NSD Python2 DAY01

1 案例1：模拟用户登陆信息系统

1.1 问题

编写login.py脚本，实现以下目标：

1. 支持新用户注册，新用户名和密码注册到字典中
2. 支持老用户登陆，用户名和密码正确提示登陆成功
3. 主程序通过循环询问进行何种操作，根据用户的选择，执行注册或是登陆操作

1.2 方案

创建空字典存储用户名、密码，用三个函数分别实现用户注册、用户登录以及判断调用函数这三个方法，完成模拟用户登录：

1.调用show_menu()函数后，利用while循环交互端输出提示，请用户input0/1/2任意数值，如果输入的值不是0/1/2，打印选择无效请重新输入并重新开始循环，如果输入的值是2，停止整个循环，如果输入的值是0/1/2通过字典键值对关联关系，调用相对应函数

2.如果输入的值是0，字典cmds中0键对应的值是register，register ()调用注册函数，函数利用if方法判断输入用户名是否存在，如果用户名在字典中，输出用户名已存在，否则用户输入密码，并将用户名与密码以键值对形式放入字典中

3.如上，如果输入的值是1，调用登录函数login()，利用if方法判断输入的用户名的对应的密码是否和字典中存储用户对应密码相同，如果不同显示登录失败，否则登录成功，此函数中导入getpass模块使用方法，作用是输入密码不可见。

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

```
[root@localhost day05]# vim login.py
#! /usr/bin/env python3

import getpass
```

Top

```
05.
06.     userdb = {}
07.
08.     def register():
09.         username = input('username: ')
10.         if username in userdb:
11.             print('\033[31;1m%s already exists.\033[0m' %username)
12.         else:
13.             password = input('password: ')
14.             userdb[username] = password
15.
16.     def login():
17.         username = input('username: ')
18.         password = getpass.getpass('password: ')
19.         # if username not in userdb or userdb['username'] != password:
20.         if userdb.get(username) != password:
21.             print('\033[31;1mLogin incorrect\033[0m')
22.         else:
23.             print('\033[32;1mLogin successful\033[0m')
24.
25.     def show_menu():
26.         prompt = """(0) register
27.     (1) login
28.     (2) quit
29.     Please input your choice(0/1/2): """
30.         cmds = {'0': register, '1': login}
31.
32.         while True:
33.             choice = input(prompt).strip()[0]
34.             if choice not in '012':
35.                 print('Invalid choice. Try again.')
36.                 continue
37.
38.             if choice == '2':
39.                 break
40.
41.             cmds[choice]()
42.
43.     if __name__ == '__main__':
44.         show_menu()
```

Top

步骤二：测试脚本执行

```
[root@localhost day05]# python3 login.py
(0) register
(1) login
(2) quit
Please input your choice(0/1/2): 0
username: a
password: 123
(0) register
(1) login
(2) quit
Please input your choice(0/1/2): 1
username: a
password:
Login successful
(0) register
(1) login
(2) quit
Please input your choice(0/1/2): 1
username: b
password:
Login incorrect
(0) register
(1) login
(2) quit
Please input your choice(0/1/2): 2
```

2 案例2：编写unix2dos的程序

2.1 问题

创建unix2dos.py脚本，要求如下：

1. Windows文本文件的行结束标志是\r\n
2. 类unix文本文件的行结束标志是\n
3. 编写程序，将unix文本文件格式转换为windows文本文件的格式

2.2 方案

更改新文件格式，利用复制文件方式，将原文件内容写入新文件

Top

将上传的实参（即unix文件名）转化为windows文本格式（即后缀增加.txt），用with方法打开原文件，用with方法以写方式打开新文件，用for循环遍历原文件将遍历结果写入新文件，如果

上诉三方面判断都结束，将符合标识符命名规则字符返回，将以上所有功能封，如下两点需要注意：

1.调用unix2dos ()函数时上传文件名实参，是利用sys.argv[]参数，在命令行调用的时候由系统传递给程序，这个变量其实是一个List列表，用于保存命令行上的参数，argv[0] 一般是“被调用的脚本文件名或全路径”，argv[1]和以后就是传入的系统命令参数

2.遍历原文件内容时， 用rstrip() 删除原文件unix文件行结束标志，增加新文件行结束标记后，再将内容写入到新文件

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

```
[root@localhost day05]# vim unix2dos.py

#! /usr/bin/env python3

import sys

def unix2dos(fname):
    dst_fname = fname + '.txt'
    with open(fname) as src_fobj:
        with open(dst_fname, 'w') as dst_fobj:
            for line in src_fobj:
                line = line.rstrip('\r\n') + '\r\n'
                dst_fobj.write(line)


if __name__ == '__main__':
    unix2dos(sys.argv[1])  # python3 unix2dos.py unix2dos.py
```

步骤二：测试脚本执行

```
[root@localhost day05]# ls
unix2dos.py
[root@localhost day05]# python3 unix2dos.py unix2dos.py
[root@localhost day04]# ls
unix2dos.py  unix2dos.py.txt
```

Top

3

4 案例3：编写类进度条程序

4.1 问题

创建railway.py文件，实现以下目标：

1. 在屏幕上打印20个#号
2. 符号@从20个#号穿过
3. 当@符号到达尾部，再从头开始

4.2 方案

利用while循环方法依次打印1个@和19个#，每循环一次，@所在位置后移一位，利用\r后内容覆盖前面内容，并利用程序休眠时间差，达到动态效果，需要注意的是：

1.\r表示将输出的内容返回到第一个指针，后面的内容会覆盖前面的内容

2.sys.stdout.flush()这句代码的意思是刷新输出，让循环结果依次显示，而不是一次性显示

3.time模块中的sleep方法让程序休眠

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

```
[root@localhost day05] #vim railway.py
#! /usr/bin/env python3

import time
import sys

l = 19
counter = 0
print('#' * (l + 1), end='')


while True:
    sys.stdout.flush()
    time.sleep(0.2)
    print('\r%s@%s' %('#' * counter, '#' * (l - counter)), end='')
    counter += 1
    if counter > l:
        counter = 0
```

Top

步骤二：测试脚本执行

```
[root@localhost day05]# python3 railway.py
####@###############
[root@localhost day05]# python3 railway.py
##########@########^Z
[3]+  已停止               python3 railway.py
```

5 案例4：简化除法判断

5.1 问题

创建mydiv.py脚本，要求如下：

1. 提示用户输入一个数字作为除数
2. 如果用户按下Ctrl+C或Ctrl+D则退出程序
3. 如果用户输入非数字字符，提示用户应该输入数字
4. 如果用户输入0，提示用户0不能作为除数

5.2 方案

首先，执行try子句（在关键字try和关键字except之间的语句），输入数字，让这个数字被100整除，

1.如果没有异常发生，忽略except子句，try子句执行后，执行else子句和finally子句，最后执行 try 语句之后的代码结束整个程序。

2.如果在执行try子句的过程中发生了异常，异常的类型和 except 之后的名称相符，那么对应的except子句将被执行。然后执行finally子句，最后执行 try 语句之后的代码结束整个程序。

需要注意的是：允许用户中断这个程序（使用Ctrl+C或Ctrl+D方法）。用户中断的信息会引发KeyboardInterrupt 和EOFError 这两种异常。

一个 try 语句可能包含多个except子句，分别来处理不同的特定的异常。最多只有一个分支会被执行。

处理程序将只针对对应的try子句中的异常进行处理，而不是其他的 try 的处理程序中的异常。

try except 语句只有一个可选的else子句，使用这个子句，必须放在所有的except子句之后。这个子句将在try子句没有发生任何异常的时候执行。

finally子句是无论异常是否发生，是否捕捉都会执行的一段代码，使用finally可以保证文件总是能正常的关闭

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

```
[root@localhost day05]# vim mydiv.py
#! /usr/bin/env python3

try:
```

Top

```
    num = int( input( "number: ") )
    result = 100 / num
except ValueError:
    print( '请输入数字')
except ZeroDivisionError:
    print( '不允许使用0')
except ( KeyboardInterrupt, EOFError):
    print( '\nBye-bye')
else:
    print( result)  #不发生异常才执行的语句
finally:
    print( 'Done')  #不管异常是否发生都要执行的语句

print( 'end of program')

#不是必须把所有的语句写全，常用的有try-except和try-finally组合
```

步骤二：测试脚本执行

```
[root@localhost day05]# python3 mydiv.py
number: 0
不允许使用0
Done
end of program
[root@localhost day05]# python3 mydiv.py
number: nighao
请输入数字
Done
end of program
[root@localhost day05]# python3 mydiv.py
number: 3
33.333333333333336
Done
end of program
[root@localhost day05]# python3 mydiv.py
number: 55^C
Bye-bye
Done
end of program
[root@localhost day05]# python3 mydiv.py
```

Top

```
22.  number:
23.  Bye-bye
24.  Done
25.  end of program
```

6

7 案例5：自定义异常

7.1 问题

创建myerror.py脚本，要求如下：

1. 编写第一个函数，函数接收姓名和年龄，如果年龄不在1到120之间，产生ValueError异常
2. 编写第二个函数，函数接收姓名和年龄，如果年龄不在1到120之间，产生断言异常

7.2 方案

两个函数，分别有引发异常及断言异常的功能：

1.当set_age()函数调用名字与年龄两个实参时，如果年龄在0-120范围内，打印“bob is 25 years old”，如果年龄在0-120范围外，利用raise 语句抛出一个指定的异常

2.当set_age2()函数调用名字与年龄两个实参时，如果年龄在0-120范围内，表达式为true，打印“bob is 20 years old”，如果年龄在0-120范围外，表达式为Flase，利用assert 断言语句抛出一个指定的异常

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

```
01.  [root@localhost day05]# vim myerror.py
02.  #! /usr/bin/env python3
03.
04.  def set_age(name, age):
05.      if not 0 < age < 120:
06.          raise ValueError("age out of range.")
07.      print("%s is %s years old" %(name, age))
08.
09.  def set_age2(name, age):
10.      assert 0 < age < 120, 'age out of range.'
11.      print("%s is %s years old" %(name, age))
12.
13.  if __name__ == '__main__':
14.      set_age('bob', 25)
15.      set_age2('bob', 20)
```

Top

步骤二：测试脚本执行

```
[root@localhost day05]# python3 myerror.py
bob is 25 years old
bob is 20 years old
[root@localhost day05]# python3 myerror.py
Traceback (most recent call last):
  File "myerror.py", line 11, in <module>
    set_age('bob', 125)
  File "myerror.py", line 3, in set_age
    raise ValueError("age out of range.")
ValueError: age out of range.
[root@localhost day05]# python3 myerror.py
bob is 25 years old
Traceback (most recent call last):
  File "myerror.py", line 12, in <module>
    set_age2('bob', 120)
  File "myerror.py", line 7, in set_age2
    assert 0 < age < 100, 'age out of range.'
AssertionError: age out of range.
```

8

9 案例6：操作文件系统

9.1 问题

创建os_module.py脚本，熟悉os模块操作,要求如下：

1. 切换到/tmp目录
2. 创建example目录
3. 切换到/tmp/example目录
4. 创建test文件，并写入字符串foo bar
5. 列出/tmp/exaple目录内容
6. 打印test文件内容
7. 反向操作，把test文件以及example目录删除

9.2 方案

用os方法查看用户当前所在位置，切换到指定目录，创建example目录，切换到创建目录下，以读写方式打开并创建一个新文件，将指定内容写入新文件中，列出目录下有指定目录下有哪些文件，指定从开始位置读取指定文件字符串并打印出来，关闭打开文件，并删除文件，删除目录。

Top

注意：读取打印文件内容时，要将字节转化为字符串读取出来。

9.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
[root@localhost day05]# vim os_module.py
#! /usr/bin/env python3
import os

#1)  切换到/tmp目录
os.getcwd()        #'/root/python代码/os'
os.chdir("/tmp")
os.getcwd()        #'/tmp'
#2)  创建example目录
os.mkdir("example")
#3)  切换到/tmp/example目录
os.chdir("/tmp/example")
os.getcwd()        #'/tmp/example'
#4)  创建test文件，并写入字符串foo bar
f=os.open("test.txt",os.O_RDWR|os.O_CREAT)      #以读写方式打开/创建并打开一个新
os.write(f,b"foo bar nihao")

#5)  列出/tmp/exaple目录内容
os.listdir("/tmp/example")    #['test.txt']
#6)  打印test文件内容
os.lseek(f,0,0)       #指定从开始位置读取字符串
str=os.read(f,100)
str = bytes.decode(str)
print("读取的字符是：",str)
os.close(f)
#7)  反向操作，把test文件以及example目录删除
os.remove("/tmp/example/test.txt")
os.removedirs("/tmp/example")
```

步骤二：测试脚本执行

```
[root@localhost day05]# python3 os_module.py
读取的字符是： foo bar
```

Top

10

11 案例7：记账程序

11.1 问题

创建account.py脚本，要求如下：

1. 假设在记账时，有一万元钱
2. 无论是开销还是收入都要进行记账
3. 记账内容包括时间、金额和说明等
4. 记账数据要求永久存储

11.2 方案

创建4个函数，分别实现记录开销、记录收入、查询收支、判断函数调用的四个方法，导入时间模块获取时间，导入os模块判断文件是否存在，导入pickle模块用来python特有类型与数据类型转换：

1.调用show_menu()函数后，先判断记录余额文件是否存在，如果不存在创建文件并写入余额，如果存在，利用while循环在交互端输出提示，请用户input0/1/2/3任意数值，如果输入的值不是0/1/2/3，打印输入值无效请重新输入并重新开始循环，如果输入的值是3，停止整个循环，如果输入的值是0/1/2通过字典键值对关联关系，调用相对应函数

2.如果输入的值是0，字典cmds中0键对应的值是spend_money，调用spend_money ()记录开销函数，让此函数实现获取当前系统日期、输入开销金额、输入开销备注信息、以二进制读方式打开记录余额文件计算本次开销后余额，以写方式打开记录余额文件将计算后开销余额写入文件，以追加方式打开记账文件，将日期、开销、备注、余额写入追加入记账文件最后

3.如果输入的值是1，字典cmds中0键对应的值是save_money，调用save_money ()记录收入函数，让此函数实现获取当前系统日期、输入收入金额、输入收入备注信息、以二进制读方式打开记录余额文件计算本次收入后余额，以写方式打开记录余额文件将计算后收入余额写入文件，以追加方式打开记账文件，将日期、开销、备注、余额写入追加入记账文件最后

4.如果输入的值是2，调用查询收支函数query ()，以二进制读方式打开记账文件，利用for循环遍历文件中数据，打印出来，打开记录余额文件读取余额并打印。

需要注意的是：为确保代码可以正常执行，while循环利用try except语句处理异常，优先匹配特殊异常，让用户按下Ctrl+C或Ctrl+D可以退出程序，遇到索引错误可以结束当次循环，重新开始选择选项。

将记录余额文件以及记账文件作为参数传入函数中

11.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
[root@localhost day05]# vim account.py
#! /usr/bin/env python3

#日期      开销      收入      余额      备注
Top

import time
```

```
07.  import os
08.  import pickle as p
09.
10.
11.  def spend_money(record, wallet):
12.      date = time.strftime('%Y-%m-%d')
13.      amount = int(input('金额: '))
14.      comment = input('备注: ')
15.      with open(wallet, 'rb') as fobj:
16.  #load从数据文件中读取数据，并转换为Python的数据结构
17.          balance = p.load(fobj) - amount
18.      with open(wallet, 'wb') as fobj:
19.          #dump将数据通过特殊形式转换为只有python语言认识的字符串，并写入文件
20.          p.dump(balance, fobj)
21.
22.      with open(record, 'a') as fobj:
23.          fobj.write(
24.              "%-15s%-8s%-8s%-10s%-20s\n" %
25.              (date, amount, 'n/a', balance, comment)
26.          )
27.
28.  def save_money(record, wallet):
29.      date = time.strftime('%Y-%m-%d')
30.      amount = int(input('金额: '))
31.      comment = input('备注: ')
32.      with open(wallet, 'rb') as fobj:
33.          balance = p.load(fobj) + amount
34.      with open(wallet, 'wb') as fobj:
35.          p.dump(balance, fobj)
36.
37.      with open(record, 'a') as fobj:
38.          fobj.write(
39.              "%-15s%-8s%-8s%-10s%-20s\n" %
40.              (date, 'n/a', amount, balance, comment)
41.          )
42.
43.  def query(record, wallet):
44.      with open(record) as fobj:
45.          for line in fobj:
46.              print(line, end='')
47.
```

Top

```
48.         with open(wallet, 'rb') as fobj:
49.             #load从数据文件中读取数据，并转换为Python的数据结构
50.             balance = p.load(fobj)
51.
52.         print('当前余额: %s' %balance)
53.
54.     def show_menu():
55.         prompt = """(0) 记录开销
56.     (1) 记录收入
57.     (2) 查询收支记录
58.     (3) 退出
59.     请选择(0/1/2/3): """
60.         cmds = {'0': spend_money, '1': save_money, '2': query}
61.         record = 'record.txt'   #记帐
62.         wallet = 'wallet.data'  #记录余额
63.
64.         if not os.path.exists(wallet):        #判断文件是否存在
65.             with open(wallet, 'wb') as fobj:
66.                 p.dump(10000, fobj)
67.
68.         while True:
69.             try:
70.                 choice = input(prompt).strip()[0]
71.             except IndexError:
72.                 continue
73.             except (KeyboardInterrupt, EOFError):
74.                 print('\nBye-bye')
75.                 choice = '3'
76.
77.             if choice not in '0123':
78.                 print('无效输入，请重试')
79.                 continue
80.
81.             if choice == '3':
82.                 break
83.
84.             cmds[choice](record, wallet)
85.
86.     if __name__ == '__main__':
87.         show_menu()
```

Top

步骤二：测试脚本执行

```
[root@localhost day05]# python3 account.py
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
(3) 退出
请选择(0/1/2/3): 0
金额: 2000
备注: huafei
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
(3) 退出
请选择(0/1/2/3): 1
金额: 1000
备注: shouru
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
(3) 退出
请选择(0/1/2/3): 2
2018-04-25    2000    n/a    28890    huafei
2018-04-25    n/a    1000    29890    shouru
当前余额: 29890
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
(3) 退出
请选择(0/1/2/3): 3
[root@localhost day05]# python3 account.py
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
(3) 退出
请选择(0/1/2/3): ^C
Bye-bye
[root@localhost day05]# python3 account.py
(0) 记录开销
(1) 记录收入
(2) 查询收支记录
```

Top

```
40.     (3) 退出
41.     请选择(0/1/2/3):
42.     Bye-bye
```

Top