



广州商学院
Guangzhou College of Commerce

实验指导书

实验课程: Python3 程序设计

编写教师: 龙君

适用专业: 计算机科学与技术专业

(蓝盾网络安全特色班)

编写日期: 2020. 2

前 言

本实验课程是《Python3程序设计》的配套实验，专供学生上机操作使用。

本实验指导书选用了32个实验项目。主要以学生训练为主，实验包括：Python安装与开发环境搭建、Python运算符、内置函数、凯撒加密、使用蒙特卡罗方法计算圆周率近似值、使用集合实现筛选法求素数、小明爬楼梯、自定义类模拟三维向量及其运算、批量生成随机信息、自定义类实现带超时功能队列结构、文本文件操作、猜数游戏、tkinter版猜数游戏、Excel文件数据导入SQLite数据库、数据分析与可视化综合实验、使用urllib、requests及beautifulsoup4库爬取网页信息、使用scrapy框架爬山东各城市天气预报；涉及的内容有：语言概述、基础知识、Python中的字符串、Python程序的流程、Python的组合数据类型、用函数实现代码复用、用类实现抽象和封装、使用模块与库编程、Python的文件操作、异常处理tkinter GUI编程、数据库编程、科学计算与图表绘制以及爬取与分析网页中的数据；实验指导书中给出了实验目的、实验内容和实验步骤，学生在做实验前一定要认真预习实验指导书，特别是注意事项。

由于编者水平有限，难免在本实验指导书中出现错误或不妥之处，望读者指正。

目录

适用教材：	5
实验一、Python 安装与开发环境搭建.....	6
实验目的：	6
实验内容：	6
实验步骤：	6
实验二、Python 运算符、内置函数.....	8
实验目的：	8
实验内容：	8
参考代码：	8
实验三、使用蒙特.卡罗方法计算圆周率近似值.....	9
实验目的：	9
实验内容：	9
参考代码：	9
实验四、使用列表实现筛选法求素数.....	10
实验目的：	10
实验内容：	10
参考代码：	10
实验五、使用集合实现筛选法求素数.....	11
实验目的：	11
实验内容：	11
参考代码：	11
实验六、理解浮点数运算的误差.....	12
实验目的：	12
实验内容：	12
提示：	12
实验七、小明爬楼梯.....	13
实验目的：	13
实验内容：	13
参考代码：	13
实验八、聪明的尼姆游戏（人机对战）	14
实验目的：	14
实验内容：	14
参考代码：	14
实验九、蒙蒂霍尔悖论游戏.....	16
实验目的：	16
实验内容：	16
参考代码：	16
实验十、猜数游戏.....	18
实验目的：	18
实验内容：	18
参考代码：	18
实验十一、抓狐狸游戏.....	20

实验目的：	20
实验内容：	20
参考代码：	20
实验十二、汉诺塔问题.....	22
实验目的：	22
实验内容：	22
参考代码：	22
实验十三、凯撒加密.....	23
实验目的：	23
实验内容：	23
参考代码：	23
实验十四、打字练习成绩评定.....	24
实验目的：	24
实验内容：	24
参考代码：	24
实验十五、批量生成随机信息.....	25
实验目的：	25
实验内容：	25
参考代码：	25
实验十六、自定义类模拟三维向量及其运算.....	29
实验目的：	29
实验内容：	29
参考代码：	29
实验十七、自定义类实现带超时功能队列结构.....	31
实验目的：	31
实验内容：	31
参考代码：	31
实验十八、文本文件操作.....	33
实验目的：	33
实验内容：	33
参考代码：	33
实验十九、磁盘垃圾文件清理器.....	34
实验目的：	34
实验内容：	34
参考代码：	34
实验二十、Excel 文件成绩处理.....	35
实验目的：	35
实验内容：	35
实验步骤：	35
参考代码：	35
实验二十一、Word 文件操作.....	38
实验目的：	38
实验内容：	38
实验步骤：	38

参考代码：	38
实验二十二、tkinter 版猜数游戏.....	39
实验目的：	39
实验内容：	39
参考代码：	39
实验二十三、使用 TCP 协议实现智能聊天机器人.....	43
实验目的：	43
实验内容：	43
参考代码：	43
服务端代码（chatServer.py）：	43
客户端代码（chatClient.py）：	44
实验二十四、使用 scrapy 框架爬取山东各城市天气预报.....	46
实验目的：	46
实验内容：	46
实验步骤：	46
实验二十五、电影打分与推荐.....	51
实验目的：	51
实验内容：	51
参考代码：	51
实验二十六、多线程快速复制目录树.....	53
实验目的：	53
实验内容：	53
参考代码：	53
进一步思考：	55
实验二十七、Excel 文件数据导入 SQLite 数据库.....	56
实验目的：	56
实验内容：	56
参考代码：	56
实验二十八、生成棋盘纹理图片.....	59
实验目的：	59
实验内容：	59
参考代码：	59
效果图：	59
实验二十九、暴力破解 MD5 值.....	60
实验目的：	60
实验内容：	60
参考代码：	60
实验三十、数据分析与可视化综合实验.....	61
实验目的：	61
实验内容：	61
参考代码：	62

适用教材：

- 1、董付国编著.《Python 程序设计（第 2 版）》，清华大学出版社，2016
- 2、董付国编著.《Python 程序设计基础（第 2 版）》，清华大学出版社，2018
- 3、董付国著.《Python 可以这样学》，清华大学出版社，2017
- 4、董付国著.《Python 程序设计开发宝典》，清华大学出版社，2017

参考书：

- 1、董付国,应根球著.《中学生可以这样学 Python》，清华大学出版社，2017
- 2、董付国编著.《玩转 Python 轻松过二级》，清华大学出版社，2018

实验一、Python 安装与开发环境搭建

实验目的：

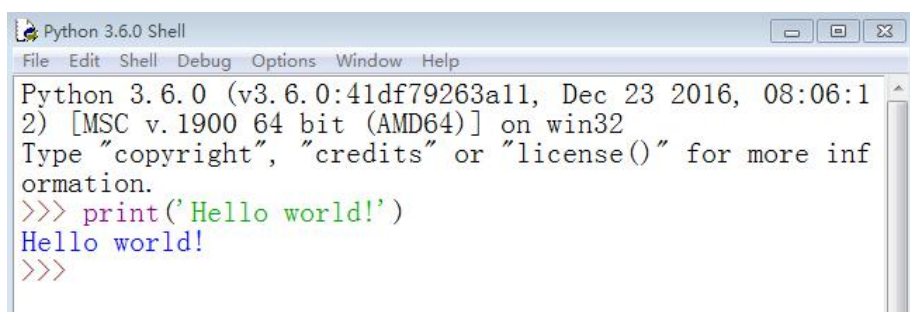
- 1、熟练掌握 Python 解释器安装与基本用法。
- 2、熟练掌握使用 pip 命令安装 Python 扩展库。
- 3、熟悉离线安装轮子文件的方法。

实验内容：

- 1、安装 Python 解释器。
- 2、安装 Python 扩展库。

实验步骤：

- 1、打开 Python 官方网站 <http://www.python.org>。
- 2、下载 Python 3.5.x 或 Python 3.6.x 或 Python 3.7.x 的最新版，至少安装其中两个。
- 3、在开始菜单中找到成功安装的 IDLE，输入下面的代码，确保 IDLE 运行正常。



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print('Hello world!')
Hello world!
>>>
```

4、在资源管理器中进入 Python 安装目录的 scripts 子目录，然后按下 Shift 键，在空白处单击鼠标右键，在弹出来的菜单中选择“在此处打开命令窗口”进入命令提示符环境。如图所示：



```
C:\Python36\Scripts>
```

5、使用 pip 命令在线安装 Python 扩展库 numpy、pandas、scipy、matplotlib、jieba、openpyxl、pillow。安装 openpyxl 的命令如图所示：



```
C:\Python36\Scripts>pip install openpyxl
Collecting openpyxl
  Downloading https://files.pythonhosted.org/packages/dc/99/9c58d83d7f093c0af5f90875f8595d2e9587fc36532a8bb347608cf0876b
/openpyxl-2.5.3.tar.gz (170kB)
    100% |#####| 174kB 666kB/s
Requirement already satisfied: jdcal in c:\python36\lib\site-packages (from openpyxl) (1.3)
Requirement already satisfied: et_xmlfile in c:\python36\lib\site-packages (from openpyxl) (1.0.1)
Building wheels for collected packages: openpyxl
  Running setup.py bdist_wheel for openpyxl ... done
  Stored in directory: C:\Users\d\AppData\Local\pip\Cache\wheels\11\7d\47\3dad56b5d260c790d9110623ba66783a2ad345eb76dd63
003b
Successfully built openpyxl
audiotools 0.1.0 requires hamlearn, which is not installed.
pyed3 0.8.0a2 requires grako, which is not installed.
Installing collected packages: openpyxl
Successfully installed openpyxl-2.5.3
```

- 6、如果遇到安装不成功的扩展库，使用浏览器打开下面的网址下载 whl 文件进行离线安装：<https://www.lfd.uci.edu/~gohlke/pythonlibs/>
- 7、在 IDLE 中使用 `import` 导入安装好的扩展库，验证是否安装成功。

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC
D64] on win32
Type "copyright", "credits" or "license()" for more informatio
>>> import openpyxl
>>> import jieba
>>> import numpy as np
>>>
```


实验二、Python 运算符、内置函数

实验目的：

- 1、熟练运用 Python 运算符。
- 2、熟练运用 Python 内置函数。

实验内容：

- 1、编写程序，输入任意大的自然数，输出各位数字之和。
- 2、编写程序，输入两个集合 `setA` 和 `setB`，分别输出它们的交集、并集和差集 `setA-setB`。
- 3、编写程序，输入一个自然数，输出它的二进制、八进制、十六进制表示形式。

参考代码：

1、

```
num = input('请输入一个自然数: ')
print(sum(map(int, num)))
```

2、

```
setA = eval(input('请输入一个集合: '))
setB = eval(input('再输入一个集合: '))
print('交集: ', setA & setB)
print('并集: ', setA | setB)
print('setA-setB: ', setA - setB)
```

3、

```
num = int(input('请输入一个自然数: '))
print('二进制: ', bin(num))
print('八进制: ', oct(num))
print('十六进制: ', hex(num))
```



实验三、使用蒙特·卡罗方法计算圆周率近似值

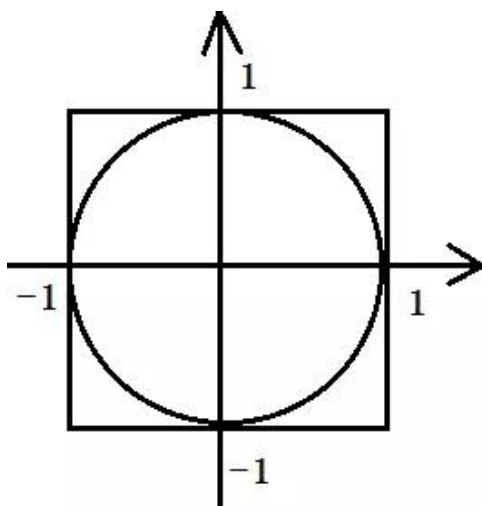
实验目的：

- 1、理解蒙特·卡罗方法原理。
- 2、理解 for 循环本质与工作原理。
- 3、了解 random 模块中常用函数。

实验内容：



蒙特·卡罗方法是一种通过概率来得到问题近似解的方法，在很多领域都有重要的应用，其中就包括圆周率近似值的计算问题。假设有一块边长为 2 的正方形木板，上面画一个单位圆，然后随意往木板上扔飞镖，落点坐标(x, y)必然在木板上（更多的时候是落在单位圆内），如果扔的次数足够多，那么落在单位圆内的次数除以总次数再乘以 4，这个数字会无限逼近圆周率的值。这就是蒙特·卡罗发明的用于计算圆周率近似值的方法，如图所示。



编写程序，模拟蒙特·卡罗计算圆周率近似值的方法，输入掷飞镖次数，然后输出圆周率近似值。

参考代码：

```
from random import random

times = int(input('请输入掷飞镖次数: '))
hits = 0
for i in range(times):
    x = random()
    y = random()
    if x*x + y*y <= 1:
        hits += 1
print(4.0 * hits/times)
```

实验四、使用列表实现筛选法求素数

实验目的：

- 1、理解筛选法求解素数的原理。
- 2、理解列表切片操作。
- 3、熟练运用内置函数 enumerate()。
- 4、熟练运用内置函数 filter()。
- 5、理解序列解包工作原理。
- 6、初步了解选择结构和循环结构。

实验内容：

编写程序，输入一个大于 2 的自然数，然后输出小于该数字的所有素数组成的列表。

参考代码：

```
maxNumber = int(input('请输入一个大于 2 的自然数: '))
lst = list(range(2, maxNumber))
#最大整数的平方根
m = int(maxNumber**0.5)
for index, value in enumerate(lst):
    #如果当前数字已大于最大整数的平方根，结束判断
    if value > m:
        break
    #对该位置之后的元素进行过滤
    lst[index+1:] = filter(lambda x: x%value != 0, lst[index+1:])
print(lst)
```



实验五、使用集合实现筛选法求素数

实验目的：

- 1、理解求解素数的筛选法原理。
- 2、理解 Python 集合对象的 `discard()` 方法。
- 3、熟练运用列表推导式。
- 4、理解 for 循环工作原理。



实验内容：

输入一个大于 2 的自然数，输出小于该数字的所有素数组成的集合。

参考代码：

```
maxNumber = int(input('请输入一个大于 2 的自然数: '))
numbers = set(range(2, maxNumber))

# 最大数的平方根，以及小于该数字的所有素数
m = int(maxNumber**0.5)+1
primesLessThanM = [p for p in range(2, m)
                    if 0 not in [p%d for d in range(2, int(p**0.5)+1)]]

# 遍历最大整数平方根之内的自然数
for p in primesLessThanM:
    for i in range(2, maxNumber//p+1):
        # 在集合中删除该数字所有的倍数
        numbers.discard(i*p)

print(numbers)
```

实验六、理解浮点数运算的误差

实验目的：

- 1、理解组合数定义式的化简。
- 2、理解浮点数运算的误差和可能带来的问题。

实验内容：

阅读并调试下面的代码，分析代码功能，发现并解决代码中的错误。

```
def cni(n,i):  
    minNI = min(i, n-i)  
    result = 1  
    for j in range(0, minNI):  
        result = result * (n-j) // (minNI-j)  
    return result
```

提示：

这段代码试图计算组合数 C_n^i ，但是由于浮点数除法时精度问题导致结果错误。



实验七、小明爬楼梯

实验目的：

- 1、理解并熟练使用序列解包。
- 2、理解递归函数工作原理。
- 3、能够编写递归函数代码解决实际问题。
- 4、理解 Python 字典的用法。



实验内容：

假设一段楼梯共 15 个台阶，小明一步最多能上 3 个台阶。编写程序计算小明上这段楼梯一共有多少种方法。要求给出递推法和递归法两种代码。

参考代码：

```
def climbStairs1(n):
    #递推法
    a = 1
    b = 2
    c = 4
    for i in range(n-3):
        c, b, a = a+b+c, c, b
    return c

def climbStairs2(n):
    #递归法
    first3 = {1:1, 2:2, 3:4}
    if n in first3.keys():
        return first3[n]
    else:
        return climbStairs2(n-1) + \
            climbStairs2(n-2) + \
            climbStairs2(n-3)

print(climbStairs1(15))
print(climbStairs2(15))
```

实验八、聪明的尼姆游戏（人机对战）

实验目的：

- 1、理解尼姆游戏规则。
- 2、了解多个函数定义与调用。
- 3、理解并熟练运用 while 循环。
- 4、理解带 else 子句的循环结构执行流程。
- 5、理解循环语句中的 break 语句的作用。
- 6、了解使用循环和异常处理结构对用户输入进行约束的用法。



实验内容：

尼姆游戏是个著名的游戏，有很多变种玩法。两个玩家轮流从一堆物品中拿走一部分。在每一步中，玩家可以自由选择拿走多少物品，但是必须至少拿走一个并且最多只能拿走一半物品，然后轮到下一个玩家。拿走最后一个物品的玩家输掉游戏。

在聪明模式中，计算机每次拿走足够多的物品使得堆的大小是 2 的幂次方减 1——也就是 3,7,15,31 或 63。除了堆的大小已经是 2 的幂次方减 1，在其他情况下这样走都是符合游戏规则的。在那种情况下，计算机就按游戏规则随机拿走一些。

编写程序，模拟聪明版本的尼姆游戏。

参考代码：

```
from math import log2
from random import randint, choice

def everyStep(n):
    half = n / 2
    m = 1
    # 所有可能满足条件的取法
    possible = []
    while True:
        rest = 2**m - 1
        if rest >= n:
            break
        if rest >= half:
            possible.append(n-rest)
        m = m+1
    # 如果至少存在一种取法使得剩余物品数量为 2^n-1
    if possible:
        return choice(possible)
```

```
# 无法使得剩余物品数量为  $2^n - 1$ , 随机取走一些
return randint(1, int(half))

def smartNimuGame(n):
    while n > 1:
        # 人类玩家先走
        print("Now it's your turn, and we have {0} left.".format(n))
        # 确保人类玩家输入合法整数值
        while True:
            try:
                num = int(input('How many do you want to take:'))
                assert 1 <= num <= n//2
                break
            except:
                print('Error.Must be between 1 and {0}'.format(n//2))
        n -= num
        if n == 1:
            return 'I fail.'
        # 计算机玩家拿走一些
        n -= everyStep(n)
    else:
        return 'You fail.'

print(smartNimuGame(randint(1, 100)))
```


实验九、蒙蒂霍尔悖论游戏

实验目的：

- 1、了解蒙蒂霍尔悖论内容。
- 2、了解游戏规则。
- 3、熟练运用字典方法和集合运算。
- 4、了解断言语句 `assert` 的用法。
- 5、熟练运用循环结构。



实验内容：

假设你正参加一个有奖游戏节目，并且有 3 道门可选：其中一个后面是汽车，另外两个后面是山羊。你选择一个门，比如说 1 号门，主持人当然知道每个门后面是什么并且打开了另一个门，比如说 3 号门，后面是一只山羊。这时，主持人会问你“你想改选 2 号门吗？”，然后根据你的选择确定最终要打开的门，并确定你获得山羊（输）或者汽车（赢）。

编写程序，模拟上面的游戏。

参考代码：

```
from random import randrange

def init():
    '''返回一个字典，键为 3 个门号，值为门后面的物品'''
    result = {i: 'goat' for i in range(3)}
    r = randrange(3)
    result[r] = 'car'
    return result

def startGame():
    # 获取本次游戏中每个门的情况
    doors = init()
    # 获取玩家选择的门号
    while True:
        try:
            firstDoorNum = int(input('Choose a door to open:'))
            assert 0<= firstDoorNum <=2
            break
        except:
            print('Door number must be between {} and {}'.format(0, 2))
    # 主持人查看另外两个门后的物品情况
    for door in doors.keys()-{firstDoorNum}:
        # 打开其中一个后面为山羊的门
```

```
    if doors[door] == 'goat':
        print('"goat" behind the door', door)
        # 获取第三个门号，让玩家纠结
        thirdDoor = (doors.keys()-{door, firstDoorNum}).pop()
        change = input('Switch to {}?(y/n)'.format(thirdDoor))
        finalDoorNum = thirdDoor if change=='y' else firstDoorNum
        if doors[finalDoorNum] == 'goat':
            return 'I Win!'
        else:
            return 'You Win.'

while True:
    print('='*30)
    print(startGame())
    r = input('Do you want to try once more?(y/n)')
    if r == 'n':
        break
```

实验十、猜数游戏



实验目的：

- 1、熟练运用选择结构与循环结构解决实际问题。
- 2、注意选择结构嵌套时代码的缩进与对齐。
- 3、理解带 else 子句的循环结构执行流程。
- 4、理解条件表达式 value1 if condition else value2 的用法。
- 5、理解使用异常处理结构约束用户输入的用法。
- 6、理解带 else 子句的异常处理结构的执行流程。

实验内容：

编写程序模拟猜数游戏。程序运行时，系统生成一个随机数，然后提示用户进行猜测，并根据用户输入进行必要的提示（猜对了、太大了、太小了），如果猜对则提前结束程序，如果次数用完仍没有猜对，提示游戏结束并给出正确答案。

参考代码：

```
from random import randint

def guessNumber(maxValue=10, maxTimes=3):
    # 随机生成一个整数
    value = randint(1,maxValue)
    for i in range(maxTimes):
        prompt = 'Start to GUESS:' if i==0 else 'Guess again:'
        # 使用异常处理结构，防止输入不是数字的情况
        try:
            x = int(input(prompt))
        except:
            print('Must input an integer between 1 and ', maxValue)
        else:
            if x == value:
                # 猜对了
                print('Congratulations!')
                break
            elif x > value:
                print('Too big')
            else:
                print('Too little')
    else:
        # 次数用完还没猜对，游戏结束，提示正确答案
        print('Game over. FAIL.')
```

```
print('The value is ', value)
```

```
guessNumber()
```

实验十一、抓狐狸游戏

实验目的：

- 1、培养分析问题并对进行建模的能力。
- 2、熟练使用列表解决实际问题。
- 3、熟练运用选择结构和循环结构解决实际问题。
- 4、理解带 else 子句的循环结构执行流程。
- 5、理解使用异常处理结构约束用户输入的用法。



实验内容：

编写程序，模拟抓狐狸小游戏。假设一共有一排 5 个洞口，小狐狸最开始的时候在其中一个洞口，然后玩家随机打开一个洞口，如果里面有狐狸就抓到了。如果洞口里没有狐狸就第二天再来抓，但是第二天狐狸会在玩家来抓之前跳到隔壁洞口里。

参考代码：

```
from random import choice, randrange

def catchMe(n=5, maxStep=10):
    '''模拟抓小狐狸，一共 n 个洞口，允许抓 maxStep 次
    如果失败，小狐狸就会跳到隔壁洞口'''
    # n 个洞口，有狐狸为 1，没有狐狸为 0
    positions = [0] * n
    # 狐狸的随机初始位置
    oldPos = randrange(0, n)
    positions[oldPos] = 1

    # 抓 maxStep 次
    while maxStep >= 0:
        maxStep -= 1
        # 这个循环保证用户输入是有效洞口编号
        while True:
            try:
                x = input('今天打算打开哪个洞口？（0-{0}）：'.format(n-1))
                # 如果输入的不是数字，就会跳转到 except 部分
                x = int(x)
                # 如果输入的洞口有效，结束这个循环，否则就继续输入
                assert 0 <= x < n, '要按套路来啊，再给你一次机会。'
                break
            except:
                # 如果输入的不是数字，就执行这里的代码
                print('要按套路来啊，再给你一次机会。')
```

```
if positions[x] == 1:
    print('成功，我抓到小狐狸啦。')
    break
else:
    print('今天又没抓到。')

# 如果这次没抓到，狐狸就跳到隔壁洞口
if oldPos == n-1:
    newPos = oldPos -1
elif oldPos == 0:
    newPos = oldPos + 1
else:
    newPos = oldPos + choice((-1, 1))
positions[oldPos], positions[newPos] = 0, 1
oldPos = newPos
else:
    print('放弃吧，你这样乱试是没有希望的。')

# 启动游戏，开始抓狐狸吧
catchMe()
```

实验十二、汉诺塔问题

实验目的：

- 1、理解函数默认值参数。
- 2、理解函数递归。
- 3、熟练运行列表对象的方法。

实验内容：

据说古代有一个梵塔，塔内有三个底座 A、B、C，A 座上有 64 个盘子，盘子大小不等，大的在下，小的在上。有一个和尚想把这 64 个盘子从 A 座移到 C 座，但每次只能允许移动一个盘子。在移动盘子的过程中可以利用 B 座，但任何时刻 3 个座上的盘子都必须始终保持大盘在下、小盘在上的顺序。如果只有一个盘子，则不需要利用 B 座，直接将盘子从 A 移动到 C 即可。编写函数，接收一个表示盘子数量的参数和分别表示源、目标、临时底座的参数，然后输出详细移动步骤和每次移动后三个底座上的盘子分布情况。

参考代码：

```
def hanoi(num, src, dst, temp=None): #递归算法
    if num < 1:
        return
    global times    #声明用来记录移动次数的变量为全局变量
    #递归调用函数自身，先把除最后一个盘子之外的所有盘子移动到临时柱子上
    hanoi(num-1, src, temp, dst)
    # 移动最后一个盘子
    print('The {0} Times move:{1}==>{2}'.format(times, src, dst))
    towers[dst].append(towers[src].pop())
    for tower in 'ABC':    #输出 3 根柱子上的盘子
        print(tower, ': ', towers[tower])
    times += 1
    #把除最后一个盘子之外的其他盘子从临时柱子上移动到目标柱子上
    hanoi(num-1, temp, dst, src)

times = 1    #用来记录移动次数的变量
n = 3        #盘子数量
towers = {'A':list(range(n, 0, -1)), #初始状态，所有盘子都在 A 柱上
          'B':[],
          'C':[]
          }
#A 表示最初放置盘子的柱子，C 是目标柱子，B 是临时柱子
hanoi(n, 'A', 'C', 'B')
```



实验十三、凯撒加密

实验目的：

- 1、了解 Python 标准库 string。
- 2、理解凯撒加密算法原理。
- 3、理解切片操作。
- 4、熟练运用字符串对象的方法。



实验内容：

编写程序，要求输入一个字符串，然后输入一个整数作为凯撒加密算法的密钥，然后输出该字符串加密后的结果。

参考代码：

```
import string
def kaisa(s, k):
    lower = string.ascii_lowercase      #小写字母
    upper = string.ascii_uppercase      #大写字母
    before = string.ascii_letters
    after = lower[k:] + lower[:k] + upper[k:] + upper[:k]
    table = ''.maketrans(before, after)  #创建映射表
    return s.translate(table)

s = input('请输入一个字符串: ')
k = int(input('请输入一个整数密钥: '))
print(kaisa(s, k))
```


实验十四、打字练习成绩评定

实验目的：

- 1、熟练运用内置函数 zip()、sum()、round()、isinstance()。
- 2、熟练运用生成器表达式。

实验内容：

编写程序，模拟打字练习程序的成绩评定。假设 origin 为原始内容，userInput 为用户练习时输入的内容，要求用户输入的内容长度不能大于原始内容的长度，如果对应位置的字符一致则认为正确，否则判定输入错误。最后成绩为：正确的字符数量/原始字符串长度，按百分制输出，要求保留 2 位小数。

参考代码：

```
def Rate(origin, userInput):
    if not (isinstance(origin, str) and isinstance(userInput, str)):
        return 'The two parameters must be strings.'
    if len(origin)<len(userInput):
        return 'Sorry. I suppose the second parameter string is shorter.'

    # 精确匹配的字符个数
    right = sum(1 for oc, uc in zip(origin, userInput) if oc==uc)
    return right/len(origin)

origin = '''Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.'''
userInput = '''Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
readability counts.'''

print(round(Rate(origin, userInput)*100, 2))
```



实验十五、批量生成随机信息

实验目的：

- 1、熟练运用标准库 random 中的函数。
- 2、了解标准库 string 中的字符串常量。
- 3、理解 Python 程序中__name__属性的作用。
- 4、了解汉字编码格式。
- 5、熟练掌握文本文件操作方法。
- 6、在文件操作时养成使用上下文管理语句 with 的习惯。

实验内容：

编写程序，生成 200 个人的模拟信息，包括姓名、性别、年龄、电话号码、家庭住址、电子邮箱地址，把生成的信息写入文本文件，每行存放一个人的信息，最后再读取生成的文本文件并输出其中的信息。

参考代码：

```
import random
import string
```

#常用汉字 Unicode 编码表，可以自行搜索

```
StringBase = '\u7684\u4e00\u4e86\u662f\u6211\u4e0d\u5728\u4eba\u4eec'\
              '\u6709\u6765\u4ed6\u8fd9\u4e0a\u7740\u4e2a\u5730\u5230'\
              '\u5927\u91cc\u8bf4\u5c31\u53bb\u5b50\u5f97\u4e5f\u548c'\
              '\u90a3\u8981\u4e0b\u770b\u5929\u65f6\u8fc7\u51fa\u5c0f'\
              '\u4e48\u8d77\u4f60\u90fd\u628a\u597d\u8fd8\u591a\u6ca1'\
              '\u4e3a\u53c8\u53ef\u5bb6\u5b66\u53ea\u4ee5\u4e3b\u4f1a'\
              '\u6837\u5e74\u60f3\u751f\u540c\u8001\u4e2d\u5341\u4ece'\
              '\u81ea\u9762\u524d\u5934\u9053\u5b83\u540e\u7136\u8d70'\
              '\u5f88\u50cf\u89c1\u4e24\u7528\u5979\u56fd\u52a8\u8fdb'\
              '\u6210\u56de\u4ec0\u8fb9\u4f5c\u5bfb\u5f00\u800c\u5df1'\
              '\u4e9b\u73b0\u5c71\u6c11\u5019\u7ecf\u53d1\u5de5\u5411'\
              '\u4e8b\u547d\u7ed9\u957f\u6c34\u51e0\u4e49\u4e09\u58f0'\
              '\u4e8e\u9ad8\u624b\u77e5\u7406\u773c\u5fd7\u70b9\u5fc3'\
              '\u6218\u4e8c\u95ee\u4f46\u8eab\u65b9\u5b9e\u5403\u505a'\
              '\u53eb\u5f53\u4f4f\u542c\u9769\u6253\u5462\u771f\u5168'\
              '\u624d\u56db\u5df2\u6240\u654c\u4e4b\u6700\u5149\u4ea7'\
              '\u60c5\u8def\u5206\u603b\u6761\u767d\u8bdd\u4e1c\u5e2d'\
              '\u6b21\u4eb2\u5982\u88ab\u82b1\u53e3\u653e\u513f\u5e38'\
              '\u6c14\u4e94\u7b2c\u4f7f\u5199\u519b\u5427\u6587\u8fd0'\
              '\u518d\u679c\u600e\u5b9a\u8bb8\u5feb\u660e\u884c\u56e0'
```



```
'\u522b\u98de\u5916\u6811\u7269\u6d3b\u90e8\u95e8\u65e0'\n'\u5f80\u8239\u671b\u65b0\u5e26\u961f\u5148\u529b\u5b8c'\n'\u5374\u7ad9\u4ee3\u5458\u673a\u66f4\u4e5d\u60a8\u6bcf'\n'\u98ce\u7ea7\u8ddf\u7b11\u554a\u5b69\u4e07\u5c11\u76f4'\n'\u610f\u591c\u6bd4\u9636\u8fde\u8f66\u91cd\u4fbf\u6597'\n'\u9a6c\u54ea\u5316\u592a\u6307\u53d8\u793e\u4f3c\u58eb'\n'\u8005\u5e72\u77f3\u6ee1\u65e5\u51b3\u767e\u539f\u62ff'\n'\u7fa4\u7a76\u5404\u516d\u672c\u601d\u89e3\u7acb\u6cb3'\n'\u6751\u516b\u96be\u65e9\u8bba\u5417\u6839\u5171\u8ba9'\n'\u76f8\u7814\u4eca\u5176\u4e66\u5750\u63a5\u5e94\u5173'\n'\u4fe1\u89c9\u6b65\u53cd\u5904\u8bb0\u5c06\u5343\u627e'\n'\u4e89\u9886\u6216\u5e08\u7ed3\u5757\u8dd1\u8c01\u8349'\n'\u8d8a\u5b57\u52a0\u811a\u7d27\u7231\u7b49\u4e60\u9635'\n'\u6015\u6708\u9752\u534a\u706b\u6cd5\u9898\u5efa\u8d76'\n'\u4f4d\u5531\u6d77\u4e03\u5973\u4efb\u4ef6\u611f\u51c6'\n'\u5f20\u56e2\u5c4b\u79bb\u8272\u8138\u7247\u79d1\u5012'\n'\u775b\u5229\u4e16\u521a\u4e14\u7531\u9001\u5207\u661f'\n'\u5bfc\u665a\u8868\u591f\u6574\u8ba4\u54cd\u96ea\u6d41'\n'\u672a\u573a\u8be5\u5e76\u5e95\u6df1\u523b\u5e73\u4f1f'\n'\u5fd9\u63d0\u786e\u8fd1\u4eae\u8f7b\u8bb2\u519c\u53e4'\n'\u9ed1\u544a\u754c\u62c9\u540d\u5440\u571f\u6e05\u9633'\n'\u7167\u529e\u53f2\u6539\u5386\u8f6c\u753b\u9020\u5634'\n'\u6b64\u6cbb\u5317\u5fc5\u670d\u96e8\u7a7f\u5185\u8bc6'\n'\u9a8c\u4f20\u4e1a\u83dc\u722c\u7761\u5174\u5f62\u91cf'\n'\u54b1\u89c2\u82e6\u4f53\u4f17\u901a\u51b2\u5408\u7834'\n'\u53cb\u5ea6\u672f\u996d\u516c\u65c1\u623f\u6781\u5357'\n'\u67aa\u8bfb\u6c99\u5c81\u7ebf\u91ce\u575a\u7a7a\u6536'\n'\u7b97\u81f3\u653f\u57ce\u52b3\u843d\u94b1\u7279\u56f4'\n'\u5f1f\u80dc\u6559\u70ed\u5c55\u5305\u6b4c\u7c7b\u6e10'\n'\u5f3a\u6570\u4e61\u547c\u6027\u97f3\u7b54\u54e5\u9645'\n'\u65e7\u795e\u5ea7\u7ae0\u5e2e\u5566\u53d7\u7cfb\u4ee4'\n'\u8df3\u975e\u4f55\u725b\u53d6\u5165\u5cb8\u6562\u6389'\n'\u5ffd\u79cd\u88c5\u9876\u6025\u6797\u505c\u606f\u53e5'\n'\u533a\u8863\u822c\u62a5\u53f6\u538b\u6162\u53d4\u80cc\u7ec6'
```

```
def getEmail():  
    # 常见域名后缀，可以随意扩展该列表  
    suffix = ['.com', '.org', '.net', '.cn']  
    characters = string.ascii_letters+string.digits+'_'  
    username = ''.join((random.choice(characters)  
                        for i in range(random.randint(6,12))))  
    domain = ''.join((random.choice(characters)  
                    for i in range(random.randint(3,6))))  
    return username+'@'+domain+random.choice(suffix)
```

```
def getTelNo():
    return ''.join((str(random.randint(0,9)) for i in range(11)))

def getNameOrAddress(flag):
    '''flag=1 表示返回随机姓名, flag=0 表示返回随机地址'''
    result = ''
    if flag==1:
        # 大部分中国人姓名在 2-4 个汉字
        rangestart, rangeend = 2, 5
    elif flag==0:
        # 假设地址在 10-30 个汉字之间
        rangestart, rangeend = 10, 31
    else:
        print('flag must be 1 or 0')
        return ''
    # 生成并返回随机信息
    for i in range(random.randrange(rangestart, rangeend)):
        result += random.choice(StringBase)
    return result

def getSex():
    return random.choice(('男', '女'))

def getAge():
    return str(random.randint(18,100))

def main(filename):
    with open(filename, 'w', encoding='utf-8') as fp:
        # 写入表头
        fp.write('Name,Sex,Age,TelNO,Address,Email\n')
        # 生成 200 个人的随机信息
        for i in range(200):
            name = getNameOrAddress(1)
            sex = getSex()
            age = getAge()
            tel = getTelNo()
            address = getNameOrAddress(0)
            email = getEmail()
            line = ','.join([name,sex,age,tel,address,email])+'\n'
            fp.write(line)

def output(filename):
    with open(filename, 'r', encoding='utf-8') as fp:
        for line in fp:
```

```
        print(line)

if __name__ == '__main__':
    filename = 'information.txt'
    main(filename)
    output(filename)
```

实验十六、自定义类模拟三维向量及其运算



实验目的：

- 1、了解如何定义一个类。
- 2、了解如何定义类的私有数据成员和成员方法。
- 3、了解如何使用自定义类实例化对象。

实验内容：

定义一个三维向量类，并定义相应的特殊方法实现两个该类对象之间的加、减运算（要求支持运算符+、-），实现该类对象与标量的乘、除运算（要求支持运算符*、/），以及向量长度的计算（要求使用属性实现）。

参考代码：

```
class Vector3:
```

```
    #构造方法，初始化，定义向量坐标
```

```
    def __init__(self, x, y, z):
```

```
        self.__x = x
```

```
        self.__y = y
```

```
        self.__z = z
```

```
    #与另一个向量相加，对应分量相加，返回新向量
```

```
    def __add__(self, anotherPoint):
```

```
        x = self.__x + anotherPoint.__x
```

```
        y = self.__y + anotherPoint.__y
```

```
        z = self.__z + anotherPoint.__z
```

```
        return Vector3(x, y, z)
```

```
    #减去另一个向量，对应分量相减，返回新向量
```

```
    def __sub__(self, anotherPoint):
```

```
        x = self.__x - anotherPoint.__x
```

```
        y = self.__y - anotherPoint.__y
```

```
        z = self.__z - anotherPoint.__z
```

```
        return Vector3(x, y, z)
```

```
    #向量与一个数字相乘，各分量乘以同一个数字，返回新向量
```

```
    def __mul__(self, n):
```

```
        x, y, z = self.__x*n, self.__y*n, self.__z*n
```

```
        return Vector3(x, y, z)
```

```
    #向量除以一个数字，各分量除以同一个数字，返回新向量
```

```
    def __truediv__(self, n):
```

```
        x, y, z = self.__x/n, self.__y/n, self.__z/n
```

```
        return Vector3(x, y, z)

#查看向量长度，所有分量平方和的平方根
@property
def length(self):
    return (self.__x**2 + self.__y**2 + self.__z**2)**0.5

def __str__(self):
    return 'Vector3({}, {}, {})'.format(self.__x,
                                          self.__y,
                                          self.__z)

#用法演示
v1 = Vector3(3, 4, 5)
v2 = Vector3(5, 6, 7)
print(v1+v2)
print(v1-v2)
print(v1*3)
print(v2/2)
print(v1.length)
```

实验十七、自定义类实现带超时功能队列结构

实验目的：

- 1、了解标准库 `time` 中 `time()` 函数的用法。
- 2、了解如何定义一个类。
- 3、理解队列结构的特点。
- 4、理解入队和出队时超时功能的实现。



实验内容：

编写程序，实现自定义类，模拟队列结构。要求实现入队、出队以及修改队列大小和判断队列是否为空、是否为满的功能，同时要求在入队时如果队列已满则等待指定时间、出队时如果队列已空则等待指定时间等辅助功能。

参考代码：

```
import time

class myQueue:
    def __init__(self, size = 10):
        self._content = []
        self._size = size
        self._current = 0

    def setSize(self, size):
        if size < self._current:
            # 如果缩小队列，应删除后面的元素
            for i in range(size, self._current)[::-1]:
                del self._content[i]
            self._current = size
        self._size = size

    def put(self, v, timeout=999999):
        # 模拟入队，在列表尾部追加元素
        # 队列满，阻塞，超时放弃
        for i in range(timeout):
            if self._current < self._size:
                self._content.append(v)
                self._current = self._current+1
                break
            time.sleep(1)
        else:
            return '队列已满，超时放弃'
```



```
def get(self, timeout=999999):
    # 模拟出队，从列表头部弹出元素
    # 队列为空，阻塞，超时放弃
    for i in range(timeout):
        if self._content:
            self._current = self._current-1
            return self._content.pop(0)
        time.sleep(1)
    else:
        return '队列为空，超时放弃'

def show(self):
    # 如果列表非空，输出列表
    if self._content:
        print(self._content)
    else:
        print('The queue is empty')

def empty(self):
    self._content = []
    self._current = 0

def isEmpty(self):
    return not self._content

def isFull(self):
    return self._current == self._size

if __name__ == '__main__':
    print('Please use me as a module.')
```

实验十八、文本文件操作

实验目的：

- 1、熟练掌握内置函数 `open()` 的用法。
- 2、熟练运用内置函数 `len()`、`max()`、`enumerate()`。
- 3、理解字符串方法 `ljust()`。
- 4、理解列表推导式。

实验内容：

编写一个程序 `demo.py`，要求运行该程序后，生成 `demo_new.py` 文件，其中内容与 `demo.py` 一样，只是在每一行的后面加上行号。要求行号以 `#` 开始，并且所有行的 `#` 符号垂直对齐。

参考代码：

```
filename = 'demo.py'
with open(filename, 'r') as fp:
    lines = fp.readlines()
maxLength = len(max(lines, key=len))

lines = [line.rstrip().ljust(maxLength)+'#+str(index)+'\n'
          for index, line in enumerate(lines)]
with open(filename[:-3]+'_new.py', 'w') as fp:
    fp.writelines(lines)
```



实验十九、磁盘垃圾文件清理器

实验目的：

- 1、熟练运用标准库 `os` 和 `os.path` 中的函数。
- 2、理解 `sys` 库中 `argv` 成员用法。
- 3、理解 Python 程序接收命令行参数的方式。
- 4、理解递归遍历目录树的原理。
- 5、了解从命令提示符环境运行 Python 程序的方式。



实验内容：

编写程序，实现磁盘垃圾文件清理功能。要求程序运行时，通过命令行参数指定要清理的文件夹，然后删除该文件夹及其子文件夹中所有扩展名为 `tmp`、`log`、`obj`、`txt` 以及大小为 `0` 的文件。

参考代码：

```
from os.path import isdir, join, splitext, getsize
from os import remove, listdir
import sys

# 指定要删除的文件类型
filetypes = ['.tmp', '.log', '.obj', '.txt']

def delCertainFiles(directory):
    if not isdir(directory):
        return
    for filename in listdir(directory):
        temp = join(directory, filename)
        if isdir(temp):
            delCertainFiles(temp)
        elif splitext(temp)[1] in filetypes or getsize(temp)==0:
            # 删除指定类型的文件或大小为 0 的文件
            remove(temp)
            print(temp, ' deleted....')

directory = sys.argv[1]
delCertainFiles(directory)
```

实验二十、Excel 文件成绩处理



实验目的:

- 1、了解扩展库 `openpyxl` 的安装与使用。
- 2、熟练运用字典结构解决实际问题。

实验内容:

假设某学校所有课程每学期允许多次考试，学生可随时参加考试，系统自动将每次成绩添加到 Excel 文件（包含 3 列：姓名，课程，成绩）中，现期末要求统计所有学生每门课程的最高成绩。

编写程序，模拟生成若干同学的成绩并写入 Excel 文件，其中学生姓名和课程名称均可重复，也就是允许出现同一门课程的多次成绩，最后统计所有学生每门课程的最高成绩，并写入新的 Excel 文件。

实验步骤:

- 1、在命令提示符环境使用 `pip install openpyxl` 命令安装扩展库 `openpyxl`。
- 2、编写代码。

参考代码:

```
from random import choice, randint
from openpyxl import Workbook, load_workbook
```

```
#生成随机数据
```

```
def generateRandomInformation(filename):
```

```
    workbook = Workbook()
```

```
    worksheet = workbook.worksheets[0]
```

```
    worksheet.append(['姓名', '课程', '成绩'])
```

```
#中文名字中的第一、第二、第三个字
```

```
first = '赵钱孙李'
```

```
middle = '伟昀琛东'
```

```
last = '坤艳志'
```

```
subjects = ('语文', '数学', '英语')
```

```
for i in range(200):
```

```
    name = choice(first)
```

```
    #按一定概率生成只有两个字的中文名字
```

```
    if randint(1,100)>50:
```

```
        name = name + choice(middle)
```

```
    name = name + choice(last)
```

```
    #依次生成姓名、课程名称和成绩
```

```
        worksheet.append([name, choice(subjects), randint(0, 100)])
#保存数据，生成 Excel 2007 格式的文件
workbook.save(filename)

def getResult(oldfile, newfile):
    #用于存放结果数据的字典
    result = dict()

    #打开原始数据
    workbook = load_workbook(oldfile)
    worksheet = workbook.worksheets[0]

    #遍历原始数据
    for row in worksheet.rows:
        if row[0].value == '姓名':
            continue
        #姓名,课程名称,本次成绩
        name, subject, grade = map(lambda cell:cell.value, row)

        #获取当前姓名对应的课程名称和成绩信息
        #如果 result 字典中不包含，则返回空字典
        t = result.get(name, {})
        #获取当前学生当前课程的成绩，若不存在，返回 0
        f = t.get(subject, 0)
        #只保留该学生该课程的最高成绩
        if grade > f:
            t[subject] = grade
            result[name] = t

    workbook1 = Workbook()
    worksheet1 = workbook1.worksheets[0]
    worksheet1.append(['姓名','课程','成绩'])

    #将 result 字典中的结果数据写入 Excel 文件
    for name, t in result.items():
        print(name, t)
        for subject, grade in t.items():
            worksheet1.append([name, subject, grade])

    workbook1.save(newfile)

if __name__ == '__main__':
    oldfile = r'd:\test.xlsx'
    newfile = r'd:\result.xlsx'
```

```
generateRandomInformation(oldfile)  
getResult(oldfile, newfile)
```

实验二十一、Word 文件操作

实验目的：

- 1、了解扩展库 `python-docx` 的安装与使用。
- 2、理解 Word 文档结构和内容组织形式。
- 3、熟练运用列表和集合解决实际问题。

实验内容：

编写程序，读取 Word 文件中的所有段落文本，然后输出其中所有红色的文本和加粗的文本以及同时具有这两种属性的文本。

实验步骤：

- 1、在命令提示符环境使用 `pip install python-docx` 命令安装扩展库 `python-docx`。
- 2、创建测试用的 Word 文档 `test.docx`，写入测试内容，并根据需要设置红色文本和加粗文本。
- 3、编写程序。

参考代码：

```
from docx import Document
from docx.shared import RGBColor

boldText = []
redText = []
doc = Document('test.docx')
for p in doc.paragraphs:
    for r in p.runs:
        # 加粗字体
        if r.bold:
            boldText.append(r.text)
        # 红色字体
        if r.font.color.rgb == RGBColor(255,0,0):
            redText.append(r.text)

result = {'red text': redText,
          'bold text': boldText,
          'both': set(redText) & set(boldText)}
# 输出结果
for title in result.keys():
    print(title.center(30, '='))
    for text in result[title]:
        print(text)
```



实验二十二、tkinter 版猜数游戏

实验目的：

- 1、理解 tkinter 标准库的用法。
- 2、熟悉创建窗体和组件的方法。
- 3、熟悉 tkinter 组件属性及其作用和设置方法。
- 4、了解如何为 tkinter 组件绑定事件处理方法。



实验内容：

使用 Python 标准库 tkinter 编写 GUI 版本的猜数游戏。每次猜数之前要启动游戏并设置猜数范围和最大猜测次数等参数，退出游戏时显示战绩（共玩几次，猜对几次）信息。

参考代码：

```
import random
import tkinter
from tkinter.messagebox import showerror, showinfo
from tkinter.simpledialog import askinteger

root = tkinter.Tk()
# 窗口标题
root.title('猜数游戏--by 董付国')
# 窗口初始大小和位置
root.geometry('280x80+400+300')
# 不允许改变窗口大小
root.resizable(False, False)

# 用户猜的数
varNumber = tkinter.StringVar(root, value='0')

# 允许猜的总次数
totalTimes = tkinter.IntVar(root, value=0)

# 已猜次数
already = tkinter.IntVar(root, value=0)

# 当前生成的随机数
currentNumber = tkinter.IntVar(root, value=0)

# 玩家玩游戏的总次数
times = tkinter.IntVar(root, value=0)

# 玩家猜对的总次数
```



```
right = tkinter.IntVar(root, value=0)

lb = tkinter.Label(root, text='请输入一个整数: ')
lb.place(x=10, y=10, width=100, height=20)

# 用户猜数并输入的文本框
entryNumber = tkinter.Entry(root,
                              width=140,
                              textvariable=varNumber)
entryNumber.place(x=110, y=10, width=140, height=20)

# 默认禁用，只有开始游戏以后才允许输入
entryNumber['state'] = 'disabled'

# 按钮单击事件处理函数
def buttonClick():
    if button['text']=='Start Game':
        # 每次游戏时允许用户自定义数值范围
        # 玩家必须输入正确的数
        # 最小数值
        while True:
            try:
                start = askinteger('允许的最小整数',
                                   '最小数（必须大于 0）',
                                   initialvalue=1)

                if start != None:
                    assert start>0
                    break
            except:
                pass

        # 最大数值
        while True:
            try:
                end = askinteger('允许的最大整数',
                                 '最大数（必须大于 10）',
                                 initialvalue=11)

                if end != None:
                    assert end>10 and end>start
                    break
            except:
                pass

        # 在用户自定义的数值范围内生成要猜的随机数
```

```
currentNumber.set(random.randint(start, end))

# 用户自定义一共允许猜几次
# 玩家必须输入正确的整数
while True:
    try:
        t = askinteger('最多允许猜几次? ',
                        '总次数（必须大于 0）',
                        initialvalue=3)
        if t != None:
            assert t>0
            totalTimes.set(t)
            break
    except:
        pass
# 已猜次数初始化为 0
already.set(0)
button['text'] = '剩余次数: ' + str(t)

# 把文本框初始化为 0
varNumber.set('0')

# 启用文本框，允许用户开始输入整数
entryNumber['state'] = 'normal'

# 玩游戏的次数加 1
times.set(times.get() + 1)
else:
    # 一共允许猜几次
    total = totalTimes.get()

    # 本次游戏的正确答案
    current = currentNumber.get()

    # 玩家本次猜的数
    try:
        x = int(varNumber.get())
    except:
        showerror('抱歉', '必须输入整数')
        return

    # 猜对了
    if x == current:
        showinfo('恭喜', '猜对啦')
```

```
        button['text'] = 'Start Game'

        # 禁用文本框
        entryNumber['state'] = 'disabled'

        # 猜对的次数加1
        right.set(right.get() + 1)
    else:
        # 本次游戏已猜次数加1
        already.set(already.get()+1)

        if x > current:
            showerror('抱歉', '猜的数太大了')
        else:
            showerror('抱歉', '猜的数太小了')

        # 可猜次数用完了
        if already.get() == total:
            showerror('抱歉',
                    '游戏结束了，正确的数是：' +
                    str(currentNumber.get()))
            button['text'] = 'Start Game'
            # 禁用文本框
            entryNumber['state'] = 'disabled'
        else:
            button['text'] = '剩余次数：' + str(total-already.get())

# 在窗口上创建按钮，并设置事件处理函数
button = tkinter.Button(root,
                        text='Start Game',
                        command=buttonClick)
button.place(x=10, y=40, width=250, height=20)

# 关闭程序时提示战绩
def closeWindow():
    message = '共玩游戏 {0} 次，猜对 {1} 次！\n 欢迎下次再玩！'
    message = message.format(times.get(), right.get())
    showinfo('战绩', message)
    root.destroy()
root.protocol('WM_DELETE_WINDOW', closeWindow)

# 启动消息主循环
root.mainloop()
```

实验二十三、使用 TCP 协议实现智能聊天机器人

实验目的：

- 1、熟悉标准库 `socket` 的用法。
- 2、熟悉 TCP 协议工作原理。
- 3、理解端口号的概念与作用。
- 4、熟悉 `Socket` 编程。

实验内容：

编写聊天程序的服务端代码和客户端代码。完成后，先启动服务端代码，然后启动客户端程序用输入问题，服务端可以返回相应的答案。要求服务端代码具有一定的智能，能够根据不完整的问题识别客户端真正要问的问题。

参考代码：

服务端代码（`chatServer.py`）：

```
import socket
from os.path import commonprefix

words = {'how are you?': 'Fine, thank you.',
        'how old are you?': '38',
        'what is your name?': 'Dong FuGuo',
        "what's your name?": 'Dong FuGuo',
        'where do you work?': 'University',
        'bye': 'Bye'}

HOST = ''
PORT = 50007
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 绑定 socket
s.bind((HOST, PORT))
# 开始监听一个客户端连接
s.listen(1)
print('Listening on port:', PORT)

conn, addr = s.accept()
print('Connected by', addr)
# 开始聊天
while True:
    data = conn.recv(1024).decode()
    if not data:
        break
```



```
print('Received message:', data)
# 尽量猜测对方要表达的真正意思
m = 0
key = ''
for k in words.keys():
    # 删除多余的空白字符
    data = ' '.join(data.split())
    # 与某个“键”非常接近，就直接返回
    if len(commonprefix([k, data])) > len(k)*0.7:
        key = k
        break
# 使用选择法，选择一个重合度较高的“键”
length = len(set(data.split())&set(k.split()))
if length > m:
    m = length
    key = k
# 选择合适的信息进行回复
conn.sendall(words.get(key, 'Sorry.').encode())
conn.close()
s.close()
```

客户端代码（chatClient.py）：

```
import socket
import sys

# 服务端主机 IP 地址和端口号
HOST = '127.0.0.1'
PORT = 50007
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    # 连接服务器
    s.connect((HOST, PORT))
except Exception as e:
    print('Server not found or not open')
    sys.exit()

while True:
    c = input('Input the content you want to send:')
    # 发送数据
    s.sendall(c.encode())
    # 从服务端接收数据
    data = s.recv(1024)
    data = data.decode()
    print('Received:', data)
    if c.lower() == 'bye':
```

```
        break  
# 关闭连接  
s.close()
```

实验二十四、使用 scrapy 框架爬取山东各城市天气预报

实验目的：

- 1、熟练安装 Python 扩展库 scrapy。
- 2、熟悉常见 HTML 标签的用法。
- 3、理解网页源代码结构。
- 4、理解 scrapy 框架工作原理。

实验内容：

安装 Python 扩展库 scrapy，然后编写爬虫项目，从网站 <http://www.weather.com.cn/shandong/index.shtml> 爬取山东各城市的天气预报数据，并把爬取到的天气数据写入本地文本 weather.txt。

实验步骤：

1. 在命令提示符环境使用 `pip install scrapy` 命令安装 Python 扩展库 scrapy。
2. 在命令提示符环境使用 `scrapy startproject sdWeatherSpider` 创建爬虫项目。
3. 进入爬虫项目文件夹，然后执行命令 `scrapy genspider everyCityinSD.py www.weather.com.cn` 创建爬虫程序。
4. 使用浏览器打开网址 <http://www.weather.com.cn/shandong/index.shtml>，找到下面位置。



5. 在页面上单击鼠标右键，选择“查看网页源代码”，然后找到与“城市预报列表”对应的位置。

```
412 <div class="gsbox" style="margin-top:0;">
413 <div class="forecast"><h1 class="weatherH1">城市预报列表
414 (2018-05-29 07:30发布)<span></span></h1>
415 <div class="forecastBox" id="forecastID">
416 <dl>
417 <dt>
418 <a title="济南天气预报" href="http://www.weather.com.cn/weather/101120101.shtml" target="_blank">济南</a>
419 </dt>
420 <dd>
421 <a href="http://www.weather.com.cn/static/html/legend.shtml" target="_blank"></a>
422 <a><span>31℃</span></a></a><b>20℃</b></a>
423 </dd>
424 </dl>
425 <dl>
426 <dt>
427 <a title="青岛天气预报" href="http://www.weather.com.cn/weather/101120201.shtml" target="_blank">青岛</a>
428 </dt>
429 <dd>
430 <a href="http://www.weather.com.cn/static/html/legend.shtml" target="_blank"></a>
431 <a><span>24℃</span></a></a><b>16℃</b></a>
432 </dd>
433 </dl>
434 <dl>
435 <dt>
436 <a title="淄博天气预报" href="http://www.weather.com.cn/weather/101120301.shtml" target="blank">淄博</a>
437 </dt>
438 <dd>
439 <a href="http://www.weather.com.cn/static/html/legend.shtml" target="_blank"></a>
440 <a><span>24℃</span></a></a><b>16℃</b></a>
```

6. 选择并打开山东省内任意城市的天气预报页面，此处以烟台为例。



7. 在页面上单击鼠标右键，选择“查看网页源代码”，找到与上图中天气预报相对应的位置。


```
267 <input type="hidden" id="fc_24h_internal_update_time" vs
268 <input type="hidden" id="update_time" value="07:30"/>
269 <ul class="t clearfix">
270 <li class="sky skyid lv2 on">
271 <h1>29日（今天）</h1>
272 <big class="png40 d01"></big>
273 <big class="png40 n01"></big>
274 <p title="多云" class="wea">多云</p>
275 <p class="tem">
276 <span>24</span>/<i>16℃</i>
277 </p>
278 <p class="win">
279 <em>
280 <span title="东北风" class="NE"></span>
281 <span title="东北风" class="NE"></span>
282 </em>
283 <i>3-4级</i>
284 </p>
285 <div class="slid"></div>
286 </li>
287 <li class="sky skyid lv2">
288 <h1>30日（明天）</h1>
289 <big class="png40 d01"></big>
290 <big class="png40 n01"></big>
291 <p title="多云" class="wea">多云</p>
292 <p class="tem">
293 <span>24</span>/<i>16℃</i>
294 </p>
295 <p class="win">
296 <em>
297 <span title="北风" class="N"></span>
298 <span title="北风" class="N"></span>
299 </em>
300 <i>3-4级</i>
301 </p>
302 <div class="slid"></div>
```

Python小图

8. 修改 items.py 文件，定义要爬取的内容。

```
import scrapy
```

```
class SdweatherspiderItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    city = scrapy.Field()
    weather = scrapy.Field()
```

9. 修改爬虫文件 everyCityinSD.py，定义如何爬取内容，其中用到的规则参考前面
对页面的分析，如果无法正常运行，有可能是网页结构有变化，可以回到前面的步骤重新
分析网页源代码。

```
from re import findall
from urllib.request import urlopen
import scrapy
from sdWeatherSpider.items import SdweatherspiderItem
```

```
class EverycityinsdSpider(scrapy.Spider):
```

```
name = 'everyCityinSD'
allowed_domains = ['www.weather.com.cn']
start_urls = []
# 遍历各城市，获取要爬取的页面 URL
url = r'http://www.weather.com.cn/shandong/index.shtml'
with urlopen(url) as fp:
    contents = fp.read().decode()
pattern = '<a title=".*?" href="(.*?)" target="_blank">(.*?)</a>'
for url in findall(pattern, contents):
    start_urls.append(url[0])

def parse(self, response):
    # 处理每个城市的天气预报页面数据
    item = SdweatherspiderItem()
    city = response.xpath('//div[@class="crumbs fl"]//a[2]//text()').extract()[0]
    item['city'] = city

    # 每个页面只有一个城市的天气数据，直接取[0]
    selector = response.xpath('//ul[@class="t clearfix"]')[0]

    # 存放天气数据
    weather = ''
    for li in selector.xpath('./li'):
        date = li.xpath('./h1//text()').extract()[0]
        cloud = li.xpath('./p[@title]//text()').extract()[0]
        high = li.xpath('./p[@class="tem"]//span//text()').extract()[0]
        low = li.xpath('./p[@class="tem"]//i//text()').extract()[0]
        wind = li.xpath('./p[@class="win"]//em//span[1]//text()').extract()[0]
        wind = wind + li.xpath('./p[@class="win"]//i//text()').extract()[0]

        weather = weather + date+':'+cloud+', '+high+r'/' +low+', '+wind+'\n'
    item['weather'] = weather
    return [item]
```

10. 修改 pipelines.py 文件，把爬取到的数据写入文件 weather.txt。

```
class SdweatherspiderPipeline(object):
    def process_item(self, item, spider):
        with open('weather.txt', 'a', encoding='utf8') as fp:
            fp.write(item['city']+'\n')
            fp.write(item['weather']+'\n\n')
        return item
```

11. 修改 settings.py 文件，分派任务，指定处理数据的程序。

```
BOT_NAME = 'sdWeatherSpider'
```

```
SPIDER_MODULES = ['sdWeatherSpider.spiders']  
NEWSPIDER_MODULE = 'sdWeatherSpider.spiders'
```

```
ITEM_PIPELINES = {  
    'sdWeatherSpider.pipelines.SdweatherspiderPipeline':1,  
}
```

12. 切换到命令提示符环境，执行 `scrapy crawl everyCityinSD` 命令运行爬虫程序。

实验二十五、电影打分与推荐

实验目的：

- 1、理解基于用户的协同过滤算法原理。
- 2、熟练运用字典和集合。
- 3、熟练运用内置函数 `sum()`、`min()`、`len()`。

实验内容：

编写程序，生成数据模拟（也可以使用真实数据）多人对多个电影的打分（1-5 分），然后根据这些数据对某用户 A 进行推荐。推荐规则为：在已有数据中选择与该用户 A 的爱好最相似的用户 B，然后从最相似的用户 B 已看过但用户 A 还没看过的电影中选择用户 B 打分最高的电影推荐给用户 A。相似度的计算标准为：1）两个用户共同打分过的电影越多，越相似；2）两个用户对共同打分的电影的打分越接近，越相似。

参考代码：

```
from random import randrange

# 模拟历史电影打分数据
data = {'user'+str(i):{'film'+str(randrange(1, 15)):randrange(1, 6)
                        for j in range(randrange(3, 10))}
        for i in range(10)}

# 当前用户打分数据
user = {'film'+str(randrange(1, 15)):randrange(1,6) for i in range(5)}
# 最相似的用户及其对电影打分情况
# 两个用户共同打分的电影最多
# 并且所有电影打分差值的平方和最小
f = lambda item:(-len(item[1].keys())&user),
                sum(((item[1].get(film)-user.get(film))**2
                      for film in user.keys()&item[1].keys()))
similarUser, films = min(data.items(), key=f)

print('known data'.center(50, '='))
for item in data.items():
    print(len(item[1].keys())&user.keys()),
          sum(((item[1].get(film)-user.get(film))**2
                for film in user.keys()&item[1].keys()))),
          item,
          sep=':')

print('current user'.center(50, '='))
```



```
print(user)

print('most similar user and his films'.center(50, '='))
print(similarUser, films, sep=':')
print('recommended film'.center(50, '='))
# 在当前用户没看过的电影中选择打分最高的进行推荐
print(max(films.keys()-user.keys(), key=lambda film: films[film]))
```

实验二十六、多线程快速复制目录树

实验目的：

- 1、熟悉标准库 `os`、`shutil` 的用法。
- 2、了解标准库 `sys` 中 `argv` 成员的含义和用法。
- 3、了解标准库 `argparse` 接收并解析命令行参数的用法。
- 4、理解函数嵌套定义的用法。
- 5、掌握标准库 `threading` 的用法。
- 6、理解多线程概念和工作原理。

实验内容：

编写程序，使用多线程技术把源文件夹及其子文件夹中所有内容都复制指定的目标文件夹中。要求该程序能够通过命令行参数来指定源文件夹和目标文件夹以及线程数量。

参考代码：

```
import os
import sys
import argparse
from queue import Queue
from threading import Thread
from shutil import copyfile

def copyFile(src, dst, num):
    '''使用 num 个线程复制 src 目录下的文件到 dst 目录中'''

    # 源文件夹必须存在
    assert os.path.isdir(src), src+' must be an existing directory.'
    # 如果目标文件夹不存在，创建一个
    if not os.path.isdir(dst):
        os.makedirs(dst)

    # 最多容纳 10 个元素的队列
    q = Queue(10)

    def add(src):
        # 把源文件夹中所有项目添加到队列中
        for f in os.listdir(src):
            f = os.path.join(src, f)
            if os.path.isfile(f):
                # 往队列中放数据，满了会自动等待
                q.put(f)
```



```
        elif os.path.isdir(f):
            q.put(f)
            # 递归
            add(f)

# 用来通知工作线程再没有文件需要复制了
for _ in range(num):
    q.put(None)

# 创建并启动往队列中存放元素的线程
t_add = Thread(target=add, args=(src,))
t_add.start()

def copy(name):
    # 工作线程函数
    while True:
        srcItem = q.get()
        if srcItem == None:
            print(name, 'quit...')
            break

        # 替换字符串，生成目标路径
        dstItem = srcItem.replace(src, dst)
        print('{0}:{1}==>{2}'.format(name, srcItem, dstItem))

        # 复制文件
        if os.path.isfile(srcItem):
            # 根据需要创建目标文件夹
            dstDir = os.path.split(dstItem)[0]
            if not os.path.isdir(dstDir):
                try:
                    os.makedirs(dstDir)
                except FileExistsError as e:
                    pass
            copyfile(srcItem, dstItem)
        elif os.path.isdir(srcItem):
            # 创建目标文件夹
            try:
                os.makedirs(dstItem)
            except FileExistsError as e:
                pass

# 创建指定数量的线程来复制文件
for _ in range(num):
```

```
t = Thread(target=copy, args=('Thread'+str(_),))
t.start()

if __name__ == '__main__':
    # 解析命令行参数
    parser = argparse.ArgumentParser(description='copy files from src
to dst')
    parser.add_argument('-s', '--src')
    parser.add_argument('-d', '--dst')
    parser.add_argument('-n', '--num', default='5')
    args = parser.parse_args()

    if args.src!=None and args.dst!=None:
        copyFile(args.src, args.dst, int(args.num))
    else:
        print('Please use the following command to see how to use:')
        print(' '+sys.argv[0]+' -h')
```

进一步思考：

尝试改写成多进程的版本，并验证是否能够提高整体速度。

实验二十七、Excel 文件数据导入 SQLite 数据库

实验目的：

- 1、熟练运用扩展库 openpyxl 操作 Excel 文件。
- 2、熟悉 SQL 语句的编写。
- 3、熟悉生成器函数的编写和使用。
- 4、理解并运用 time 模块测试代码运行时间。
- 5、熟悉标准库 string、os、os.path、sqlite3、time 的用法。



实验内容：

编写程序，生成 50 个 Excel 文件，每个文件中包含 5 列数据，其中每个单元格内的内容随机生成，并且每个 Excel 文件的数据行数不相同。然后创建一个 SQLite 数据库，其结构与 Excel 文件相符合，最后把前面生成的 50 个 Excel 文件中的数据导入到这个数据库中。

参考代码：

```
from random import choice, randrange
from string import digits, ascii_letters
from os import listdir, mkdir
from os.path import isdir
import sqlite3
from time import time
from openpyxl import Workbook, load_workbook

def generateRandomData():
    ''' 生成测试数据，共 50 个 Excel 文件，每个文件有 5 列随机字符串'''
    # 如果不存在子文件夹 xlsxs，就创建
    if not isdir('xlsxs'):
        mkdir('xlsxs')

    # total 表示记录总条数
    global total

    # 候选字符集
    characters = digits+ascii_letters

    # 生成 50 个 Excel 文件
    for i in range(50):
        xlsName = 'xlsxs\\'+str(i)+'.xlsx'
```

```
# 随机数，每个xlsx文件的行数不一样
totalLines = randrange(10**4)

# 创建Workbook，获取第1个Worksheet
wb = Workbook()
ws = wb.worksheets[0]

# 写入表头
ws.append(['a', 'b', 'c', 'd', 'e'])
# 随机数据，每行5个字段，每个字段30个字符
for j in range(totalLines):
    line = [''.join((choice(characters)
                      for ii in range(30)))
            for jj in range(5)]
    ws.append(line)
    total += 1

# 保存xlsx文件
wb.save(xlsName)

def eachXlsx(xlsxFn):
    '''针对每个xlsx文件的生成器'''
    # 打开Excel文件，获取第1个Worksheet
    wb = load_workbook(xlsxFn)
    ws = wb.worksheets[0]
    for index, row in enumerate(ws.rows):
        # 忽略表头
        if index == 0:
            continue
        yield tuple(map(lambda x:x.value, row))

def xlsx2sqlite():
    '''从批量Excel文件中导入数据到SQLite数据库'''
    # 获取所有xlsx文件名
    xlsxs = ('xlsxs\\'+fn for fn in listdir('xlsxs'))

    # 连接数据库，创建游标
    with sqlite3.connect('dataxlsx.db') as conn:
        cur = conn.cursor()
        for xlsx in xlsxs:
            # 批量导入，减少提交事务的次数，可以提高速度
            sql = 'INSERT INTO fromxlsx VALUES(?,?,?,?,?)'
            cur.executemany(sql, eachXlsx(xlsx))
            conn.commit()
```

```
# 用来记录生成的随机数据的总行数
total = 0

# 生成随机数据
generateRandomData()

# 导入数据，并测试速度
start = time()
xlsx2sqlite()
delta = time()-start

print('导入用时: ', delta)
print('导入速度（条/秒）: ', total/delta)
```

实验二十八、生成棋盘纹理图片

实验目的：

- 1、熟悉 Python 扩展库 pillow 的安装方法。
- 2、熟悉 Python 扩展库 pillow 的简单使用。
- 3、理解棋盘网格纹理的生成原理。

实验内容：

编写程序，绘制棋盘网格，要求棋盘的宽度和高度、交替的两种颜色以及网格数量都可以通过参数指定，并且两种颜色交替出现，水平方向和垂直方向上的网格数量相同。

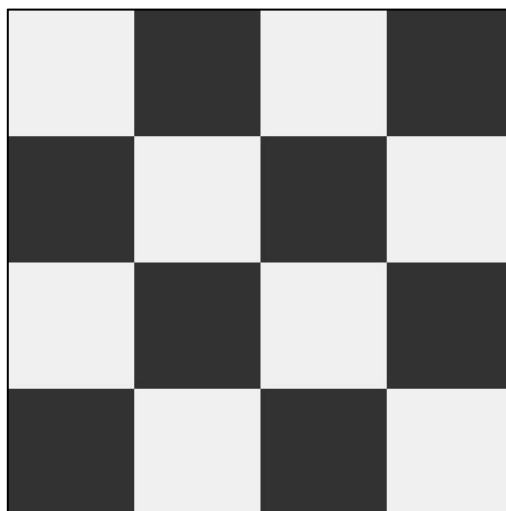
参考代码：

```
from PIL import Image
import math

def qipan(width, height, color1, color2, interval):
    im = Image.new('RGB', (width, height))
    hInterval = height / interval
    wInterval = width / interval
    for h in range(height):
        for w in range(width):
            if (int(h / hInterval) + int(w / wInterval)) % 2 == 1:
                im.putpixel((w, h), color1)
            else:
                im.putpixel((w, h), color2)
    im.show()

qipan(500, 500, (50, 50, 50), (240, 240, 240), 4)
```

效果图：



实验二十九、暴力破解 MD5 值

实验目的：

- 1、理解 MD5 算法原理。
- 2、熟练运用内置函数 print() 的 end 参数。
- 3、了解标准库 hashlib、itertools、time 的用法。
- 4、熟练运用字符串的 join() 和 encode() 方法。

实验内容：

编写程序，使用暴力测试的方法破解一个 MD5 值对应的明文。假设该 MD5 值是对一个长度大于等于 5 且小于 10 的字符串使用 UTF8 编码之后得到的字节串进行加密的结果，要求输出代码运行时间，也就是破解该 MD5 值所需要的时间。

参考代码：

```
from hashlib import md5
from string import ascii_letters, digits
from itertools import permutations
from time import time

all_letters = ascii_letters + digits + '.,;' #候选字符集

def decrypt_md5(md5_value):
    if len(md5_value) != 32: #破解 32 位 MD5 值
        print('error')
        return

    md5_value = md5_value.lower() #转换为小写 MD5 值
    for k in range(5,10): #预期密码长度
        for item in permutations(all_letters, k): #暴力测试
            item = ''.join(item)
            print('.', end='') #显示进度
            if md5(item.encode()).hexdigest() == md5_value:
                return item

md5_value = 'b932ae9220e9a413b39d9782605fee8f'
start = time()
result = decrypt_md5(md5_value)
if result:
    print('\nSuccess: ' + md5_value + '==>' + result)
print('Time used:', time()-start)
```



实验三十、数据分析与可视化综合实验



实验目的：

- 1、熟悉 Python 标准库 csv 的用法。
- 2、熟悉 CSV 和 TXT 文件操作。
- 3、熟练安装扩展库 numpy、pandas、matplotlib。
- 4、熟悉使用扩展库 pandas 进行数据分析的基本操作。
- 5、熟悉使用扩展库 matplotlib 进行数据可视化的基本操作。

实验内容：

(1) 运行下面的程序，在当前文件夹中生成饭店营业额模拟数据文件 data.csv。

```
import csv
import random
import datetime

fn = 'data.csv'

with open(fn, 'w') as fp:
    # 创建 csv 文件写入对象
    wr = csv.writer(fp)
    # 写入表头
    wr.writerow(['日期', '销量'])

    # 生成模拟数据
    startDate = datetime.date(2017, 1, 1)

    # 生成 365 个模拟数据，可以根据需要调整
    for i in range(365):
        # 生成一个模拟数据，写入 csv 文件
        amount = 300 + i*5 + random.randrange(100)
        wr.writerow([str(startDate), amount])
        # 下一天
        startDate = startDate + datetime.timedelta(days=1)
```

(2) 然后完成下面的任务：

- 1) 使用 pandas 读取文件 data.csv 中的数据，创建 DataFrame 对象，并删除其中所有缺失值；
- 2) 使用 matplotlib 生成折线图，反应该饭店每天的营业额情况，并把图形保存为本地文件 first.jpg；
- 3) 按月份进行统计，使用 matplotlib 绘制柱状图显示每个月份的营业额，并把图形

保存为本地文件 `second.jpg`;

4) 按月份进行统计, 找出相邻两个月最大涨幅, 并把涨幅最大的月份写入文件

`maxMonth.txt`;

5) 按季度统计该饭店 2017 年的营业额数据, 使用 `matplotlib` 生成饼状图显示 2017 年 4 个季度的营业额分布情况, 并把图形保存为本地文件 `third.jpg`。

参考代码:

```
import pandas as pd
import matplotlib.pyplot as plt

# 读取数据, 丢弃缺失值
df = pd.read_csv('data.csv', encoding='cp936')
df = df.dropna()

# 生成营业额折线图
plt.figure()
df.plot(x=df['日期'])
plt.savefig('first.jpg')

# 按月统计, 生成柱状图
plt.figure()
df1 = df[:]
df1['month'] = df1['日期'].map(lambda x: x[:x.rindex('-')])
df1 = df1.groupby(by='month', as_index=False).sum()
df1.plot(x=df1['month'], kind='bar')
plt.savefig('second.jpg')

# 查找涨幅最大的月份, 写入文件
df2 = df1.drop('month', axis=1).diff()
m = df2['销量'].nlargest(1).keys()[0]
with open('maxMonth.txt', 'w') as fp:
    fp.write(df1.loc[m, 'month'])

# 按季度统计, 生成饼状图
plt.figure()
one = df1[:3]['销量'].sum()
two = df1[3:6]['销量'].sum()
three = df1[6:9]['销量'].sum()
four = df1[9:12]['销量'].sum()
plt.pie([one, two, three, four], labels=['one', 'two', 'three', 'four'])
plt.savefig('third.jpg')
```