#### 1.判断循环语句

前面我们学习的内容解决了一个核心问题就是变量=值，我们将了合格的变量是什么样的，合格的值是什么样的，以及各种各样值之间能做哪些变换，下面就学习一下判断和循环语句。

**重点**：编写判断语句，编写循环语句。

#### 2.判断语句

判断语句用于根据条件的真假执行不同的代码块

1. `if`语句：  
 `if`语句用于执行一段代码块，当条件为真时执行。可以与可选的`elif`和`else`语句一起使用来处理多个条件分支。  
 isRaining = True  
 if isRaining:  
 print("It's raining!")  
 else:  
 print("It's not raining.")  
  
2. `elif`语句：  
 `elif`语句用于在`if`语句中添加额外的条件分支。如果前面的条件不满足，将检查`elif`语句，并在满足条件时执行相应的代码块。  
 temperature = 25  
   
 if temperature > 30:  
 print("It's hot!")  
 elif temperature > 20:  
 print("It's warm.")  
 else:  
 print("It's cool.")  
  
3. `else`语句：  
 `else`语句用于在`if`语句中处理所有其他不满足前面条件的情况。当前面的条件都不满足时，将执行`else`语句中的代码块。  
 is\_raining = False  
   
 if is\_raining:  
 print("It's raining!")  
 else:  
 print("It's not raining.")

#### 3.循环语句

在Python中，循环语句用于重复执行一段代码块，直到满足特定条件。Python提供了两种主要的循环语句：for循环和while循环。

1. `for`循环：`for`循环用于遍历一个可迭代对象（如列表、元组、字符串等），并对其中的每个元素执行相同的操作。  
例如，下面的代码使用`for`循环计算并打印数字列表中每个数字的平方：  
numbers = [1, 2, 3, 4, 5]  
for num in numbers:  
 square = num \*\* 2  
 print(square)  
  
 输出结果：  
 1  
 4  
 9  
 16  
 25  
  
2. `while`循环：`while`循环在满足特定条件的情况下重复执行代码块，直到条件不再满足。  
例如，下面的代码使用`while`循环计算并打印从1到10的数字的平方：  
num = 1  
while num <= 10:  
 square = num \*\* 2  
 print(square)  
 num += 1  
  
 输出结果：  
 1  
 4  
 9  
 16  
 25  
 36  
 49  
 64  
 81  
 100

##### 3.1for循环

for循环是Python中用于遍历可迭代对象的循环结构。

它的语法形式如下：  
for 变量 in 可迭代对象:  
 # 循环体代码块  
  
`for`循环的执行过程是将可迭代对象中的每个元素依次赋值给变量，然后执行循环体中的代码块。当遍历完所有元素时，循环终止。  
  
下面是对`for`循环的各个部分进行详细说明：  
  
1. `变量`：在每次迭代中，将可迭代对象中的一个元素赋值给变量。这个变量会在循环体中使用。  
  
2. `可迭代对象`：一个可以被迭代遍历的对象，例如列表、元组、字符串等。可迭代对象是`for`循环的关键部分。  
  
3. 循环体代码块：在每次迭代中，执行的代码块。这是您希望在每个元素上执行的操作。  
  
以下是一些使用`for`循环的示例：  
  
1. 遍历列表：  
 fruits = ["apple", "banana", "cherry"]  
 for fruit in fruits:  
 print(fruit)  
  
 输出结果：  
 apple  
 banana  
 cherry  
  
2. 遍历字符串：  
 message = "Hello"  
 for char in message:  
 print(char)  
  
 输出结果：  
 H  
 e  
 l  
 l  
 o  
  
3. 遍历范围：  
 for num in range(1, 6):  
 print(num)  
  
  
 输出结果：  
 1  
 2  
 3  
 4  
 5  
  
4. 结合条件语句：  
 numbers = [1, 2, 3, 4, 5]  
 for num in numbers:  
 if num % 2 == 0:  
 print(f"{num}, is even")  
 else:  
 print(num, "is odd")  
  
 输出结果：  
 1 is odd  
 2 is even  
 3 is odd  
 4 is even  
 5 is odd

range函数

`range()` 函数是一个内置函数，用于生成一个整数序列。它常用于循环中，特别是在 `for` 循环中，用于指定循环的次数或迭代的范围。  
  
`range()` 函数的语法形式如下：  
range(start, stop, step)  
  
`range()` 函数接受三个参数：  
- `start`（可选）：生成整数序列的起始值。默认为 0。  
- `stop`：生成整数序列的结束值，但不包含该值。生成的序列将从 `start` 开始，逐步递增，直到小于 `stop`。  
- `step`（可选）：生成整数序列的步长（即相邻两个数之间的差）。默认为 1。  
  
`range()` 函数返回一个可迭代对象，可以通过迭代器逐个访问序列中的整数元素。  
  
  
1. 指定结束值：  
for num in range(5):  
 print(num)  
输出结果：  
0  
1  
2  
3  
4  
  
2. 指定起始值和结束值：  
for num in range(2, 8):  
 print(num)  
输出结果：  
2  
3  
4  
5  
6  
7  
  
3. 指定起始值、结束值和步长：  
for num in range(1, 10, 2):  
 print(num)  
输出结果：  
1  
3  
5  
7  
9  
  
4. 使用 `range()` 函数生成一个列表：  
numbers = list(range(1, 6))  
print(numbers)  
输出结果：  
[1, 2, 3, 4, 5]

##### 3.2while循环

while 循环是Python中的一种迭代结构，用于根据特定条件重复执行一段代码块。while 循环会在每次迭代之前检查指定的条件，只要条件为真，循环就会继续执行。

while 条件:  
 # 循环体代码块  
  
  
`条件` 是一个布尔表达式，当条件为真（即非零或非空）时，循环继续执行。一旦条件为假（即零或空），循环就会终止，程序将跳出循环并继续执行后续的代码。  
  
以下是对 `while` 循环的各个部分进行详细说明：  
  
1. `条件`：一个布尔表达式，用于控制循环的执行。如果条件为真，循环将继续执行；如果条件为假，循环将终止。  
  
2. 循环体代码块：在每次迭代中执行的代码块。这是您希望在每次迭代时执行的操作。  
  
  
1. 基本的 `while` 循环：  
 count = 0  
 while count < 5:  
 print("Count:", count)  
 count += 1  
  
 输出结果：  
 Count: 0  
 Count: 1  
 Count: 2  
 Count: 3  
 Count: 4  
  
2. 结合条件语句和控制语句：  
 num = 1  
 while num <= 10:  
 if num % 2 == 0:  
 print(num, "is even")  
 else:  
 print(num, "is odd")  
 num += 1  
  
 输出结果：  
 1 is odd  
 2 is even  
 3 is odd  
 4 is even  
 5 is odd  
 6 is even  
 7 is odd  
 8 is even  
 9 is odd  
 10 is even

1. 使用 break 和 continue 控制循环：

num = 1  
while num <= 5:  
 if num == 3:  
 num += 1  
 continue  
 if num == 5:  
 break  
 print(num)  
 num += 1  
  
输出结果：  
1  
2  
4

##### 3.3break和continue

break 和 continue 是用于控制循环行为的关键字。

1. break：当在循环中遇到 break 关键字时，循环会立即终止，即使循环条件仍然为真。程序将跳出循环并继续执行循环之后的代码。

以下是使用 break 的示例：

for num in range(1, 6):  
 if num == 4:  
 break  
 print(num)  
  
输出结果：  
1  
2  
3

在上面的示例中，当 num 的值等于 4 时，break 被执行，循环立即终止。因此，数字 4 不会被打印出来。

1. continue：当在循环中遇到 continue 关键字时，循环会跳过当前迭代的剩余代码，并开始下一次迭代。换句话说，continue 会中断当前迭代，然后继续执行下一次迭代。

以下是使用 continue 的示例：

for num in range(1, 6):  
 if num == 3:  
 continue  
 print(num)  
  
输出结果：  
1  
2  
4  
5

在上面的示例中，当 num 的值等于 3 时，continue 被执行，当前迭代的剩余代码被跳过，然后开始下一次迭代。因此，数字 3 不会被打印出来。

break 可以用于在满足特定条件时提前终止循环，而 continue 可以用于跳过某些迭代。