NSD Python1 DAY01

1 案例1：准备python开发环境

1.1 问题

1. 下载最新版本的python3
2. 下载pycharm社区版
3. 安装python3，使其支持Tab键补全
4. 配置pycharm，使其符合自己的习惯

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：下载最新版python3

首先去python官网下载python3的源码包，网址：https://www.python.org/

进去之后点击导航栏的Downloads，也可以鼠标放到Downloads上弹出菜单选择Source code，表示源码包，这里选择最新版本3.6.4，这里选择第一个下载即可，下载的就是源码包：Python-3.6.4.tar.gz，下载好之后上传到linux系统，准备安装，如图-1所示：

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64, not Itanium processors
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64, not Itanium processors
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64, not Itanium processors
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	

图-1

步骤二：Linux下安装python3

1)python安装之前需要一些必要的模块，如果没有这些模块后来使用会出现一些问题，输入以下命令提前预装依赖包：

Top

```
01. [root@localhost ~]# yum install -y gcc gcc-c++ zlib-devel openssl-devel readline-devel lib
```

释放文件：

```
[root@localhost python]# tar -xzf Python-3.6.4.tar.gz
```

2)进入Python-3.6.4目录：

```
[root@localhost python]# cd Python-3.6.4
[root@localhost Python-3.6.4]#ls        #此时Python-3.6.4文件夹中没有makefile文件
```

3)配置安装目录：

configure是用来进行用户个性配置的工具，--prefix是说软件安装目录设置在哪里，=/usr/local就是你给出的安装目录

```
[root@localhost Python-3.6.4]# ./configure --prefix=/usr/local
[root@localhost Python-3.6.4]# ls        #此时Python-3.6.4文件夹中生成了makefile文件
aclocal.m4     Doc             Makefile        PCbuild         python-config.py
build          Grammar         Makefile.pre    Programs        python-gdb.py
config.guess   Include         Makefile.pre.in pybuilddir.txt  README.rst
config.log     install-sh      Misc            pyconfig.h      setup.py
config.status  Lib             Modules         pyconfig.h.in   Tools
config.sub     libpython3.6m.a Objects         python
configure      LICENSE         Parser          Python
configure.ac   Mac             PC              python-config
```

4)接下来编译源码：

```
[root@localhost Python-3.6.4]# make
```

5)执行安装：

```
[root@localhost Python-3.6.4]# make install
```

Top

整个过程大约5-10分钟，安装成功

步骤三：下载并安装Pycharm社区版

网址：https://www.jetbrains.com/pycharm/download，这里选择下图红框下载即可，下载好之后上传到linux系统，准备安装，如图-2所示：

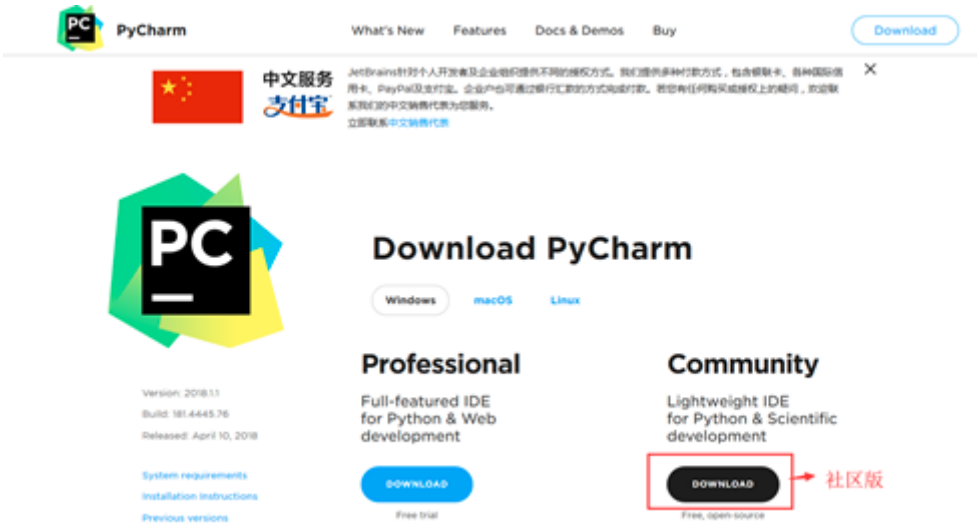


图-2

2)释放文件：

```
[root@localhost ~]# tar -xzf pycharm-community-2018.1.1.tar.gz
```

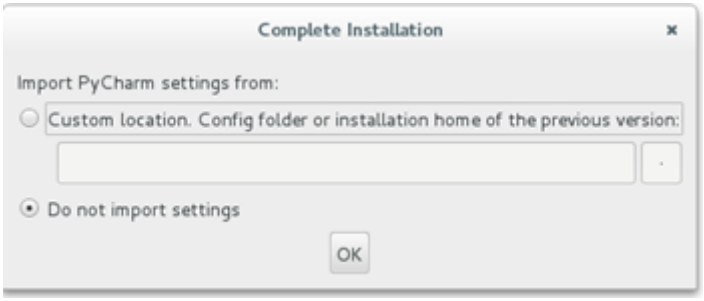
3)运行下面的命令进入PyCharm 目录：

```
[root@localhost pycharm-community-2018.1.1]# cd pycharm-community-2018.1.1/bin
```

4)通过运行下面的命令来运行PyCharm进入图形化安装界面：

```
[root@localhost bin]# sh pycharm.sh &
```

5)Pycharm打开后，如果你需要导入之前安装版本的设置的话，可以选择第一个选项，如果没有的话，选择(Do not import settings)默认不导入设置，点击/同意，就可以进入pycharm进行配置，如图-3所示：



Top

图-3

6)激活Pycharm：在弹出的激活窗口中，选择“License serveer”输入激活服务器地址“http://127.0.0.1:1017”，之后点击‘Activate’，完成pycharm激活，如图-4所示：

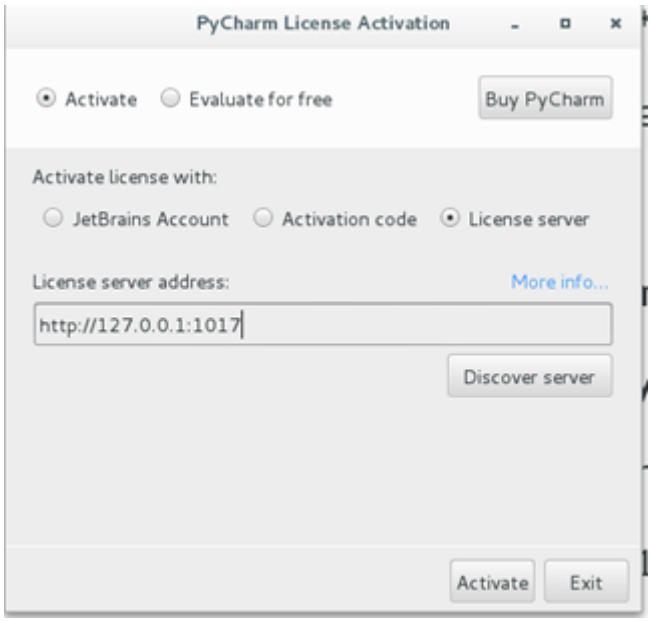


图-4

7)启动完成进入欢迎界面，如图-5所示：

图-5

2 案例2：配置git

2.1 问题

1. 安装git版本控制软件
2. 设置用户信息，如用户名、email等
3. 设置默认编辑器为vim
4. 查看用户配置

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：安装git版本控制软件

Top

```
[root@localhost ~]# yum install -y git
已安装:
  git.x86_64 0:1.8.3.1-11.el7

作为依赖被安装:
  perl-Error.noarch 1:0.17020-2.el7          perl-Git.noarch 0:1.8.3.1-11.el7
  perl-TermReadKey.x86_64 0:2.30-20.el7

完毕！
[root@localhost ~]# git –version       #查看版本
git version 1.8.3.1
```

步骤二：设置用户信息

Git 提供了一个叫做 git config 的工具，专门用来配置或读取相应的工作环境变量。

```
[root@localhost ~]# git config --global user.name "Mr.Zhang"
[root@localhost ~]# git config --global user.email zhangzg@tedu.cn
```

步骤三：设置默认编译器为vim

```
[root@localhost ~]# git config --global core.editor vim
```

步骤四：查看配置

```
[root@localhost ~]# git config --list
user.name=Mr.Zhang
user.email=zhangzg@tedu.cn
core.editor=vim
```

3 案例3：git本地操作

3.1 问题

1. 创建devops目录
2. 为devops创建git仓库
3. 新建文件hello.py，并将文件初始化到仓库中
4. 修改hello.py并将其更新到仓库

Top

5. 从他库中删除hello.py

3.2 方案

Git 使用 git init 命令来初始化一个 Git 仓库，Git 的很多命令都需要在 Git 的仓库中运行，所以git init是使用 Git 的第一个命令。添加文件第一步使用git add是将文件添加进暂存区，第二部 git commit提交更改，实际上将暂存区的所有内容提交到仓库。

3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建目录初始化

此时创建了一个空仓库，可以发现当前目录下有一个隐藏的目录.git，此目录为Git来跟踪管理版本库，建议不要修改内部文件，以免Git仓库遭到破坏。

```
[root@localhost ~]# mkdir devops
[root@localhost ~]# cd devops/
[root@localhost devops]# git init        #通过命令把目录变成Git可以管理的仓库
初始化空的 Git 版本库于 /root/devops/.git/
[root@localhost devops]# git init devops
初始化空的 Git 版本库于 /root/devops/devops/.git/
[root@localhost devops]# ls -a
. .. devops .git
```

步骤二：新建文件hello.py，并将文件初始化到仓库中

1)添加指定文件hello.py到暂存区

```
[root@localhost devops]# echo 'print( "hello world! ") ' > hello.py
[root@localhost devops]# git add hello.py        #将文件添加到暂存区
[root@localhost devops]# git status        #查看状态
# 位于分支 master
#
# 初始提交
#
# 要提交的变更：
#   (使用 "git rm --cached <file>..." 撤出暂存区)
#
#   新文件：    hello.py
```

Top

2)将暂存区文件初始化到仓库中

```
[root@localhost devops]# git commit -m "初始化仓库"      #把暂存区所有内容提交到分
[master 8e6e22a] 初始化仓库
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
[root@localhost devops]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

步骤三：修改hello.py并将其更新到仓库

```
[root@localhost devops]# echo 'print("done.")' >> hello.py
[root@localhost devops]# git commit -am "向hello.py添加新行"
[master 1ca03d5] 向hello.py添加新行
 1 file changed, 1 insertion(+)
```

步骤四：从库中删除hello.py文件

要从 Git 中移除某个文件，就必须要从已跟踪文件清单中移除

```
[root@localhost devops]# git ls-files       #查看版本库中文件
hello.py
niha.py
[root@localhost devops]# git rm hello.py
rm 'hello.py'
[root@localhost devops]# git commit -m '删除hello.py'
[master a37ff34] 删除hello.py
 1 file changed, 2 deletions(-)
 delete mode 100644 hello.py
```

4 案例4：使用自建gitlab服务器

4.1 问题

1. 通过docker搭建gitlab服务器
2. 新建群组devops
3. 新建项目core_py
4. 新建用户，他/她在devops组中是主程序员
5. 新用户上传版本库到gitlab
6. 熟悉git远程操作方法

Top

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：通过docker搭建gitlab服务器

1)从ftp://172.40.50.116/pub/docker/images/处获取gitlab_zh.tar镜像文件，导入中文版gitlab镜像

```
[root@localhost devops]# docker load < /path/to/gitlab_zh.tar
a94e0d5a7c40: Loading layer [==================================================>]
88888b9b1b5b: Loading layer [==================================================>]
52f389ea437e: Loading layer [==================================================>]
52a7ea2bb533: Loading layer [==================================================>]
db584c622b50: Loading layer [==================================================>]
62786ff6a243: Loading layer [==================================================>]
71bc04f4b7c7: Loading layer [==================================================>]
26e083d332d8: Loading layer [==================================================>]
2c02e58e96b8: Loading layer [==================================================>]
589c7a23de2a: Loading layer [==================================================>]
44474d2cdcd1: Loading layer [==================================================>]
41c94e16b901: Loading layer [==================================================>]
04cafa6a1534: Loading layer [==================================================>]
Loaded image: gitlab_zh:latest
```

2)将物理主机ssh端口改为2022后，输入如下命令起动容器：

```
[root@localhost devops]# docker run -d -h gitlab --name gitlab -p 443:443 -p 80:80 -p 2
b9dc65e0def51a4d09d2a597b2b929490e972a34f3de993439d2f7cc22039b77
此时端口成功启动
```

步骤二：新建群组

1)在浏览器地址栏中输入启动容器ip地址即可登录GitLab的界面，第一次登录使用的用户名为root，首次登录会强制用户修改密码。密码修改成功后，输入新密码进行登录，如图-6所示：

Top

图-6

2)进入网站后点击菜单栏-工具图标打开管理区域，创建群组，使用群组管理项目和成员，如图-7、图-8所示：

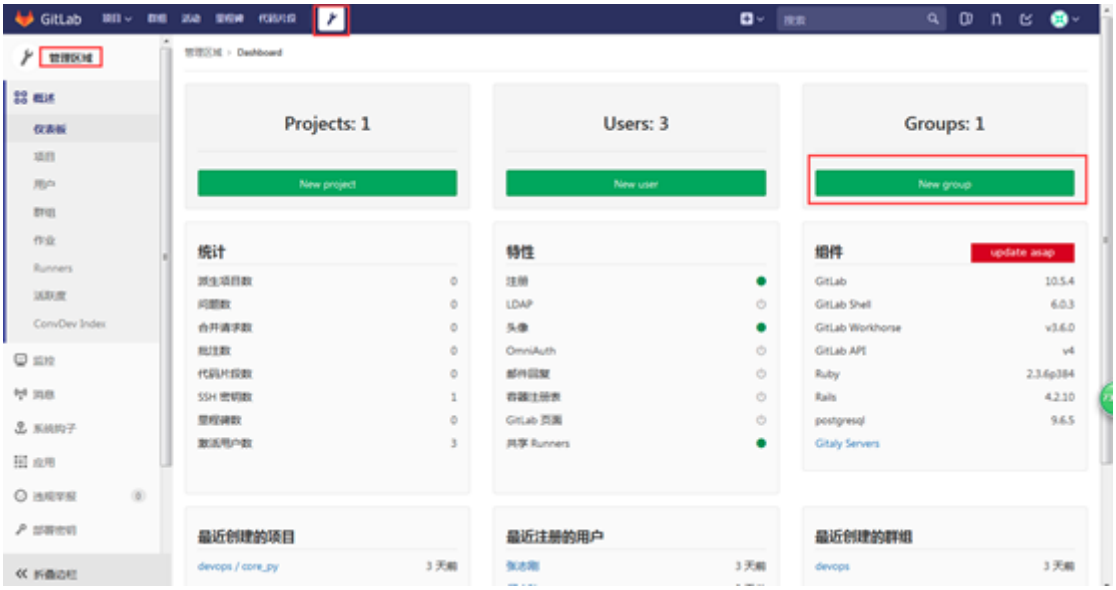
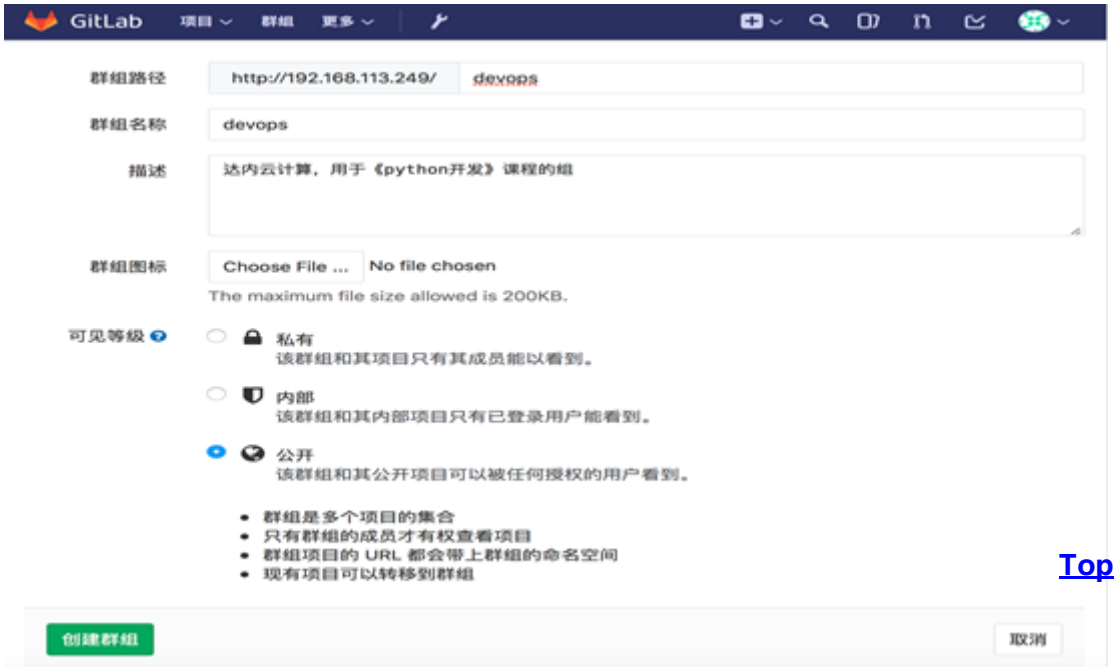


图-7



Top

图-8

步骤三：在Gitlab主页中新建一个项目

显示如图-9所示：

图-9

步骤四：在Gitlab主页中新建一个用户

1)在Gitlab主页中新建一个用户，如图-10所示：

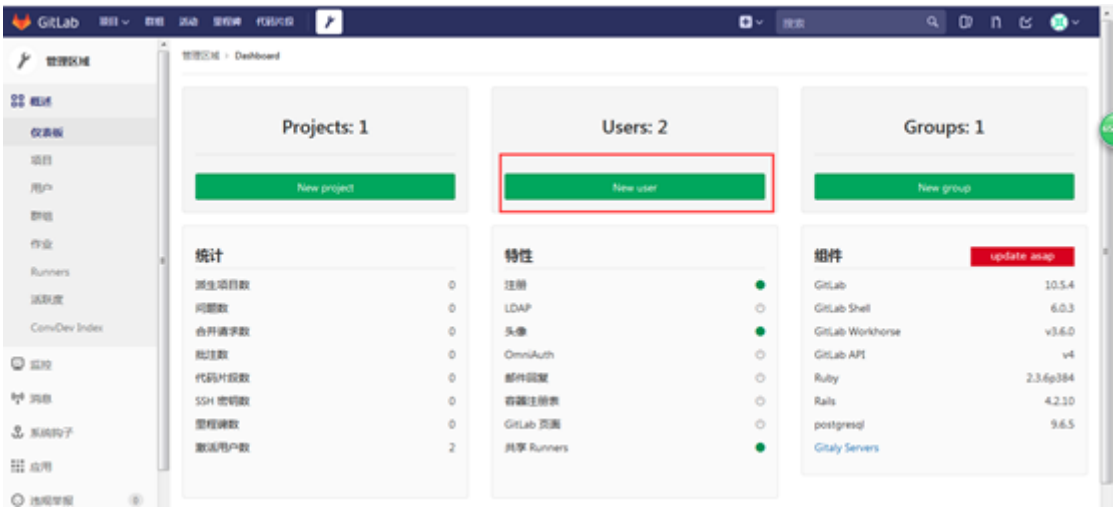


图-10

2)设置账号基本信息，其他均可为默认，如图-11所示：

Top

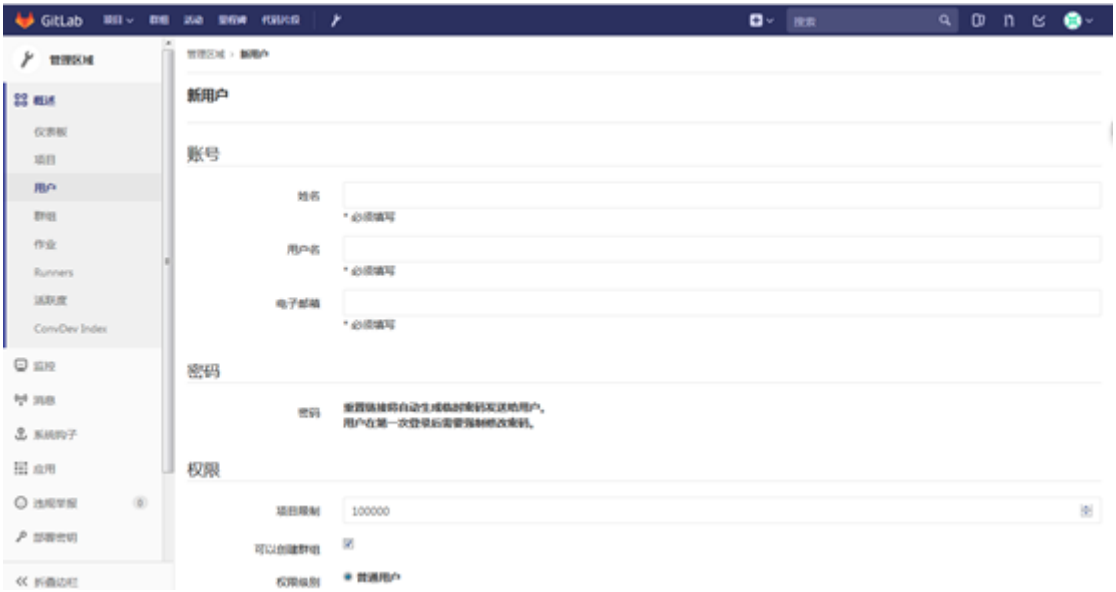


图-11

3)创建用户后，再次编辑可设置密码，如图-12所示：

图-12

4)root用户将新用户加入组中，点击devops进入群组，设置管理权限，在群组中添加成员并设置新成员为“主程序员”，如图-13、图-14、图-15所示：

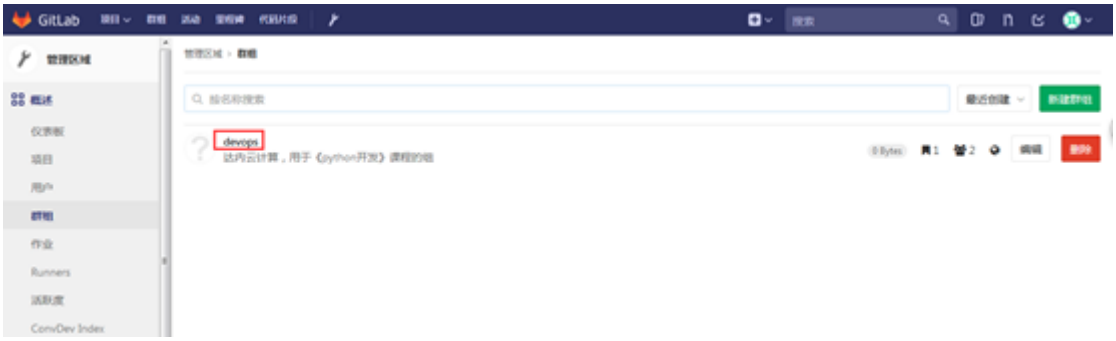


图-13

Top

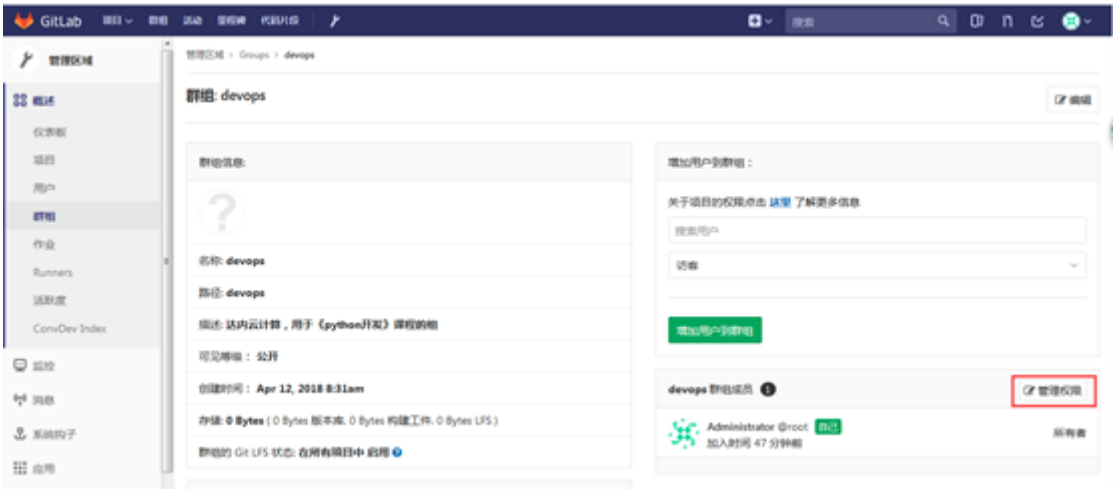


图-14

图-15

5)在终端中输入ssh-keygen命令，一路enter，可以生成缺省的rsa方式的sshkey，将/root/.ssh/id_rsa.pub中生成的rsa公钥内容拷到gitlab中，如图-16所示，进入设置页面对ssh进行配置：

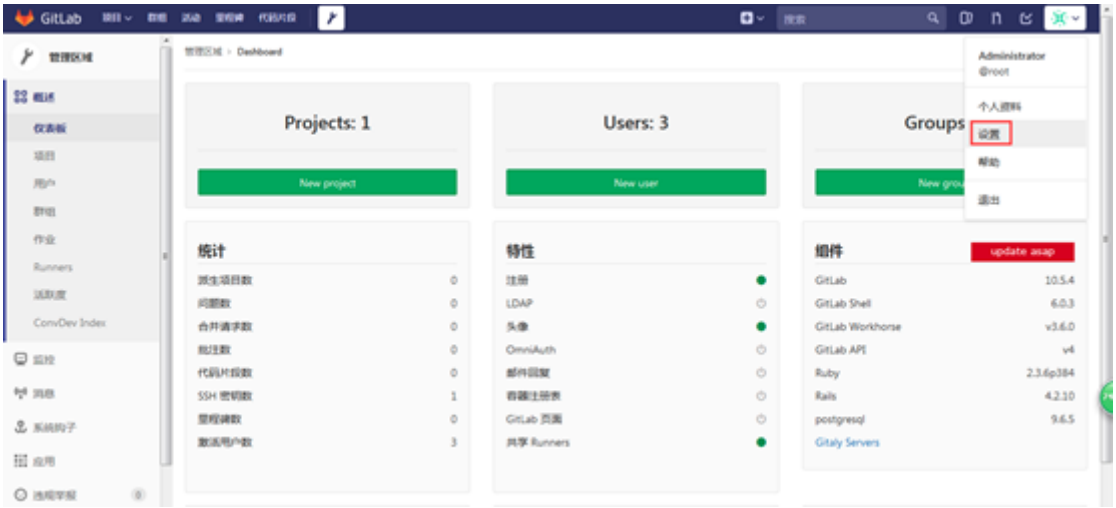


图-16

1. SSH是一种安全协议，在你的电脑与GitLab服务器进行通信时，我们使用SSH密钥（SSH Keys）认证的方式来保证通信安全。你可以在网络上搜索到关于SSH密钥的更多介绍；下面我们重点讲解如何创建 SSH密钥，并将密钥中的公钥添加到GitLab，以便我们通过SSH协议来访问Git仓库。
2. 显示如图-17所示：

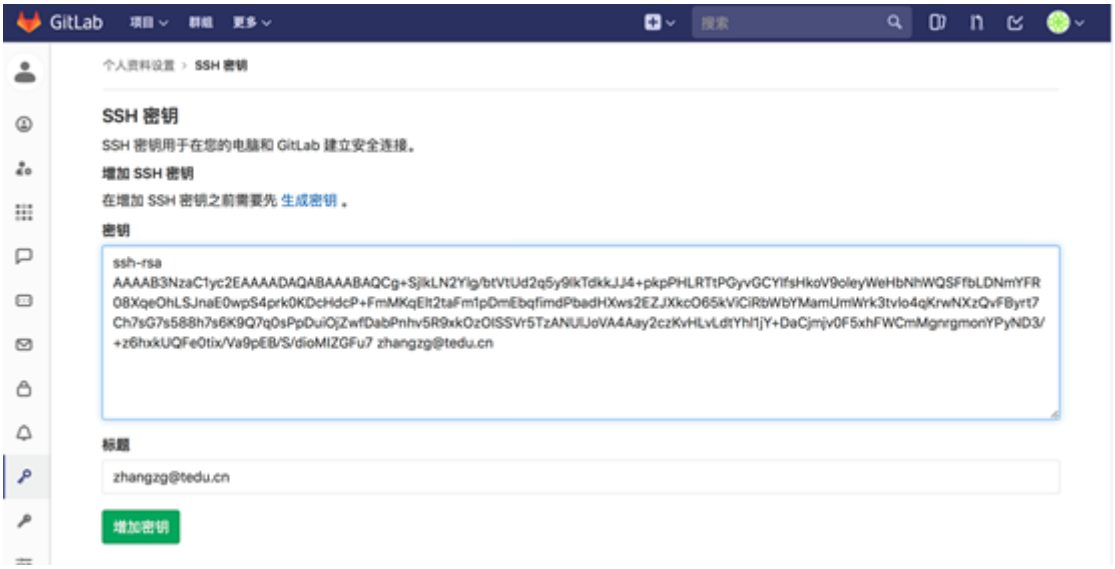


图-17

步骤五：简单远程操作及新用户上传版本库到gitlab

1)克隆远程库代码到本地

```
# git clone git@192.168.113.249:devops/core_py.git
```

2)创建一个文件

```
# cd /root/whsir/whsir
# echo "Hello" > hello.py
```

3)将文件添加到仓库

```
# git add hello.py
```

4)提交文件到仓库，输出信息如图-18所示：

```
# git commit -m "hello文件"
```

Top

```
[root@vm2 whsir]# git commit -m "1"
[master (root-commit) b498a0e] 1
 Committer: root <root@vm2.(none)>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

If the identity used for this commit is wrong, you can fix it with:

    git commit --amend --author='Your Name <you@example.com>'

 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

图-18

5)关联远程库

```
01.  # git remote add origin git@192.168.113.249:root/whsir.git
```

6)最后推送到gitlab上，输出信息如图-19所示：

```
01.  # git push origin master
```

```
[root@vm2 whsir]# git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 203 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@192.168.0.80:root/whsir.git
 * [new branch]      master -> master
```

图-19

5 案例5：模拟用户登陆

5.1 问题

编写login.py脚本，实现以下目标：

1. 创建名为login.py的程序文件
2. 程序提示用户输入用户名
3. 用户输入用户名后，打印欢迎用户

5.2 方案

编写程序时，很多情况下都需要程序与用户交互。在python3中，主要通过input()获取用户输入信息，使用print()打印信息。

通常当想看变量内容时，会在代码中使用print()语句输出。不过在交互式解释器中，可以用print语句显示变量的字符串表示，或者仅使用变量名查看该变量的原始值。

Top

从用户那里得到数据输入的最容易的方法是使用input()内建函数。它读取标准输入，并将读取到的数据赋值给指定的变量。需要注意的是，input()函数读入的数据全部是以字符串的方式存储的。如果用户输的是数字，那么python也将其保存为字符串，当将字符串与数字做数学运算是将会出现TypeError的错误。

初学者在需要显示信息或得到用户输入时，很容易想到使用print()语句和input()内建函数。不过在此建议函数应该保持其清晰性，也就是它只应该接受参数，返回结果。从用户那里得到需要的数据， 然后调用函数处理， 从函数得到返回值，然后显示结果给用户。这样你就能够在其它地方也可以使用你的函数而不必担心自定义输出的问题。这个规则的一个例外是，如果函数的基本功能就是为了得到用户输出，或者就是为了输出信息，这时在函数体使用print()语句或input()也未尝不可。更重要的，将函数分为两大类，一类只做事，不需要返回值（比如与用户交互或设置变量的值）， 另一类则执行一些运算，最后返回结果。如果输出就是函数的目的，那么在函数体内使用print()语句也是可以接受的选择。

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

本次练习的脚本文件是/root/bin/login.py。

```
[root@localhost day01]# vim login.py
#! /usr/bin/env python3

username = input('username: ')       #使用变量username接收用户输入的字符

print('Welcome', username)              #输出欢迎信息，字符串和变量名之间用逗号
                                    #隔开，两者之间自动会加上空格
```

步骤二：测试脚本执行

```
[root@localhost day01]# python3 login.py
username: bob                       #输入用户名
Welcome bob
```

Top

NSD Python1 DAY02

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。

```
[root@localhost day02]# vim login2.py

#! /usr/bin/env python3

username = input('username: ')
password = input('password: ')

if username == 'bob':
    if password == '123456':
        print('Login successful')
    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 day02]# 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 day02]# python3 login2.py
02.  username: bob
03.  password: 123456
04.  Login successful
05.  [root@localhost day02]# python3 login2.py
06.  username: bob
07.  password: abcd
08.  Login incorrect
09.  [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。

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：

```
[root@localhost day02]# vim login2.py

#! /usr/bin/env python3

import getpass    #调用该函数可以在命令行窗口里面无回显输入密码

username = input('username: ')
password = getpass.getpass('password: ')

if username == 'bob' and password == '123456':
    print('\033[32;1mLogin successful!\033[0m')        #绿色粗体显示
else:
    print('\033[31;1mLogin incorrect!\033[0m')        #红色粗体显示
```

测试脚本执行：

```
[root@localhost day02]# python3 login2.py
username: bob
password: 123456              #此处所填写的密码将不在屏幕上显示
Login successful
[root@localhost day02]# python3 login2.py
username: tom
password: 123456              #此处所填写的密码将不在屏幕上显示
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

Top

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

```
[root@localhost day02]# vim grade.py

#! /usr/bin/env python3
#coding: utf8                    #为了程序可以支持中文，指定UTF8编码

score = int(input('成绩：'))

if score >= 90:
    print('优秀')
elif score >= 80:
    print('好')
elif score >= 70:
    print('良')
elif score >= 60:
    print('及格')
else:
    print('你要努力了！')
```

或将上面的代码改为以下写法：

```
score = int(input('成绩：'))

if score >= 60 and score < 70:
    print('及格')
elif 70 <= score < 80:
    print('良')
elif 80 <= score < 90:
    print('好')
elif score >= 90:
    print('优秀')
else:
    print('你要努力了！')
```

Top

步骤二：测试脚本执行

```
[root@localhost day02]# python3 grade.py
成绩： 59
你要努力了！
[root@localhost day02]# python3 grade.py
成绩： 88
好
[root@localhost day02]# python3 grade.py
成绩： 64
及格
[root@localhost day02]# python3 grade.py
成绩： 75
良
[root@localhost day02]# python3 grade.py
成绩： 97
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

```
[root@localhost day02]# vim game.py
#! /usr/bin/env python3

import random

#1. 提示并获取用户的输入
player = int( input( "请输入 0剪刀 1石头 2布:" ) )

#2. 让电脑出一个随机数
computer = random.randint( 0,2)

#3. 判断用户的输入,然后显示对应的结果
#if 玩家获胜的条件:
if ( player==0 and computer==2) or ( player==1 and computer==0) or ( player==2 and comp
    print( "赢了,,,,可以去买奶粉了....." )
#elif 玩家平局的条件:
elif player==computer:
    print( "平局了,,,洗洗手决战到天亮...." )
else:
    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

```
import random

computer = random.choice( [ '石头', '剪刀', '布' ] )
player = input( '请出拳( 石头/剪刀/布) : ' )

# print( '您出了:', player, '计算机出的是:', computer)
print( '您出了: %s, 计算机出的是: %s' %( player, computer) )
if player == '石头':
    if computer == '石头':
        print( '平局' )
    elif computer == '剪刀':
        print( 'You WIN! ! ! ' )
    else:
        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

```
[root@localhost day02]# python3 game.py
请出拳(石头/剪刀/布):布
您出了:布,计算机出的是:剪刀
You LOSE!!!
[root@localhost day02]# python3 game.py
请出拳(石头/剪刀/布):石头
您出了:石头,计算机出的是:剪刀
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’

```
import random

all_choices=['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_choices)
ind=int(input(prompt))
player=all_choices[ind]

print('您出了:%s,计算机出的是:%s'%(player,computer))
if player==computer:
    print('\033[32;1m平局\033[0m')
elif [player,computer] in win_list:
    print('\033[31;1mYou WIN!!!\033[0m')
else:
    print('\033[31;1mYou LOSE!!!\033[0m')
```

测试脚本执行:

Top

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

```
(1) 剪刀
(2) 布
请选择(0/1/2) ：2
您出了: 布, 计算机出的是: 布
平局
[root@localhost day02]# python3 game2.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) ：1
您出了: 剪刀, 计算机出的是: 剪刀
平局
[root@localhost day02]# python3 game2.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) ：0
您出了: 石头, 计算机出的是: 石头
平局
[root@localhost day02]# python3 game2.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) ：1
您出了: 剪刀, 计算机出的是: 石头
You LOSE!!!
[root@localhost day02]# python3 game2.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) ：2
您出了: 布, 计算机出的是: 剪刀
You LOSE!!!
[root@localhost day02]# python3 game2.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) ：1
您出了: 剪刀, 计算机出的是: 石头
You LOSE!!!
[root@localhost day02]# python3 game2.py
```

Top

```
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2) :0
您出了: 石头, 计算机出的是: 剪刀
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

```
[root@localhost day02]# vim game2.py
#!/usr/bin/env python3

import random

i = 1           #游戏次数
win = 0         #赢局次数
while i <= 3:
    #1. 提示并获取用户的输入
    player = int(input("请输入 0剪刀 1石头 2布:"))

    #2. 让电脑出一个随机数
    computer = random.randint(0,2)
#让用户输入合法
```

Top

```
    if player==0 or player==1 or player==2:
        #3. 判断用户的输入,然后显示对应的结果
        if (player==0 and computer==2) or (player==1 and computer==0) or (player==2 and
            print("第"+str(i)+"局"+"赢了")
            win += 1
        elif player==computer:
            print("第"+str(i)+"局"+"平局")
        else:
            print("第"+str(i)+"局"+"输了")
        i += 1
    else:
        print("请重新输入合法数字")
#4. 判断最终猜拳结果：3局两胜
if win >= 2:
    print("恭喜，你赢了！！")
else:
    print("你输了！！")
```

步骤二：测试脚本执行

```
[root@localhost day02]# python3 game2.py
请输入 0剪刀 1石头 2布:3
请重新输入合法数字
请输入 0剪刀 1石头 2布:1
第1局赢了
请输入 0剪刀 1石头 2布:2
第2局赢了
请输入 0剪刀 1石头 2布:3
请重新输入合法数字
请输入 0剪刀 1石头 2布:2
第3局平局
恭喜，你赢了！！
```

步骤三：改进脚本

Top

```
import random

all_choices = ['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): """
cwin = 0
pwin = 0

while cwin < 2 and pwin < 2:
    computer = random.choice(all_choices)
    ind = int(input(prompt))
    player = all_choices[ind]

    print("Your choice: %s, Computer's choice: %s" %(player, computer))
    if player == computer:
        print('\033[32;1m平局\033[0m')
    elif [player, computer] in win_list:
        pwin += 1
        print('\033[31;1mYou WIN!!!\033[0m')
    else:
        cwin += 1
        print('\033[31;1mYou LOSE!!!\033[0m')
```

测试脚本执行：

```
[root@localhost day02]# python3 game3.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2): 1
Your choice: 剪刀, Computer's choice: 剪刀
平局
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2): 2
Your choice: 布, Computer's choice: 石头
You WIN!!!
(0) 石头
(1) 剪刀
```

Top

```
(2) 布
请选择(0/1/2):0
Your choice: 石头, Computer's choice: 剪刀
You WIN!!!
[root@localhost day02]# python3 game3.py
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2):0
Your choice: 石头, Computer's choice: 布
You LOSE!!!
(0) 石头
(1) 剪刀
(2) 布
请选择(0/1/2):1
Your choice: 剪刀, Computer's choice: 石头
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

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

```

Top

```
import random

secret = random.randint(1,100)        #生成随机数

time = 5       #猜数字的次数

print("---------欢迎来到猜数字的地方，请开始---------")
while time > 0:
    guess = int(input("*数字区间0-100，请输入你猜的数字:"))
    print("你输入数字是：",guess)
    if 0 <= guess < 100:
        if guess == secret:
            print("猜对了，真厉害")
        else:
            print("太遗憾了，你猜错了，你还有",time-1,"次机会")
        time -= 1
    else:
        print("输入非法，请重新输入")
print("游戏结束，正确的结果是：",secret)
```

步骤二：测试脚本执行

```
[root@localhost day02]# python3 guess.py
---------欢迎来到猜数字的地方，请开始---------
*数字区间0-100，请输入你猜的数字:100
你输入数字是： 100
输入非法，请重新输入
*数字区间0-100，请输入你猜的数字:0
你输入数字是： 0
太遗憾了，你猜错了，你还有 4 次机会
*数字区间0-100，请输入你猜的数字:-1
你输入数字是： -1
输入非法，请重新输入
*数字区间0-100，请输入你猜的数字:12
你输入数字是： 12
太遗憾了，你猜错了，你还有 3 次机会
*数字区间0-100，请输入你猜的数字:34
你输入数字是： 34
太遗憾了，你猜错了，你还有 2 次机会
*数字区间0-100，请输入你猜的数字:56
```

Top

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

步骤三：改进脚本

```
import random

num = random.randint( 1, 100)
counter = 0

while counter < 5:
    answer = int( input( 'guess the number: ') )
    if answer > num:
        print( '猜大了')
    elif answer < num:
        print( '猜小了')
    else:
        print( '猜对了')
        break
    counter += 1
else:  # 循环被break就不执行了，没有被break才执行
    print( 'the number is:', num)
```

测试脚本执行：

```
[ root@localhost day02] # python3 guess2.py
猜大了
guess the number: 30
猜小了
guess the number: 50
猜小了
guess the number: 70
猜小了
guess the number: 78
猜小了
```

Top

```
the number is: 88
[root@localhost day02]# python3 guess2.py
guess the number: 16
猜小了
guess the number: 90
猜大了
guess the number: 50
猜大了
guess the number: 30
猜对了
```

Top

NSD Python1 DAY03

1 案例1：斐波那契数列

1.1 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

1.2 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

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 day03]# vim fib.py

#! /usr/bin/env python3

a, b = 0, 1

for i in range(10):
    print(a)
    a, b = b, a + b
```

或将上面的代码改为以下写法：

Top

```
[root@localhost day03]# vim fib2.py
```

```
02.
03.  #! /usr/bin/env python3
04.
05.  fib = [ 0, 1]
06.
07.  l = int( input( "数列长度: ") )
08.  for i in range( l - 2) :
09.      fib.append( fib[ - 1] + fib[ - 2] )
10.
11.  print( fib)
```

或将上面的代码改为以下写法：

```
01.  [ root@localhost day03] # vim fib_func.py
02.
03.  #! /usr/bin/env python3
04.
05.  def gen_fib( l) :
06.      fib = [ 0, 1]
07.
08.      for i in range( l - len( fib) ) :
09.          fib.append( fib[ - 1] + fib[ - 2] )
10.
11.      return fib  # 返回列表，不返回变量fib
12.
13.  a = gen_fib( 10)
14.  print( a)
15.  print( '- ' * 50)
16.  n = int( input( "length: ") )
17.  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

```
01.  [ root@localhost day03] # python3 fib.py
02.  0
03.  1
04.  1
05.  2
06.  3
```

Top

```
5
8
13
21
34
[root@localhost day03]# python3 fib2.py
数列长度: 9
[0, 1, 1, 2, 3, 5, 8, 13, 21]
[root@localhost day03]# python3 fib_func.py
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
--------------------------------------------------
length: 9
[0, 1, 1, 2, 3, 5, 8, 13, 21]
```

2 案例2：九九乘法表

2.1 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

2.2 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

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 day03]# vim mtable.py

#! /usr/bin/env python3

for i in range(1,10):           # [0, 1, 2]
    for j in range(1,i+1):      # i->0:[0], i->1: [0, 1], i->2: [0, 1, 2]
```

Top

```
            print('%sX%s=%s' %(j, i, i*j), end=' ')
        print()

[root@localhost day03]# vim mtable.py

#! /usr/bin/env python3

i=1
while i<=9:
    j=1
    while j<=i:
        print("%d*%d=%d" %(j,i,j*i),end=" ")
        j+=1
    print("")
    i+=1
```

步骤二：测试脚本执行

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

3 案例3：模拟cp操作

3.1 问题

创建cp.py文件，实现以下目标：

1. 将/bin/ls“拷贝”到/root/目录下
2. 不要修改原始文件

3.2 方案

Top

获取用户原文件名和新文件名，打开原文件，打开新文件，从打开的原文件中读取数据，写入到打开的新文件中，关闭两个文件

cp代码的过程中，需要注意的部分在于：

如果一个文件过大，你将无法直接读取数据到内存，此时，使用while循环语句，分次读取数据，每次读4096字节，读取数据为空时，结束循环，将数据写入到目标文件

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

```
[root@localhost day03]# vim cp.py
#! /usr/bin/env python3

f1 = open('/bin/ls', 'rb')
f2 = open('/root/ls', 'wb')

data = f1.read()
f2.write(data)

f1.close()
f2.close()
```

或将上面的代码改为以下写法：
循环读取文件中数据，避免读取数据过大

```
[root@localhost day03]# vim cp2.py
#! /usr/bin/env python3

src_fname = '/bin/ls'
dst_fname = '/tmp/ls'

src_fobj = open(src_fname, 'rb')
dst_fobj = open(dst_fname, 'wb')

while True:
    data = src_fobj.read(4096)    # 每次读4096字节
    if data == b'':               # 读不到数据意味着读写完毕，中断循环
        break
    dst_fobj.write(data)          # 将数据写到目标文件

src_fobj.close()
dst_fobj.close()
```

Top

或将上面的代码改为以下写法：

With打开文件读取数据或写入数据后，文件会直接关闭

```
[root@localhost day03]# vim cp3.py
#! /usr/bin/env python3

src_fname = '/bin/ls'
dst_fname = '/root/ls'

with open(src_fname, 'rb') as src_fobj:
    with open(dst_fname, 'wb') as dst_fobj:
        while True:
            data = src_fobj.read(4096)
            if not data:
                break
            dst_fobj.write(data)
```

或将上面的代码改为以下写法：

sys.argv方法表示空列表，执行脚本时输入命令： python3 cp_func.py /bin/ls /root/ls，表示sys.argv=[cp_func.py， ‘/bin/ls’ ， ‘/root/ls’]，所以，调用copy函数时，列表切片方式获取实参为（ ‘/bin/ls’ ， ‘/root/ls’ ）

```
[root@localhost day03]# vim cp_func.py
#! /usr/bin/env python3

import sys

def copy(src_fname, dst_fname):
    src_fobj = open(src_fname, 'rb')
    dst_fobj = open(dst_fname, 'wb')

    while True:
        data = src_fobj.read(4096)
        if not data:
            break
        dst_fobj.write(data)

    src_fobj.close()
    dst_fobj.close()
```

Top

```
18.
19.     copy(sys.argv[1], sys.argv[2])
```

步骤二：测试脚本执行

```
01.  [root@localhost day03]# python3 cp.py
02.  [root@localhost day03]# cd /root
03.  [root@localhost ~]# ls
04.  core  ls
05.
06.  [root@localhost day03]# python3 cp2.py
07.  [root@localhost day03]# cd /root
08.  [root@localhost ~]# ls
09.  core  ls
10.
11.  [root@localhost day03]# python3 cp3.py
12.  [root@localhost day03]# cd /root
13.  [root@localhost ~]# ls
14.  core  ls
15.
16.  [root@localhost day03]# python3 cp_func.py  /bin/ls /root/ls
17.  [root@localhost day03]# cd /root
18.  [root@localhost ~]# ls
19.  core  ls
```

4 案例4：生成随机密码

4.1 问题

创建randpass.py脚本，要求如下：

1. 编写一个能生成8位随机密码的程序
2. 使用random的choice函数随机取出字符
3. 改进程序，用户可以自己决定生成多少位的密码

4.2 方案

导入random模块，通过random静态对象调用choice()方法，从自定义字符串all_chs中获取随机项，将获取到的随机字符ch与原result值进行拼接，将最终字符串结果返回给函数，for循环每循环一次获取一个随机字符，密码位数由循环次数决定，循环次数由传递参数值决定。

此程序需要注意的部分在于： 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.将整个生成随机密码的代码封装进gen_pass()函数中，当模块文件直接执行时，调用函数即可输出结果

3.参数传递问题：调用函数无实参时，函数调用默认参数，有实参时，函数调用实际参数

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 day03]# vim randpass.py
#! /usr/bin/env python3
import random
import string

all_chs = string.digits + string.ascii_letters

def gen_pass(n=8):
    result = ''

    for i in range(n):
        ch = random.choice(all_chs)
        result += ch
    return result

if __name__ == '__main__':
    print(gen_pass())
    print(gen_pass(4))
```

或将上面的代码改为以下写法：

利用列表推导式更简洁输出数据

```
[root@localhost day03]# vim randpass2.py
#! /usr/bin/env python3

from random import choice
from string import ascii_letters, digits

all_chs = ascii_letters + digits


```

Top

```
10.  def randpass(n=8):
11.      result = [choice(all_chs) for i in range(n)]
12.      return ''.join(result)  #将列表的字符拼接起来
13.
14.
15.  if __name__ == '__main__':
16.      print(randpass())
17.      print(randpass(4))
```

步骤二：测试脚本执行

```
01.  [root@localhost day03]# python3 randpass.py
02.  82wi2gOP
03.  XzMi
04.  [root@localhost day03]# python3 randpass.py
05.  5wMbDEgC
06.  BDpc
07.  [root@localhost day03]# python3 randpass.py
08.  Ige2VGod
09.  Az0z
10.  [root@localhost day03]# python3 randpass2.py
11.  eajAocMH
12.  edW1
```

Top

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

```
user information:
username: a
password: hD31SmTS
user information:
username: b
password: DztS7ycn
user information:
username: a
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）：

```
[root@localhost day04]# vim fmtoutput.py
#! /usr/bin/env python3
from mktxtfile import get_contents

width = 48

contents = get_contents()
```

Top

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

步骤二：测试脚本执行

```
[root@localhost day04]# python3 fmtoutput.py
请输入内容，结束请输入end。
> nihao
> my name zhangzhigang
> bye
> end
+************************************************+
+                    nihao                       +
+            my name zhangzhigang                +
+                    bye                         +
+************************************************+
```

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