

NSD Python1 DAY02

1. [案例1：判断合法用户](#)
2. [案例2：编写判断成绩的程序](#)
3. [案例3：编写石头剪刀布小游戏](#)
4. [案例4：完善石头剪刀布小游戏](#)
5. [案例5：猜数程序](#)

1 案例1：判断合法用户

1.1 问题

编写login2.py脚本，实现以下目标：

1. 提示用户输入用户名和密码
2. 将用户名和密码分别保存在变量中
3. 如果用户名为bob并且密码为123456，则输出Login successful，否则输出Login inorrect

1.2 方案

本题主要是复合的判断语句，写法有如下两种：

- 1.使用两个判断语句，先判断用户名，如果用户名正确再判断密码是否正确
- 2.在一个判断语句中，同时判断两个条件是否全部成立

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

在很多语言中，if后面的判断条件需要使用圆括号或方括号，但是python并不强制，可以直接将判断条件写在if后面，并不会产生错误。

有些时候，判断条件可能有多多个（使用and或or连接），为了更好的可读性，建议在这种环境下，将多个条件分别用圆括号括起来。

```
01. [ root@localhost day 02] # vim login2.py
02.
03.  #! /usr/bin/env python3
04.
05.  username = input('username: ')
06.  password = input('password: ')
07.
08.  if username == 'bob':
09.      if password == '123456':
10.          print('Login successful')
11.      else:
```

[Top](#)

```
12.         print('Login incorrect')
13.     else:
14.         print('Login incorrect')
```

或将上面的代码改为以下写法：

```
01. [ root@localhost day 02] # vim login2.py
02.
03.     #!/usr/bin/env python3
04.
05.     username = input('username: ')
06.     password = input('password: ')
07.
08.     if username == 'bob' and password == '123456':
09.         print('Login successful')
10.     else:
11.         print('Login incorrect')
```

步骤二：测试脚本执行

```
01. [ root@localhost day 02] # python3 login2.py
02.     username: bob
03.     password: 123456
04.     Login successful
05. [ root@localhost day 02] # python3 login2.py
06.     username: bob
07.     password: abcd
08.     Login incorrect
09. [ root@localhost day 02] # python3 login2.py
10.     username: tom
11.     password: 123456
12.     Login incorrect
```

步骤三：改进脚本

脚本程序在运行时，应该将敏感密码隐藏，不要显示在屏幕上。为了实现这个功能，可以使用getpass模块中的getpass方法。

[Top](#)

getpass可以像Linux处理密码一样，屏幕上不出现任何字符，但是用户的输入可以保存到相应的变量中。

上面的代码可以改写为：

```

01. [ root@localhost day 02] # vim login2.py
02.
03.  #! /usr/bin/env python3
04.
05.  import getpass  #调用该函数可以在命令行窗口里面无回显输入密码
06.
07.  username = input('username: ')
08.  password = getpass.getpass('password: ')
09.
10.  if username == 'bob' and password == '123456':
11.      print('\033[ 32;1mLogin successful! \033[ 0m')    #绿色粗体显示
12.  else:
13.      print('\033[ 31;1mLogin incorrect! \033[ 0m')    #红色粗体显示

```

测试脚本执行：

```

01. [ root@localhost day 02] # python3 login2.py
02.  username: bob
03.  password: 123456          #此处所填写的密码将不在屏幕上显示
04.  Login successful
05. [ root@localhost day 02] # python3 login2.py
06.  username: tom
07.  password: 123456          #此处所填写的密码将不在屏幕上显示
08.  Login incorrect!

```

2 案例2：编写判断成绩的程序

2.1 问题

编写grade.py脚本，根据用户输入的成绩分档，要求如下：

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”

2.2 方案

[Top](#)

本题需要注意的是逻辑顺序。在分支的if语句中，自顶向下逐步匹配，一旦匹配则执行相应的子语句，其他语句将不再执行。

因此，在编写代码时要注意逻辑，成绩是100分也大于60分，如果把判断较小分数的语句写在前面，那么是凡大于60分的成绩都是输出“及格”，那么只有第一个判断语句会执行，所以应该把分值更高的判断写在上面。

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
01. [ root@localhost day02] # vim grade.py
02.
03. #! /usr/bin/env python3
04. #coding: utf8                                #为了程序可以支持中文，指定UTF8编码
05.
06. score = int(input('成绩：'))
07.
08. if score >= 90:
09.     print('优秀')
10. elif score >= 80:
11.     print('好')
12. elif score >= 70:
13.     print('良')
14. elif score >= 60:
15.     print('及格')
16. else:
17.     print('你要努力了！')
```

或将上面的代码改为以下写法：

```
01. score = int(input('成绩：'))
02.
03. if score >= 60 and score < 70:
04.     print('及格')
05. elif 70 <= score < 80:
06.     print('良')
07. elif 80 <= score < 90:
08.     print('好')
09. elif score >= 90:
10.     print('优秀')
11. else:
12.     print('你要努力了！')
```

[Top](#)

步骤二：测试脚本执行

```
01. [root@localhost day 02] # python3 grade.py
02. 成绩：59
03. 你要努力了！
04. [root@localhost day 02] # python3 grade.py
05. 成绩：88
06. 好
07. [root@localhost day 02] # python3 grade.py
08. 成绩：64
09. 及格
10. [root@localhost day 02] # python3 grade.py
11. 成绩：75
12. 良
13. [root@localhost day 02] # python3 grade.py
14. 成绩：97
15. 优秀
```

3 案例3：编写石头剪刀布小游戏

3.1 问题

编写game.py脚本，实现以下目标：

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

3.2 方案

引用random模块生成0-2的随机数，提示并获取用户的整数输入值，应用if扩展语句对随机数与输入值进行对比判断，满足指定条件，输出结果

为简化代码，玩家获胜条件中用and和or两个逻辑运算符进行多个条件内容的判断，用括号来区分运算优先级，所以用户获胜条件为以下3项中任意一项：

1. 用户输入剪刀并且随机数是布
2. 用户输入石头并且随机数是剪刀
3. 用户输入布并且随机数是石头

3.3 步骤

实现此案例需要按照如下步骤进行。

[Top](#)

步骤一：编写脚本

```

01. [ root@localhost day02] # vim game.py
02. #!/usr/bin/env python3
03.
04. import random
05.
06. #1. 提示并获取用户的输入
07. player = int( input( "请输入 0剪刀 1石头 2布:" ))
08.
09. #2. 让电脑出一个随机数
10. computer = random.randint( 0,2)
11.
12. #3. 判断用户的输入,然后显示对应的结果
13. #if 玩家获胜的条件:
14. if ( player==0 and computer==2) or ( player==1 and computer==0) or ( player==2 and comp
15.     print( "赢了,,,可以去买奶粉了.....")
16. #elif 玩家平局的条件:
17. elif player==computer:
18.     print( "平局了,,,洗洗手决战到天亮....")
19. else:
20.     print( "输了,,,回家拿钱 再来....")

```

或将上面的代码改为以下写法：

引用random模块choice方法随机生成‘石头’、‘剪刀’、‘布’中任意一项，提示并获取用户的输入字符，应用if扩展语句对随机数与输入值进行对比判断，满足指定条件，输出结果问题结果

```

01. import random
02.
03. computer = random.choice(['石头', '剪刀', '布'])
04. player = input( '请出拳( 石头/剪刀/布) :')
05.
06. # print( '您出了:', player, '计算机出的是:', computer)
07. print( '您出了: %s, 计算机出的是: %s' %( player, computer))
08. if player == '石头':
09.     if computer == '石头':
10.         print( '平局')
11.     elif computer == '剪刀':
12.         print( 'You WIN!!!')
13.     else:
14.         print( 'You LOSE!!!')

```

[Top](#)

```

15.     elif player == '剪刀':
16.         if computer == '石头':
17.             print('You LOSE!!!')
18.         elif computer == '剪刀':
19.             print('平局')
20.         else:
21.             print('You WIN!!!')
22.     else:
23.         if computer == '石头':
24.             print('You WIN!!!')
25.         elif computer == '剪刀':
26.             print('You LOSE!!!')
27.         else:
28.             print('平局')

```

步骤二：测试脚本执行

```

01. [root@localhost day02] # python3 game.py
02. 请输入 0剪刀 1石头 2布:1
03. 平局了,,,洗洗手决战到天亮....
04. [root@localhost day02] # python3 game.py
05. 请输入 0剪刀 1石头 2布:0
06. 赢了,,,可以去买奶粉了.....
07. [root@localhost day02] # python3 game.py
08. 请输入 0剪刀 1石头 2布:2
09. 平局了,,,洗洗手决战到天亮....
10. [root@localhost day02] # python3 game.py
11. 请输入 0剪刀 1石头 2布:1
12. 赢了,,,可以去买奶粉了.....
13. [root@localhost day02] # python3 game.py
14. 请输入 0剪刀 1石头 2布:1
15. 输了,,,回家拿钱 再来....
16. [root@localhost day02] # python3 game.py
17. 请出拳( 石头/剪刀/布) : 石头
18. 您出了: 石头, 计算机出的是: 石头
19. 平局
20. [root@localhost day02] # python3 game.py
21. 请出拳( 石头/剪刀/布) : 剪刀
22. 您出了: 剪刀, 计算机出的是: 剪刀
23. 平局

```

[Top](#)

```

24. [root@localhost day02] # python3 game.py
25. 请出拳( 石头/剪刀/布) : 布
26. 您出了: 布, 计算机出的是: 剪刀
27. You LOSE!!!
28. [root@localhost day02] # python3 game.py
29. 请出拳( 石头/剪刀/布) : 石头
30. 您出了: 石头, 计算机出的是: 剪刀
31. You WIN!!!

```

步骤三：改进脚本

执行代码后，在终端显示中，根据提示输入‘石头、剪刀、布’对应数值，通过列表切片获取用户输入字符，引用random模块choice方法电脑随机生成‘石头’、‘剪刀’、‘布’中任意一项字符，将可赢组合放入列表中，如果随机生成电脑值与用户获取字符在可赢列表中，则为可赢组合，输出‘you win’，否则，输出‘you lose’

```

01. import random
02.
03. all_choices = ['石头', '剪刀', '布']
04. win_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]
05. prompt = '''(0) 石头
06. (1) 剪刀
07. (2) 布
08. 请选择(0/1/2) : '''
09. computer = random.choice( all_choices)
10. ind = int( input( prompt) )
11. player = all_choices[ind]
12.
13. print('您出了: %s, 计算机出的是: %s' %( player, computer))
14. if player == computer:
15.     print('\033[32;1m平局\033[0m')
16. elif [ player, computer] in win_list:
17.     print('\033[31;1mYou WIN!!! \033[0m')
18. else:
19.     print('\033[31;1mYou LOSE!!! \033[0m')

```

测试脚本执行：

```

01. [root@localhost day02] # python3 game2.py
02. (0) 石头

```

[Top](#)

03. (1) 剪刀
04. (2) 布
05. 请选择(0/1/2) : 2
06. 您出了: 布, 计算机出的是: 布
07. 平局
08. [root@localhost day 02] # python3 game2.py
09. (0) 石头
10. (1) 剪刀
11. (2) 布
12. 请选择(0/1/2) : 1
13. 您出了: 剪刀, 计算机出的是: 剪刀
14. 平局
15. [root@localhost day 02] # python3 game2.py
16. (0) 石头
17. (1) 剪刀
18. (2) 布
19. 请选择(0/1/2) : 0
20. 您出了: 石头, 计算机出的是: 石头
21. 平局
22. [root@localhost day 02] # python3 game2.py
23. (0) 石头
24. (1) 剪刀
25. (2) 布
26. 请选择(0/1/2) : 1
27. 您出了: 剪刀, 计算机出的是: 石头
28. You LOSE!!!
29. [root@localhost day 02] # python3 game2.py
30. (0) 石头
31. (1) 剪刀
32. (2) 布
33. 请选择(0/1/2) : 2
34. 您出了: 布, 计算机出的是: 剪刀
35. You LOSE!!!
36. [root@localhost day 02] # python3 game2.py
37. (0) 石头
38. (1) 剪刀
39. (2) 布
40. 请选择(0/1/2) : 1
41. 您出了: 剪刀, 计算机出的是: 石头
42. You LOSE!!!
43. [root@localhost day 02] # python3 game2.py

[Top](#)

- 44. (0) 石头
- 45. (1) 剪刀
- 46. (2) 布
- 47. 请选择(0/1/2) : 0
- 48. 您出了: 石头, 计算机出的是: 剪刀
- 49. You WIN!!!

4 案例4：完善石头剪刀布小游戏

4.1 问题

编写game2.py脚本，实现以下目标：

1. 基于上节game.py程序
2. 实现循环结构，要求游戏三局两胜

4.2 方案

用while循环语句让游戏执行3次，在判断输赢之前用if嵌套方式先判断用户输入的值是否合法，如果合法进行输赢判断，如果不合法重新执行循环语句，三次游戏结束后，即循环结束后，用if语句判断赢了几次，赢得次数大于等于2次，获得最终胜利，否则为输

此程序需要注意的部分在于：

1. 要对每次赢局结果进行记录（即赢局次数加1）
2. 每局输赢判断之后，游戏次数一定要加1，否则游戏次数将永无休止

4.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```

01. [root@localhost day 02] # vim game2.py
02. #!/usr/bin/env python3
03.
04. import random
05.
06. i = 1      #游戏次数
07. win = 0    #赢局次数
08. while i <= 3:
09.     #1 提示并获取用户的输入
10.     player = int(input("请输入 0剪刀 1石头 2布:"))
11.
12.     #2 让电脑出一个随机数
13.     computer = random.randint(0,2)
14.     #让用户输入合法

```

[Top](#)

```

15.     if player==0 or player==1 or player==2:
16.         #3. 判断用户的输入,然后显示对应的结果
17.         if ( player==0 and computer==2) or ( player==1 and computer==0) or ( player==2 and
18.             print( "第"+str(i)+"局"+"赢了")
19.             win += 1
20.         elif player==computer:
21.             print( "第"+str(i)+"局"+"平局")
22.         else:
23.             print( "第"+str(i)+"局"+"输了")
24.             i += 1
25.     else:
26.         print( "请重新输入合法数字")
27. #4. 判断最终猜拳结果：3局两胜
28. if win >= 2:
29.     print( "恭喜，你赢了！！")
30. else:
31.     print( "你输了！！")

```

步骤二：测试脚本执行

```

01. [root@localhost day02] # python3 game2.py
02. 请输入 0剪刀 1石头 2布:3
03. 请重新输入合法数字
04. 请输入 0剪刀 1石头 2布:1
05. 第1局赢了
06. 请输入 0剪刀 1石头 2布:2
07. 第2局赢了
08. 请输入 0剪刀 1石头 2布:3
09. 请重新输入合法数字
10. 请输入 0剪刀 1石头 2布:2
11. 第3局平局
12. 恭喜，你赢了！！

```

步骤三：改进脚本

```

01. import random
02.
03. all_choices = ['石头', '剪刀', '布']

```

[Top](#)

```

04. win_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]
05. prompt = ""( 0) 石头
06. ( 1) 剪刀
07. ( 2) 布
08. 请选择( 0/1/2): ""
09. cwin = 0
10. pwin = 0
11.
12. while cwin < 2 and pwin < 2:
13.     computer = random.choice( all_choices)
14.     ind = int( input( prompt) )
15.     player = all_choices[ ind]
16.
17.     print( "Your choice: %s, Computer's choice: %s" %( player, computer) )
18.     if player == computer:
19.         print( '\033[ 32; 1m平局\033[ 0m' )
20.     elif [ player, computer] in win_list:
21.         pwin += 1
22.         print( '\033[ 31; 1mYou WIN!! \033[ 0m' )
23.     else:
24.         cwin += 1
25.         print( '\033[ 31; 1mYou LOSE!! \033[ 0m' )

```

测试脚本执行：

```

01. [ root@localhost day02] # python3 game3.py
02. ( 0) 石头
03. ( 1) 剪刀
04. ( 2) 布
05. 请选择( 0/1/2): 1
06. Your choice: 剪刀, Computer's choice: 剪刀
07. 平局
08. ( 0) 石头
09. ( 1) 剪刀
10. ( 2) 布
11. 请选择( 0/1/2): 2
12. Your choice: 布, Computer's choice: 石头
13. You WIN!!
14. ( 0) 石头
15. ( 1) 剪刀

```

[Top](#)

```
16.  (2) 布
17.  请选择(0/1/2): 0
18.  Your choice: 石头, Computer's choice: 剪刀
19.  You WIN!!!
20.  [root@localhost day02] # python3 game3.py
21.  (0) 石头
22.  (1) 剪刀
23.  (2) 布
24.  请选择(0/1/2): 0
25.  Your choice: 石头, Computer's choice: 布
26.  You LOSE!!!
27.  (0) 石头
28.  (1) 剪刀
29.  (2) 布
30.  请选择(0/1/2): 1
31.  Your choice: 剪刀, Computer's choice: 石头
32.  You LOSE!!!
```

5 案例5：猜数程序

5.1 问题

编写guess.py脚本，实现以下目标：

1. 系统随机生成100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

5.2 方案

引用random模块生成1-100的随机数，用while循环语句让猜数字次数大于0，提示并获取用户输入整数值，在进行猜数字对错判断前先用if嵌套判断方式确定输入值是否合法，如果合法进行猜数字对错判断，判断结束后猜数字次数需减1，如果不合法重新进入循环，此时循环次数不减少

此程序需要注意的部分在于：

每局对错判断之后，猜数字次数一定要减1，这样猜数字次数等于0的时候，循环就结束了

5.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
01.  [root@localhost day02] # vim guess.py
02.  #!/usr/bin/env python3
03.
```

[Top](#)

```

04. import random
05.
06. secret = random.randint( 1,100)    #生成随机数
07.
08. time = 5    #猜数字的次数
09.
10. print( "------ 欢迎来到猜数字的地方，请开始----- ")
11. while time > 0:
12.     guess = int( input( "*数字区间0-100，请输入你猜的数字:" ) )
13.     print( "你输入数字是：" ,guess)
14.     if 0 <= guess < 100:
15.         if guess == secret:
16.             print( "猜对了，真厉害")
17.         else:
18.             print( "太遗憾了，你猜错了，你还有",time-1,"次机会")
19.             time -= 1
20.     else:
21.         print( "输入非法，请重新输入")
22. print( "游戏结束，正确的结果是：" ,secret)

```

步骤二：测试脚本执行

```

01. [ root@localhost day 02] # python3 guess.py
02. ----- 欢迎来到猜数字的地方，请开始 -----
03. *数字区间0-100，请输入你猜的数字:100
04. 你输入数字是： 100
05. 输入非法，请重新输入
06. *数字区间0-100，请输入你猜的数字:0
07. 你输入数字是： 0
08. 太遗憾了，你猜错了，你还有 4 次机会
09. *数字区间0-100，请输入你猜的数字:-1
10. 你输入数字是： -1
11. 输入非法，请重新输入
12. *数字区间0-100，请输入你猜的数字:12
13. 你输入数字是： 12
14. 太遗憾了，你猜错了，你还有 3 次机会
15. *数字区间0-100，请输入你猜的数字:34
16. 你输入数字是： 34
17. 太遗憾了，你猜错了，你还有 2 次机会
18. *数字区间0-100，请输入你猜的数字:56

```

[Top](#)

19. 你输入数字是：56
20. 太遗憾了，你猜错了，你还有 1 次机会
21. *数字区间0-100，请输入你猜的数字:89
22. 你输入数字是：89
23. 太遗憾了，你猜错了，你还有 0 次机会
24. 游戏结束，正确的结果是：47

步骤三：改进脚本

```
01. import random
02.
03. num = random.randint( 1, 100)
04. counter = 0
05.
06. while counter < 5:
07.     answer = int( input( 'guess the number: ' ))
08.     if answer > num:
09.         print( '猜大了')
10.     elif answer < num:
11.         print( '猜小了')
12.     else:
13.         print( '猜对了')
14.         break
15.     counter += 1
16. else: # 循环被break就不执行了，没有被break才执行
17.     print( 'the number is:', num)
```

测试脚本执行：

```
01. [ root@localhost day 02] # py thon3 guess2.py
02. 猜大了
03. guess the number: 30
04. 猜小了
05. guess the number: 50
06. 猜小了
07. guess the number: 70
08. 猜小了
09. guess the number: 78
10. 猜小了
```

[Top](#)

11. the number is: 88
12. [root@localhost day02] # python3 guess2.py
13. guess the number: 16
14. 猜小了
15. guess the number: 90
16. 猜大了
17. guess the number: 50
18. 猜大了
19. guess the number: 30
20. 猜对了

[Top](#)