：
# 1. 定义一个条件变量，一般赋值为0
# 2. while 判断条件：
    # 3. 满足条件的循环代码块
    # 4. if 满足条件：
        # 4.1 打印提示信息：累了，结束战斗
        # 4.2 break跳出循环
    # 5. 条件变量的改变【非常重要】
"""
```

```python
# 1. 定义一个条件变量，一般赋值为0
i = 0
# 2. while 判断条件:
while i < 10:
    # 3. 满足条件的循环代码块
    print("跑了 %d 圈" % (i + 1))
    # 4. if 满足条件:
    if i == 4:
        # 4.1 打印提示信息：累了，结束战斗
        print("累了，结束战斗")
        # 4.2 break跳出循环
        break
    # 5. 条件变量的改变【非常重要】
    i += 1

print("循环体外部")
```

4.2【重点】continue的作用

- **跳过**本次循环，进入下次循环继续
- 很多时候，`continue`前面修改条件变量，不然，很容易导致死循环

```
# continue:    跳过循环，不再执行本次循环体中后续重复的代码，但进入下一次循环判断

"""
需求：跑步10圈，到第5圈休息一下，第6圈继续

步骤流程：
# 1. 定义一个条件变量，一般赋值为0
# 2. while 判断条件：
    # 3. if 满足条件：
        # 3.1 打印：累了，休息1圈
        # 3.2 条件变量的改变【非常重要】，如果没有，导致死循环
        # 3.3 continue跳过本次循环

    # 4. 打印跑步第几圈
    # 5. 条件变量的改变【非常重要】
"""

# 1. 定义一个条件变量，一般赋值为0
i = 0
# 2. while 判断条件：
while i < 10:
    # 3. if 满足条件：
    if i == 4:
        # 3.1 打印：累了，休息1圈
```

```
            print("累了，休息1圈")
            # 3.2 条件变量的改变【非常重要】，如果没有，导致死循环
            i += 1
            # 3.3 continue跳过本次循环
            continue

        # 4. 打印跑步第几圈
        print("跑了%d圈" % (i+1))
        # 5. 条件变量的改变【非常重要】
        i += 1
```

4.3【知道】循环的else执行流程

- if...else：
  - 不满足if条件的前提下，执行else操作
  - 两者只会执行一个分支
- 循环 else：
  - 循环里面**没有遇到break语句**，while执行完后，则会执行else的分支
  - 也就是说，没有遇到break，不仅会执行while的语句，也会执行else的语句

```
# 循环里面没有遇到break语句，while执行完后，则会执行else的分支

i = 0
while i < 5:
    print("跑了 %d 圈" % (i+1))
    # if i == 1:
    #     break
    i += 1
else:
    print("while循环中没有break,while结束后执行else代码")
```