NSD Python1 DAY04

1 案例1：创建文件

1.1 问题

编写mktxtfile.py脚本，实现以下目标：

1. 编写一个程序，要求用户输入文件名
2. 如果文件已存在，要求用户重新输入
3. 提示用户输入数据，每行数据先写到列表中
4. 将列表数据写入到用户输入的文件名中

1.2 方案

用三个函数分别实现文件名获取、文件内容获取、将获取到的文件内容写入get_fname()函数获取的文件中 这三个方法，最终调用三个函数，完成文件创建：

1.获取文件名函数get_fname()：利用while语句循环判断文件名是否存在，input文件名，如果不存在，循环停止，返回用户输入的文件名，如果存在，提示已存在，重新进入循环，直至文件名不存在为止，返回文件名用户输入的文件名

2.文件内容获取函数get_contents()：创建空列表存储获取到的数据，利用while语句让用户循环输入数据，如果输入的数据是end，循环停止，返回列表中内容，如果输入的数据不是end，将输入的数据追加到列表结尾，返回列表中内容

3.wfile()函数：用with语句将获取到的文件以写方式打开，这样打开代码块结束后文件会自动关闭，将get_contents()函数返回内容写入到已打开文件中

4.最终当用户cat文件名时，可以看到写入结果

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 day04]# vim mktxtfile.py
#! /usr/bin/env python3

import os

def get_fname():
    while True:
```

Top

```
08.             filename = input('请输入文件名：')
09.             if not os.path.exists(filename):
10.                 break
11.             print('%s 已存在，请重试。' %filename)
12.
13.         return filename
14.
15.
16.     def get_contents():
17.         contents = []
18.
19.         print('请输入内容，结束请输入end。')
20.         while True:
21.             line = input('> ')
22.             if line == 'end':
23.                 break
24.             contents.append(line)
25.
26.         return contents
27.
28.
29.     def wfile(fname, contents):
30.         with open(fname, 'w') as fobj:
31.             fobj.writelines(contents)
32.
33.
34.     if __name__ == '__main__':
35.         fname = get_fname()
36.         contents = get_contents()
37.         contents = ['%s\n' %line for line in contents]
38.         wfile(fname, contents)
```

步骤二：测试脚本执行

```
01.     [root@localhost day04]# ls
02.     adduser.py     format_str2.py  list_method.py  mylist.py      string_op.py
03.     checkid.py     format_str.py   mkseq.py        randpass2.py
04.     fmtoutput.py   get_val.py      mktxtfile.py    seq_func.py
05.     [root@localhost day04]# python3 mktxtfile.py
06.     请输入文件名：passwd
```

Top

```
请输入内容，结束请输入end。
> nihao,welcom
> woshi
> end
[root@localhost day04]# python3 mktxtfile.py
请输入文件名：mkseq.py
mkseq.py 已存在，请重试。
请输入文件名：randpass.py
请输入内容，结束请输入end。
> my name
> end
[root@localhost day04]# cat passwd
nihao,welcom
woshi
[root@localhost day04]# cat randpass.py
my name
[root@localhost day04]# ls
adduser.py    format_str2.py  list_method.py  mylist.py     randpass.py
checkid.py    format_str.py   mkseq.py        passwd        seq_func.py
fmtoutput.py  get_val.py      mktxtfile.py    randpass2.py  string_op.py
```

2 案例2：检查标识符

2.1 问题

创建checkid.py脚本，要求如下：

1. 程序接受用户输入
2. 判断用户输入的标识符是否合法
3. 用户输入的标识符不能使用关键字
4. 有不合法字符，需要指明第几个字符不合法

2.2 方案

本题主要利用标识符命名规则从三方面判断用户输入标识符是否合法，

首先，如果用户输入的第一个字符（用切片方式拿出idt第一个字符）不是以大小写字母或下划线开头，返回'第一个字符不合法'

接下来，利用for循环逐个判断其他字符是否合法，这里的判断范围除大小写字母、下划线外增加了0-9数字，如果其他字符不在判断范围之内，返回'第几个字符非法'

最后，判断idt是否是关键字，如果是返回'idt是关键字，不能作为自定义的标识符'

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，将以上所有功能封装入函数，调用函数即可，需要注意的是：

Top

1.导入String模块，其中ascii_letters是生成所有字母，从a-z和A-Z，digits是生成所有数字0-9

2.导入keyword模块，iskeyword(idt)是用来查看某一个字符串是否是关键字

3.enumerate()函数是python的内置函数，enumerate(idt[1:])最终会返回参数的索引和值，利用索引值输出是第几个字符不合法

4.标识符的命名规则有三项，以大小写字母或下划线开头，可包括字母、下划线和数字，如 ‘and’ ‘if’ ‘import’ 等关键字不可为标识符

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 day04]# vim checkid.py

#! /usr/bin/env python3

import string
import keyword

first_chs = string.ascii_letters + '_'
all_chs = first_chs + string.digits

def check_id(idt):    # abc@123
    if idt[0] not in first_chs:
        return '第一个字符不合法'

    for ind, ch in enumerate(idt[1:]):  # bc@123 [(0, b), (1, c)...]
        if ch not in all_chs:
            return '第%s个字符%s非法' %(ind+2, ch)

    if keyword.iskeyword(idt):
        return '%s是关键字，不能作为自定义的标识符' %idt

    return '%s是合法的标识符' %idt


if __name__ == '__main__':
    idt = input('请输入待检查的标识符：')
    print(check_id(idt))
```

Top

步骤二：测试脚本执行

```
[root@localhost day05]# python3 checkid.py
请输入待检查的标识符：abc@123
第4个字符@非法
[root@localhost day04]# python3 checkid.py
请输入待检查的标识符：bc@123
第3个字符@非法
[root@localhost day04]# python3 checkid.py
请输入待检查的标识符：and
and是关键字，不能作为自定义的标识符
[root@localhost day04]# python3 checkid.py
请输入待检查的标识符：_Ance
_Ance是合法的标识符
[root@localhost day04]# python3 checkid.py
请输入待检查的标识符：nice_en*- ni
第8个字符*非法
```

3 案例3：创建用户

3.1 问题

创建adduser.py文件，实现以下目标：

1. 编写一个程序，实现创建用户的功能
2. 提示用户输入用户名
3. 随机生成8位密码
4. 创建用户并设置密码
5. 将用户相关信息写入指定文件

3.2 方案

创建add_user()函数，让函数具有创建用户、创建密码、将用户密码写入到指定文件三种方法，因此为函数设置3个参数，分别是用户名、密码及用户名密码存放文件，最终通过函数调用上传实参的方式，完成用户创建

1.利用subprocess.call函数运行用户创建命令

2.subprocess.call函数运行密码设置命令

3.用with语句将指定的文件以追加模式打开，这样打开代码块结束后文件会自动关闭，将用户密码用指定格式写入指定文件

4.调用add_user()函数时上传的用户名实参，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

3.3 步骤

Top

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

将randpass文件的代码以模块形式导入以下代码中,直接调用gen_pass()函数获取返回值（即获取随机生成的密码）：

```
[root@localhost day04]# vim adduser.py
#! /usr/bin/env python3

import sys
import subprocess
from randpass import gen_pass

def add_user(username, password, fname):
    info = """user information:
username: %s
password: %s
"""
    subprocess.call('useradd %s' %username, shell=True)
    subprocess.call(
        'echo %s | passwd --stdin %s' %(password, username),
        shell=True
    )

    with open(fname, 'a') as fobj:
        fobj.write(info %(username, password))

if __name__ == '__main__':
    username = sys.argv[1]
    password = gen_pass()
    fname = '/tmp/users.txt'
    add_user(username, password, fname)
```

步骤二：测试脚本执行

```
[root@localhost day04]# python3 adduser.py b c d
更改用户 b 的密码 。
passwd：所有的身份验证令牌已经成功更新。
[root@localhost day04]# python3 adduser.py a c d
useradd：用户“a”已存在
更改用户 a 的密码 。
passwd：所有的身份验证令牌已经成功更新。
[root@localhost day04]# cat /tmp/users.txt
```

Top

```
09.     user information:
10.     username: a
11.     password: hD31SmTS
12.     user information:
13.     username: b
14.     password: DztS7ycn
15.     user information:
16.     username: a
17.     password: f2iH0Znt
```

4 案例4：格式化输出

4.1 问题

创建fmtoutput.py脚本，要求如下：

1. 提示用户输入（多行）数据
2. 假定屏幕的宽度为50，用户输入的多行数据如图-1所示（文本内容居中）：

图-1

4.2 方案

利用for循环方式遍历获取到的用户输入数据列表，将用户输入的每一条数据依次遍历出来

通过format()方法，把遍历得到的字符串当作一个模版，通过传入的参数进行格式化。这个用来格式化的模版使用大括号({,})作为特殊字符，其中^代表居中对齐、48代表宽度。

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

将mktxtfile文件的代码以模块形式导入以下代码中,直接调用get_contents ()函数获取返回值（即获取用户输入数据列表）：

```
01.     [root@localhost day04]# vim fmtoutput.py
02.     #! /usr/bin/env python3
03.     from mktxtfile import get_contents
04.
05.     width = 48
06.
07.     contents = get_contents()
```

Top

```
08.     print('+%s+' %('*' * 48))
09.     for line in contents:
10.         print('+{:^48}+'.format(line))
11.     print('+%s+' %('*' * 48))
```

步骤二：测试脚本执行

```
01.     [root@localhost day04]# python3 fmtoutput.py
02.     请输入内容，结束请输入end。
03.     > nihao
04.     > my name zhangzhigang
05.     > bye
06.     > end
07.     +------------------------------------------------+
08.     +                     nihao                      +
09.     +              my name zhangzhigang              +
10.     +                     bye                        +
11.     +------------------------------------------------+
```

5 案例5：用列表构建栈结构

5.1 问题

创建stack.py脚本，要求如下：

1. 栈是一个后进先出的结构
2. 编写一个程序，用列表实现栈结构
3. 需要支持压栈、出栈、查询功能

5.2 方案

创建空列表存储数据，创建4个函数，分别实现压栈、出栈、查询以及判断函数调用的方法。

此程序需要注意的是堆栈的结构特点，先进后出，后进先出：

1.调用show_menu()函数后，利用while循环交互端输出提示，请用户input0/1/2/3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push_it，push_it()调用压栈函数，压栈函数利用stack.append()方法将输入数据追加到列表结尾

3.如上，如果输入的值是1，调用出栈函数pop_it()，出栈函数如果stack列表中有数据，弹出列表最后一个元素（根据堆栈结构特点stack.pop()中参数为空），如果stack列表没有数据，输出空列表

Top

4.如果输入的值是2，调用查询函数view_it()，显示当前列表

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

让输出的文字带颜色：\033[31;1m高亮度红色字体、\033[31;1m高亮度绿色字体、\033[0m关闭所有属性

```
[root@localhost day04]# vim stack.py
#! /usr/bin/env python3

stack = []
def push_it():
    item = input('item to push: ')
    stack.append(item)

def pop_it():
    if stack:
        print("\033[31;1mPopped %s\033[0m" % stack.pop())
    else:
        print('\033[31;1mEmpty stack\033[0m')

def view_it():
    print("\033[32;1m%s\033[0m" % stack)

def show_menu():
    prompt = """(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): """

    cmds = {'0': push_it, '1': pop_it, '2': view_it}

    while True:
        # strip() 方法用于移除字符串头尾指定的字符（默认为空格）
        choice = input(prompt).strip()[0]
        if choice not in '0123':
            print('Invalid input. Try again.')
            continue    #结束本次循环

        if choice == '3':
            break       #结束整个循环
```

Top

```
36.
37.             cmds[ choice] ( )    # push_it( )  pop_it( )
38.             # if choice == '0':
39.             #     push_it( )
40.             # elif choice == '1':
41.             #     pop_it( )
42.             # elif choice == '2':
43.             #     view_it( )
44.
45.
46.     if __name__ == '__main__':
47.         show_menu( )
```

步骤二：测试脚本执行

```
01.  [ root@localhost day04] # python3 stack.py
02.  ( 0)  push_it
03.  ( 1)  pop_it
04.  ( 2)  view_it
05.  ( 3)  quit
06.  Please input your choice( 0/1/2/3) : 6
07.  Invalid input. Try again.
08.  ( 0)  push_it
09.  ( 1)  pop_it
10.  ( 2)  view_it
11.  ( 3)  quit
12.  Please input your choice( 0/1/2/3) : 0
13.  item to push: nihao
14.  ( 0)  push_it
15.  ( 1)  pop_it
16.  ( 2)  view_it
17.  ( 3)  quit
18.  Please input your choice( 0/1/2/3) : 1
19.  Popped nihao
20.  ( 0)  push_it
21.  ( 1)  pop_it
22.  ( 2)  view_it
23.  ( 3)  quit
24.  Please input your choice( 0/1/2/3) : 2
25.  [ ]
```

Top

```
(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): 0
item to push: a
(0) push_it
(1) pop_it
(2) view_it
Please input your choice(0/1/2/3): 0
item to push: b
(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): 0
item to push: c
(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): 1
Popped c
(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): 2
['a', 'b']
(0) push_it
(1) pop_it
(2) view_it
(3) quit
Please input your choice(0/1/2/3): 3
(3) quit
```

Top