day02 课堂笔记

课程之前

复习和反馈

```
数据类型转换:  数据本来的类型不是我们计算使用想要的类型
int()  将其他类型转换为 int 类型(小数, 整数类型的字符串)
float() 将其他类型转换为 float 类型(整数, 数字类型的字符串)
str() 将其他类型转换为 str 类型(任意类型)

函数: 作用 语法
type(变量)  可以获取变量的数据类型

input() 获取键盘的输入
变量 = input('提示性信息')    # 提示性信息可以随便写,目的是让别人知道要做什么事
```

作业

```
# 书写代码
# 获取用户输入的数字,类型是 str
num1 = input('请输入第一个数字:')
num2 = input('请输入第二个数字:')
# 求和, 需要将 str类型转换为数字类型
num = int(num1) + int(num2)
# 打印求和的结果
print(num)
```

今日内容

- 字符串格式化的补充
- 运算符
- 判断语句 `if elif else`
- 循环语句基础`while`

字符串格式化的补充

```
字符串.format()  可以在任意版本中使用

1. 在需要使用 变量的地方使用 {} 占位
2. '{}, {}, ...'.format(变量, 变量, ...)
```

```
# 定义变量 姓名  年龄  身高
```

```python
name = '小明'  # 可以使用 input 输入
age = 18   # 可以使用 input 输入
height = 1.71  # 可以使用 input 输入
stu_num = 1  # 学号
num = 90  # 及格率

# print('我的名字是 xx，年龄是 xx，身高是 xx m，学号 xx，本次考试的及格率为 xx%')
print(f'我的名字是 {name}，年龄是 {age}，身高是 {height} m，学号 {stu_num}，本次考试的及格率为 {num}%')
# 一般不会有这样的需求
print(f'我的名字是 {name}，年龄是 {age}，身高是 {height:.3f} m，学号 {stu_num:06d}，本次考试的及格率为 {num}%')

# 字符串.format()
print('我的名字是 {}，年龄是 {}，身高是 {} m，学号 {}，本次考试的及格率为 {}%'.format(name, age, height, stu_num, num))
print('我的名字是 {}，年龄是 {}，身高是 {:.3f} m，学号 {:06d}，本次考试的及格率为 {}%'.format(name, age, height, stu_num, num))

```

运算符

逻辑运算符

```
逻辑运算符 可以连接多个条件，在判断和循环中使用

and  逻辑与 和,并且  and 连接两个条件,都必须为 True，整体结果才为 True，即一假为假（当第一个条件为 False 的时候,第二个条件就不再判断）

or  逻辑或  或者    or 连接的两个条件，只要有一个条件为 True ，整体结果就为 True，即 一真为真（当第一个条件为 True的时候,第二个条件就不再判断）

not 逻辑非  取反    not 后边的条件，如果本来是 True,变为 False，本来是 False,变为 True
```

```
>>> a = 5
>>> b = 3
>>> a > b and  a == 5  # True  and True
True
>>> a >= b and b >= 3  # True  and True
True
>>> a != b  and b != 3  # True  and False
False
>>> a != b  or b != 3  # True or False
True
>>> not  a > b  # not True
False
```

赋值运算符

```
1  赋值运算符 =，作用就是将等号右边的值保存到等号左边的变量中
2
3  复合赋值运算符(将算术运算符和赋值运算符进行结合)
4  += -= *= /= //=  %=
5
6  a += b  ===> a = a + b
```

运算符优先级

```
1  不需要刻意去记忆优先级,因为可以使用 () 改变优先级
```

判断

```
1  日常生活中说的 如果 ... 否则 .... ,这个就是判断,在程序代码中需要使用 if(如果)  elif(如果)  else(否则) 三个关键字来实现
2
3  在代码中有判断语句，待会不会全部执行，会有一部分不会执行
```

if 的基本结构

```
1  即 只有 如果的情况，  如果的条件成立，会执行的代码,会做的事
```

- 基本语法

```
if 判断条件:
    书写条件成立(真),执行的代码
    书写条件成立(真),执行的代码

顶格书写,没有缩进的代码,和 if无关, 不管条件是否成立,都会执行

# 1. if 是一个关键字, 和后续的判断条件之间需要一个空格
# 2. 判断条件后边需要一个冒号,不要少了
# 3. 冒号之后,回车,代码需要缩进, 在 pycharm 中会自动进行缩进, 一般是 4 个空格 或者 一个 tab 键
# 4. 所有在 if 代码下方的缩进中书写的代码,属于 if 语句的代码块, 判断条件为 True 的时候会执行
# 5. if 代码块中的代码,要么都执行,要么都不执行
# 6. if 代码块结束之后, 代码要顶格书写(不再有缩进), 表示是和 if 无关的代码
```

- 代码案例

```
1. 使用 input 获取用户的年龄
2. 判断年龄是否满足 18 岁
3. 如果年龄大于等于(满足)18 岁, 输出 '满 18 岁了,可以进入网吧为所欲为了'
```

```
# 1. 使用 input 获取用户的年龄, 类型是 str
age = input('请输入你的年龄:')
# 2. 判断年龄是否满足 18 岁
if int(age) >= 18:  # 字符串和 int 类型不能比大小, 先类型转换,再比大小
    # 3. 如果年龄大于等于(满足)18 岁, 输出 '满 18 岁了,可以进入网吧为所欲为了'
    print('满 18 岁了,可以进入网吧为所欲为了')


print('我和 if 判断没有关系,不管怎样,都会执行')
```

- 练习

```
1. 获取用户输入的用户名信息
2. 如果用户名信息是 admin, 就在控制台输出出来
```

```
# 1. 获取用户输入的用户名信息
name = input('请输入你的用户名:')
# 2. 如果用户名信息是 admin, 就在控制台输出出来
if name == 'admin':
    print('欢迎 admin')
```

if else 结构

```
如果 条件成立 做什么事 否则(条件不成立) 做另一件事
```

- 基本语法

```
if 判断条件:
    书写条件成立(真),执行的代码
    书写条件成立(真),执行的代码
else:
    书写条件不成立(假), 执行的代码
    书写条件不成立(假), 执行的代码


# 1. else 是关键字, 后边需要冒号
# 2. 冒号之后回车,同样需要缩进
# 3. 处于 else 代码下方缩进中的内容,属于 else 的代码块
# 4. if  和 else 的代码块, 只会执行其中的一个
# 5. else 需要结合 if 使用
# 6. if else 之间不能有其他顶格书写的内容(不提 elif)
```

- 代码案例

```
1. 使用 input 获取用户的年龄
2. 判断年龄是否满足 18 岁
3. 如果年龄大于等于(满足)18 岁, 输出 '满 18 岁了,可以进入网吧为所欲为了'
4. 如果不满足, 输出 '不满 18 岁,回去写作业吧'
```

```
# 1. 使用 input 获取用户的年龄, 类型是 str
age = input('请输入你的年龄:')
# 2. 判断年龄是否满足 18 岁
if int(age) >= 18:  # 字符串和 int 类型不能比大小, 先类型转换,再比大小
    # 3. 如果年龄大于等于(满足)18 岁, 输出 '满 18 岁了,可以进入网吧为所欲为了'
    print('满 18 岁了,可以进入网吧为所欲为了')
# 4. 如果不满足, 输出 '不满 18 岁,回去写作业吧'
else:
    print('不满 18 岁,回去写作业吧')

```

- 练习

```
1. 获取用户输入的用户名信息
2. 如果用户名信息是 admin, 就在控制台输出出来
3. 如果用户名信息不是 admin, 就在控制台输出"用户名错误!"
```

```python
# 1. 获取用户输入的用户名信息
name = input('请输入你的用户名:')
# 2. 如果用户名信息是 admin, 就在控制台输出出来
if name == 'admin':
    print('欢迎 admin')
# 3. 如果用户名信息不是 admin, 就在控制台输出"用户名错误!"
else:
    print('用户名错误!')

```

if 和逻辑运算符结合使用

- 案例一

```
1. 获取用户输入的用户名和密码
2. 判断用户名是 admin 并且密码是 123456 时, 在控制台输出: 登录成功!
3. 否则在控制台输出: 登录信息错误!
```

```python
# 1. 获取用户输入的用户名和密码
name = input('请输入用户名:')
pwd = input('请输入密码:')

# 2. 判断用户名是 admin 并且密码是 123456 时, 在控制台输出: 登录成功!
if name == 'admin' and pwd == '123456':
    print('登录成功!')
# 3. 否则在控制台输出: 登录信息错误!
else:
    print('登录信息错误!')

```

- 案例二

```
1. 获取用户输入的用户名
2. 判断用户名是 admin 时, 在控制台输出: 欢迎 admin 登录!
3. 用户名是 test 时, 在控制台输出: 欢迎 test 登录!
4. 如果是其他信息, 在控制台输出: 查无此人!
```

```python
# 1. 获取用户输入的用户名
username = input('请输入用户名:')
# 2. 判断用户名是 admin 时, 在控制台输出: 欢迎 admin 登录!
# 3. 用户名是 test 时, 在控制台输出: 欢迎 test 登录!
if username == 'admin' or username == 'test':
    print(f'欢迎 {username} 登录!')
# 4. 如果是其他信息, 在控制台输出: 查无此人!
else:
    print('查无此人!')
```

```
# username == 'admin'  或者 'test'(一直 True, 空字符串是 False)
if username == 'admin' or 'test':
    pass   # pass  关键字, 占位,
```

if elif else 结构

```
如果某个判断条件有多个, 此时建议使用 if elif else 结构来实现
```

- 语法

```
if 判断条件1:
    判断条件1成立,执行的代码
elif 判断条件2:  # 只有判断条件1不成立,才会判断 判断条件2
    判断条件2成立执行的代码
else:
    以上条件都不成立,执行的代码

# 1. elif 也是关键字,后边和判断条件之间需要一个空格,判断条件之后需要冒号
# 2. 冒号之后回车需要缩进, 处在这个缩进中的的代码表示是 elif 的代码块
# 3. 在一个 if判断中,可以有很多个 elif
# 4. 只有 if 的条件不成立,才会去判断 elif 的条件
# 5. 在一个if 中, 如果有多个 elif , 只要有一个条件成立,后续的所有都不再判断
# 6. if elif else 结构, 和 if 的缩进相同的只能是 elif  和 else,如果是其他的,就表示 这个判断结构结束了

if 判断条件1:
    执行的代码
if 判断条件2:
    执行的代码
if 判断条件3:
    执行的代码

# 多个 if 的结构, 每个 if 都会进行判断,之间没有关联系
```

- 案例

```
1. 定义 score 变量记录考试分数
2. 如果分数是大于等于90分应该显示优
3. 如果分数是大于等于80分并且小于90分应该显示良
4. 如果分数是大于等于70分并且小于80分应该显示中
5. 如果分数是大于等于60分并且小于70分应该显示差
6. 其它分数显示不及格
```

```
# 3. 如果分数是大于等于80分并且小于90分应该显示良
elif score >= 80 and score < 90:
    print('良') and 连接两个条件， 怕你不知道优先级， 想让你加括号
# 4. 如果分数是大于等于70分并且小于80分应该显示中
# 5. 如果分数是大于等于60分并且小于70分应该显示差
```

- elif 实现

```
# 1. 定义 score 变量记录考试分数
score = int(input('请输入你的分数'))  # int  float
# 2. 如果分数是大于等于90分应该显示优
if score >= 90:
    print('优')
# 3. 如果分数是大于等于80分并且小于90分应该显示良
elif (score >= 80) and score < 90:
    print('良')
# 4. 如果分数是大于等于70分并且小于80分应该显示中
# and score < 80 可以不写的， 原因只有上边一个判断条件不成立(一定满足 score<80),才会执行这个
elif score >= 70:
    print('中')
# 5. 如果分数是大于等于60分并且小于70分应该显示差
elif score >= 60:
    print('差')
# 6. 其它分数显示不及格
else:
    print('不及格')
```

- 多个 if 实现

```
# 1. 定义 score 变量记录考试分数
score = int(input('请输入你的分数'))  # int  float
# 2. 如果分数是大于等于90分应该显示优
if score >= 90:
    print('优')
# 3. 如果分数是大于等于80分并且小于90分应该显示良
if (score >= 80) and score < 90:
    print('良')
# 4. 如果分数是大于等于70分并且小于80分应该显示中
if (score >= 70) and score < 80:
    print('中')
# 5. 如果分数是大于等于60分并且小于70分应该显示差
if (score >= 60) and score < 70:
    print('差')
# 6. 其它分数显示不及格
if score < 60:
    print('不及格')

```

Debug 调试代码

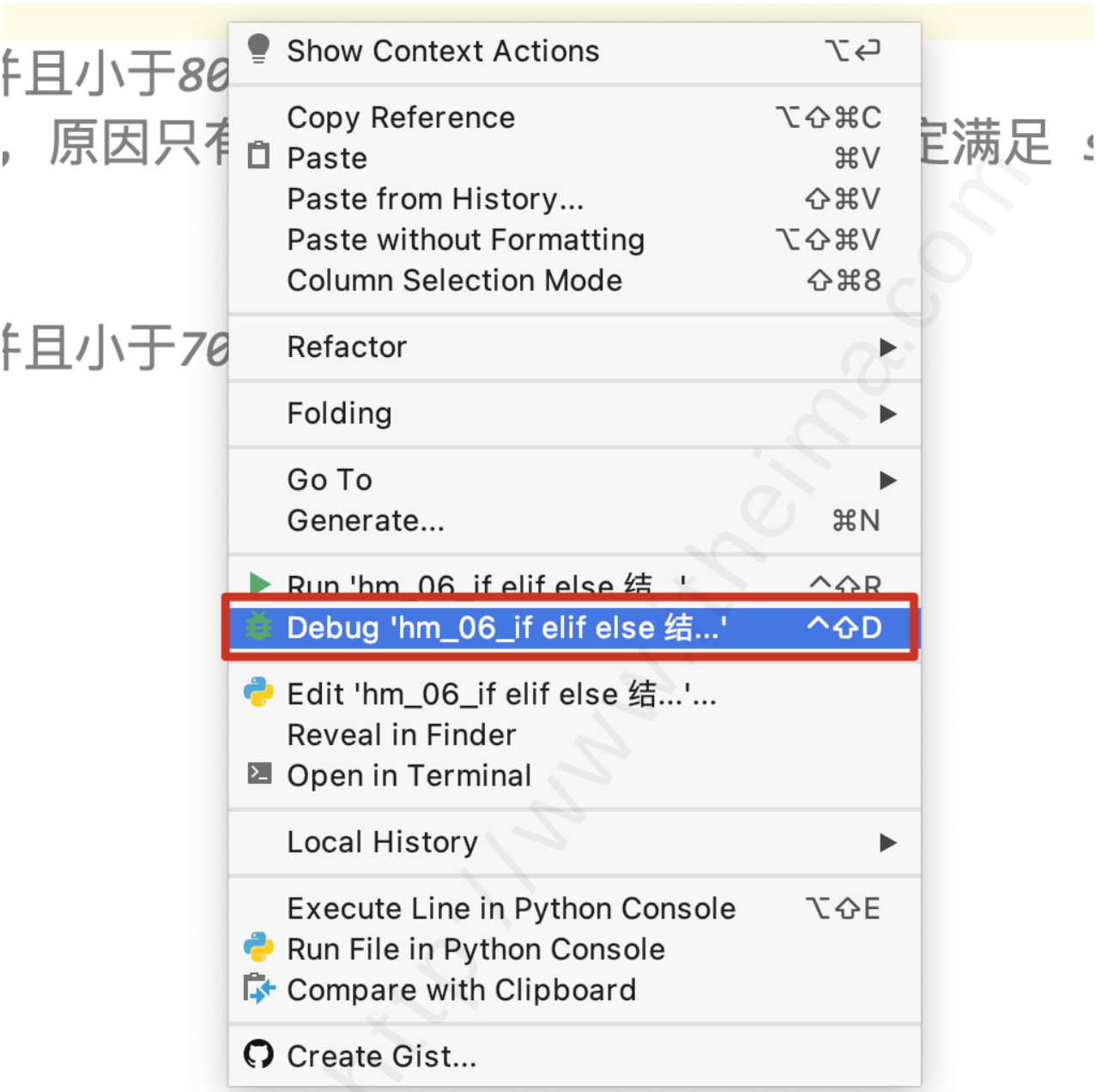
```
debug 在代码中出现问题错误(bug)，可以使用 debug 来调试代码,查找错误.
我们使用 debug 主要用来 查看代码的执行步骤
```

1. 打断点

```
在 pycharm 中，代码和行号之间进行点击，出现小红点即打断点，再次点击小红点会取消断点

断点的位置,一般来说会在代码的第一行(在程序运行的时候,想要在什么地方停下来)

注意点: 可能会出现的 bug(pycharm 软件的问题): 代码中只有一个断点的时候不能 debug 调试查看代码执行过程，解决方案,在代码其他任意地方多加一个断点
```

```python
# 1. 定义 score 变量记录考试分数
score = int(input('请输入你的分数'))  # int  float
# 2. 如果分数是大于等于90分应该显示优
if score >= 90:
    print('优')
# 3. 如果分数是大于等于80分并且小于90分应该显示良
elif (score >= 80) and score < 90:
    print('良')
# 4. 如果分数是大于等于70分并且小于80分应该显示中
# and score < 80 可以不写的，原因只有上边一个判断条件不成立(一定满足 score<
elif score >= 70:
    print('中')
# 5. 如果分数是大于等于60分并且小于70分应该显示差
elif score >= 60:
    print('差')
# 6. 其它分数显示不及格
else:
    print('不及格')

```

2. 右键 `debug` 运行代码



3. 单步执行代码

if 嵌套

```
if 嵌套，是指在一个 if(elif else) 中嵌套另一个 if.
使用场景：判断条件存在递进关系(只有第一个条件满足了,才会判断第二个条件)
```

- 语法

```
if 判断条件1:
    判断条件1成立,执行的代码
    if 判断条件2:
        判断条件2成立,执行的代码
    else:
        判断条件2不成立,执行的代码
else:
    判断条件1不成立,执行的代码
```

- 代码案例

```
取款机取钱的过程，假定 你的密码是：123456，  账户余额为 1000

1. 提示用户输入密码
2. 判断密码是否正确
3. 密码正确后,提示输入取款的金额，
4. 判断取款的金额和余额的关系
```

```
# 取款机取钱的过程，假定 你的密码是：123456，  账户余额为 1000
pwd = '123456'  # 可以预先定义变量,也可以在判断的时候直接使用
money = 1000  # 可以预先定义变量,也可以在判断的时候直接使用

```

```python
# 1. 提示用户输入密码
password = input('请输入密码:')
# 2. 判断密码是否正确
if password == pwd:
    print('密码正确,登录成功')
    # 3. 密码正确后,提示输入取款的金额,
    get_money = int(input('请输入要取款的金额:'))
    # 4. 判断取款的金额和余额的关系
    if money >= get_money:
        print('取款成功')
    else:
        print('余额不足')
else:
    print('密码有误,请再次尝试')

```

- 练习

```
假定某网站用户名固定为 'admin', 密码固定为'123456', 验证码 固定为 '8888'
1. 获取用户输入的用户名,密码和验证码
2. 先判断验证码是否正确,如果正确打印输出验证码正确,再判断用户名和密码是否正确
3. 如果验证吗不正确,直接输出 验证码不正确,请重新输入
```

```python
# 假定某网站用户名固定为 'admin', 密码固定为'123456', 验证码 固定为 '8888'
# 1. 获取用户输入的用户名,密码和验证码
username = input('请输入用户名:')
pwd = input('请输入密码:')
code = input('请输入验证码:')
# 2. 先判断验证码是否正确,如果正确打印输出验证码正确,再判断用户名和密码是否正确
if code == '8888':
    print('验证码正确')
    # 再判断用户名和密码是否正确
    if username == 'admin' and pwd == '123456':
        print('用户名密码正确,登录成功')
    else:
        print('用户名或者密码错误,请再次尝试')
# 3. 如果验证吗不正确,直接输出 验证码不正确,请重新输入
else:
    print('验证码不正确')

```

案例: 猜拳游戏

```
剪刀 石头 布
剪刀 赢 布
石头 赢 剪刀
布 赢 石头

```

```
案例的步骤:
1. 自己出拳(石头(1)/剪刀(2)/布(3)) input  (player)
2. 电脑随机出拳 (使用随机数模块(工具)完成)  (computer)
3. 判断输赢
3.1 玩家胜利
3.1.1 player==1 and computer == 2
or
3.1.2 player==2 and computer == 3
or
3.1.3 player==3 and computer == 1
3.2 平局  player == computer
3.3 玩家输了 else
```

随机出拳

```
案例中需要电脑随机出拳,即随机出 1 2 3
在 Python 中想要随机获得整数数字可以使用如下方法
# 1. 导入随机数工具包
import random
# 2. 使用工具包中的工具产生指定范围内的数字
random.randint(a, b)  # 产生[a, b] 之间的随机整数,包含 a b 的
```

```
import random  # 这行代码习惯性的放在第一行

num = random.randint(1, 3)
print(num)
```

代码

```
import random

# 1. 自己出拳(石头(1)/剪刀(2)/布(3)) input  (player)
player = int(input('请出拳石头(1)/剪刀(2)/布(3):'))   # 不要忘了类型转换
# 2. 电脑随机出拳 (使用随机数模块(工具)完成)  (computer)
computer = random.randint(1, 3)
# 3. 判断输赢
# 3.1 玩家胜利
if (player == 1 and computer == 2) or (player == 2 and computer == 3) or (player == 3 and computer == 1):
    print('恭喜你获得胜利')
# 3.2 平局  player == computer
elif player == computer:
    print('平局')
# 3.3 玩家输了 else
else:
    print('输了, 不要放弃, 再来一局')

```

循环

```
程序开发中(写代码)，有三大流程(三大结构)：
1，顺序，代码从上到下,全部执行
2，分支，判断语句,代码有选择性的执行
3，循环，重复执行某一部分的代码


循环的作用就是让 指定的代码 重复的执行
```

- 语法

```
1. 设置循环的初始条件(计数器)

2. 书写循环的判断条件
while 判断条件:
    # 3. 需要重复执行的代码
    # 4. 改变循环的初始条件(计数器)

# 1. while 是关键字
```

- 代码

```
# 1. 设置循环的初始条件 (记录说了几遍 我错了)
i = 0

# 2. 书写判断条件
while i < 5:
    # 3. 重复执行的代码
    print('媳妇，我错了')
    # 4. 改变初始条件，
    # i = i + 1
    i += 1


```

死循环和无限循环

```
死循环：一般是由写代码的人不小心造成的 bug，代码一直不停的运行下去

无限循环：写代码的人故意让代码无限制的去执行,代码一直不停的运行下去
无限循环的使用场景：在书写循环的时候,不确定循环要执行多少次
无限循环的使用一般会在循环中添加一个 if 判断，当 if 条件成立,使用关键字 break 来终止循环

```

```
while True:
    重复执行的代码  # 可以在 if 的上边
    if 判断条件:
        break  # 关键字的作用就是终止循环， 当代码执行遇到 break,这个循环就不再执行了
    重复执行的代码     # 可以在 if 的下边

```

```
书写循环的技巧:
确定这行代码执行几次,如果执行多次,就放在循环的缩进中,如果只执行一次,就不要放在循环的缩进中
```

```
import random

while True:  # 不知道你想玩几次,所以使用 无限循环
    # 1. 自己出拳(石头(1)/剪刀(2)/布(3)) input  (player)
    player = int(input('请出拳石头(1)/剪刀(2)/布(3)/退出(0):'))   # 不要忘了类型转换
    if player == 0:
        break  # 代码遇到 break,就会终止循环
    # 2. 电脑随机出拳 (使用随机数模块(工具)完成)  (computer)
    computer = random.randint(1, 3)
    # 3. 判断输赢
    # 3.1 玩家胜利
    if (player == 1 and computer == 2) or (player == 2 and computer == 3) or (player
== 3 and computer == 1):
        print('恭喜你获得胜利')
    # 3.2 平局  player == computer
    elif player == computer:
        print('平局')
    # 3.3 玩家输了 else
    else:
        print('输了，不要放弃，再来一局')

```

使用循环求 1-100 之间数字的和

```
1 + 2 + 3 + ... + 99 + 100
```

```
# 1. 定义变量 保存求和的结果
num = 0
# 2  定义循环获取 1- 100 之间的数字并求和
# 2.1 定义初始条件
i = 1
# 2.2 书写循环的条件
while i <= 100:
    # 2.3 书写重复执行的代码
    # 0 + 1 + 2 + 3 + 4 + .. + 99 + 100
    num += i  # 因为每一次 i 的值会变化
    # 2.4 改变计数器
    i += 1
# 3  打印求和的结果
print('求和的结果是:', num)
```

使用循环求 1-100 之间偶数的和

方法一

```
直接找到 1-100 之间所有的偶数,进行相加
2 4 6 8 .. 98 100
```

```
# 1. 定义变量 保存求和的结果
num = 0
# 2  定义循环获取 1- 100 之间的偶数并求和
# 2.1 定义初始条件
i = 2
# 2.2 书写循环的条件
while i <= 100:
    # 2.3 书写重复执行的代码
    num += i  # 因为每一次 i 的值会变化  num  = num + i
    # print(i)
    # 2.4 改变计数器
    i += 2  # 每次加 2，才可以保证所有的数字都是偶数
# 3  打印求和的结果
print('求和的结果是:', num)
```

方法二

```
1. 找 1-100 之间所有的数字
2. 判断这个数字是不是偶数，如果是偶数再求和
偶数：能被 2 整除的数，（数字除 2 的余数是 0）
i % 2 == 0
```

```
num = 0

i = 1
while i <= 100:
    # 判断该数字是不是偶数,如果是偶数再求和
    if i % 2 == 0:
        # print(i)
        num += i

    # 改变计数器
    i += 1

print('求和的结果是:', num)

```