项目 3　循环结构

项目目标

- 熟练掌握循环结构
- 熟练掌握 while 循环结构和 for 循环的使用场景及相互转换
- 掌握循环结构中 continue 和 break 的区别
- 掌握循环的嵌套结构

项目目标

- **知识目标**

本项目将学习循环的概念，及 while、while-else、for、for-else 循环结构特点及执行流程。循环流程改变语句 continue 和 break 的区别，以及在单重循环和循环嵌套中的作用。

- **技能目标**

能够使用循环结构解决实际问题。

- **素养目标**

学习、生活和工作往往是简单的事情重复多次。

项目描述

本项目首先简要介绍两种基本的循环语句，接着介绍循环流程跳转的两种语句 break 和 continue 及其区别，最后重点介绍循环的嵌套结构。

任务列表

任 务 名 称	任 务 描 述
任务 1 抓娃娃游戏——while 循环	while 循环结构
任务 2 猜数字谜游戏——循环流程控制	循环流程控制语句 break、continue；while-else 循环结构
任务 3 模拟发红包游戏——for 循环、列表	for 循环结构、range、random

把重复执行一组“相同”或”相似”操作的流程结构称为循环，该组“相同”或”相似”的操作语句集合被称为循环体。Python 语言主要支持 while、for 等两种形式的循环结构。

3.1　任务 1 抓娃娃游戏——while 循环

任务目标

➢ 掌握 while 循环结构

➢ 能够使用循环结构解决实际问题

【任务描述】抓娃娃游戏玩一局需要 3 元钱，根据充值卡中的金额判断还能玩几局。

【任务分析】当（while）充值卡中的余额（balance）大于等于 3 元时，游戏一直进行（重复），同时每次支付 3 元，即 balance-=3，直到 balance 小于 3 元时，游戏终止。

【任务类型】本任务属于当满足一定条件时就一直执行某操作的情景，故可使用循环结构实现。

3.1.1 知识点：while 循环结构

1. 语法格式

```
while Exp_cntrl:
    语句组 A
```

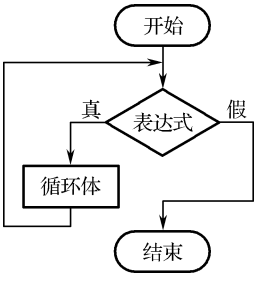

图 3-1 while 循环流程图

2. 执行流程

首先判断循环控制表达式 Exp_cntrl 的值，当该表达式的值为逻辑真（True）时，会一直执行循环体（语句组 A），直到表达式的值为逻辑假（False）时，结束循环体执行。while 循环流程图，如图 3-1 所示。

通常把循环控制表达式 Exp_cntrl 中含有的变量，称为循环控制变量。为了避免程序陷入死循环，必须要有能改变循环控制变量的操作，使循环控制表达式 Exp_cntrl 的值为逻辑假，以终止循环。

【示例】计算 100 以内所有奇数的和。

【分析】题目要求是求和，故定义求和变量为 s，并初始化为 0。遍历 100 以内所有的奇数，并累加到求和变量 s 中，设当前奇数为 i，并初始化为第一个奇数 1，则 i 为循环控制变量。当 i 小于等于 100 时，程序一直进行两个操作（循环体）：①把当前的奇数 i 累加到求和变量 s 上；②把 i 更新为下一个奇数，即循环控制变量的增量。

【参考代码】

```
s = 0                    #求和变量 s，初始化为 0
i = 1                    #循环控制变量 i 的初始化
while i <= 100 :         #循环控制表达式的判断
    s += i
    i += 2               #循环控制变量的增量
print("sum = %d" % s)    #输出求和结果
```

【运行结果】

```
sum = 2500
```

3.1.2 任务实施

【分析】

（1）定义变量 balance 表示充值卡余额，其金额从键盘输入，cost 表示每局所需费用，默认为 3 元，该费用后续可调整，n 用来统计总共玩了几局，初始值为 0。

```
balance = eval(input('充值卡余额（元）：'))
cost = 3
n = 0
```

（2）当余额 balance 够玩一局的花费 cost，即 balance >= cost 时，游戏一直进行，即可采用 while 循环结构，每次游戏均执行三项操作（循环体）：游戏次数加 1，即 n += 1，从充值卡中扣除一次游戏的费用 cost，即 balance -= cost，打印每局提示及好运信息。

（3）当余额 balance 不足一次游戏费用 cost 时，整个游戏结束，打印游戏次数及卡上当前余额等信息。

【参考代码】

```
balance = eval(input('充值卡余额（元）：'))     #输入当前余额
cost = 3                                        #每局游戏费用，可调整
n = 0                                           #统计共玩的局数
while balance >= cost :                         #只要余额大于等于 cost，一直进行
    n += 1                                      #累加次数
    balance -= cost                             #从余额中扣除费用
    print('第{}局游戏开始，祝您好运'.format(n))
#循环结束
print('-------------------------')
print('您共玩了{}局'.format(n))
print('充值卡余额{}元，余额不足！'.format(balance))
```

【运行结果】

```
充值卡余额（元）：23
第 1 局游戏开始，祝您好运
第 2 局游戏开始，祝您好运
第 3 局游戏开始，祝您好运
第 4 局游戏开始，祝您好运
第 5 局游戏开始，祝您好运
第 6 局游戏开始，祝您好运
第 7 局游戏开始，祝您好运
-------------------------
您共玩了 7 局
充值卡余额 2 元，余额不足！
```

3.1.3　巩固案例

【案例 1】输入任意一个十进制正整数，将其“反序”后输出（若输入：1234，则输出：4321）。

【分析】经分析本题主要涉及两个操作：一是把原数从最低位到最高位逐位分离，如 4、3、2、1；二是按照分离出的顺序，用分离出的数字组成新的十进制整数 4321。

1．把 n=1234 按从低到高位逐位分离

方法和步骤如下。

（1）分离个位数字：4。用 n（1234）除以 10 取余，即 n % 10，得个位数 4，然后再用 n（1234）除以 10 取整，即 n = 1234 // 10 = 123。

（2）分离十位数字：3。用 n（123）除以 10 取余，即 n % 10，得个位数 3，然后再用 n（123）除以 10 取整，即 n = 123 // 10 = 12。

（3）分离百位数字：2。用 n（12）除以 10 取余，即 n % 10，得个位数 2，然后再用 n（12）除以 10 取整，即 n = 12 // 10 = 1。

（4）分离千位数字：1。用 n（1）除以 10 取余，即 n % 10，得个位数 1，然后再用 n（1）除以 10 取整，即 n = 1 // 10 = 0，即当 n=0 时，原数的各位数字均已分离出来。

分析可知，上述各位数字的分离过程是重复执行 t = n % 10 和 n = n // 10 两条语句，直到 n = 0 为止。故可采用 while 循环结构，代码框架如下：

```
while n != 0 :
    t = n % 10  #t:当前 n 分离出的低位数字
    …
    n //= 10    #去除已分离出的当前低位后的数值，为下一次分离次低位做好准备
```

2．把各位数字按分离出的顺序组成一个新的十进制整数

先分离出的位（原数据的低位）作为新十进制数的高位，后分离出的位（原数据的高位）作为新十进制数的低位。

若用逐位分离出来的 4、3、2、1 组成一个新的十进制整数 m，则组建 m 的步骤如下。

（1）设新组成的十进制数 m 初始化为 0。

（2）m（0）扩大 10 倍，即原 m 的各位均向左（高位）移动 1 位，再加上刚分离出的数位 4，即 m = m * 10 + 4 = 0 * 10 + 4 = 0 + 4 = 4。

（3）m（4）扩大 10 倍，即原 m 的各位均向左（高位）移动 1 位，再加上刚分离出的数位 3，即 m = m * 10 + 3 = 4 * 10 + 3 = 40 + 3 = 43。

（4）m（43）扩大 10 倍，即原 m 的各位均向左（高位）移动 1 位，再加上刚分离出的数位 2，即 m = m * 10 + 2 = 43 * 10 + 3 = 430 + 2 = 432。

（5）m（432）扩大 10 倍，即原 m 的各位均向左（高位）移动 1 位，再加上刚分离出的数位 1，即 m = m * 10 + 1 = 432 * 10 + 1 = 4320 + 1 = 4321。

完善上述代码框架如下：

```
m = 0
while n != 0 :
    t = n % 10       #分离 n 的当前低位数字
    m = m * 10 + t   #原 m 的各位作为高位，刚分离出的 t 作为低位，组成新 m
    n //= 10         #去除已分离出的当前低位，为下一次分离次低位作准备
```

【参考代码】

```
m = 0   #反序后的数，初始化为 0
n = eval(input('反序前：'))
```

```
while n != 0 :      #直到 n 分解为 0 循环才终止
    t = n % 10
    m = m * 10 + t
    n //= 10
print("反序后：{}".format(m))
```

【运行结果】

```
反序前：12345
反序后：54321
```

【案例 2】计算并输出 1-3+5-7+…-99 的值。

【分析】本题是重复执行“把当前数据项 item，如 1、-3、5、-7…等累加到求和变量 s 上”的相似操作，故采用循环结构。循环算法的关键是要确定循环条件表达式和循环体语句。

每个数据项 item 由符号位 sign 和数值位 n 两部分组成。

循环控制变量及初始条件确定：由题意可知，数据项的数值位 n 作为该题的循环控制变量，初值为 n=1。其他变量初始值为求和变量 s=0，符号位 sign。由于第一个数据项为 1 即+1，故符号位初始为 sign=1。

循环条件表达式的确定：循环控制变量 n 的起止值，起始值为 1，终止值为 99。故循环条件表达式为 n<=99。

循环体确定：该题的循环体语句包括三部分操作：

（1）组建当前数据项 item（由符号位 sign 和数值位 n 组成），即 item =sign* n。

（2）把各个当前数据项累加到求和变量 s 上，即 s+=item。

（3）然后改变下一个数据项的符号位 sign 及数值位 n，符号位与前一项相反，即 sign*=-1，数值位 n 的改变也就是循环控制变量的增量部分，比前一项大 2，即 n+=2。

案例 2 执行流程图如图 3-2 所示。

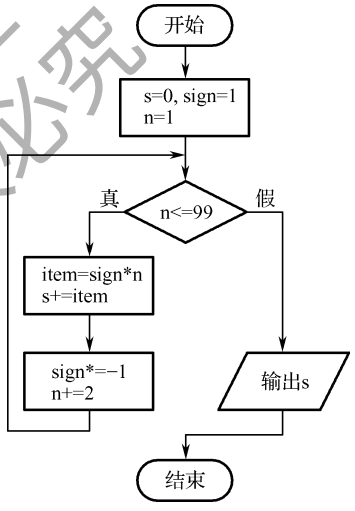

图 3-2　案例 2 执行流程

【参考代码】

```
s = 0                       #求和变量，初值为 0
sign , n = 1, 1             #首项的符号位和数值位
while n <= 99 :
    item = sign * n         #组建当前项
    s += item               #累加当前项
    sign *= -1              #改变下一项的符号位
    n += 2                  #改变下一项的数值位
print('sum = {}'.format(s))
```

【运行结果】

```
sum = -50
```

【动手练一练】

1．输入一个 n 值，求算术表达式 1+1/2+1/3+…+1/n 的值。

2．输入一个十进制正整数，计算并输出该整数是几位数。如 123 是 3 位数，1000 是 4 位数。

3.2 任务 2 猜数字谜游戏——循环流程控制

Python 控制程序流程跳转的语句通常有 break、continue、return 等。本节主要讲解前两种流程跳转语句，return 语句通常用在被调函数执行结束后，返回调用者处的流程跳转，将在函数项目中讲解。

任务目标

- 掌握 break、continue 等流程跳转语句在循环流程控制中的应用
- 掌握 while–else 循环结构
- 能够使用循环流程控制结构解决实际问题

【任务描述】设计一个猜数字游戏，随机生成 1 到 100 之间的一个整数，玩家总共有 5 次机会，每次输入所猜数字，程序给出相应的提示信息（“再大一点”、“再小一点”或“$恭喜您猜中了$”），如果猜中了，提示“$恭喜您猜中了$”，“总共猜了*次”，游戏结束，否则游戏继续；总共有 5 次机会，若均没猜中，则提示“很遗憾，今天运气不好！”并退出循环。

【任务分析】猜数字游戏是一直执行相似的操作，直到猜中为止，故采用循环结构。但是循环有两种终止情况，一种是 5 次机会全部用完均为猜中，正常停止；还有一种是猜中了提前停止。

【任务类型】本任务需要用到改变循环执行流程的语法。

3.2.1 知识点 1：break 语句

当执行到循环体中的 break 语句时，将终止 break 所在层的循环，从该层循环体之后的语句继续执行。

单重循环情况：这里选用 while 循环结构示意，对 for 循环结构也同样适用。循环嵌套中 break 的使用方法将在后面项目中讲解。

1．语法格式

```
while 循环判断表达式:
    循环体内 break 前的语句组
    if 条件表达式 :
        break
    循环体内 break 后的语句组
循环结构后的语句组
```

2．执行流程

在循环体中，当执行到 break 语句时，终止 break 所在层的循环，即“循环体内 break 后的语句组”部分将不再被执行，程序执行流程从“循环结构后的语句组”处，继续往后执行。

【示例】

【示例 1】分析以下程序，输出其运行结果。

【程序代码】

```
n = 0
print('按约定工作 5 天')
print('------------')
while n < 5 :
    n += 1
    print('第{}天打卡'.format(n))
print('------------')
print('完成约定')
```

【分析】

循环控制变量 n 的初值为 0，只要 n<5 就一直执行，故 n 的取值范围为 0、1、2、3、4，故循环体共执行 5 次。执行 5 次后循环结构正常终止，接着继续执行循环结构后的打印分割线及“完成约定”信息的语句。

【运行结果】

```
按约定工作 5 天
------------
第 1 天打卡
第 2 天打卡
第 3 天打卡
第 4 天打卡
第 5 天打卡
------------
完成约定
```

【示例 2】分析以下程序，输出其运行结果。

【程序代码】

```
n = 0
print('按约定工作 5 天')
print('------------')
while n < 5 :
    n += 1
    if n == 4 :
        print('发现合同有问题')
        break
    print('第{}天打卡'.format(n))
print('------------')
print('提前终止合作')
```

【分析】

若去掉循环体中 if 结构，则是普通的 while 循环结构，与上例相似。

```
if n == 4 :
    print('发现合同有问题')
    break
```

当 n==4 时，出现“特殊情况”，先执行打印“发现合同有问题”这行代码，然后执行 break 语句。在单层循环中，当执行到 break 时，立刻终止执行整个循环结构，故“第 4 天打卡”、“第 5 天打卡”均不会被打印出来，执行流程跳转到循环结构后的第一条语句，即打印分界线处继续往后执行。

【运行结果】

```
按约定工作 5 天
------------
第 1 天打卡
第 2 天打卡
第 3 天打卡
发现合同有问题
------------
提前终止合作
```

3.2.2 知识点 2：continue 语句

在循环体中设置 continue 语句，同样可以改变循环的执行流程，只是它不像 break 那样结束整个循环体，而是仅结束本次循环体的执行，提前进入下一次循环。

仍选用 while 循环结构示意。

1. 语法格式

```
while 循环判断表达式:
    循环体内 continue 前的语句组
    if 条件表达式 :
        continue
    循环体内 continue 后的语句组
循环结构后的语句组
```

2. 执行流程

在循环体中，当执行到 continue 语句时，本次循环体的执行流程将跳过“循环体内 continue 后的语句组”，继续执行“循环判断表达式”，即提前进入下一次的循环准备工作。

【示例】分析以下程序，输出其运行结果。

【程序代码】

```
n = 0
print('按约定工作 5 天')
print('------------')
while n < 5 :
    n += 1
    if n == 4 :
        print('身体不舒服，请假一天')
```