

day02 课堂笔记

课程之前

复习和反馈

多行注释

- 1, 一般放在文件的开头, 标明整个代码文件做什么事, 或者其他的信息, 版本号, 作者等
- 2, 文档注释(函数)

作业

今日内容

流程控制:

判断(如果...否则): if(如果) elif(如果) else(否则)
循环(重复做某些事): while(直到) for in(在) break(终止) continue(继续)
pass 空语句, 可以占位

判断语句

判断语句 对应的就是 如果 否则
条件成立 做什么事, 条件不成立做什么事
很多的条件怎么处理

if 语句的基本结构

- 介绍

单独的 `if` 语句,就是 如果 条件成立,做什么事

- 语法

`if` 判断条件:

判断条件成立,执行的代码

判断条件成立,执行的代码

这行代码和`if` 判断无关

1, `if` 语句后边需要一个 冒号

2, 冒号之后 需要回车缩进(在 `pycharm` 中会自动的添加)

3, 处在 `if` 语句的缩进中的代码 可以称为是 `if` 语句的代码块(多行代码)

4, `if` 语句的代码块中的代码,要么都执行,要么都不执行

5, 如果某行代码 和`if` 的判断无关,就不需要写在 `if` 的缩进中

- 代码需求

需求:

1. 定义一个整数变量记录输入的年龄
2. 判断是否满 18 岁 (\geq)
3. 如果满 18 岁, 允许进网吧嗨皮

- 代码

```
# 1. 定义一个整数变量记录输入的年龄  input -->str
age = int(input('请输入年龄:'))      #
# 2. 判断是否满 18 岁 ( $\geq$ )
if age  $\geq$  18:
    # 3. 如果满 18 岁, 允许进网吧嗨皮
    print('哥满 18 岁了, 可以进入网吧 为所欲为...')

print('这行代码 和 if 判断无关')  # 1 会  2 不会
```

- 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 "欢迎admin 登录"

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

if else 结构

- 语法

```
# if else 如果 ... 否则 ....
if 判断条件:
    判断条件成立,执行的代码
else:
    判断条件不成立,执行的代码
```

1, else 是关键字, 后面需要 冒号
2, 存在冒号,就需要回车 和缩进
3, 处于 else 缩进中的代码 ,称为是 else 语句的代码块
4, else 不能单独使用 必须配合 if 使用, 并且 else
要和 if 对齐
5, if 和 else 之间不能有顶格书写的东西

- 需求

需求：

1. 输入用户年龄
2. 判断是否满 18 岁 (\geq)
3. 如果满 18 岁，允许进网吧嗨皮
4. 如果未满 18 岁，提示回家写作业

● 代码

```
# 1. 输入用户年龄
age = int(input('请输入年龄:'))
# 2. 判断是否满 18 岁 ( $\geq$ )
if age >= 18:
    # 3. 如果满 18 岁，允许进网吧嗨皮 条件成立
    print('可以进网吧嗨皮')
else:
    # 4. 如果未满 18 岁，提示回家写作业
    print('回家写作业....')
```

● 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 "欢迎admin 登录"
3. 如果用户名信息不是 admin，就在控制台输出 "用户名错误!"

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

if 和逻辑运算符结合

逻辑运算符 and or not

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!

```
# 1. 获取用户输入的用户名和密码    input
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. 定义两个整数变量python_score、c_score，使用 input 获取成绩 编写代码判断成绩
2. 要求只要有一门成绩 > 60 分就输出合格

```
# 1. 定义两个整数变量 python_score、c_score，使用
input 获取成绩 编写代码判断成绩
python_score = int(input('请输入 Python 成绩:'))
c_score = int(input('请输入 C 成绩:'))
# 2. 要求只要有一门成绩 > 60 分就输出合格
if python_score > 60 or c_score > 60:
    print('合格')
```

案例三

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！

```
# 1. 获取用户输入的用户名
name = 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 name == 'admin' or name == 'test':
    print(f'欢迎 {name} 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!')
```

if elif else 结构

- 语法

```
# if elif else 如果 ... 如果 ... 否则 ....
# 多个如果之间存在关系

if 判断条件1:
```


判断条件1成立,执行的代码

`elif` 判断条件2: # 判断条件1 不成立

判断条件2成立,执行的代码

`elif`:

`pass`

`else`:

以上 判断条件都不成立,才会执行的代码

1, `elif` 是关键字, 后边需要冒号, 回车 和 缩进

2, `if elif else` 的代码结构, 如果某一个条件成立, 其他的条件就都不再判断

● 需求

1. 定义 `score` 变量记录考试分数

2. 如果分数是 大于等于 90分 显示 优

3. 如果分数是 大于等于 80分 并且 小于 90分 显示 良

4. 如果分数是 大于等于 70分 并且 小于 80分 显示 中

5. 如果分数是 大于等于 60分 并且 小于 70分 显示 差

6. 其它分数显示 不及格

● 代码

1. 定义 `score` 变量记录考试分数

`score = int(input('请输入分数'))`

2. 如果分数是 大于等于 90分 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 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 显示 中
elif (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 显示 差
elif (score >= 60) and score < 70:
    print('差')
# 6. 其它分数显示 不及格
else:
    print('不及格')
```

```
# 1. 定义 score 变量记录考试分数
score = int(input('请输入分数'))
# 2. 如果分数是 大于等于 90分 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 显示 良
elif score >= 80:    # 想要判断这个条件,一定是上边的
                    # 条件不成立即 score < 90
    print('良')
# 4. 如果分数是 大于等于 70分 并且 小于 80分 显示 中
elif score >= 70:
    print('中')
# 5. 如果分数是 大于等于 60分 并且 小于 70分 显示 差
elif score >= 60:
```

```
    print('差')
# 6. 其它分数显示 不及格
else:
    print('不及格')
```

- 使用 多个 if 实现

```
# 1. 定义 score 变量记录考试分数
score = int(input('请输入分数'))
# 2. 如果分数是 大于等于 90分 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 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 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 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

我们使用 `debug` 的目的，认为就是查看代码的执行过程的。

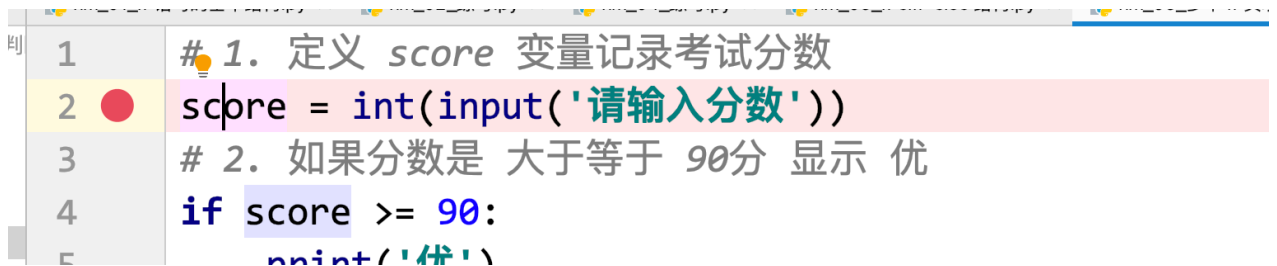
- 步骤

1. 打断点

断点的意义是，`debug` 运行的时候，代码会在断点处停下来不执行

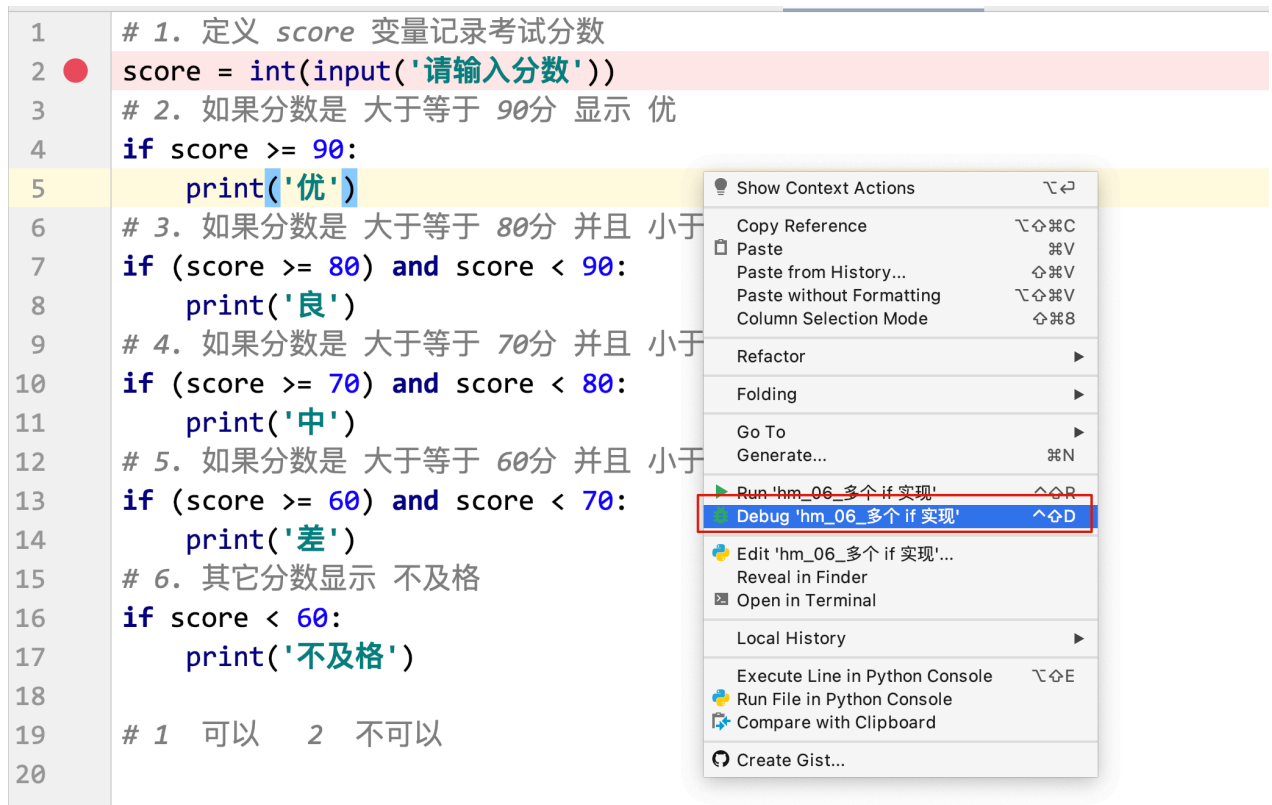
如果是想要查看代码的执行过程，建议将断点放在第一行在代码 和行号之间 点击，出现的红色圆点 就是断点，再次点击可以取消

`pycharm` 软件存在一个问题，想要 `debug` 运行，可能至少需要两个断点。

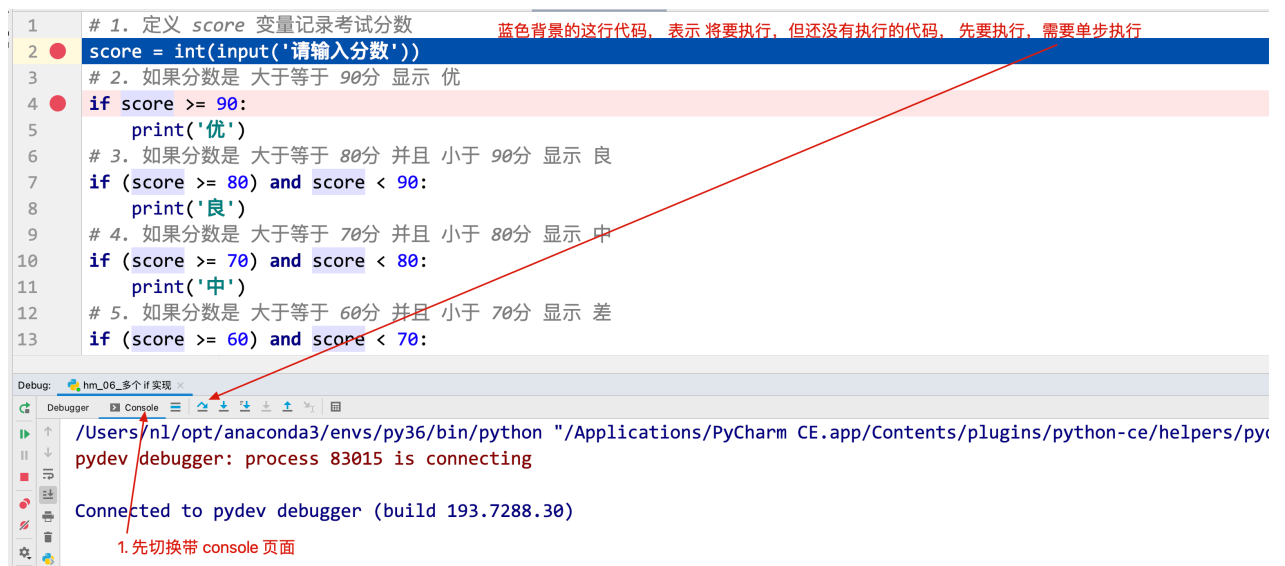


```
1 # 1. 定义 score 变量记录考试分数
2 ● score = int(input('请输入分数'))
3 # 2. 如果分数是 大于等于 90分 显示 优
4 if score >= 90:
5     print('优')
```

2. 右键 `DEBUG` 运行



3. 单步执行,查看执行过程



if 嵌套

在一个 `if(elif else)` 语句中 嵌套另一个 `if(elif else)` 语句

判断条件存在递进关系才会使用。即 只有第一个条件成立,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

- 需求

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

- 代码

```
# 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
pwd = input('请输入密码:')
# 2. 判断密码是否正确
if pwd == '123456':
    print('密码输入正确')
    # 3. 密码正确后,提示输入取款的金额,
    money = int(input('请输入取款的金额:'))
    # 4. 判断取款的金额和余额的关系
    if money > 1000:
        print('余额不足')
    else:
        print('取款中,请稍后....')
        print('请收好你的钱....')
else:
    print('密码输入错误,请重试')
```

- 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. 如果验证码不正确,直接输出 验证码不正确,请重新输入

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:')
```

```
password = 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')
```

```
    if username == 'admin' and password ==  
'123456':
```

```
        print('登录成功')
```

```
    else:
```

```
        print('用户名或密码错误')
```

3. 如果验证码不正确,直接输出 验证码不正确,请重新输入

```
else:
```

```
    print('验证码不正确,请重新输入')
```


案例 - 石头剪刀布

```
1. 控制台出拳(石头1/剪刀2/布3)  player = input()  
2. 电脑出拳  computer = 电脑的结果  
3. 判断胜负  
3.1 玩家胜利  
3.1.1 玩家出石头, 电脑出剪刀  player == 1 and  computer  
== 2  
or  
3.1.2 玩家出剪刀, 电脑出 布  player == 2 and  computer  
== 3  
or  
3.1.3 玩家出布, 电脑出 石头  player == 3 and computer  
== 1  
3.2 平局  玩家和电脑出的内容一样,  player == cpmputer  
3.3 电脑胜利  else:
```

随机数

让电脑随机出拳，即让电脑随机产生 1 2 3 中 任意一个数字，随机数

1, 导随机数包(工具包)

```
import random
```

2, 使用工具包中的 工具产生随机数

```
random.randint(a, b) # 产生 [a, b]之间的随机整数，包含 a 和 b
```

1. 导包(放在最上方)

```
import random
```

2. 产生 随机数

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

```
import random
```

1. 控制台出拳(石头1/剪刀2/布3)

```
player = int(input('请出拳 石头1/剪刀2/布3:'))
```

2. 电脑出拳 computer = 电脑的结果

```
computer = random.randint(1, 3)
```

3. 判断胜负

3.1 玩家胜利

3.1.1 玩家出石头，电脑出剪刀

3.1.2 玩家

出剪刀，电脑出 布

3.1.3 玩家出布，电脑出 石头

```
if (player == 1 and computer == 2) or (player == 2
and computer == 3) or (player == 3 and computer ==
1):
    print('玩家胜利')
# 3.2 平局 玩家和电脑出的内容一样,
elif player == computer:
    print('平局')
# 3.3 电脑胜利
else:
    print('电脑胜利')
```

循环

让指定的代码重复的执行。

while 循环

- 语法

1. 循环的初始条件(计数器)

2. 循环的终止条件

while 判断条件:

判断条件成立执行的代码

判断条件成立执行的代码

判断条件成立执行的代码

3. 计数器加1

1, 处于 **while** 缩进中的代码,称为是 **while** 的循环体

2, 执行顺序 1 2 3 2 3 2 3 2 3 2(条件不成立, 结束)

● 代码

1. 定义计数器 (说了几遍 我错了)

i = 0

2. 循环的终止条件

while **i** < 5:

print('媳妇, 我错了...')

 # 3. 计数器 +1

i = **i** + 1

死循环 & 无限循环

死循环和无限循环 在程序执行层面上看起来是一样的，都是代码一直执行不能停止。

死循环：是由于 写代码的人 不小心造成。 bug

无限循环：是写代码的人 故意这么写。

无限循环中，一般会存在一个 `if` 判断语句，当这个判断语句的条件成立，执行 `break` 语句，来终止循环。

关键字 `break`：当程序代码执行遇到 `break`，`break` 所在的循环就会被终止执行。(终止循环)

关键字 `continue`：当程序代码执行遇到 `continue`，`continue` 后续的代码不执行，但是会继续下一次的循环(结束本次循环，继续下一次循环)

`break` 和 `continue` 只能用在循环中。

```
while True:
    xxxx
    xxxx
    if xxxx:
        break
    xxxx
```

循环版本的石头剪刀布

- 1, 只需要确定哪些代码需要重复执行
- 2, 将需要重复执行的代码 写在循环的缩进中(循环体)

```
import random

while True:
    # 1. 控制台出拳(石头1/剪刀2/布3)
    player = int(input('请出拳 石头1/剪刀2/布3:'))
    # 2. 电脑出拳 computer = 电脑的结果
    computer = random.randint(1, 3)
    # 3. 判断胜负
    # 3.1 玩家胜利
    # 3.1.1 玩家出石头, 电脑出剪刀 # 3.1.2
    # 3.1.3 玩家出布, 电脑出石头
    if (player == 1 and computer == 2) or (player ==
2 and computer == 3) or (player == 3 and computer ==
1):
        print('玩家胜利')
    # 3.2 平局 玩家和电脑出的内容一样,
    elif player == computer:
        print('平局')
    # 3.3 电脑胜利
    else:
```

```
print('电脑胜利')
```

```
import random
```

```
while True:
```

```
    # 1. 控制台出拳(石头1/剪刀2/布3)
```

```
    player = int(input('请出拳 石头1/剪刀2/布3/退出0:'))
```

```
    if player == 0:
```

```
        # continue # 结束本次循环,继续下一次循环
```

```
        # 结束循环
```

```
        print('欢迎下次游戏')
```

```
        break
```

```
    # 2. 电脑出拳 computer = 电脑的结果
```

```
    computer = random.randint(1, 3)
```

```
    # 3. 判断胜负
```

```
    # 3.1 玩家胜利
```

```
    # 3.1.1 玩家出石头, 电脑出剪刀 # 3.1.2
```

```
    玩家出剪刀, 电脑出 布 # 3.1.3 玩家出布, 电脑出 石头
```

```
    if (player == 1 and computer == 2) or (player == 2 and computer == 3) or (player == 3 and computer == 1):
```

```
        print('玩家胜利')
```

```
    # 3.2 平局 玩家和电脑出的内容一样,
```

```
    elif player == computer:
```

```
        print('平局')
```

```
# 3.3 电脑胜利
else:
    print('电脑胜利')
```

案例 求 1- 100 之间的累加和

$1 + 2 + 3 + \dots + 99 + 100$
使用循环产生 1-100 之间的数字

```
# 1. 定义计数器
i = 1
# 定义变量 保存求和的结果
num = 0

# 2. 书写判断条件
while i <= 100:
    # 求和
    num = num + i # 0 + 1 + 2 + 3 + ... + 100
    i += 1

# 求和的结果 只需要打印一次，所以放在循环的外边
print('求和的结果为:', num)
```


for 循环

for 循环 也称为是 **for 遍历**，也可以做指定次数的循环
遍历：是从容器中将数据逐个取出的过程。
容器：字符串/列表/元组/字典

for 循环遍历字符串

for 变量 **in** 字符串：
重复执行的代码

- # 1，字符串中存在多少个字符，代码就执行多少次
- # 2，每次循环 会从字符串中取出一个字符保存到前边的变量中
- # 3，**for** 和 **in** 都是关键字

```
str1 = 'abcd'

for i in str1:
    print('媳妇，我错了', i)
```

for 指定次数的循环

for 变量 **in range(n)**： # **n** 就是要循环的次数
重复执行的代码

- # 1，**range(n)** 可以生成 **[0, n)** 的整数的序列，不包含 **n**

```
for i in range(5): # [0, 1, 2, 3, 4]
    print('媳妇，我错了', i)
```

1-100 之间的类加和

使用 for 循环 找 1- 100 之间的数字

```
for i in range(101): [0, 100)
```

定义变量保存求和的结果

```
num = 0
```

```
for i in range(101):
```

```
    num += i
```

```
print('求和的结果为：', num)
```

案例

1. 提示用户输入登录系统的用户名和密码
2. 校验用户名和密码是否正确(正确的用户名:admin、密码:123456)
3. 如果用户名和密码都正确, 打印“登录成功!”, 并结束程序
4. 如果用户名或密码错误, 打印“用户名或密码错误!”, 提示用户继续输入用户名和密码登录
5. 如果用户输入的用户名为“exit”, 则退出程序

```
while True:
    # 1. 提示用户输入登录系统的用户名和密码
    username = input('请输入用户名:')
    if username == 'exit':
        break
    password = input('请输入密码:')
    # 2. 校验用户名和密码是否正确(正确的用户名:admin、密码:123456)
    if username == 'admin' and password == '123456':
        # 3. 如果用户名和密码都正确, 打印“登录成功!”, 并结束程序
        print('登录成功')
        break
    else:
        # 4. 如果用户名或密码错误, 打印“用户名或密码错误!”, 提示用户继续输入用户名和密码登录
        print('用户名或密码错误! 请再次输入')
```

