【白话python连载（6）】python的循环语句--变繁为简的捷径

大多数接触过的程序语言，都会使用比较经典的猜数字游戏来介绍循环的具体使用方法。本次我们介绍python的循环语句也不例外，首先来看一下通过python如何实现猜数字的小游戏。

* 从小入手-python小应用

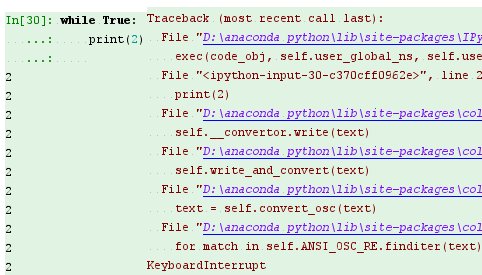
#问题描述：猜数字游戏

|  |
| --- |
| *#!/usr/bin/python # -\*- coding: utf-8 -\*-* **import** random  **def new\_guess**()**: #定义每次猜测的一个1-100之间的随机数** number **=** random.**randint**(0, 100)  *# print(number)* **return** number **def new\_game**()**:** number **= new\_guess**() #首轮游戏默认随机数为猜测数字，在指定次数内猜对重新生成。  n **=** 0  **while True:** n **+=** 1 #统计一共游戏的次数  **try:** guess\_num **=** int(input(**'请输入你要尝试的次数：'**))  **except** ValueError**:** guess\_num **=** 10  print(**'输入的次数不合法，默认最多猜10次'**)  **for** i **in** range(guess\_num)**:** guess **=** int(input(**'请输入你猜的数字：'**))  **if** guess **>** number**:** print(**'你猜的数字大了，不要灰心，继续加油！'**)  **elif** guess **<** number**:** print(**'你猜的数字小了，不要灰心，继续努力！'**)  **else:  if** i **<** 3**:** print(**'真厉害，这么快就猜对了！'**)  **else:** print(**'终于猜对了，恭喜恭喜！'**)  number **= new\_guess**() #猜对后将调用生成函数再次生成新的数字  **break  if** i **==** guess\_num**-**1**:** print(**'很遗憾！在{}次中你均未猜出'**.**format**(guess\_num))  next\_game **=** input(**'是否继续游戏？请输入 yes or no：'**)  **if** next\_game **== 'no': #根据用户输入判断是否开始新一轮游戏  break  else:** print(**'开始下一轮游戏！'**)  **continue** print(**'你一共玩了{}轮游戏哦'**.**format**(n))   **if** \_\_name\_\_ **== '\_\_main\_\_':** print(**'猜数字小游戏!'**)  **new\_game**() |
| *"D:\anaconda python\python3.6.exe" D:/vernacular/coding\_ziyi/circulate.py*  *猜数字小游戏!*  *请输入你要尝试的次数：5*  *请输入你猜的数字：3*  *你猜的数字小了，不要灰心，继续努力！*  *很遗憾！在5次中你均未猜出*  *是否继续游戏？请输入 yes or no：yes*  *开始下一轮游戏！*  *请输入你猜的数字：94*  *终于猜对了，恭喜恭喜！*  *是否继续游戏？请输入 yes or no：yes*  *开始下一轮游戏！*  *请输入你要尝试的次数：a*  *输入的次数不合法，默认最多猜10次*  *请输入你猜的数字：16*  *你猜的数字小了，不要灰心，继续努力！*  *请输入你猜的数字：17*  *真厉害，这么快就猜对了！*  *是否继续游戏？请输入 yes or no：no*  *你一共玩了3轮游戏哦*  *Process finished with exit code 0* |

* 使用python处理重复的小任务

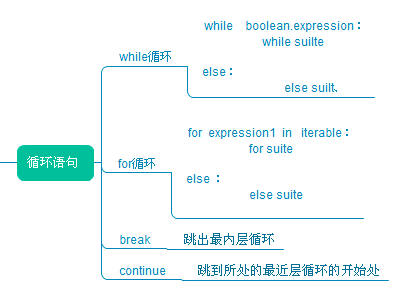
在上面的猜数字小实例之中，就使用了python循环语句两种典型的循环while和for，以及关键字的continue和break的作用。

|  |  |  |
| --- | --- | --- |
| 循环语句 | 基本表达式 | 实例 |
| while：  （循环次数不确定） | while bool\_expression:  suite  else:  suite |  |
| for：  （已知循环次数） | for expression in iterable:  suite  else:  suite |  |
| continue | 跳到所处的最近层循环的开始处。  会接着完成接下来的循环 |  |
| break | 跳出最内层循环  不会再执行break之后的所有循环操作 |  |

循环是让计算机做重复任务的有效的方法，但是如果判断条件写的不合理，如上述应用中用while True：进行循环，判断条件始终为True，如果后续没有额外的break语句，猜数字的游戏将会无限进行下去。这种循环为‘死循环’，可以用Ctrl+C退出该循环程序，或者强制结束Python进程。

总结

1. python的循环语句通常会与条件语句同时使用，进行循环开始和结束的判断处理。如果可以提前明确循环的次数建议使用for循环防止出现无限循环。
2. 灵活的应用break和continue可以很好得快控制程序功能。
3. 出现无限循环后，使用ctrl+c或者强制退出python进程，结束循环。



如果已知循环次数可以使用for 或while循环

但是不知道循环次数则使用while循环