

第一章

一、编程题

- 1、编写程序：hello.py，要求用户输入姓名并打印“你好，姓名！”

```
name = input("请输入您的姓名：")
print("您好，", name, "！")
```

- 2、编写程序：calc.py 要求用户输入 1 到 100 之间数字并判断，输入符合要求打印“你妹好漂亮”，不符合要求则打印“你大爷好丑”

```
temp = input("请输入1-100之间的数字：")
num = int(temp)
if 1<=num<=100:
    print("你妹好漂亮^_^")
else:
    print("你大爷好丑T_T")
```

- 3、如果非要在原始字符串结尾输入反斜杠，可以如何灵活处理？

```
>>>str = r'c:\program Files\Tarena\Good'\'
```

第二章

一、编程题

- 1、Python 可以计算很大很大的数据，但是.....真正的大数据计算可是要靠的硬件滴，写一个小代码，让你的计算机为之崩溃？

答：print(3 ** 3 ** 32)

一般很多机子都会在一会儿之后：Memory Overflow，内存不够用。

设计到幂操作，结果都是惊人滴。

- 2、写一个程序，判断给定年份是否为闰年。

这样定义闰年的：能被 4 整除但不能被 100 整除，或者能被 400 整除都是闰年。

```

year = int(input("请输入一个年份: "))
if ((year % 4) == 0 and (year % 100) != 0
    or (year % 400) == 0):
    print("{0}是闰年".format(year))
else:
    print("{0}不是闰年".format(year))

```

3、请写一个程序打印出 0~100 所有的奇数。

答：

```

i = 0
while i <= 100:
    if i % 2 != 0:
        print(i, end=' ')
        i += 1
    else:
        i += 1

```

4、爱因斯坦曾出过这样一道有趣的数学题：有一个长阶梯，若每步上 2 阶，最后剩 1 阶；若每步上 3 阶，最后剩 2 阶；若每步上 5 阶，最后剩 4 阶；若每步上 6 阶，最后剩 5 阶；只有每步上 7 阶，最后刚好一阶也不剩。（温馨提示：步子太大真的容易扯着蛋~~~）

题目：请编程求解该阶梯至少有多少阶？

答：

```

x = 7
i = 1
flag = 0
while i <= 100:
    if (x%2 == 1) and (x%3 == 2) and (x%5 == 4) and (x%6==5):
        flag = 1
    else:
        x = 7 * (i+1) # 根据题意，x 一定是 7 的整数倍，所以每次乘以 7
        i += 1
if flag == 1:
    print('阶梯数是: ', x)
else:
    print('在程序限定的范围内找不到答案! ')

```

第三章

一、编程题

1、编写一个进制转换程序，程序演示如下（提示，十进制转换二进制可以用 bin() 这个 BIF）：

答:

```
q = True
while q:
    num = input('请输入一个整数(输入 Q 结束程序): ')
    if num != 'Q':
        num = int(num)
        print('十进制 -> 十六进制 : %d -> 0x%x' % (num, num))
        print('十进制 -> 八进制 : %d -> 0o%o' % (num, num))
        print('十进制 -> 二进制 : %d -> %s' % num, bin(num))
    else:
        q = False
```

2、请写一个密码安全性检查的脚本代码: check.py

```
# 密码安全性检查代码
# 低级密码要求:
# 1. 密码由单纯的数字或字母组成
# 2. 密码长度小于等于 8 位
# 中级密码要求:
# 1. 密码必须由数字字母或特殊字符 (仅限: ~!@#%&*()_-=/,.?<>;:[]{}|\) 任意两种组合
# 2. 密码长度不能低于 8 位
# 高级密码要求:
# 1. 密码必须由数字、字母及特殊字符 (仅限: ~!@#%&*()_-=/,.?<>;:[]{}|\) 三种组合
# 2. 密码只能由字母开头
# 3. 密码长度不能低于 16 位
```

答:

```
symbols = r"!@#%&*()_+~/*{}[]|'";:/,.<>"
chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
nums = '0123456789'
passwd = input('请输入需要检查的密码组合: ')
# 判断长度
length = len(passwd)
while (passwd.isspace() or length == 0):
    passwd = input("您输入的密码为空 (或空格), 请重新输入: ")
if length <= 8:
    flag_len = 1
elif 8 < length < 16:
    flag_len = 2
else:
    flag_len = 3
flag_con = 0
# 判断是否包含特殊字符
for each in passwd:
    if each in symbols:
        flag_con += 1
    break
```

```

# 判断是否包含字母
for each in passwd:
    if each in chars:
        flag_con += 1
        break
# 判断是否包含数字
for each in passwd:
    if each in nums:
        flag_con += 1
        break
# 打印结果
while 1:
    print("您的密码安全级别评定为: ", end="")
    if flag_len == 1 or flag_con == 1:
        print("低")
    elif flag_len == 2 or flag_con == 2:
        print("中")
    else:
        print("高")
        print("请继续保持")
        break
    print("请按以下方式提升您的密码安全级别: \n\
\t1. 密码必须由数字、字母及特殊字符三种组合\n\
\t2. 密码只能由字母开头\n\
\t3. 密码长度不能低于 16 位")
    break

```

第四章

一、编程题

1、猜想一下 min() 这个 BIF 的实现过程

答:

```

def min(x):
    least = x[0]
    for each in x:
        if each < least:
            least = each
    return least

print(min('123456789'))

```

2、自己动手试试看，并分析在这种情况下，向列表添加数据应当采用哪种方法比较好？

假设给定以下列表：

```
member = ['tarena', 'python', '迷途', '怡静', '秋舞斜阳']
```

要求将列表修改为：

```
member = ['tarena ', 88, 'python', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]
```

方法一：使用 insert() 和 append() 方法修改列表。

方法二：重新创建一个同名字的列表覆盖。

答：

方法一：

```
member.insert(1, 88)
member.insert(3, 90)
member.insert(5, 85)
member.insert(7, 90)
member.append(88)
```

方法二：

```
member = ['tarena ', 88, 'python', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]
```

对于这种情况，明显是第二种方法看起来要好一些嘛。

不过对于大型的列表，第一种方法可能更合适，所以我们说永远没有最好的，只有最合适的。

3、按照 100 分制，90 分以上成绩为 A，80 到 90 为 B，60 到 80 为 C，60 以下为 D，写一个程序，当用户输入分数，自动转换为 ABCD 的形式打印。Tarena 使用 if elif else 在大多数情况下效率要比全部使用 if 要高，但根据一般的统计规律，一个班的成绩一般服从正态分布，也就是说平均成绩一般集中在 70~80 分之间，因此根据统计规律，我们还可以改进下程序以提高效率。

答：

```
score = int(input('请输入一个分数: '))
if 80 > score >= 60:
    print('C')
elif 90 > score >= 80:
    print('B')
elif 60 > score >= 0:
    print('D')
elif 100 >= score >= 90:
    print('A')
else:
    print('输入错误!')
```

4、Python 的作者在很长一段时间不肯加入三元操作符就是怕跟 C 语言一样搞出国际乱码大赛，蛋疼的复杂度让初学者望而生畏，不过，如果你一旦搞清楚了三元操作符的使用技巧，或许一些比较复杂的问题反而迎刃而解。

请将以下代码修改为三元操作符实现：

```
x, y, z = 6, 5, 4
if x < y:
    small = x
```

```

        if z < small:
            small = z
    elif y < z:
        small = y
    else:
        small = z

```

答:

```
small = x if (x < y and x < z) else (y if y < z else z)
```

第五章

一、编程题

1、三色球问题

有红、黄、蓝三种颜色的球，其中红球 3 个，黄球 3 个，绿球 6 个。先将这 12 个球混合放在一个盒子中，从中任意摸出 8 个球，编程计算摸出球的各种颜色搭配。

答:

```

print('red\tyellow\tblue')
for red in range(0, 4):
    for yellow in range(0, 4):
        for green in range(2, 7):
            if red + yellow + green == 8:
                # 注意，下边不是字符串拼接，因此不用“+”哦~
                print(red, '\t', yellow, '\t', green)

```

注释: range(2, 7)是产生[2, 3, 4, 5, 6]5 个数，绿球不能是 1 个，因为如果绿球是 1 个的话，红球 + 黄球需要有 7 个才能符合题意，而红球和黄球每种只有 3 个，因此是 range(2, 7)

2、编写一个程序，求 100~999 之间的所有水仙花数。

如果一个 3 位数等于其各位数字的立方和，则称这个数为水仙花数。例如: $153 = 1^3 + 5^3 + 3^3$ ，因此 153 就是一个水仙花数。

如果一个 3 位数等于其各位数字的立方和，则称这个数为水仙花数。例如: $153 = 1^3 + 5^3 + 3^3$ ，因此 153 就是一个水仙花数

答:

```

for i in range(100, 1000):
    sum = 0
    temp = i
    while temp:
        sum = sum + (temp%10) ** 3
        temp //= 10    # 注意这里要使用地板除哦~
    if sum == i:
        print(i)

```

第六章

一、编程题

1、编写一个将十进制转换为二进制的函数，要求采用“除 2 取余”（脑补链接）的方式，结果与调用 `bin()` 一样返回字符串形式。

```
1. def Dec2Bin(dec):
2.     temp = []
3.     result = ''
4.
5.     while dec:
6.         quo = dec % 2
7.         dec = dec // 2
8.         temp.append(quo)
9.
10.    while temp:
11.        result += str(temp.pop())
12.
13.    return result
14.
15. print(Dec2Bin(62))
```

2、编写一个函数 `power()` 模拟内建函数 `pow()`，即 `power(x, y)` 为计算并返回 `x` 的 `y` 次幂的值。

```
1. def power(x, y):
2.     result = 1
3.
4.     for i in range(y):
5.         result *= x
6.
7.     return result
8.
9. print(power(2, 3))
```

3、编写一个函数，利用欧几里得算法（[脑补链接](#)）求最大公约数，例如 gcd(x, y) 返回值为参数 x 和参数 y 的最大公约数。

```
1. def gcd(x, y):
2.     while y:
3.         t = x % y
4.         x = y
5.         y = t
6.
7.     return x
8.
9. print(gcd(4, 6))
```

第七章

一、编程题

1、还记得求回文字符串那道题吗？现在让你使用递归的方式来求解。

解题思路：方法很多，朴素性的做法是利用递归每次索引前后两个字符进行对比，当 start>end 的时候，也正是首尾下标“碰面”的时候，即作为结束递归的条件。

参考代码：

```
01. def is_palindrome(n, start, end):
02.     if start > end:
03.         return 1
04.     else:
05.         return is_palindrome(n, start+1, end-1) if n[start] == n[end] else 0
06.
07. string = input('请输入一串字符串：')
08. length = len(string)-1
09.
10. if is_palindrome(string, 0, length):
11.     print('\n%s\''是回文字符串!' % string)
12. else:
13.     print('\n%s\''不是回文字符串!' % string)
14.
```

[复制代码](#)

2、使用递归编程求解以下问题：

有 5 个人坐在一起，问第五个人多少岁？他说比第四个人大二岁，第四个人比第三个人大两岁，第三个人又比第二个人大两岁，第二个人又比第一个人大两岁。最后问第一个人，他说自己是 10 岁。请问第五个人多大？

解题思路：利用递归的方法，递归分为回推和递推两个阶段。要想知道第五个人的岁数，以此类推，推到第一个人(10 岁)，再往回推。

参考代码：

```
01. def age(n):
02.     if n == 1:
03.         return 10
04.     else:
05.         return age(n-1) + 2
06.
07. print('哈哈，我知道了，第五个人的年龄是 %d 岁，啧啧脆！' % age(5))
08.
```

[复制代码](#)

3、写一个函数 `get_digits(n)`，将参数 `n` 分解出每个位的数字并按顺序存放在列表当中。举例：`get_digits(12345) ==> [1, 2, 3, 4, 5]`

解题思路：利用除以 10 取余数的方式，每次调用 `get_digits(n//10)`，并将余数存放到列表中即可。要注意的是结束条件设置正确。

参考代码：

```
01. result = []
02. def get_digits(n):
03.     if n > 0:
04.         result.insert(0, n%10)
05.         get_digits(n//10)
06.
07. get_digits(12345)
08. print(result)
09.
```

[复制代码](#)

4、使用递归编写一个十进制转换为二进制的函数(要求采用“取 2 取余”的方式，结果与调用 `bin()` 一样返回字符串形式)。

参考代码：

```
01. def Dec2Bin(dec):
02.     result = ''
03.
04.     if dec:
05.         result = Dec2Bin(dec//2)
06.         return result + str(dec%2)
07.     else:
08.         return result
09.
10. print(Dec2Bin(62))
11.
```

[复制代码](#)

第八章

一、编程题

1、尝试利用字典的特性编写一个通讯录程序吧，功能如图：

动动手：

0. 尝试利用字典的特性编写一个通讯录程序吧，功能如图：

```
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
|--- 欢迎进入通讯录程序 ---|
|--- 1: 查询联系人资料 ---|
|--- 2: 插入新的联系人 ---|
|--- 3: 删除已有联系人 ---|
|--- 4: 退出通讯录程序 ---|

请输入相关的指令代码：2
请输入联系人姓名：
请输入用户联系电话：020-88974651

请输入相关的指令代码：1
请输入联系人姓名：
小甲鱼：020-88974651

请输入相关的指令代码：2
请输入联系人姓名：
您输入的姓名在通讯录中已存在 -->>      : 020-88974651
是否修改用户资料 (YES/NO) : YES
请输入用户联系电话：020-88974563
```

参考代码：

```
01.
02. print('|--- 欢迎进入通讯录程序 ---|')
03. print('|--- 1: 查询联系人资料 ---|')
04. print('|--- 2: 插入新的联系人 ---|')
05. print('|--- 3: 删除已有联系人 ---|')
06. print('|--- 4: 退出通讯录程序 ---|')
07.
08. contacts = dict()
09.
10. while 1:
11.     instr = int(input('\n请输入相关的指令代码：'))
12.
13.     if instr == 1:
14.         name = input('请输入联系人姓名：')
15.         if name in contacts:
16.             print(name + '：' + contacts[name])
17.         else:
18.             print('您输入的姓名不再通讯录中！')
19.
20.     if instr == 2:
21.         name = input('请输入联系人姓名：')
22.         if name in contacts:
```

```

23.         print('您输入的姓名在通讯录中已存在 -->> ', end='')
24.         print(name + ' : ' + contacts[name])
25.         if input('是否修改用户资料 (YES/NO) : ') == 'YES':
26.             contacts[name] = input('请输入用户联系电话 : ')
27.         else:
28.             contacts[name] = input('请输入用户联系电话 : ')
29.
30.     if instr == 3:
31.         name = input('请输入联系人姓名 : ')
32.         if name in contacts:
33.             del(contacts[name])          # 也可以使用dict.pop()
34.         else:
35.             print('您输入的联系人不存在。')
36.
37.     if instr == 4:
38.         break
39.
40. print('|--- 感谢使用通讯录程序 ---|')
41.
42.

```

复制代码

第九章

一、编程题

1、尝试编写一个用户登录程序(这次尝试将功能封装成函数)，程序实现如图：

动动手：

0. 尝试编写一个用户登录程序（这次尝试将功能封装成函数），程序实现如图：

```

Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit
D64] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>

|--- 新建用户 : N/n ---|
|--- 登录账号 : E/e ---|
|--- 推出程序 : Q/q ---|
|--- 请输入指令代码 : N
请输入用户名 :
请输入密码 :
注册成功，赶紧试试登录吧^_^

|--- 新建用户 : N/n ---|
|--- 登录账号 : E/e ---|
|--- 推出程序 : Q/q ---|
|--- 请输入指令代码 : n
请输入用户名 :
此用户名已经被使用，请重新输入：
请输入密码 :
注册成功，赶紧试试登录吧^_^

|--- 新建用户 : N/n ---|
|--- 登录账号 : E/e ---|
|--- 推出程序 : Q/q ---|
|--- 请输入指令代码 : E
请输入用户名 :

```

您输入的用户名不存在，请重新输入：
请输入密码：
欢迎进入xxoo系统，请点击右上角的x结束程序！

参考代码：

```
01.
02.     user_data = {}
03.
04.     def new_user():
05.         prompt = '请输入用户名：'
06.         while True:
07.             name = input(prompt)
08.             if name in user_data:
09.                 prompt = '此用户名已经被使用，请重新输入：'
10.                 continue
11.             else:
12.                 break
13.
14.         passwd = input('请输入密码：')
15.         user_data[name] = passwd
16.         print('注册成功，赶紧试试登录吧^_^')
17.
18.     def old_user():
19.         prompt = '请输入用户名：'
20.         while True:
21.             name = input(prompt)
22.             if name not in user_data:
```

第十章

一、编程题

1、尝试将一个音频文件（openme.mp3）打印到屏幕上

答：直接使用打开文本文件的形式打开即可，至于为什么？稍后便知

```
01.     f = open('OpenMe.mp3')
02.     for each_line in f:
03.         print(each_line, end='')
04.     f.close()
05.
```

复制代码

2、编写代码，将上一题中的文件 openme.mp3 保存为新文件 openme.txt

答

:

```
01.     f1 = open('OpenMe.mp3')
02.     f2 = open('OpenMe.txt', 'x')          # 使用"x"打开更安全
03.     f2.write(f1.read())
04.     f2.close()
05.     f1.close()
06.
```

复制代码

第十一章

一、编程题

1、编写一个程序比较用户输入的两个文件，如果不同，显示出所有不同处的行号与第一个不同字符的位置，程序实现如图：

```
>>> ===== RESTART =====
>>>
请输入需要比较的头一个文件名: something.txt
请输入需要比较的另一个文件名: something2.txt
两个文件共有【4】处不同:
第 4 行不一样
第 7 行不一样
第 16 行不一样
第 17 行不一样
>>> |
```

```
01. def file_compare(file1, file2):
02.     f1 = open(file1)
03.     f2 = open(file2)
04.     count = 0 # 统计行数
05.     differ = [] # 统计不一样的数量
06.
07.     for line1 in f1:
08.         line2 = f2.readline()
09.         count += 1
10.         if line1 != line2:
11.             differ.append(count)
12.
13.     f1.close()
14.     f2.close()
15.     return differ
16.
17. file1 = input('请输入需要比较的头一个文件名:')
18. file2 = input('请输入需要比较的另一个文件名:')
19.
20. differ = file_compare(file1, file2)
21.
22. if len(differ) == 0:
23.     print('两个文件完全一样! ')
24. else:
25.     print('两个文件共有【%d】处不同:' % len(differ))
26.     for each in differ:
27.         print('第 %d 行不一样' % each)
28.
复制代码
```

2、编写一个程序，当用户输入文件名和行号数（N）后，将该文件的前 N 行内容打印到屏幕上，程序实现如下：

请输入要打开的文件：/home/tarena/a.txt

请输入需要显示该文件前几行：12

文件/home/tarena/a.txt 的前 12 行的内容如下：

小二：老王，今天有客户问你有没有女朋友？

老王：咦？？

小二：我跟她说你已经有女朋友啦！

老王：。。。。。。

小二：她让你分手后考虑下她！然后我说：“您要买个优盘，我帮您留意下~”

老王：然后呢？

小二：她买了两个，说发一个货就好~

老王：呃。。。你真牛！

小二：那是，谁让我是最可爱的小二嘛~

老王：下次有人想调戏你我不阻止~

小二：滚!!!

```
01. def file_view(file_name, line_num):
02.     print('\n文件%s的前%s的内容如下:\n' % (file_name, line_num))
03.     f = open(file_name)
04.     for i in range(int(line_num)):
05.         print(f.readline(), end= '')
06.
07.     f.close()
08.
09. file_name = input(r'请输入要打开的文件(C:\test.txt): ')
10. line_num = input('请输入需要显示该文件前几行: ')
11. file_view(file_name, line_num)
12.
```

3、呃，不得不说我们的用户变得越来越刁钻啦。要在上一题的基础上扩展，用户可以随意输入需要显示的行数。（如输入 13:21 打印 13 行到 21 行，输入:21 打印前 21 行，输入:21 打印从 21 行开始到文件结尾所有内容）

```
01. def file_view(file_name, line_num):
02.     if line_num.strip() == ':':
03.         begin = '1'
04.         end = '-1'
05.
06.     (begin, end) = line_num.split(':')
07.
08.     if begin == '':
09.         begin = '1'
10.     if end == '':
11.         end = '-1'
12.
13.     if begin == '1' and end == '-1':
14.         prompt = '的全文'
15.     elif begin == '1':
16.         prompt = '从开始到%s' % end
17.     elif end == '-1':
18.         prompt = '从%s到结束' % begin
19.     else:
20.         prompt = '从第%s行到第%s行' % (begin, end)
21.
```

```

22.     print('\n文件%s%s的内容如下:\n' % (file_name, prompt))
23.
24.     begin = int(begin) - 1
25.     end = int(end)
26.     lines = end - begin
27.
28.     f = open(file_name)
29.
30.     for i in range(begin): # 用于消耗掉begin之前的内容
31.         f.readline()
32.
33.     if lines < 0:
34.         print(f.read())
35.     else:
36.         for j in range(lines):
37.             print(f.readline(), end='')
38.
39.     f.close()
40.
41.     file_name = input(r'请输入要打开的文件(C:\\test.txt): ')
42.     line_num = input('请输入需要显示的行数【格式如 13:21 或 :21 或 21: 或 :】: ')
43.     file_view(file_name, line_num)

```

4、编写一个程序，实现“全部替换”功能，程序实现如图：

```

>>> ===== RESTART =====
>>>
请输入文件名: something.txt
请输入需要替换的单词或字符: 愿
请输入新的单词或字符: 希望

文件 something.txt 中共有4个【愿】
您确定要把所有的【愿】替换为【希望】吗?
【YES/NO】: yes

```

```

01.
02. def file_replace(file_name, rep_word, new_word):
03.     f_read = open(file_name)
04.
05.     content = []
06.     count = 0
07.
08.     for eachline in f_read:
09.         if rep_word in eachline:
10.             count = eachline.count(rep_word) #count感觉应该用这个
11.             eachline = eachline.replace(rep_word, new_word)
12.             content.append(eachline)
13.
14.     decide = input('\n文件 %s 中共有%s个【%s】\n您确定要把所有的【%s】替换为【%s】吗? \n【YES/NO】: ' \
15.                    % (file_name, count, rep_word, rep_word, new_word))
16.
17.     if decide in ['YES', 'Yes', 'yes']:
18.         f_write = open(file_name, 'w')
19.         f_write.writelines(content)
20.         f_write.close()
21.
22.     f_read.close()
23.
24.

```

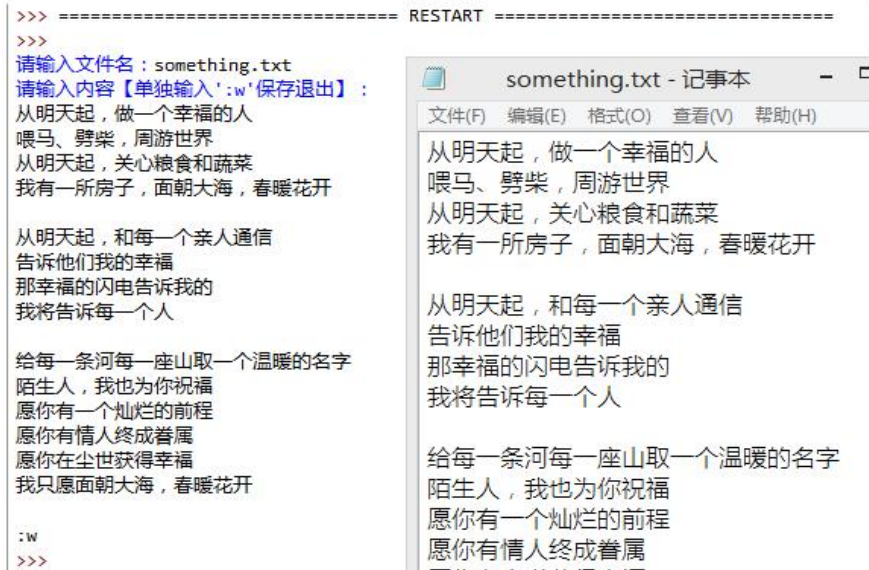


```

25. file_name = input('请输入文件名:')
26. rep_word = input('请输入需要替换的单词或字符:')
27. new_word = input('请输入新的单词或字符:')
28. file_replace(file_name, rep_word, new_word)
29.

```

5、编写一个程序，接收用户的输入并保存为新的文件，程序实现如图：



The image shows a Python REPL on the left and a Notepad window titled 'something.txt - 记事本' on the right. The REPL shows the user inputting the filename 'something.txt', the word to replace ':w', and then pasting a poem. The Notepad window shows the same poem, indicating it has been saved or replaced.

```

>>> ===== RESTART =====
>>>
请输入文件名: something.txt
请输入内容【单独输入':w'保存退出】:
从明天起，做一个幸福的人
喂马、劈柴，周游世界
从明天起，关心粮食和蔬菜
我有一所房子，面朝大海，春暖花开

从明天起，和每一个亲人通信
告诉他们我的幸福
那幸福的闪电告诉我的
我将告诉每一个人

给每一条河每一座山取一个温暖的名字
陌生人，我也为你祝福
愿你有一个灿烂的前程
愿你有情人终成眷属
愿你在尘世获得幸福
我只愿面朝大海，春暖花开

:w
>>>

```

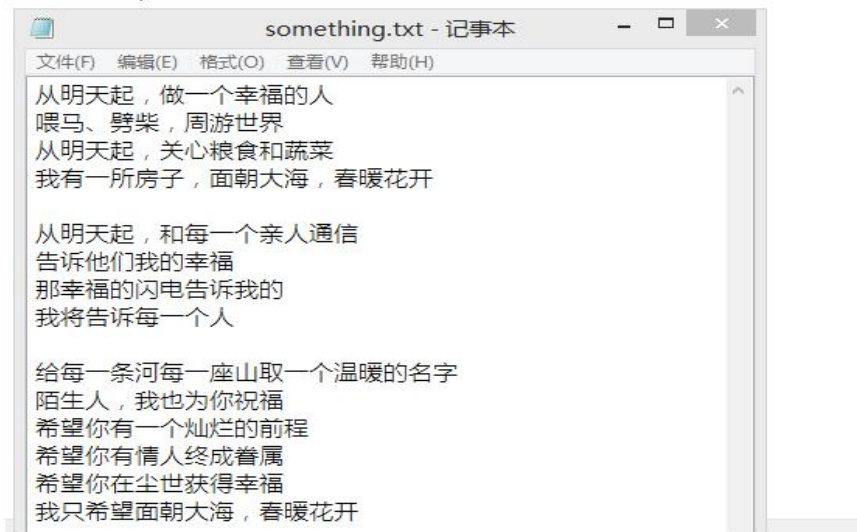
something.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

从明天起，做一个幸福的人
喂马、劈柴，周游世界
从明天起，关心粮食和蔬菜
我有一所房子，面朝大海，春暖花开

从明天起，和每一个亲人通信
告诉他们我的幸福
那幸福的闪电告诉我的
我将告诉每一个人

给每一条河每一座山取一个温暖的名字
陌生人，我也为你祝福
愿你有一个灿烂的前程
愿你有情人终成眷属
愿你在尘世获得幸福



The image shows a Notepad window titled 'something.txt - 记事本' with the same poem content as the previous screenshot, confirming it has been saved.

something.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

从明天起，做一个幸福的人
喂马、劈柴，周游世界
从明天起，关心粮食和蔬菜
我有一所房子，面朝大海，春暖花开

从明天起，和每一个亲人通信
告诉他们我的幸福
那幸福的闪电告诉我的
我将告诉每一个人

给每一条河每一座山取一个温暖的名字
陌生人，我也为你祝福
希望你有一个灿烂的前程
希望你情人终成眷属
希望你在尘世获得幸福
我只希望面朝大海，春暖花开


```

01. def file_write(file_name)
02.     f = open(file_name, 'w')
03.     print('请输入内容【单独输入\':w\'保存退出】:')
04.
05.     while True:
06.         write_some = input()
07.         if write_some != ':w':
08.             f.write('%s\n' % write_some)
09.         else:
10.             break
11.
12.     f.close()
13.
14. file_name = input('请输入文件名:')
15. file_write(file_name)
16.

```

第十二章

一、编程题

1、编写一个程序，统计当前目录下每个文件类型的文件数，程序实现如图：

```

>>> ===== RESTART =====
>>>
该文件夹下共有类型为【.txt】的文件 1 个
该文件夹下共有类型为【.png】的文件 2 个
该文件夹下共有类型为【.py】的文件 3 个
该文件夹下共有类型为【.docx】的文件 2 个
该文件夹下共有类型为【文件夹】的文件 2 个
>>>

```

```

01. import os
02.
03. all_files = os.listdir(os.getcwd()) # 使用os.getcwd表示当前目录更标准
04. type_dict = dict()
05.
06. for each_file in all_files:
07.     if os.path.isdir(each_file):
08.         type_dict.setdefault('文件夹', 0)
09.         type_dict['文件夹'] += 1
10.     else:
11.         ext = os.path.splitext(each_file)[1]
12.         type_dict.setdefault(ext, 0)
13.         type_dict[ext] += 1
14.
15. for each_type in type_dict.keys():
16.     print('该文件夹下共有类型为【%s】的文件 %d 个' % (each_type, type_dict[each_type]))
17.

```

2、编写一个程序，计算当前文件夹下所有文件的大小，程序实现如图：

```

Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
0.png [8450Bytes]
vediolist.txt [503Bytes]
2.png [6791Bytes]
3.png [45800Bytes]
homework_3.py [760Bytes]
5.png [26154Bytes]
homework_1.py [354Bytes]
测试1.txt [402Bytes]
homework_0.py [528Bytes]
~$课后作业.docx [162Bytes]
课后作业.docx [17195Bytes]
homework_5.py [19248Bytes]
homework_2.py [576Bytes]
>>>

```

```

01. import os
02.
03. all_files = os.listdir(os.curdir) # 使用os.curdir表示当前目录更标准
04. file_dict = dict()
05.
06. for each_file in all_files:
07.     if os.path.isfile(each_file):
08.         file_size = os.path.getsize(each_file)
09.         file_dict[each_file] = file_size
10.
11. for each in file_dict.items():
12.     print('%s【%dBytes】' % (each[0], each[1]))
13.

```

3、编写一个程序，用户输入文件名以及开始搜索的路径，搜索文件是否存在，如遇到文件夹，则进入文件夹继续搜索，程序实现如图：

```

>>> ===== RESTART =====
>>>
请输入待查找的初始目录：E:\\TestFolder
请输入需要查找的目标文件：测试3.txt
E:\\TestFolder\\SubFolder1\\SubFolder3\\测试3.txt
E:\\TestFolder\\SubFolder2\\测试3.txt
>>>

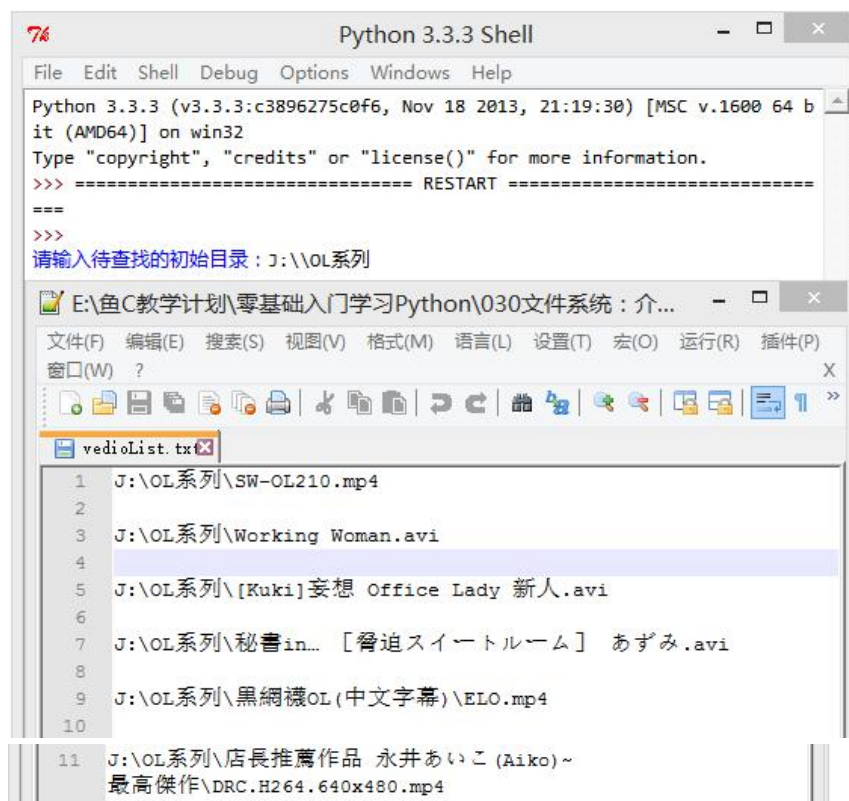
```

```

01. import os
02.
03. def search_file(start_dir, target) :
04.     os.chdir(start_dir)
05.
06.     for each_file in os.listdir(os.curdir) :
07.         if each_file == target :
08.             print(os.getcwd() + os.sep + each_file) # 使用os.sep是程序更标准
09.         if os.path.isdir(each_file) :
10.             search_file(each_file, target) # 递归调用
11.         os.chdir(os.pardir) # 递归调用后切记返回上一层目录
12.
13. start_dir = input('请输入待查找的初始目录：')
14. target = input('请输入需要查找的目标文件：')
15. search_file(start_dir, target)
16.

```

4、编写一个程序，用户输入开始搜索的路径，查找该路径下（包含子文件夹内）所有的视频格式文件（要求查找 mp4rmvb, avi 的格式即可），并把创建一个文件（vedioList.txt）存放所有找到的文件的路径，程序实现如图：



```

01. import os
02.
03. def search_file(start_dir, target) :
04.     os.chdir(start_dir)
05.
06.     for each_file in os.listdir(os.getcwd()) :
07.         ext = os.path.splitext(each_file)[1]
08.         if ext in target :
09.             vedio_list.append(os.getcwd() + os.sep + each_file + os.linesep) # 使用os.sep是程序更标准
10.
11.         if os.path.isdir(each_file) :
12.             search_file(each_file, target) # 递归调用
13.             os.chdir(os.pardir) # 递归调用后切记返回上一层目录
14.
15. start_dir = input('请输入待查找的初始目录:')
16. program_dir = os.getcwd()
17. target = ['.mp4', '.avi', '.rmvb']
18. vedio_list = []

```

```

19.
20. search_file(start_dir, target)
21.
22. f = open(program_dir + os.sep + 'vediolist.txt', 'w')
23. f.writelines(vedio_list)
24. f.close()
25.

```

5、编写一个程序，用户输入关键字，查找当前文件夹内（如果当前文件夹内包含文件夹，则进入文件夹继续搜索）所有该关键字的文本文件（.txt 后缀），要求显示该文件所在的位置以及关键字在文件中的具体位置（第几行第几个字符）程序实现如图：

```
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
请将该脚本放于待查找的文件夹内，请输入关键字：tedu
请问是否需要打印关键字【tedu】在文件中的具体位置（YES/NO）：yes
=====
在文件【E:\TestFolder\测试1.txt】中找到关键字【tedu】
关键字出现在第 13 行，第 [3, 10, 18] 个位置。
=====
在文件【E:\TestFolder\SubFolder1\SubFolder3\测试3.txt】中找到关键字【tedu】
关键字出现在第 6 行，第 [3, 7] 个位置。
关键字出现在第 12 行，第 [3, 7, 13] 个位置。
关键字出现在第 16 行，第 [1] 个位置。
=====
在文件【E:\TestFolder\SubFolder2\测试2.txt】中找到关键字【tedu】
关键字出现在第 2 行，第 [3, 7] 个位置。
关键字出现在第 3 行，第 [1] 个位置。
>>>
```

```
01. import os
02.
03. def print_pos(key_dict):
04.     keys = key_dict.keys()
05.     keys = sorted(keys) # 由于字典是无序的，我们这里对行数进行排序
06.     for each_key in keys:
07.         print('关键字出现在第 %s 行，第 %s 个位置。' % (each_key, str(key_dict[each_key])))
08.
09.
10. def pos_in_line(line, key):
11.     pos = []
12.     begin = line.find(key)
13.     while begin != -1:
14.         pos.append(begin + 1) # 用户的角度是从1开始数
15.         begin = line.find(key, begin+1) # 从下一个位置继续查找
16.
17.     return pos
18.
```

```

19.
20. def search_in_file(file_name, key):
21.     f = open(file_name)
22.     count = 0 # 记录行数
23.     key_dict = dict() # 字典，用户存放key所在具体行数对应具体位置
24.
25.     for each_line in f:
26.         count += 1
27.         if key in each_line:
28.             pos = pos_in_line(each_line, key) # key在每行对应的位置
29.             key_dict[count] = pos
30.
31.     f.close()
32.     return key_dict
33.
34.
35. def search_files(key, detail):
36.     all_files = os.walk(os.getcwd())
37.     txt_files = []
38.
39.     for i in all_files:
40.         for each_file in i[2]:
41.             if os.path.splitext(each_file)[1] == '.txt': # 根据后缀判断是否文本文件
42.                 each_file = os.path.join(i[0], each_file)
43.                 txt_files.append(each_file)
44.
45.
46.         for each_txt_file in txt_files:
47.             key_dict = search_in_file(each_txt_file, key)
48.             if key_dict:
49.                 print('=====')
50.                 print('在文件【%s】中找到关键字【%s】' % (each_txt_file, key))
51.                 if detail in ['YES', 'Yes', 'yes']:
52.                     print_pos(key_dict)
53.
54. key = input('请将脚本放于待查找的文件夹内，请输入关键字：')
55. detail = input('请问是否需要打印关键字【%s】在文件中的具体位置（YES/NO）：' % key)
56. search_files(key, detail)
57.

```

第十三章

一、编程题

1、猜数字游戏


```

01. import random
02.
03. secret = random.randint(1,10)
04. print('-----我爱 达内 -----')
05. temp = input("不妨猜一下 老王 现在心里想的是哪个数字：")
06. try:
07.     guess = int(temp)
08. except ValueError:
09.     print('输入错误! ')
10.     guess = secret
11. while guess != secret:
12.     temp = input("哎呀，猜错了，请重新输入吧：")
13.     guess = int(temp)
14.     if guess == secret:
15.         print("我草，你是 老王 心里的蛔虫吗?! ")
16.         print("哼，猜中了也没有奖励! ")
17.     else:
18.         if guess > secret:
19.             print("哥，大了大了~~~")
20.         else:
21.             print("嘿，小了，小了~~~")
22. print("游戏结束，不玩啦^_^")
23.
复制代码

```

2、还记得我们第一个小游戏吗？只要用户输入非整数，程序立刻就会蹦出不和谐的异常信息然后崩溃，请使用刚学的异常处理方法修改以下程序，提高用户体验。

2. 尝试一个新的函数 `int_input()`，当用户输入整数的时候正常返回，否则提示出错并要求重新输入。

程序实现如图：

`int_input('请输入一个整数：')`

```

Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:19:30) [MSC v.1600 64 bit
D64] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
请输入一个整数：FishC
出错，您输入的不是整数！
请输入一个整数：54321
>>>

```

```

01. def int_input(prompt=''):
02.     while True:
03.         try:
04.             int(input(prompt))
05.             break
06.         except ValueError:
07.             print('出错，您输入的不是整数! ')
08.
09. int_input('请输入一个整数：')
10.

```

3. 把文件关闭放在 **finally** 语句块中执行还是会出现问题，像下边这个代码，当前文件夹中并不存在 "My_File.txt" 这个文件，那么程序执行起来会发生什么事情呢？你有办法解决这个问题吗？

答：由于 finally 语句块里试图去关闭一个并没有成功打开的文件，因此会弹出错误内容如下：

```
01. >>> 出错啦：[Errno 2] No such file or directory: 'My_File.txt'
02. Traceback (most recent call last):
03.   File "C:\Users\FishC000\Desktop\test.py", line 7, in <module>
04.     f.close()
05. NameError: name 'f' is not defined
06.
    复制代码
```

我们这么修正：

```
01. try:
02.     f = open('My_File.txt') # 当前文件夹中并不存在 "My_File.txt" 这个文件T_T
03.     print(f.read())
04. except OSError as reason:
05.     print('出错啦：' + str(reason))
06. finally:
07.     if 'f' in locals(): # 如果文件对象变量存在当前局部变量符号表的话，说明打开成功
08.         f.close()
09.
    复制代码
```

3、在 python 中，else 语句能跟哪些语句进行搭配？

答：在 python 中，else 语句不仅能跟 if 语句搭配，构成“要么怎么样，要么不怎么样的语境”，它还能循环语句，构成“干完了能怎样，干不完就别想怎样”，的语境；其实 else 语句还能够跟我们刚刚讲的异常处理进行搭配，构成“没有问题 那就干吧”的语境。

1、请问以下例子中，循环中的 break 语句会跳过 else 语句吗？

```
01. def showMaxFactor(num):
02.     count = num // 2
03.     while count > 1:
04.         if num % count == 0:
05.             print('%d最大的约数是%d' % (num, count))
06.             break
07.         count -= 1
08.     else:
09.         print('%d是素数!' % num)
10.
11. num = int(input('请输入一个数：'))
12. showMaxFactor(num)
```

答：会，因为如果将 else 语句与循环语句（while 和 for 语句）进行搭配，那么只有在循环正常执行完成后才会执行 else 语句块的内容。

4、以下代码会打印什么内容？

```

01. try:
02.     print('ABC')
03. except:
04.     print('DEF')
05. else:
06.     print('GHI')
07. finally:
08.     print('JKL')
09.

```

答：只有 except 语句中的内容不打印，因为 try 语句块中并没有异常，则 else 语句块也会被执行。

ABC

GHI

JKL

5、先练练手，把我们的刚开始的那个猜数字小游戏加上界面吧？



答案：代码，

```

01. import random
02. import easygui as g
03.
04. g.msgbox("嗨，欢迎进入第一个界面小游戏^_^")
05. secret = random.randint(1,10)
06.
07. msg = "不妨猜一下达内现在心里想的是哪个数字 (1-10) : "
08. title = "数字小游戏"
09. guess = g.integerbox(msg, title, lowerbound=1, upperbound=10)
10.
11. while True:
12.     if guess == secret:
13.         g.msgbox("我草，你是达内心里的蛔虫吗?! ")
14.         g.msgbox("哼，猜中了也没有奖励! ")
15.         break
16.     else:
17.         if guess > secret:
18.             g.msgbox("哥，大了大了~~~")
19.         else:
20.             g.msgbox("嘿，小了，小了~~~")
21.         guess = g.integerbox(msg, title, lowerbound=1, upperbound=10)
22.
23. g.msgbox("游戏结束，不玩啦^_^")
24.

```


6、如下图，实现一个用于登记用户账号信息的界面（如果是带*号的必填项，要求一定要有输入并且不能是空格）。

74 账号中心

【*真实姓名】为必填项。

【*手机号码】为必填项。

【*E-mail】为必填项。

*用户名 达内

*真实姓名

固定电话

*手机号码

QQ

*E-mail

OK Cancel

答案：

```
01. import easygui as g
02.
03. msg = "请填写以下联系方式"
04. title = "账号中心"
05. fieldNames = ["*用户名", "*真实姓名", "固定电话", "*手机号码", "QQ", "*E-mail"]
06. fieldValues = []
07. fieldValues = g.multenterbox(msg, title, fieldNames)
08.
09. while 1:
10.     if fieldValues == None:
11.         break
12.     errmsg = ""
13.     for i in range(len(fieldNames)):
14.         option = fieldNames[i].strip()
15.         if fieldValues[i].strip() == "" and option[0] == "*":
16.             errmsg += ('【%s】为必填项。\\n\\n' % fieldNames[i])
17.     if errmsg == "":
18.         break
19.     fieldValues = g.multenterbox(errmsg, title, fieldNames, fieldValues)
20.
21. print("用户资料如下：%s" % str(fieldValues))
22.
```

7、提供一个文件夹浏览框，让用户选择需要打开的文本文件，打开并显示文件内容。



答案：

```
01. import easygui as g
02. import os
03.
04. file_path = g.fileopenbox(default="*.txt")
05.
06. with open(file_path) as f:
07.     title = os.path.basename(file_path)
08.     msg = "文件【%s】的内容如下：" % title
09.     text = f.read()
10.     g.textbox(msg, title, text)
```

8、在上一题的基础上增强功能：当用户点击“OK”按钮的时候，比较当前文件是否修改过，如果修改过，则提示“覆盖保存”、“放弃保存”或“另存为…”并实现相应的功能。

（提示：解决这道题可能需要点耐心，因为你有可能被一个小问题卡主，但请坚持，自己想办法找到这个小问题所在并解决它！）



答案：这道题会出现的一个小问题就是 `easygui.textbox` 函数会在返回的字符串后面追加一个行结束符（“\n”），因此在比较字符串是否发生改变的时候我们需要人工将这个行结束符忽略。

代码：

```

01. import easygui as g
02. import os
03.
04. file_path = g.fileopenbox(default="*.txt")
05.
06. with open(file_path) as old_file:
07.     title = os.path.basename(file_path)
08.     msg = "文件【%s】的内容如下：" % title
09.     text = old_file.read()
10.     text_after = g.textbox(msg, title, text)
11.
12. if text != text_after[:-1]:
13.     # textbox 的返回值会追加一个换行符
14.     choice = g.buttonbox("检测到文件内容发生改变，请选择以下操作：" , "警告", ("覆盖保存", "放弃保存", "另存为..."))
15.     if choice == "覆盖保存":
16.         with open(file_path, "w") as old_file:
17.             old_file.write(text_after[:-1])
18.     if choice == "放弃保存":
19.         pass
20.     if choice == "另存为...":
21.         another_path = g.filesavebox(default=".txt")
22.         if os.path.splitext(another_path)[1] != '.txt':
23.             another_path += '.txt'
24.         with open(another_path, "w") as new_file:
25.             new_file.write(text_after[:-1])
26.

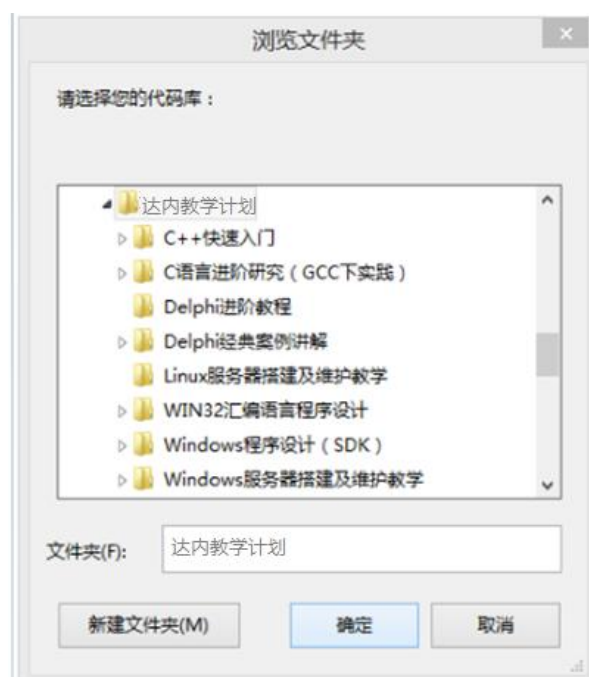
```

9、写一个程序统计你当前代码量的总和，并显示离十万行代码量还有多远？

要求一：递归搜索各个文件夹

要求二：显示各个类型的源文件和源代码量。 要求三：显示总行数与百分比

截图一：





答案：

```
01. import easygui as g
02. import os
03.
04. def show_result(start_dir):
05.     lines = 0
06.     total = 0
07.     text = ""
08.
09.     for i in source_list:
10.         lines = source_list[i]
11.         total += lines
12.         text += "【%s】源文件 %d 个, 源代码 %d 行\n" % (i, file_list[i], lines)
13.     title = '统计结果'
14.     msg = '您目前共累积编写了 %d 行代码，完成进度：%.2f %%\n离 10 万行代码还差 %d 行，请继续努力！' % (total, total/1000, 100000-total)
15.     g.textbox(msg, title, text)
16.
17. def calc_code(file_name):
18.
19.     lines = 0
20.     with open(file_name) as f:
21.         print('正在分析文件：%s ...' % file_name)
22.         try:
23.             for each_line in f:
24.                 lines += 1
25.         except UnicodeDecodeError:
26.             pass # 不可避免会遇到格式不兼容的文件，这里忽略掉.....
27.     return lines
28.
29. def search_file(start_dir) :
30.
31.     for each_file in os.listdir(os.curdir) :
32.         ext = os.path.splitext(each_file)[1]
33.         if ext in target :
34.             lines = calc_code(each_file) # 统计行数
35.             # 还记得异常的用法吗？如果字典中不存，抛出 KeyError，则添加字典键
36.             # 统计文件数
37.             try:
38.                 file_list[ext] += 1
39.             except KeyError:
40.                 file_list[ext] = 1
41.             # 统计源代码行数
42.             try:
43.                 source_list[ext] += lines
44.             except KeyError:
```

```

45.         source_list[ext] = lines
46.
47.         if os.path.isdir(each_file) :
48.             search_file(each_file) # 递归调用
49.             os.chdir(os.pardir) # 递归调用后切记返回上一层目录
50.
51.     target = ['.c', '.cpp', '.py', '.cc', '.java', '.pas', '.asm']
52.     file_list = {}
53.     source_list = {}
54.
55.     g.msgbox("请打开您存放所有代码的文件夹.....", "统计代码量")
56.     path = g.diropenbox("请选择您的代码库:")
57.
58.     search_file(path)
59.     show_result(path)

```

第十四章

一、编程题

1、按照以下提示尝试定义一个矩形类并生成类实例对象。

属性：长和宽

方法： 设置长和宽 -> setRect(self), 获得长和宽 -> getRect(self), 获得面积 -> getArea(self)

提示：方法中对属性的引用形式需加上 self, 如 self.width

程序截图：

```

>>> ----- RESTART -----
>>>
>>> rect = Rectangle()
>>> rect.getRect()
这个矩形的长是：5.00，宽是：4.00
>>> rect.setRect()
请输入矩形的长和宽...
长：3.5
宽：5
>>> rect.getRect()
这个矩形的长是：3.50，宽是：5.00
>>> rect.getArea()
17.5

```

答案：

```

01. class Rectangle:
02.     length = 5
03.     width = 4
04.
05.     def setRect(self):
06.         print("请输入矩形的长和宽...")
07.         self.length = float(input('长: '))
08.         self.width = float(input('宽: '))
09.
10.     def getRect(self):
11.         print('这个矩形的长是: %.2f, 宽是: %.2f' % (self.length, self.width))
12.
13.     def getArea(self):
14.         return self.length * self.width

```

2、按照以下提示尝试定义一个 **Person** 类并生成类实例对象。

属性：姓名（默认姓名为“达内”）

方法：打印姓名

提示：方法中对属性的引用形式需加上 **self**, 如 **self.name**

答案：

```

01. class Person:
02.     name = "tarena"
03.
04.     def printName(self):
05.         print(self.name)
06.

```

3、游戏编程：按以下要求定义一个麻雀类和鸚鵡类并尝试编写游戏。（初学者不一定可以完整实现，但请务必先自己动手，你会从中学习到很多知识的）

假设游戏场景为范围（x,y）为 $0 \leq x \leq 10, 0 \leq y \leq 10$

游戏生成 1 只麻雀和 10 只鸚鵡

他们的移动方向均随机

麻雀的最大移动能力是 2（Ta 可以随机选择 1 还是 2 移动），鸚鵡的最大移动能力是 1

当移动到场景边缘，自动向反方向移动

麻雀初始化体力为 100（上限）

麻雀每移动一次，体力消耗 1

当麻雀和鸚鵡坐标重叠，麻雀吃掉鸚鵡，麻雀体力增加 20

鸚鵡暂不计算体力

当麻雀体力值为 0（挂掉）或者鸚鵡的数量为 0 游戏结束

答案：参考代码附详细注释，希望自己认真完成，你会从中学习到很多知识的。

代码：

```

01. import random as r
02.
03. legal_x = [0, 10]
04. legal_y = [0, 10]
05.
06. class Sparrow:
07.     def __init__(self):
08.         # 初始体力
09.         self.power = 100
10.         # 初始位置随机
11.
12.         self.x = r.randint(legal_x[0], legal_x[1])
13.         self.y = r.randint(legal_y[0], legal_y[1])
14.
15.     def move(self):
16.         # 随机计算方向并移动到新的位置(x, y)
17.         new_x = self.x + r.choice([1, 2, -1, -2])
18.         new_y = self.y + r.choice([1, 2, -1, -2])
19.         # 检查移动后是否超出场景x轴边界
20.         if new_x < legal_x[0]:
21.             self.x = legal_x[0] - (new_x - legal_x[0])
22.         elif new_x > legal_x[1]:
23.             self.x = legal_x[1] - (new_x - legal_x[1])
24.         else:
25.             self.x = new_x
26.         # 检查移动后是否超出场景y轴边界
27.         if new_y < legal_y[0]:
28.             self.y = legal_y[0] - (new_y - legal_y[0])
29.         elif new_y > legal_y[1]:
30.             self.y = legal_y[1] - (new_y - legal_y[1])
31.         else:
32.             self.y = new_y
33.         # 体力消耗
34.         self.power -= 1
35.         # 返回移动后的新位置
36.         return (self.x, self.y)
37.
38.     def eat(self):

```



```

38.         self.power += 20
39.         if self.power > 100:
40.             self.power = 100
41.
42.     class Parrot:
43.         def __init__(self):
44.             self.x = r.randint(legal_x[0], legal_x[1])
45.             self.y = r.randint(legal_y[0], legal_y[1])
46.
47.         def move(self):
48.             # 随机计算方向并移动到新的位置 (x, y)
49.             new_x = self.x + r.choice([1, -1])
50.             new_y = self.y + r.choice([1, -1])
51.             # 检查移动后是否超出场景x轴边界
52.             if new_x < legal_x[0]:
53.                 self.x = legal_x[0] - (new_x - legal_x[0])
54.             elif new_x > legal_x[1]:
55.                 self.x = legal_x[1] - (new_x - legal_x[1])
56.             else:
57.                 self.x = new_x
58.             # 检查移动后是否超出场景y轴边界
59.             if new_y < legal_y[0]:
60.                 self.y = legal_y[0] - (new_y - legal_y[0])
61.             elif new_y > legal_y[1]:
62.                 self.y = legal_y[1] - (new_y - legal_y[1])
63.             else:
64.                 self.y = new_y
65.             # 返回移动后的新位置
66.             return (self.x, self.y)
67.
68.     turtle = Sparrow ()
69.     fish = []
70.     for i in range(10):
71.         new_fish = Parrot()
72.         parrot.append(new_parrot)
73.
74.     while True:
75.         if not len(parrot):
76.             print("鹦鹉都吃完,游戏结束!")
77.             break
78.         if not sparrow.power:
79.             print("麻雀体力耗尽,挂了")
80.             break
81.
82.         pos = sparrow.move()
83.         # 在迭代器中删除列表元素是非常危险的,经常会出现意想不到的问题,因为迭代器是直接引用列表的数据进行引用
84.         # 这里我们把列表拷贝给迭代器,然后对原列表进行删除操作就不会有问题了^_^
85.         for each_parrot in parrot[:]:
86.             if each_parrot.move() == pos:
87.                 # 鹦鹉被吃了
88.                 sparrow.eat()

```



```

89.         parrot.remove(each_parrot)
90.         print("有一只鹦鹉被吃了...")
91.

```

4、按照以下要求定义一个游乐园门票的类，并尝试计算 2 个成人+1 个小孩平日计价。

- . 平日票价 100 元
- . 周末票价为平日的 120%
- . 儿童半票

答案：面向对象编程的难点在于思维的转换。

代码：

```

01. class Ticket():
02.     def __init__(self, weekend=False, child=False):
03.         self.exp = 100
04.         if weekend:
05.             self.inc = 1.2
06.         else:
07.             self.inc = 1
08.         if child:
09.             self.discount = 0.5
10.         else:
11.             self.discount = 1
12.     def calcPrice(self, num):
13.         return self.exp * self.inc * self.discount * num
14.
15. >>> adult = Ticket()
16. >>> child = Ticket(child=True)
17. >>> print("2个成人 + 1个小孩平日票价为：%.2f" % (adult.calcPrice(2) + child.calcPrice(1)))
18. 2个成人 + 1个小孩平日票价为：250.00
19.

```

第十五章

一、编程题

1、定义一个点（Point）类和直线（Line）类，使用 getLen 方法可以获取直线的长度。

提示：

. 设点（X1，Y1）、点（X2,Y2），则两点构成的直线长度 $|AB| = \sqrt{(X1-X2)^2 + (Y1-Y2)^2}$

Python 中计算开根号可使用 math 模块中的 sqrt 函数

直线需有两点构成，因此初始化时需要有两个点（Point）对象作为参数

答案：代码，

```

01. import math
02.
03. class Point():
04.     def __init__(self, x=0, y=0):
05.         self.x = x
06.         self.y = y
07.
08.     def getX(self):
09.         return self.x
10.
11.     def getY(self):
12.         return self.y
13.
14. class Line():
15.     def __init__(self, p1, p2):
16.         self.x = p1.getX() - p2.getX()
17.         self.y = p1.getY() - p2.getY()
18.         self.len = math.sqrt(self.x*self.x + self.y*self.y)
19.

```

```

20.     def getLen(self):
21.         return self.len
22.
23. >>> p1 = Point(1, 1)
24. >>> p2 = Point(4, 5)
25. >>> line = Line(p1, p2)
26. >>> line.getLen()
27. 5.0
28.

```

2、通过学习的知识内容，动手在一个类中定义一个变量，用于跟踪该类有多少个实例被创建（当实例化一个对象，这个变量+1，当销毁一个变量，这个变量自动-1）。

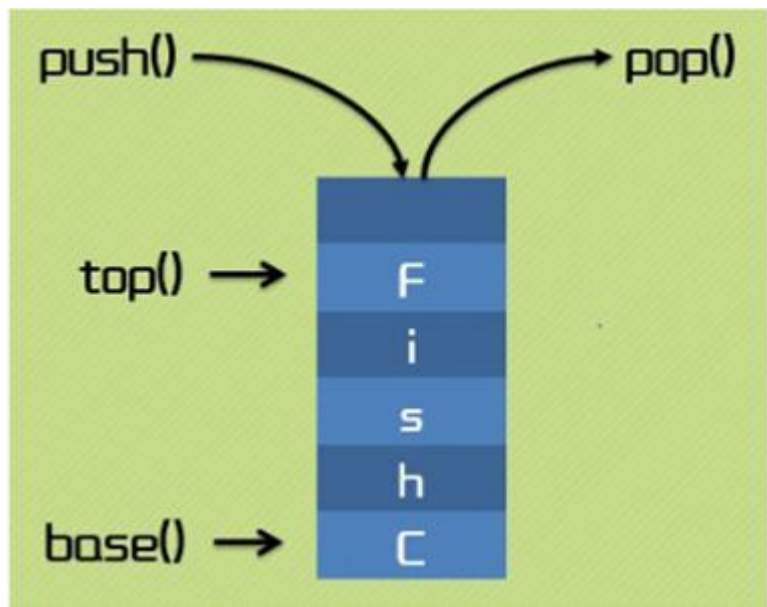
答案：代码，

```

01. class C:
02.     count = 0
03.
04.     def __init__(self):
05.         C.count += 1
06.
07.     def __del__(self):
08.         C.count -= 1
09.
10. >>> a = C()
11. >>> b = C()
12. >>> c = C()
13. >>> C.count
14. 3
15. >>> del a
16. >>> C.count
17. 2
18. >>> del b, c
19. >>> C.count
20. 0
21.

```

3、定义一个栈（Stack）类，用于模拟一种具有后进先出（LIFO）特性的数据结构。至少需要有以下方法：



方法名	含义
isEmpty()	判断当前栈是否为空（返回 True 或 False）
push()	往栈的顶部压入一个数据项
pop()	从栈顶弹出一个数据项（并在栈中删除）
top()	显示当前栈顶的一个数据项
bottom()	显示当前栈底的一个数据项

答案：代码，

```
01. class Stack:
02.     def __init__(self, start=[]):
03.         self.stack = []
04.         for x in start:
05.             self.push(x)
06.
07.     def isEmpty(self):
08.         return not self.stack
09.
10.     def push(self, obj):
11.         self.stack.append(obj)
12.
13.     def pop(self):
14.         if not self.stack:
15.             print('警告：栈为空！')
```

4、 我们知道在 Python 中，两个字符相加会自动拼接字符串，但遗憾的是两个字符串相减却抛出异常。因此，现在我们要求定义一个 Nstr 类，支持字符串的相减操作:A-B，从 A 中去除所有 B 的子字符串。

```

01. >>> a = Nstr('I love tarenaC.com!iiiiiii')
02. >>> b = Nstr('i')
03. >>> a - b
04. 'I love tarenaC.com'

```

示例：

答：只需要重载__sub__魔法方法即可

```

01. class Nstr(str):
02.     def __sub__(self, other):
03.         return self.replace(other, '')

```

5、移位操作符是应用于二进制操作数的，现在需要你定义一个新的类 Nstr，也支持移位操作符的运算：

```

01. >>> a = Nstr('I love tarenaC.com!')
02. >>> a << 3
03. 'ove tarenaC.com!I l'
04. >>> a >> 3
05. 'I love tarenaC.com!'

```

答：只需要重载__lshift__和__rshift__魔法方法即可。

```

01. class Nstr(str):
02.     def __lshift__(self, other):
03.         return self[other:] + self[:other]
04.
05.     def __rshift__(self, other):
06.         return self[:-other] + self[-other:]

```

```

16.         else:
17.             return self.stack.pop()
18.
19.     def top(self):
20.         if not self.stack:
21.             print('警告：栈为空! ')
22.         else:
23.             return self.stack[-1]
24.
25.     def bottom(self):
26.         if not self.stack:
27.             print('警告：栈为空! ')
28.         else:
29.             return self.stack[0]
30.

```

第十六章

一、编程题

1、定义一个类，当实例化该类的时候，自动判断传入了多少个参数，并显示出来。

```
01. class C:
02.     def __init__(self, *args):
03.         if not args:
04.             print("并没有传入参数")
05.         else:
06.             print("传入了 %d 个参数，分别是：" % len(args), end='')
07.             for each in args:
08.                 print(each, end=' ')
```

2、定义一个单词(Word)类继承字符串，重写比较操作符，当两个 Word 类对象进行比较时，根据单词的长度来比较大小。

要求：实例化时如果传入的是带空格的字符串，则取第一个空格前的单词作为参数

答：可以通过重载__new__来实现(因为字符串是不可变类型)，通过重写__gt__、__lt__、__ge__、__le__方法来定义 Word 类在比较操作中的表现。

注意，我们没有定义__eq__和__ne__方法。这是因为将会产生一些怪异不符合逻辑的结果

```
01. class Word(str):
02.     '''存储单词的类，定义比较单词的几种方法'''
03.
04.     def __new__(cls, word):
05.         # 注意我们必须要用到 __new__ 方法，因为 str 是不可变类型
06.         # 所以我们要在创建的时候将它初始化
07.         if ' ' in word:
08.             print "Value contains spaces. Truncating to first space."
09.             word = word[:word.index(' ')] #单词是第一个空格之前的所有字符
10.         return str.__new__(cls, word)
11.
12.     def __gt__(self, other):
13.         return len(self) > len(other)
14.     def __lt__(self, other):
15.         return len(self) < len(other)
16.     def __ge__(self, other):
17.         return len(self) >= len(other)
18.     def __le__(self, other):
19.         return len(self) <= len(other)
```

3、按照课堂中的程序，开始计时的时间是(2022 年 2 月 22 日 16:30:30),停止时间是(2025 年 1 月 23 日 15::30:30)，那按照我们用停止时间减开始时间的计算方式

就出现负数，你应该对比做一些转换。

相信大姐已经意识到不对劲了，为什么一个月一定要 31 天，也有可能是 29 天或者 30 天啊。没错，如果得到正确月份的天数，我们还需要考虑是否是闰年，还有每月的最大天数，所以，我们使用更优秀的解决方案：用 `time` 模块的 `perf_counter()` 和 `process_time()` 来计算，其中，`perf_counter()` 返回计时器的精准时间(系统的运行时间)；`process_time()` 返回当前进程执行 CPU 的时间总和。

题目：改进我们的例子，这次使用 `perf_counter()` 和 `process_time()` 作为计时器。另外增加一个 `set_timer()` 方法，用于设置默认计时器，(默认是 `perf_counter()`，可以通过此方法修改为 `process_time()`)。

既然都做到了这一步，那么在深入一下，改进我们的代码，让他能够统计一个函数运行若干次的时间

要求一：函数调用的次数可以设置(默认是 1000000 次)

要求二：新增一个 `timing` 方法，用于启动计时器。

示例：

2. 既然咱都做到了这一步，那不如再深入一下。再次改进我们的代码，让它能够统计一个函数运行若干次的时间。

要求一：函数调用的次数可以设置 (默认是 1000000 次)

要求二：新增一个 `timing()` 方法，用于启动计时器

函数演示：

```
01. >>> ===== RESTART =====
02. >>>
03. >>> def test():
04.         text = "I love tarenaC.com!"
05.         char = 'o'
06.         if char in text:
07.             pass
08.
09.
10. >>> t1 = MyTimer(test)
11. >>> t1.timing()
12. >>> t1
13. 总共运行了 0.27 秒
14. >>> t2 = MyTimer(test, 100000000)
15. >>> t2.timing()
16. >>> t2
17. 总共运行了 25.92 秒
18. >>> t1 + t2          tarenaC
19. '总共运行了 26.19 秒'
```

[复制代码](#)

答案：

```

01. import time as t
02.
03. class MyTimer:
04.     def __init__(self):
05.         self.unit = ['年', '月', '天', '小时', '分钟', '秒']
06.         self.borrow = [0, 12, 31, 24, 60, 60]
07.         self.prompt = "未开始计时！"
08.         self.lasted = []
09.         self.begin = 0
10.         self.end = 0
11.
12.     def __str__(self):
13.         return self.prompt
14.
15.     __repr__ = __str__
16.
17.     def __add__(self, other):
18.         prompt = "总共运行了"
19.         result = []
20.         for index in range(6):
21.             result.append(self.lasted[index] + other.lasted[index])
22.             if result[index]:
23.                 prompt += (str(result[index]) + self.unit[index])
24.         return prompt
25.
26.     # 开始计时
27.     def start(self):
28.         self.begin = t.localtime()
29.         self.prompt = "提示：请先调用 stop() 停止计时！"
30.         print("计时开始...")
31.
32.     # 停止计时
33.     def stop(self):
34.         if not self.begin:
35.             print("提示：请先调用 start() 进行计时！")
36.         else:
37.             self.end = t.localtime()
38.             self._calc()
39.             print("计时结束！")
40.
41.     # 内部方法，计算运行时间
42.     def _calc(self):
43.         self.lasted = []
44.         self.prompt = "总共运行了"
45.         for index in range(6):
46.             temp = self.end[index] - self.begin[index]

```



```

48.         # 低位不够减，需向高位借位
49.         if temp < 0:
50.             # 测试高位是否有得“借”，没得借的话向再高位借.....
51.             i = 1
52.             while self.lasted[index-i] < 1:
53.                 self.lasted[index-i] += self.borrow[index-i] - 1
54.                 self.lasted[index-i-1] -= 1
55.                 i += 1
56.
57.                 self.lasted.append(self.borrow[index] + temp)
58.                 self.lasted[index-1] -= 1
59.             else:
60.                 self.lasted.append(temp)
61.
62.         # 由于高位随时会被借位，所以打印要放在最后
63.         for index in range(6):
64.             if self.lasted[index]:
65.                 self.prompt += str(self.lasted[index]) + self.unit[index]
66.
67.         # 为下一轮计时初始化变量
68.         self.begin = 0
69.         self.end = 0

```

```

01. import time as t
02.
03. class MyTimer:
04.     def __init__(self):
05.         self.prompt = "未开始计时！"
06.         self.lasted = 0.0
07.         self.begin = 0
08.         self.end = 0
09.         self.default_timer = t.perf_counter
10.
11.     def __str__(self):
12.         return self.prompt
13.
14.     __repr__ = __str__
15.
16.     def __add__(self, other):
17.         result = self.lasted + other.lasted
18.         prompt = "总共运行了 %0.2f 秒" % result
19.         return prompt
20.
21.     # 开始计时
22.     def start(self):
23.         self.begin = self.default_timer()
24.         self.prompt = "提示：请先调用 stop() 停止计时！"

```



```

25.         print("计时开始...")
26.
27.     # 停止计时
28.     def stop(self):
29.         if not self.begin:
30.             print("提示：请先调用 start() 进行计时！")
31.         else:
32.             self.end = self.default_timer()
33.             self._calc()
34.             print("计时结束！")
35.
36.     # 内部方法，计算运行时间
37.     def _calc(self):
38.         self.lasted = self.end - self.begin
39.         self.prompt = "总共运行了 %0.2f 秒" % self.lasted
40.
41.     # 为下一轮计时初始化变量
42.     self.begin = 0
43.     self.end = 0
44.
45.     # 设置计时器(time.perf_counter() 或 time.process_time())
46.     def set_timer(self, timer):
47.         if timer == 'process_time':
48.             self.default_timer = t.process_time
49.         elif timer == 'perf_counter':
50.             self.default_timer = t.perf_counter

```

```

51.         else:
52.             print("输入无效，请输入 perf_counter 或 process_time")

```

```

01. import time as t
02.
03. class MyTimer:
04.     def __init__(self, func, number=1000000):
05.         self.prompt = "未开始计时！"
06.         self.lasted = 0.0
07.         self.default_timer = t.perf_counter
08.         self.func = func
09.         self.number = number
10.
11.     def __str__(self):
12.         return self.prompt
13.
14.     __repr__ = __str__
15.
16.     def __add__(self, other):
17.         result = self.lasted + other.lasted
18.         prompt = "总共运行了 %0.2f 秒" % result
19.         return prompt
20.
21.     # 内部方法，计算运行时间
22.     def timing(self):
23.         self.begin = self.default_timer()
24.         for i in range(self.number):
25.             self.func()

```

```

26.         self.end = self.default_timer()
27.         self.lasted = self.end - self.begin
28.         self.prompt = "总共运行了 %0.2f 秒" % self.lasted
29.
30.         # 设置计时器(time.perf_counter() 或 time.process_time())
31.         def set_timer(self, timer):
32.             if timer == 'process_time':
33.                 self.default_timer = t.process_time
34.             elif timer == 'perf_counter':
35.                 self.default_timer = t.perf_counter
36.             else:
37.                 print("输入无效，请输入 perf_counter 或 process_time")

```

为一个很短的代码计时都很复杂，因为你不知道处理器有多少时间用于运行这个代码？有什么在后台运行？小小的疏忽可能破坏你的百年大计，后台服务偶尔被“唤醒”在最后千分之一秒做一些像查收信件，连接计时通信服务器，检查应用程序更新，扫描病毒，查看是否有磁盘被插入光驱之类很有意义的事。在开始计时测试之前，把一切都关掉，断开网络的连接。再次确定一切都关上后关掉那些不断查看网络是否恢复的服务等等。

接下来是计时框架本身引入的变化因素。Python 解释器是否缓存了方法名的查找？是否缓存代码块的编译结果？正则表达式呢？你的代码重复运行时副作用吗？不要忘记，你的工作结果将以比秒更小的单位呈现，你的计时框架中的小错误将会带来不可挽回的结果扭曲。

Python 社区有句俗语：“Python 自己带着电池。”别自己写计时框架。Python 具备一个叫做 timeit 的完美计时工具。

第十七章

一、编程题

1、按要求重写魔法方法:当访问一个不存在的属性时，不报错且提示“该属性不存在！”

```

01.     >>> class Demo:
02.         def __getattr__(self, name):
03.             return '该属性不存在!'
04.
05.
06.     >>> demo = Demo()
07.     >>> demo.x
08.     '该属性不存在!'

```

2、编写一个 Demo 类，使得下面代码可以正常执行。

```
01. >>> demo = Demo()
02. >>> demo.x
03. 'tarenaC'
04. >>> demo.x = "X-man"
05. >>> demo.x
06. 'X-man'
```

答：

```
01. >>> class Demo:
02.         def __getattr__(self, name):
03.             self.name = 'tarenaC'
04.             return self.name
```

3、修改上边【简答题】的第三题，使之可以正常运行:编写一个 Counter 类，用于实时检测对象有多少个属性

程序实现如下：

```
01. >>> c = Counter()
02. >>> c.x = 1
03. >>> c.counter
04. 1
05. >>> c.y = 1
06. >>> c.z = 1
07. >>> c.counter
08. 3
09. >>> del c.x
10. >>> c.counter
11. 2
```

```

01. class Counter:
02.     def __init__(self):
03.         super().__setattr__('counter', 0)
04.     def __setattr__(self, name, value):
05.         super().__setattr__('counter', self.counter + 1)
06.         super().__setattr__(name, value)
07.     def __delattr__(self, name):
08.         super().__setattr__('counter', self.counter - 1)
09.         super().__delattr__(name)

```

4、按要求编写描述符 MyDes:当类的属性被设置、访问或修改的时候，做出提醒
答案：

```

class MyDes:
    def __init__(self, initval=None, name=None):
        self.val = initval
        self.name = name

    def __get__(self, instance, owner):
        print('正在获取变量', self.name)
        return self.val

    def __set__(self, instance, value):
        print('正在修改变量', self.name)
        self.val = value

    def __delete__(self, instance):
        print('正在删除变量', self.name)
        print('这个变量不能删除')

```

通过代码可以发现，我们这里描述符起到的作用就是间接的对指定变量的操作

5、编写描述符 MyDes,使用文件来存储属性,属性的值会直接存储到对应的 pickle 文件中，如果属性被删除，文件同时也会被删除，属性的名字也会被注销
答案：

```

import os
import pickle
class MyDes:
    saved = []
    def __init__(self, name=None):
        self.name = name
        self.file_name = self.name + '.pkl'
    def __get__(self, instance, owner):
        if self.name not in MyDes.saved:
            raise AttributeError('%s 这个属性还没有赋值' % self.name)
        with open(self.file_name, 'rb') as f:
            value = pickle.load(f)
        return value
    def __set__(self, instance, value):

```

```

        with open(self.file_name, 'wb') as f:
            pickle.dump(value, f)
            MyDes.saved.append(self.name)
    def __delete(self, instance):
        os.remove(self.file_name)
        MyDes.saved.remove(self.name)

```

6、定制一个列表,同样要求记录列表中每个元素被访问的次数。这一次我们希望定制的列表功能更加全面一些,比如支持 `append()`、`pop()`、`extend()` 原生列表所拥有的方法。

要求 1:实现获取、设置和删除一个元素的行为(删除一个元素的时候对应的计数器也会被删除)

要求 2:增加 `counter(index)` 方法,返回 `index` 参数所指定的元素记录的访问次数

要求 3:实现 `append()`、`pop()`、`remove()`、`insert()`、`clear()` 和 `reverse()` 方法(重写这些方法的时候注意考虑计数器的对应改变)

答案:为了实现这么多功能,我们不能再用字典存放计数了因为对于列表来说,如果你删除其中一个元素那么其他元素的下标都会发生相应变化(利用下标作为字典的键肯定就不行了),因此,我们改用一个列表来存放对应的元素的计数,

下面的 `CountList` 类继承并严重依赖其父类(`list`)的行为,并按要求重写了一些方法。

```

class CountList(list):
    def __init__(self, *args):
        super().__init__(args)
        self.count = []
        for i in args:
            self.count.append(0)
    def __len__(self):
        return len(self.count)
    def __getitem__(self, key):
        self.count[key] += 1
        return super().__getitem__(key)
    def __setitem__(self, key, value):
        self.count[key] += 1
        super().__setitem__(key, value)
    def __delitem__(self, key):
        del self.count[key]
        super().__delitem__(key)
    def counter(self, key):
        return self.count[key]
    def append(self, value):
        self.count.append(0)
        super().append(value)
    def pop(self):
        del self.count[-1]
        return super().pop(-1)

```

```

def remove(self, value):
    key = super().index(value)
    del self.count[key]
    super().remove(value)
def insert(self, key, value):
    self.count.insert(key, 0)
    super().insert(key, value)
def clear(self):
    self.count.clear()
    super().clear()
def reverse(self):
    self.count.remove()
    super().reverse()

```

第十八章

一、编程题

1、Python 支持常量吗？相信很多同学的答案是否定的，但实际上，Python 内建的命名空间是支持小部分常量的，例如 True、False、None 等，只是 Python 没有提供定义常量的直接方式而已。现在要求创建一个 const 模块，功能是让 Python 支持常量，举个例子：

以下是我们的测试代码：

```

# const 模块用于让python支持常量操作
import const
const.NAME = "Tarena"
print(const.NAME)
try:
    # 尝试修改常量
    const.NAME = "Tarena.com"
except TypeError as Err:
    print(Err)
try:
    # 变量名需要大写
    const.name = "Tarena"
except TypeError as Err:
    print(Err)

```

执行后的结果是：

Tarena

常量无法改变！

常量名必须由大写字母组成！

在 const 模块中我们到底做了什么，使得这个模块这么有“魔力”呢？大家跟着我的提示，一步一步来做你就懂了：

- 提示一：我们需要一个 Const 类
- 提示二：重写 Const 类的某一个魔法方法，指定当实例对象的属性被修改时的行为
- 提示三：检查该属性是否已经存在
- 提示四：检查该属性的名字是否为大写
- 提示五：细心的同学可能发现了，怎么我们这个 const 模块导入之后就把它当对象来使用（const.NAME = “Tarena”）了呢？难道模块也可以是一个对象？没错啦，在 python 中无处不对象，到处都是你的对象。使用以下方法可以将你的模块与类 A 的对象挂钩。

```
'''
sys.modules 是一个字典，它包含了从python开始运行起，被导入的所有模块。键就是模块名，值就是模块对象
'''
```

```
import sys
sys.modules[__name__] = A()
```

呃……。好像说的有点太多了，大家一定要自己动手先尝试完成哦
代码清单：

```
# 该模块用于让python 支持常量操作
class Const:
    def setattr (self,name,value):
        if name in self.__dict__:
            raise TypeError("常量无法改变")
        if not name.isupper():
            raise TypeError("常量名必须由大写字母组成")
        self.__dict__[name] = value
import sys
sys.modules[ __name__ ] = A()
```

2、请使用 while 语句实现与以下语句相同的功能

```
for each in range(5):
    print(each)
```

答案：

```
a_list = range(5)
it = iter(a_list)
while True:
    try:
        print(next(it))
    except StopIteration:
        break
```

3、写一个迭代器，要求输出至今为止所有的闰年，如：

```
leapYears = LeapYear()
for i in leapYears:
```



```

        if i >= 2000:
            print(i)
        else:
            break
>>>
2016
2012
2008
2004
2000

```

*提示：闰年判定法 (year%4==0 and year%100!=0) or (year%400==0)

答案：

```
import datetime as dt
```

```

class LeapYear:
    def __init__(self):
        self.now = dt.date.today().year
    def is_leap_year(self, year):
        if (year%4 == 0 and year%100 !=0) or (year%400 == 0):
            return True
        else:
            return False
    def __iter__(self):
        return self
    def __next__(self):
        while not self.is_leap_year(self.now):
            self.now -= 1
        temp = self.now
        self.now -= 1
        return temp

```

4、要求写一个 MyRev 类，功能与 `reversed()` 相同（内置函数 `reversed(seq)` 是返回一个迭代器，是序列 `seq` 的逆序显示），例如：

```
myRev = MyRev('Tarena')
```

```

for i in myRev:
    print(i, end='')

```

答案：

```

class MyRev:
    def __init__(self, data):
        self.data = data
        self.index = len(data)
    def __iter__(self):
        return self
    def __next__(self):

```

```

    if self.index == 0:
        raise StopIteration
    self.index = self.index - 1
    return self.data[self.index]

```

5、要求实现一个功能与 reversed() 相同（内置函数 reversed(seq) 是返回一个迭代器，是 seq 的逆序显示）的生成器，例如：

```

for i in myRev('Tarena'):
    print(i, end='')

```

>>>aneraT

答案：

```

def myRev(data):
    for index in range(len(data)-1, -1, -1):
        yield data[index]

```

6、10 以内的素数之和是 2+3+5+7=17，请编写程序，计算 2000000 以内的素数和
答案：

```

import math
def is_prime(number):
    if number > 1:
        if number == 2:
            return True
        if number % 2 == 0:
            return False
        for current in range(3, int(math.sqrt(number)+1), 2):
            if number % current == 0:
                return False
        return True
    return False
def get_primes(number):
    while True:
        if is_prime(number):
            yield number
        number += 1
def solve():
    total = 2
    for next_prime in get_primes(3):
        if next_prime < 2000000:
            total += next_prime
        else:
            print(total)
            return
if __name__ == '__main__':
    solve()

```