[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "page_top_case)

# NSD Python1 DAY02

1. [案例1：判断合法用户](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case1)
2. [案例2：编写判断成绩的程序](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case2)
3. [案例3：编写石头剪刀布小游戏](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case3)
4. [案例4：完善石头剪刀布小游戏](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case4)
5. [案例5：猜数程序](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case5)
6. [案例6：斐波那契数列](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case6)
7. [案例7：九九乘法表](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY02/CASE/01/index.html" \l "case7)

## 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。

1. [root@localhost day02]# vim login2.py
2. #!/usr/bin/env python3
3. username = input('username: ')
4. password = input('password: ')
5. if username == 'bob':
6. if password == '123456':
7. print('Login successful')
8. else:
9. print('Login incorrect')
10. else:
11. print('Login incorrect')

或将上面的代码改为以下写法：

1. [root@localhost day02]# vim login2.py
2. #!/usr/bin/env python3
3. username = input('username: ')
4. password = input('password: ')
5. if username == 'bob' and password == '123456':
6. print('Login successful')
7. else:
8. print('Login incorrect')

步骤二：测试脚本执行

1. [root@localhost day02]# python3 login2.py
2. username: bob
3. password: 123456
4. Login successful
5. [root@localhost day02]# python3 login2.py
6. username: bob
7. password: abcd
8. Login incorrect
9. [root@localhost day02]# 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，应该将敏感的密码隐藏，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。

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：

1. [root@localhost day02]# vim login2.py
2. #!/usr/bin/env python3
3. import getpass    #调用该函数可以在命令行窗口里面无回显输入密码
4. username = input('username: ')
5. password = getpass.getpass('password: ')
6. if username == 'bob' and password == '123456':
7. print('\033[32;1mLogin successful!\033[0m')        #绿色粗体显示
8. else:
9. print('\033[31;1mLogin incorrect!\033[0m')        #红色粗体显示

测试脚本执行：

1. [root@localhost day02]# python3 login2.py
2. username: bob
3. password: 123456 #此处所填写的密码将不在屏幕上显示
4. Login successful
5. [root@localhost day02]# python3 login2.py
6. username: tom
7. password: 123456 #此处所填写的密码将不在屏幕上显示
8. 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. 如果成绩大分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

本题需要注意的是逻辑顺序。在多分支的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

1. [root@localhost day02]# vim grade.py
2. #!/usr/bin/env python3
3. #coding: utf8 #为了程序可以支持中文，指定UTF8编码
4. score = int(input('成绩：'))
5. if score >= 90:
6. print('优秀')
7. elif score >= 80:
8. print('好')
9. elif score >= 70:
10. print('良')
11. elif score >= 60:
12. print('及格')
13. else:
14. print('你要努力了！')

或将上面的代码改为以下写法：

1. score = int(input('成绩：'))
2. if score >= 60 and score < 70:
3. print('及格')
4. elif 70 <= score < 80:
5. print('良')
6. elif 80 <= score < 90:
7. print('好')
8. elif score >= 90:
9. print('优秀')
10. else:
11. print('你要努力了！')

步骤二：测试脚本执行

1. [root@localhost day02]# python3 grade.py
2. 成绩： 59
3. 你要努力了！
4. [root@localhost day02]# python3 grade.py
5. 成绩： 88
6. 好
7. [root@localhost day02]# python3 grade.py
8. 成绩： 64
9. 及格
10. [root@localhost day02]# python3 grade.py
11. 成绩： 75
12. 良
13. [root@localhost day02]# 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。

步骤一：编写脚本

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

1. import random
2. computer = random.choice(['石头', '剪刀', '布'])
3. player = input('请出拳(石头/剪刀/布)：')
4. # print('您出了:', player, '计算机出的是:', computer)
5. print('您出了: %s, 计算机出的是: %s' % (player, computer))
6. if player == '石头':
7. if computer == '石头':
8. print('平局')
9. elif computer == '剪刀':
10. print('You WIN!!!')
11. else:
12. print('You LOSE!!!')
13. elif player == '剪刀':
14. if computer == '石头':
15. print('You LOSE!!!')
16. elif computer == '剪刀':
17. print('平局')
18. else:
19. print('You WIN!!!')
20. else:
21. if computer == '石头':
22. print('You WIN!!!')
23. elif computer == '剪刀':
24. print('You LOSE!!!')
25. else:
26. print('平局')

步骤二：测试脚本执行

1. [root@localhost day02]# python3 game.py
2. 请输入 0剪刀 1石头 2布:1
3. 平局了,,,洗洗手决战到天亮....
4. [root@localhost day02]# python3 game.py
5. 请输入 0剪刀 1石头 2布:0
6. 赢了,,,,可以去买奶粉了.....
7. [root@localhost day02]# python3 game.py
8. 请输入 0剪刀 1石头 2布:2
9. 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
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’

1. import random
2. all\_choices = ['石头', '剪刀', '布']
3. win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]
4. prompt = '''(0) 石头
5. (1) 剪刀
6. (2) 布
7. 请选择(0/1/2)：'''
8. computer = random.choice(all\_choices)
9. ind = int(input(prompt))
10. player = all\_choices[ind]
11. print('您出了: %s, 计算机出的是: %s' % (player, computer))
12. if player == computer:
13. print('\033[32;1m平局\033[0m')
14. elif [player, computer] in win\_list:
15. print('\033[31;1mYou WIN!!!\033[0m')
16. else:
17. print('\033[31;1mYou LOSE!!!\033[0m')

测试脚本执行：

1. [root@localhost day02]# python3 game2.py
2. (0) 石头
3. (1) 剪刀
4. (2) 布
5. 请选择(0/1/2)：2
6. 您出了: 布, 计算机出的是: 布
7. 平局
8. [root@localhost day02]# python3 game2.py
9. (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 day02]# 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 day02]# 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 day02]# 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 day02]# 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 day02]# python3 game2.py
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

1. [root@localhost day02]# vim game2.py
2. #!/usr/bin/env python3
3. import random
4. i = 1            #游戏次数
5. win = 0        #赢局次数
6. while i <= 3:
7. #1. 提示并获取用户的输入
8. player = int(input("请输入 0剪刀 1石头 2布:"))
10. #2. 让电脑出一个随机数
11. computer = random.randint(0,2)
12. #让用户输入合法
13. if player==0 or player==1 or player==2:
14. #3. 判断用户的输入,然后显示对应的结果
15. if (player==0 and computer==2) or (player==1 and computer==0) or (player==2 and computer==1):
16. print("第"+str(i)+"局"+"赢了")
17. win += 1
18. elif player==computer:
19. print("第"+str(i)+"局"+"平局")
20. else:
21. print("第"+str(i)+"局"+"输了")
22. i += 1
23. else:
24. print("请重新输入合法数字")
25. #4. 判断最终猜拳结果：3局两胜
26. if win >= 2:
27. print("恭喜，你赢了！！")
28. else:
29. print("你输了！！")

步骤二：测试脚本执行

1. [root@localhost day02]# python3 game2.py
2. 请输入 0剪刀 1石头 2布:3
3. 请重新输入合法数字
4. 请输入 0剪刀 1石头 2布:1
5. 第1局赢了
6. 请输入 0剪刀 1石头 2布:2
7. 第2局赢了
8. 请输入 0剪刀 1石头 2布:3
9. 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

1. import random
2. all\_choices = ['石头', '剪刀', '布']
3. win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]
4. prompt = """(0) 石头
5. (1) 剪刀
6. (2) 布
7. 请选择(0/1/2): """
8. cwin = 0
9. pwin = 0
10. while cwin < 2 and pwin < 2:
11. computer = random.choice(all\_choices)
12. ind = int(input(prompt))
13. player = all\_choices[ind]
14. print("Your choice: %s, Computer's choice: %s" % (player, computer))
15. if player == computer:
16. print('\033[32;1m平局\033[0m')
17. elif [player, computer] in win\_list:
18. pwin += 1
19. print('\033[31;1mYou WIN!!!\033[0m')
20. else:
21. cwin += 1
22. print('\033[31;1mYou LOSE!!!\033[0m')

测试脚本执行：

1. [root@localhost day02]# python3 game3.py
2. (0) 石头
3. (1) 剪刀
4. (2) 布
5. 请选择(0/1/2): 1
6. Your choice: 剪刀, Computer's choice: 剪刀
7. 平局
8. (0) 石头
9. (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
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

1. [root@localhost day02]# vim guess.py
2. #!/usr/bin/env python3
3. import random
4. secret = random.randint(1,100)        #生成随机数
5. time = 5        #猜数字的次数
6. print("---------欢迎来到猜数字的地方，请开始---------")
7. while time > 0:
8. guess = int(input("\*数字区间0-100，请输入你猜的数字:"))
9. print("你输入数字是：",guess)
10. if 0 <= guess < 100:
11. if guess == secret:
12. print("猜对了，真厉害")
13. else:
14. print("太遗憾了，你猜错了，你还有",time-1,"次机会")
15. time -= 1
16. else:
17. print("输入非法，请重新输入")
18. print("游戏结束，正确的结果是：",secret)

步骤二：测试脚本执行

1. [root@localhost day02]# python3 guess.py
2. ---------欢迎来到猜数字的地方，请开始---------
3. \*数字区间0-100，请输入你猜的数字:100
4. 你输入数字是： 100
5. 输入非法，请重新输入
6. \*数字区间0-100，请输入你猜的数字:0
7. 你输入数字是： 0
8. 太遗憾了，你猜错了，你还有 4 次机会
9. \*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
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

1. import random
2. num = random.randint(1, 100)
3. counter = 0
4. while counter < 5:
5. answer = int(input('guess the number: '))
6. if answer > num:
7. print('猜大了')
8. elif answer < num:
9. print('猜小了')
10. else:
11. print('猜对了')
12. break
13. counter += 1
14. else: # 循环被break就不执行了，没有被break才执行
15. print('the number is:', num)

测试脚本执行：

1. [root@localhost day02]# python3 guess2.py
2. 猜大了
3. guess the number: 30
4. 猜小了
5. guess the number: 50
6. 猜小了
7. guess the number: 70
8. 猜小了
9. guess the number: 78
10. 猜小了
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

## 6 案例6：斐波那契数列

### 6.1 问题

编写fib.py脚本，实现以下目标：

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

### 6.2 方案

本题主要是for循环语句，写法有如下两种：

1.输入一个变量确定列表长度，for循环用内置函数range确定循环次数，利用切片方法将列表fib最后两数之和追加到列表中，每循环一次追加一个值

2.for循环用内置函数range确定循环次数，每循环一次执行：将变量b的值赋值给变量a，并且将a b之和赋值给b，此时，a的新值是前一个b的值，b的新值是前面a b之和，让a成为数列中的值

### 6.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

1. [root@localhost day03]# vim fib.py
2. #!/usr/bin/env python3
3. a, b = 0, 1
4. for i in range(10):
5. print(a)
6. a, b = b, a + b

或将上面的代码改为以下写法：

1. [root@localhost day03]# vim fib2.py
2. #!/usr/bin/env python3
3. fib = [0, 1]
4. l = int(input("数列长度: "))
5. for i in range(l - 2):
6. fib.append(fib[-1] + fib[-2])
7. print(fib)

或将上面的代码改为以下写法：

1. [root@localhost day03]# vim fib\_func.py
2. #!/usr/bin/env python3
3. def gen\_fib(l):
4. fib = [0, 1]
5. for i in range(l - len(fib)):
6. fib.append(fib[-1] + fib[-2])
7. return fib # 返回列表，不返回变量fib
8. a = gen\_fib(10)
9. print(a)
10. print('-' \* 50)
11. n = int(input("length: "))
12. print(gen\_fib(n)) # 不会把变量n传入，是把n代表的值赋值给形参

步骤二：测试脚本执行

1. [root@localhost day03]# python3 fib.py
2. 0
3. 1
4. 1
5. 2
6. 3
7. 5
8. 8
9. 13
10. 21
11. 34
12. [root@localhost day03]# python3 fib2.py
13. 数列长度: 9
14. [0, 1, 1, 2, 3, 5, 8, 13, 21]
15. [root@localhost day03]# python3 fib\_func.py
16. [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
17. --------------------------------------------------
18. length: 9
19. [0, 1, 1, 2, 3, 5, 8, 13, 21]

## 7 案例7：九九乘法表

### 7.1 问题

创建mtable.py脚本，要求如下：

1. 程序运行后，可以在屏幕上打印出九九乘法表
2. 修改程序，由用户输入数字，可打印任意数字的乘法表

### 7.2 方案

本题主要用for循环双层嵌套方式编写脚本，需要注意的是：

1.外层for循环用内置函数range，将1~9范围内的每个数字，依次装入自定义变量i中，此时，变量i被循环赋值9次

2.内层for循环将1~变量i范围内的每个数字，依次装入变量j中，此时变量j被循环赋值i次，此时外层for循环每循环一次，内层for循环i次

3.内层for循环range取值节点应是外层变量i加1，这样内层变量j可以取到i的值

4.程序最后print()相当于回车，每完成一次外部循环，执行回车，作用在于美化执行结果

### 7.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

1. [root@localhost day03]# vim mtable.py
2. #!/usr/bin/env python3
3. for i in range(1, 10):         # [0, 1, 2]
4. for j in range(1, i+1):         # i->0:[0], i->1: [0, 1], i->2: [0, 1, 2]
5. print('%sX%s=%s' % (j, i, i\*j), end=' ')
6. print()
7. [root@localhost day03]# vim mtable.py
8. #!/usr/bin/env python3
9. i=1
10. while i<=9:
11. j=1
12. while j<=i:
13. print("%d\*%d=%d" % (j,i,j\*i),end=" ")
14. j+=1
15. print("")
16. i+=1

步骤二：测试脚本执行

1. [root@localhost day03]# python3 mtable.py
2. 1X1=1
3. 1X2=2 2X2=4
4. 1X3=3 2X3=6 3X3=9
5. 1X4=4 2X4=8 3X4=12 4X4=16
6. 1X5=5 2X5=10 3X5=15 4X5=20 5X5=25
7. 1X6=6 2X6=12 3X6=18 4X6=24 5X6=30 6X6=36
8. 1X7=7 2X7=14 3X7=21 4X7=28 5X7=35 6X7=42 7X7=49
9. 1X8=8 2X8=16 3X8=24 4X8=32 5X8=40 6X8=48 7X8=56 8X8=64
10. 1X9=9 2X9=18 3X9=27 4X9=36 5X9=45 6X9=54 7X9=63 8X9=72 9X9=81