写用python脚本 统计每个开发人员的代码量：

# encoding: utf-8  
import git  
import subprocess  
import os  
import pymysql  
import getopt  
import sys  
import datetime  
import requests  
  
  
# 定义脚本使用方法  
def usage():  
 print(  
 """  
 usage: python3 [{0}] ... [ -s 2019-9-30 | -e 2019-10-10 | -d 1,2,3... | -h | -r |-c /root/mytest/allproject ] ..  
 参数说明:   
 -h : usage  
 -r : 是否 dry run, 缺省为否  
 -s : 代码开始统计时间  
 -e : 代码结束统计时间 ,默认统计昨天代码量  
 -d : 指定统计最近 n 天的代码量  
 -c : 指定将拉取的项目存入哪一个目录下面,默认在/root/mytest/allproject  
 """.format(sys.argv[0]))  
  
  
yesterday = (datetime.datetime.now() - datetime.timedelta(days=1)).strftime("%Y-%m-%d")  
# 没有参数默认统计昨天的代码量  
starttime = yesterday + " 00:00:00"  
endtime = yesterday + " 23:59:59"  
# 默认目标路径为/root/mytest/allproject  
targetPath = "/root/mytest/allproject"  
opts, args = getopt.getopt(sys.argv[1:], "hs:e:c:d:r", ["help", "dry-run"])  
for op, value in opts:  
 if op in ("-h", "--help"):  
 usage()  
 sys.exit()  
 if op in ("-r", "--dry-run"):  
 print("python3 {0}".format(sys.argv[0]))  
 sys.exit()  
 elif op == '-s':  
 starttime = value + " 00:00:00"  
 elif op == '-e':  
 endtime = value + " 23:59:59"  
 elif op == '-d':  
 starttime = (datetime.datetime.now() - datetime.timedelta(days=eval(value.strip()))).strftime(  
 "%Y-%m-%d") + " 00:00:00"  
 elif op == '-c':  
 targetPath = value  
 else:  
 pass  
  
# 需要爬取的url  
url = "https://git.jczh56.com/api/v4/projects?private\_token=65cx2shbYxsJgWpQcpCN&per\_page=20000"  
# 用一个二维矩阵存放projects info  
projects\_info\_matrix = []  
  
  
# 依据代码开始统计时间从gitlab上获取项目信息，包括项目名，git url,更新时间  
def getProject\_info(project\_url):  
 r = requests.get(project\_url)  
 data = r.json()  
 for info in data:  
 # 存放当前项目相关信息: 项目名，git url,更新时间  
 project\_infos = []  
 # 2019-10-11T08:55:13.982Z 截取时间 2019-10-11 08:55:13  
 project\_mtime = info['last\_activity\_at'][0:10] + " " + info['last\_activity\_at'][11:19]  
 # # 根据项目提交时间，判断是否需要缓存，以便后续统计代码量  
 # if project\_mtime >= starttime:  
 project\_infos.append(info['name'])  
 project\_infos.append(info['http\_url\_to\_repo'])  
 project\_infos.append(project\_mtime)  
 projects\_info\_matrix.append(project\_infos)  
  
  
getProject\_info(url)  
  
# 数据库初始化  
connect = pymysql.Connect(  
 host='192.168.1.11',  
 port=3307,  
 user='git\_user',  
 passwd='GitTest@456PC',  
 db='gitstats',  
 charset='utf8'  
)  
  
# 获取游标  
cursor = connect.cursor()  
  
# 每次在进行代码统计前先将表中数据清空  
truncsql = "truncate table author\_codecount"  
# 将所有开发者的代码量归零  
to\_zerosql = "update developer\_info SET code\_number = 0"  
# 插入数据,将每个人的代码量统计放入author\_codecount(作为临时表，因为有些author用英文，方便后期对照)  
codecountsql = "INSERT INTO author\_codecount (author,codecount) values('%s','%d')"  
# 将统计的代码量存入developer\_info中(将存在的用户名对应代码量存入developer\_info表中)  
updateCountsql = "update developer\_info SET code\_number = %d WHERE author\_name = '%s'"  
query\_namesql = "SELECT author\_name FROM developer\_info WHERE author\_name = '%s'"  
try:  
 cursor.execute(truncsql)  
 cursor.execute(to\_zerosql)  
 connect.commit()  
except Exception as aa:  
 pass  
  
# 用于统计每个人的代码量  
personal\_code\_count = {}  
# 用于统计代码量的shell脚本  
# codecountcmd = """  
# git log --after="2019-9-30 00:00:00" --before="2019-10-10 23:59:59" --format='%aN' | sort -u | while read name; do echo -en "||$name\t" ; git log --author="$name" --after="2019-9-30 00:00:00" --before="2019-10-10 23:59:59" --pretty=tformat: --numstat | awk '{ add += $1; subs += $2; loc += $1 - $2 } END { printf -en "added lines: %s\tremoved lines: %s\ttotal lines: %s", add, subs, loc }' -; done  
# """  
  
codecountcmd = """  
 git log --after=\"""" + starttime + """\" --before=\"""" + endtime + """\" --format='%aN' | sort -u | while read name; do echo -en "||$name\t"; git log --author="$name" --after=\"""" + starttime + """\" --before=\"""" + endtime + """\" --pretty=tformat: --numstat | awk '{ add += $1; subs += $2; loc += $1 - $2 } END { printf -en "added lines: %s\tremoved lines: %s\ttotal lines: %s", add, subs, loc }' -; done  
"""  
  
  
# 对每个分支进行代码统计  
def codecount():  
 status = subprocess.getstatusoutput(codecountcmd)  
 # 字段处理，将不需要的内容处理掉  
 resultAll = status[1]  
 results = resultAll.split("||")  
 results.pop(0)  
 for result in results:  
 # 解析 result,把人名和总代码量取出来  
 person\_code = result.split("\t")  
 person = person\_code[0].strip()  
 code\_count = person\_code[1].split(": ")[1].strip()  
 if code\_count != "":  
 personal\_code\_count[person] = personal\_code\_count.get(person, 0) + eval(code\_count)  
  
  
for project\_info in projects\_info\_matrix:  
 # 依据项目名称创建本地仓库  
 localpath = f"{targetPath}/{project\_info[0]}"  
 # 如果该目录存在则删除该目录  
 if os.path.exists(localpath):  
 rmcmd = f"rm -rf {localpath}"  
 subprocess.getstatusoutput(rmcmd)  
 os.makedirs(localpath)  
 # 先切换到本地仓库，才能使用git log 统计代码量  
 os.chdir(localpath)  
 # 将代码clone到本地  
 url = r"https://wwf:WwfOracle@git.jczh56.com" + project\_info[1].split(r"//")[1][13:]  
 try:  
 clone\_repo = git.Repo.clone\_from(url, localpath)  
 except Exception as e:  
 continue  
 # 克隆到本地后再将各分支代码拉过来并统计代码量  
 repo = git.Repo(localpath)  
 # 获取所有远程分支  
 repo\_branch = repo.git.branch("-r").split("\n")  
 # 删除本地当前分支，不进行统计  
 repo\_branch.pop(0)  
 # 判断当前项目含有多少分支，只有master分支，统计master分支代码量;否则统计其他分支代码量  
 if len(repo\_branch) == 1:  
 for ref in repo\_branch:  
 codecount()  
 else:  
 for ref in repo\_branch:  
 # 获取远程分支对应名称  
 refname = ref.split("/")[1].strip()  
 # 如果是master分支，则略过  
 if refname == "master":  
 continue  
 # 如果已创建本地分支，先删除该分支  
 delgitcmd = f"git branch -d {refname}"  
 subprocess.getstatusoutput(delgitcmd)  
 # 在本地创建refname分支并关联远程ref分支  
 gitbcmd = f"git checkout -b {refname} {ref}"  
 subprocess.getstatusoutput(gitbcmd)  
 # 对当前分支的代码量进行统计  
 codecount()  
 # 如果该目录存在则删除该目录,清除拉下来的项目，减少系统空间  
 if os.path.exists(localpath):  
 rmcmd = f"rm -rf {localpath}"  
 subprocess.getstatusoutput(rmcmd)  
  
for key, value in personal\_code\_count.items():  
 print(key, "\t\t", value)  
 tmpdata = (key, value)  
 query\_name\_data = (key,)  
 try:  
 cursor.execute(codecountsql % tmpdata)  
 connect.commit()  
 cursor.execute(query\_namesql % query\_name\_data)  
 # 如果存在该author,则将该代码量更新到结果表中  
 if cursor.fetchall():  
 result\_data = (value, key)  
 cursor.execute(updateCountsql % result\_data)  
 connect.commit()  
 except Exception as e:  
 connect.rollback() # 事务回滚  
 print('事务处理失败', e)  
  
# 关闭连接  
cursor.close()  
connect.close()