**17Python\_02判断语句+while循环+for循环**

**一 判断语句-if语句**

**1.1 if语句语法结构**

**• 标准if条件语句的语法**

**if expression:**

**if\_suite #[swiːt]一套**

**else:**

**else\_suite**

**• 如果表达式的值非0\非空\为布尔值True, 则代码组if\_suite被执行;否则就去执行else\_suite**

**• 代码组是一个python术语,它由一条或多条语句组成,表示一个子代码块**

**1.2 if语句示例解析**

**• 只要表达式数字为非零值即为True**

**• 空字符串、空列表、空元组,空字典的值均为False**

py\_str = 'python'

if 'th' in py\_str:

print('th in python')

if -0.0:

print('表达式值为0,**不打印**')

if ' ': #有个空格

print('空格也是一个字符,为真,打印')

if []:

print('空列表为假,**不打印**')

if (1, 2):

print('非空元组为真,打印')

if {'name': 'tom'}:

print('非空字典为真,打印')

if not {}:

print('空字典为假,取反为真,打印')

if not None:

print('None表示空对象,类似mysql中的null,为假,取反为真,打印')

if [ ]:

print('空列表,为假,**不打印**')

if['']:

print('非空列表,为真,打印')

**1.3 条件表达式**

• Python 在很长的一段时间里没有条件表达式(C ? X:Y),或称**三元运算符**,因为范·罗萨姆一直拒绝加入这样的功能

• 从Python 2.5集成的语法确定为: X if C else Y

>>> x = 3

>>> y = 4

>>> bigger = x if x > y else y

>>> print(bigger)

4

**案例1:判断合法用户**

1. 创建login2.py文件;2. 提示用户输入用户名和密码

3. 获得到相关信息后,将其保存在变量中

4. 如果用户输的用户名为bob,密码为123456,则输出Login successful,否则输出Login incorrect

**import getpass**  #导入getpass模块

username = input('please input username:')

password = **getpass.getpass**('please input password:')

if username == 'bob' and password == '123456':

print('**\033[32;1m**login successful**\033[0m**')

else: #32是绿色,31是红色,前景色;40以上是背景色;\033[0m是关闭颜色

print('\033[31;1mlogin incorrect\033[0m')

**二 判断语句-扩展if语句**

**2.1 扩展if语句结构**

• 扩展if语句结构

if expression1:

if\_suite

elif expression2:

elif\_suite

else:

else\_suite

**2.2 扩展if语句示例解析**

案例2:编写判断成绩的程序

• 创建grade.py脚本,根据用户输入的成绩分档,要求

1. 如果成绩大于60分,输出“及格”

2. 如果成绩大于70分,输出“良”

3. 如果成绩大于80分,输出“好”

4. 如果成绩大于90分,输出“优秀”

5. 否则输出“你要努力了”

grade = int(input('成绩:'))

if grade >= 90 : #匹配即停止,倒着写

print('\033[31;1m优秀\033[0m')

elif grade >= 80 :

print('\033[32;1m好\033[0m')

elif grade >= 70 :

print('\033[33;1m良\033[0m')

elif grade >= 60 :

print('\033[34;1m及格\033[0m')

else:

print('\033[35;1m你要努力了\033[0m')

#或:

if 70 > grade >= 60 :

print('\033[34;1m及格\033[0m')

elif 80 > grade >= 70 :

print('\033[33;1m良\033[0m')

elif 90 > grade >= 80 :

print('\033[32;1m好\033[0m')

elif grade >= 90 :

print('\033[31;1m优秀\033[0m')

else :

print('\033[35;1m你要努力了\033[0m')

**案例3:编写石头剪刀布小游戏**

• 编写game.py,要求如下:

1. 计算机随机出拳

2. 玩家自己决定如何出拳

3. 代码尽量简化

import random #导入random模块

player = input('请你出拳(石头/剪刀/布):')

**all\_choice = ['石头', '剪刀', '布'] #定义列表**

**computer = random.choice(all\_choice) #函数传参给变量computer赋值**

**print('你出拳: %s, 计算机出拳: %s' %(player, computer))**

if player == '石头' :

if computer == '石头' :

print('平局!')

elif computer == '剪刀' :

print('你赢了!')

else :

print('你输了!')

elif player == '剪刀' :

if computer == '石头' :

print('你输了!')

elif computer == '剪刀' :

print('平局!')

else :

print('你赢了!')

else :

if computer == '石头' :

print('你赢了!')

elif computer == '剪刀' :

print('你输了!')

else :

print('平局!')

或:

import random

player = input('请你出拳(石头/剪刀/布):')

all\_choice = ['石头', '剪刀', '布']

computer = random.choice(all\_choice)

win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]

print('你出拳: %s, 计算机出拳: %s' %(player, computer))

if player == computer :

print('平局!')

elif [player, computer] in win\_list :

print('你赢了!')

else :

print('你输了!')

**三 while循环-循环语句基础**

**3.1 循环概述**

• 一组被重复执行的语句称之为循环体,能否继续重复,决定于循环的终止条件

• Python中的循环有while循环和for循环

• 循环次数未知的情况下,建议采用while循环

• 循环次数可以预知的情况下,建议采用for循环

**3.2 while循环语法结构**

• 当需要语句不断的重复执行时,可以使用while循环

**while expression:**

**while\_suite**

• 语句while\_suite会被连续不断的循环执行,直到表达式的值变成0或False

sum100 = 0

counter = 0

while counter <= 100:

sum100 += counter

counter += 1

print ("result is %d" % sum100)

**案例4: 石头剪刀布改进**

import random

all\_choice = ['石头', '剪刀', '布']

win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]

**prompt = '''(0)石头**

**(1)剪刀**

**(2)布**

**请选择(0/1/2)'''**

computer = random.choice(all\_choice)

**ind = int(input(prompt)) #将用户输入的数字字符转换为数字**

**player = all\_choice[ind] #将数字作为下标从列表中取出元素**

print('你出拳: %s, 计算机出拳: %s' % (player, computer))

if player == computer:

print('平局!')

elif [player, computer] in win\_list:

print('你赢了!')

else:

print('你输了!')

**案例5: 石头剪刀布3局2胜**

import random

pwin = 0 #人的计分板

cwin = 0 #计算机计分板

all\_choice = ['石头', '剪刀', '布']

win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]

while pwin < 2 and cwin < 2 : #人和计算机都没赢够2次

player = input('请你出拳(石头/剪刀/布):')

computer = random.choice(all\_choice)

print('你出拳: %s, 计算机出拳: %s' %(player, computer))

if player == computer :

print('平局!')

elif [player, computer] in win\_list :

print('你赢了!')

pwin += 1

else :

print('你输了!')

cwin += 1

**四 while循环-循环语句进阶**

**4.1 break语句**

**• break语句结束当前循环体,然后跳转到下条语句**

• 写程序的时候,应尽量避免重复的代码,在这种情况下可以使用while-break结构

name = input('username: ')

while name != 'tom':

name = input('username: ')

#可以替换为

while True:

name = input('username: ')

if name == 'tom':

break

**案例: 猜数程序**

• 编写guess.py,要求如下:

1. 系统随机生成50以内的数字; 2. 要求用户猜生成的数字是多少

3. 最多猜5次,猜对结束程序; 4. 如果5次全部猜错,则输出正确结果

import random

number = **random.randint(1,50)** #随机生成1~50内的整数

counter = 0

while counter < 5 :

counter += 1

answer = int(input('guess(1-50):'))

if answer > number :

print('bigger')

elif answer < number :

print('smaller')

else :

print('good')

break

**else :**  # while循环也有else语句

print('the exact number is %d' % number)

**4.2 continue语句**

• **当遇到continue语句时,程序会终止当前循环,并忽略剩余的语句,然后回到循环的顶端**

**• 然后查看循环条件,如果仍然满足循环条件,循环体内语句继续执行,否则退出循环**

**计算1~100内偶数的和:**

sum100 = 0 #创建变量用于保存累加的和

counter = 0 #创建计数器,将其值累加到sum100中

while counter < 100 :

counter += 1

if counter % 2 : #结果只有1或0,奇数时1为真,偶数时0为假

continue

sum100 += counter

print ("result is %d" % sum100)

**4.3 else语句**

• python中的while语句也支持else子句

• else子句只在循环完成后执行

• break语句也会跳过else块(见4.1案例)

**五 for循环**

**5.1 for循环语法结构**

**• python中的for接受可迭代对象(例如序列或迭代器)作为其参数,每次迭代其中一个元素**

for iter\_var in iterable:

suite\_to\_repeat

**• 与while循环一样,支持break、continue、else语句**

**• 一般情况下,循环次数未知采用while循环,循环次数已知,采用for循环**

s1 = 'hello'

nums = [10, 20, 15, 80]

names = ('tom', 'jerry')

user = {'name': 'bob', 'age': '20'}

for ch in s1 :

print(ch)

print('\*' \* 50)

for i in nums :

print(i)

print('\*' \* 50)

for name in names :

print(name)

print('\*' \* 50)

for key in user :

print(key, user[key])

h

e

l

l

o

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10

20

15

80

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tom

jerry

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name bob

age 20

**5.2 range函数**

**• for循环常与range函数一起使用**

**• range函数提供循环条件**

**• range函数的完整语法为:**

**range(start, end, step = n)**

**range只是一个对象,潜在可以生成很多数**

**若range没有给起始数字,从0 开始,不包含结束数字**

for i in range(10) :

print(i)

**将range转换成列表再打印**

print(list(range(10)))

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

**打印6到10的列表**

print(list(range(6, 10)))

[6, 7, 8, 9]

**打印1,3,5,7,9**

print(list(range(1, 10, 2)))

[1, 3, 5, 7, 9]

**打印10,9,8,7,6,5,4,3,2,1**

print(list(range(10, 0, -1)))

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

**案例6:斐波那契数列**

1. 斐波那契数列就是某一个数,总是前两个数之和,比如0,1,1,2,3,5,8

2. 使用for循环和range函数编写一个程序,计算有10个数字的斐波那契数列

3. 改进程序,要求用户输入一个数字,可以生成用户需要长度的斐波那契数列

fib = [0, 1]

for i in range(8):

fib.append(fib[-1] + fib[-2])

print(fib)

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

fib = [0, 1]

n = int(input('长度: '))

for i in range(n-2):

fib.append(fib[-1] + fib[-2])

print(fib)

**案例7:九九乘法表**

1. 创建mtable.py程序

2. 乘序运行后,可以在屏幕上打印出九九乘法表

3. 修改程序,由用户输入数字,可打印任意数字的乘法表

n = int(input('输入:'))

for i in range(1, n + 1):

for j in range(1,i + 1):

print(str(j)+'\*'+str(i)+'='+str(j\*i), '\t', end = '')

print() #换行

**5.3 列表解析**

• 它是一个非常有用、简单、而且灵活的工具,可以用来动态地创建列表

• 语法:

**[expr for iter\_var in iterable]**

• 这个语句的核心是for循环,它迭代iterable对象的所有条目

• expr应用于序列的每个成员,最后的结果值是该表达式产生的列表

>>> [5]

[5]

>>> [5 + 2] #将表达式到列表

[7]

>>> [5 + 2 for i in range(1, 11)] #循环决定表达式计算的次数

[7, 7, 7, 7, 7, 7, 7, 7, 7, 7]

>>> [5 + i for i in range(1, 11)] #在表达式中使用循环的标量

[6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

>>> [5 + i for i in range(1, 11) if i % 2 == 1]

[6, 8, 10, 12, 14]

#判断条件作为过滤条件

>>> ['192.168.1.%s' % i for i in range(3)]

['192.168.1.0', '192.168.1.1', '192.168.1.2']