# 零基础入门学习Python

**目录**

[零基础入门学习Python 1](#_Toc15867)

[● P2 我和Python的第一次亲密接触 2](#_Toc5015)

[● P3 用Python设计第一个游戏 3](#_Toc14361)

[● P4 小插曲之变量和字符串 4](#_Toc4873)

[● P5 改进我们的小游戏 6](#_Toc16262)

[● P6 闲聊之Python的数据类型 8](#_Toc32274)

[● P7 Python之常用操作符 10](#_Toc17039)

[● P8 了不起的分支和循环1、2 13](#_Toc20522)

[● P10 了不起的分支和循环3 15](#_Toc17370)

[● List常用方法说明与示例 19](#_Toc18998)

[● P11：一个打了激素的数组1 20](#_Toc11788)

[● P12：一个打了激素的数组2 23](#_Toc3135)

[● P13：一个打了激素的数组3 24](#_Toc18742)

[● P14：戴上枷锁的列表 27](#_Toc28049)

[● P15：各种奇葩的内置方法 28](#_Toc2286)

[● P16：字符串：格式化 32](#_Toc3836)

[● P17：序列！序列 33](#_Toc24588)

[● P18：函数，Python的乐高积木 35](#_Toc16413)

[● P19：函数 灵活即强大 37](#_Toc28505)

[● P20：函数— 我的地盘听我的 40](#_Toc22475)

[● P21：内嵌函数和闭包 42](#_Toc20149)

[● P22：函数-lambda表达式 45](#_Toc24867)

[● P23：递归 46](#_Toc27782)

[● P24：递归：这帮小兔崽子、汉诺塔 47](#_Toc18890)

[● P25：字典：当索引不好用时 48](#_Toc7524)

[● P26：字典：当索引不好用时2 50](#_Toc25433)

[● P27：集合：在我的世界里，你就是唯一 52](#_Toc1630)

[● P28：文件：因为懂你，所以永恒 53](#_Toc22802)

[● P31：永久存储：腌制一缸美味的泡菜 54](#_Toc7769)

[● P32：异常处理，你不可能总是对的 55](#_Toc1101)

[● P33：异常处理，你不可能总是对的2 57](#_Toc10056)

[● P34：丰富的else语句及简洁的with语句 58](#_Toc7487)

[● P35：图形用户界面入门：EasyGui 58](#_Toc18007)

[● P36：类和对象：给大家介绍对象 59](#_Toc29447)

[● P37：类和对象：给大家介绍对象 60](#_Toc23103)

## ● P2 我和Python的第一次亲密接触

IDLE：python自带编辑器，python与人的交互工具

Windows操作系统快捷键：

Alt + N 第一次输入的信息

Alt + P 上一次输入的信息

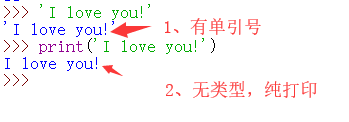
课后作业：

****测试题：****  
  
****0. Python 是什么类型的语言？**** Python是一种计算机程序设计语言。是一种动态的、面对对象的脚本语言  
****1. IDLE 是什么？**** IDLE：python自带编辑器，python与人的交互工具  
****2. print() 的作用是什么？**** 打印符合条件的语句  
****3. Python 中表示乘法的符号是什么？**** \*(星号)  
****4. 为什么 >>>print('I love fishc.com ' \* 5) 可以正常执行，但 >>>print('I love fishc.com ' + 5) 却报错？**** TypeError：can only concatenate str (not “int”) to str

解释：用单引号和双引号包括的是字符串类型，“+”(连接符)，相同类型可以直接使用”+”连接  
****5. 如果我需要在一个字符串中嵌入一个双引号，正确的做法是？**** 用转义字符 \ 语法：\”   
****6. 为什么我们要使用 Python3？Python2到底有什么问题？看起来很多程序员依然都在使用Python2？**** 个人理解：Python2 比较老旧，很多新特性不能支持，Python3更适合当时互联网节奏  
  
****动动手：****  
  
****0. 动手试试直接输入>>>5+8 与输入>>>print(5+8) 有何不同？**** 前者更简单，易懂，后者突出了print关键字，在某些环境下更容易识别

参考答案：

不妨试试直接直接输入 >>>'I love fishc.com!' 与输入 >>>print('I love fishc.com!') 有何不同？  
没错，直接输入是将结果及类型打印到屏幕上，而print是将结果打印到屏幕上，自己试试并观察结果！

截图：

****1. 在交互模式中，使用 Python 计算一年有多少秒？**** 平年：365\*24\*60\*60=31536000

闰年：366\*24\*60\*60=31622400

使用print进行打印  
****2. 设置你的操作系统的环境变量，以便可以轻松进入 Python 环境：****

2.1 右击”此电脑”—选择”属性”—选择”高级系统设置”—选择”环境变量”

2.2 聚焦系统变量列表—选中变量”path”—点击”编辑”—点击”新建”

2.3 输入python路径 如：D:\Program Files\Python37\ 最后点击确认即可

标准答案：

[https://fishc.com.cn/forum.php?mod=viewthread&tid=36000&extra=page%3D1%26filter%3Dtypeid%26typeid%3D398](https://fishc.com.cn/forum.php?mod=viewthread&tid=36000&extra=page=1&filter=typeid&typeid=398)

## ● P3 用Python设计第一个游戏

****测试题：****  
  
****0. 什么是BIF？****

BIF是缩写，全称叫 built in funciton

中文名：内置函数  
****1. 用课堂上小甲鱼教的方法数一数 Python3 提供了多少个 BIF？**** 用 dir(\_\_builtins\_\_) 可列出所有的 built in function

****2. 在 Python 看来：'FishC' 和 'fishc' 一样吗？**** 不一样  
****3. 在小甲鱼看来，Python 中什么是最重要的？你赞同吗？**** 缩进

流程规范，逻辑思路，需要画流程图，开始—参数—判断—正常—异常—结束  
****4. 这节课的例子中出现了“=”和“==”，他们表示不同的含义，你在编程的过程中会不小心把“==”误写成“=”吗？有没有好的办法可以解决这个问题呢？****

=：是赋值

==：对比  
  
****5. 你听说过“拼接”这个词吗？****

听说过，字符串拼接 用“+”号来表示

****动动手：****  
  
****0. 编写程序：hello.py，要求用户输入姓名并打印“你好，姓名！”****

Print(“---------------打招呼---------------------”)

Name = input(“请输入您的名字：”)

Print(“您好，”+name + “!”)

1. ****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”****

print(“------------1到100之间数字并判断------------------------”)

temp = input(“请输入1到100之间的数字：”)

number = int(temp)

if number >0 and number <= 100:

print(“你妹好漂亮^\_^”)

else:

print(“你大爷好丑T\_T”)

1. ****请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P4 小插曲之变量和字符串

****测试题：****  
  
****0. 以下哪个变量的命名不正确？为什么？****  
    (A) MM\_520  (B) \_MM520\_  (C) 520\_MM  (D) \_520\_MM

（C）选项不正确，因为Python中的变量名不能以数字开头。  
  
****1. 在不上机的情况下，以下代码你能猜到屏幕会打印什么内容吗？****

>>>myteacher = '小甲鱼'

>>>yourteacher = myteacher

>>>yourteacher = '黑夜'

>>>print(myteacher)

小甲鱼

****2.在不上机的情况下，以下代码你能猜到屏幕会打印什么内容吗？****

>>> myteacher = '小甲鱼'

>>> yourteacher = myteacher

>>> myteacher = '黑夜'

>>> print(yourteacher)

小甲鱼

****3.在不上机的情况下，以下代码你能猜到屏幕会打印什么内容吗？****

>>>first = 520

>>>second = '520'

>>>first = second

>>>print(first)

‘520’

会打印：520，注：这里没有单引号括起来，但是他****是一个字符串****，因为使用 print() 打印，所以不包含单引号。可以试试直接>>>first，它是一个字符串的。

****4.除了使用反斜杠（\）进行字符转义，还有什么方法可以打印：Let's go! 这个字符串？****

****4.1、”Let’s go!”****

****4.2、r”Let’s go!”****

****4.3、”””Let’s go!”””****

1. ****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\FishC\Good''\\'****

1. ****在这一讲中，我们说变量的命名需要注意一些地方，但小甲鱼在举例的时候貌似却干了点儿“失误”的事儿，你能看得出小甲鱼例子中哪里有问题吗？****

****英文单词写错了****

****变量命名最好顾名思义，一看到这个变量名就大概知道他的意思****

在例子中小甲鱼起了个 str 名字的变量，但事实上我们发现，str 的颜色跟普通变量貌似不同？没错，str() 和 print() 都是内置函数，但 Python 貌似不介意我们对内置函数进行赋值操作，所以这点我们以后就要注意啦。

****动动手：****  
  
****0. 还记得我们第一讲的动动手的题目吗？这一次要求使用变量，计算一年有多少秒？****   
提示：可以以 DaysPerYear（每年天数），HoursPerDay（每天小时数），MinutesPerHour（每小时分钟数），SecondsPerMinute（每分钟秒数）为变量名。  
print(----------------------一年多少秒-------------------)

DaysPerYear = 365

HoursPerDay = 24

MinutesPerHour = 60

SecondsPerMinute = 60

date = DayPerYear \* HoursPerDay \* MinuterPerHour \* SecondsPerMinute

print(“一年有 ”+date +”秒！”)  
  
****1. 关于最后提到的长字符串（三重引号字符串）其实在 Python3 还可以这么写，不妨试试，然后比较下哪种更方便？****

>>> string = (

"我爱鱼C，\n"

"正如我爱小甲鱼，\n"

"他那呱唧呱唧的声音，\n"

"总缠绕于我的脑海，\n"

"久久不肯散去……\n")

1. ****请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****
2. 可以用”****Let's go!****”
3. 可以用”Let\’s go!”
4. 可以用str = (r’Let’s go!’)

4、可以用str = ("我爱鱼C，\n"

"正如我爱小甲鱼，\n"

"他那呱唧呱唧的声音，\n"

"总缠绕于我的脑海，\n"

"久久不肯散去……\n")

5、可以用str = (”””正如我爱小甲鱼，

他那呱唧呱唧的声音，

总缠绕于我的脑海，

久久不肯散去……””)

## ● P5 改进我们的小游戏

****测试题：****  
  
****0. 请问以下代码会打印多少次“我爱鱼C！”****

while 'C':

print('我爱鱼C!')

无数次

****1. 请问以下代码会打印多少次“我爱鱼C！”****

i = 10

while i:

print('我爱鱼C!')

i = i - 1

10次

****2. 请写出与 10 < cost < 50 等价的表达式**** cost > 10 and cost < 50

(10 < cost) and (cost < 50)  
  
****3. Python3 中，一行可以书写多个语句吗？****

可以，语句之间用分号隔开即可，不妨试试：  
>>> print('I love fishc');print('very much!')

****4. Python3 中，一个语句可以分成多行书写吗？**** 可以，一行过长的语句可以使用反斜杠或者括号分解成几行，不妨试试：

>>> 3 > 4 and \

1 < 2  
  
****5. 请问Python的 and 操作符 和C语言的 && 操作符 有何不同？【该题针对有C或C++基础的朋友】**** 有不同

c\c++

printf(“%d,$d\n”, 1 && 3, 0 && 1);

运行结果：1, 0

Python

1 and 3 -------------3

0 and 1 -------------0  
   
****6. 听说过“短路逻辑（short-circuit logic）”吗？****

没有

****动动手：****  
  
****0. 完善第二个改进要求（为用户提供三次机会尝试，机会用完或者用户猜中答案均退出循环）并改进视频中小甲鱼的代码。****

**import random**

**times = 3**

**secret = random.randint(1,10)**

**print('-----------------------我爱鱼C工作室---------------------')**

**# 这里先给guess赋值（赋一个绝对不等于secret的值）**

**guess = 0**

**# print()默认是打印完字符串会自动添加一个换行符，end=" "参数告诉print()用空格代替换行**

**# 嗯，小甲鱼觉得富有创意的你应该会尝试用end="JJ"?**

**print("不妨猜一下小甲鱼现在心里想的是哪个数字：",end=" ")**

**while (guess != secret) and (times > 0):**

**temp = input()**

**guess = int(temp)**

**times = times -1 #用户每输入一次，可用机会就-1**

**if guess == secret:**

**print("卧槽，你是小甲鱼心里的蛔虫吗？！")**

**print("哼，猜中了也没有奖励！")**

**else:**

**if guess > secret:**

**print("哥，大了大了~~~")**

**else:**

**print("嘿，小了，小了~~~")**

**if times > 0:**

**print("再试一次吧：",end=" ")**

**else:**

**print("机会用光咯T\_T")**

**print("游戏结束，不玩啦^\_^")**

****1. 尝试写代码实现以下截图功能：****

temp = input("请输入一个整数：")

number = int(temp)

for i in range(number):

print(i+1)

temp = input('请输入一个整数:')

number = int(temp)

i = 1

while number:

print(i)

i = i + 1

number = number - 1

****2. 尝试写代码实现以下截图功能：****

temp = input("请输入一个整数：")

number = int(temp)

while number:

print(number\*" "+"\*"\*number)

number= number -1

**temp = input('请输入一个整数:')**

**number = int(temp)**

**while number:**

**i = number - 1**

**while i:**

**print(' ', end = '')**

**i = i - 1**

**j = number**

**while j:**

**print('\*', end = '')**

**j = j - 1**

**print()**

**number = number - 1**

****3. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P6 闲聊之Python的数据类型

****测试题：****  
  
****0. 在 Python 中，int 表示整型，那你还记得 bool、float 和 str 分别表示什么吗？****

Bool ：布尔类型，以TRUE，FALSE

Float：浮点型，类似与数学里面的小数

str：字符串，用双引号，单引号包括起来的数字，汉字，字母等  
  
****1. 你知道为什么布尔类型(bool)的 True 和 False 分别用 1 和 0 来代替吗？**** True：代表真

false：代表假

一般用于分支语句，循环语句判断

你可能听说过计算机是很”笨”的，究其根本是因为它只认识二进制数，所以所有的编程语言最终都会转换成简单的二进制序列给CPU按照一定的规则解析。

由于二进制只有两个数：0和1，因此用0和1来表示false和True再适合不过了，因为不用浪费资源在转换的过程上。

****2. 使用 int() 将小数转换为整数，结果是向上取整还是向下取整呢？**** 向上取整，小数点后面的全部清掉，保留整数

小数取整会采用比较暴力d的截断方式，即向下取整。（注：5.5向上取整为6，向下取整为5）

对于负数则刚好相反———是向上取整。  
   
****3. 我们人类思维是习惯于“四舍五入”法，你有什么办法使得 int() 按照“四舍五入”的方式取整吗？**** int(2.5+0.5)

int(2.4+0.5)

int()固然没那么”聪明”，但机器是死的，鱼油是活的！

5.4”四舍五入”结果为：5，int(5.4+0.5) == 5

5.6”四舍五入”结果为：6，int(5.6+0.5) == 6  
  
****4. 取得一个变量的类型，视频中介绍可以使用 type() 和 isinstance()，你更倾向于使用哪个？**** type()

建议使用isinstance(),因为它的返回结果比较直接，另外type()其实并没有你想象的那么简单，我们后面会讲到。  
  
****5. Python3 可以给变量命名中文名，知道为什么吗？**** Python3 源码文件默认使用UTF-8编码（支持中文），这表示中文代码是合法的

>>>河蟹 = ‘spring’

>>>print(河蟹)

>>>spring  
  
****6. 【该题针对零基础的鱼油】你觉得这个系列教学有难度吗？**** 目前不难  
  
****动动手：****  
  
****0. 针对视频中小甲鱼提到的小漏洞，再次改进我们的小游戏：当用户输入错误类型的时候，及时提醒用户重新输入，防止程序崩溃。****

如果你尝试过以下做法，请举下小手：

或者可能这样：

以上方法的思路是正确的，不过似乎忽略了一点儿：就是 input() 的返回值始终是字符串，所以 type(temp) 永远是 <class 'str'>！    
     
其实有蛮多的做法可以实现的，不过就目前我们学习过的内容来看，还不足够。  
  
所以，在让大家动手完成这道题之前，小甲鱼介绍一点新东西给大家！

好了，文字教程就到这里，大家赶紧趁热打铁，改造我们的小游戏吧！

s为字符串

s.isalnum() 所有字符都是数字或者字母，为真返回True，否则返回False。

s.isalpha() 所有字符都是字母，为真返回True，否则返回False。

s.isdigit() 所有字符都是数字，为真返回True，否则返回False。

s.islower() 所有字符都是小写，为真返回True，否则返回False。

s.isupper() 所有字符都是大写，为真返回True，否则返回False。

s.istitle() 所有单词都是首字母大写，为真返回True，否则返回False。

s.isspace() 所有字符都是空白字符，为真返回True，否则返回False。

例如：

>>>s = ‘I LOVE YOU’

>>>s.isupper()

>>>True  
****1. 写一个程序，判断给定年份是否为闰年。（注意：请使用已学过的 BIF 进行灵活运用）****  
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。  
 temp = input(“请输入年份：”)

while not temp.isdigit():

temp = input(“抱歉，您输入有误，请输入一个整数：”)

year = int(temp)

if year/400 == int(year/400):

print(temp + ‘是闰年！’)

else:

if year/4 == int(year/4) and year/100 != int(year/100):

print(temp + ‘是闰年！’)

else:

print(temp + ‘不是闰年！’)

****2. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P7 Python之常用操作符

****测试题：****  
  
****0. Python 的 floor 除法现在使用 “//” 实现，那 3.0 // 2.0 您目测会显示什么内容呢？**** >>> 3.0//2.0

>>>1.0  
  
****1. a < b < c 事实上是等于？****

a < b and b < c

(a < b) and (b < c)  
  
****2. 不使用 IDLE，你可以轻松说出 5 \*\* -2 的值吗？****

>>> 5 \*\* -2

>>> 50

>>> 5\*\*-2

>>>0.04

也就是1/25  
  
****3. 如何简单判断一个数是奇数还是偶数？**** 用它模除2，余下1的是奇数，余下0的是偶数

能被2整除的为偶数，所以 x % 2 == 0,否则为奇数  
****4. 请用最快速度说出答案：not 1 or 0 and 1 or 3 and 4 or 5 and 6 or 7 and 8 and 9****

false

4

not or and 的优先级是不同的：not > and > or

我们按照优先级给它们加上括号：(not 1) or (0 and 1) or (3 and 4) or (5 and 6) or (7 and 8 and 9)

== 0 or 0 or 4 or 6 or 9

== 4

为啥是4？

大家还记得第四讲作业提到的“短路逻辑”吗？ 3 and 4 == 4,而3 or 4 == 3。

所以答案是：4

温馨提示：为了更好的表达你的程序，spring 再次呼吁有些括号还是不能省下的，毕竟不是所有程序员都跟你一样都将优先级烂透于心的。  
  
****5. 还记得我们上节课那个求闰年的作业吗？如果还没有学到“求余”操作，还记得用什么方法可以“委曲求全”代替“%”的功能呢？****

因为上节课我们还没有教大家“求余”的操作，所以spring使用了”if year/400 == int(year/400)”这样的方式来”委屈求全”

****动动手：****  
  
****0. 请写一个程序打印出 0~100 所有的奇数。****  
 print("----------打印出0-100所有的奇数-----------")

num = 100

while num:

if num % 2 ==1:

print(num)

num-=1

else:

num-=1

****1. 我们说过现在的 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刚刚的硬件滴，不妨写一个小代码，让你的计算机为之崩溃？****  
 print(2 \*\* 2 \*\* 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. 爱因斯坦的难题****

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。  
（小甲鱼温馨提示：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？

number = 1

switch = 1

while switch:

if number % 2 == 1 and number % 3 == 2 and number % 5 == 4 and number % 6 == 5 and number % 7 == 0:

print(number)

switch = 0

else:

print(number)

number +=1

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！')

****3. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P8 了不起的分支和循环1、2

****测试题：****  
     
****0. if not (money < 100): 上边这行代码相当于？****   如：money <100，为真

not (money < 100)，就为假

if money >= 100:  
  
****1. assert 的作用是什么？****

  断言，如果为假，抛出一个异常

assert这个关键字我们称之为“断言”，当这个关键字后面的条件为假的时候，程序自动崩溃并抛出AssertionError的异常。

什么情况下我们需要这样的代码呢？ 当我们在测试程序的时候就很好用，因为与其让错误的条件导致程序进后莫名奇妙的崩溃，不如在错误的条件出现的那一瞬间我们实现“自爆”。

一般来说我们可以用Ta再程序中置入检查点，当需要确保程序中的某个条件一定为真才能让程序正常工作的话，assert关键字就非常有用了。  
  
****2. 假设有 x = 1，y = 2，z = 3，请问如何快速将三个变量的值互相交换？****   新增一个变量：middle，交换时用到这个变量

x,y,z = z,y,x  
  
****3. 猜猜 (x < y and [x] or [y])[0] 实现什么样的功能？****     
  
****4. 你听说过成员资格运算符吗？****   in ：如果在指定的序列中找到值返回True，否则返回False

not in ：如果在指定的序列中没有找到值返回True，否则返回False

>>> name = ‘小甲鱼’

>>> ‘鱼’ in name

True

>>> ‘肥鱼’ in name

false  
    
****动动手：****  
    
****0. 视频中小甲鱼使用 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。****

题目备忘：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。

temp= input(“请输入学生的成绩：”)

success = int(temp)

if success >=60 and success <80:

print(“C”)

elif success <60:

print(“D”)

elif success >=80 and success <90:

print(“B”)

elif success >=90 and success <100:

print(“A”)

else:

print(“输入有误，请稍后再试”)

score = int(input(‘请输入学生分数：’))

if 80 > score > = 80:

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，请重新输入！”)

****1. 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

x,y,z = 6,5,4

small = (x if x < y else y)

small = (z if small > z else small)

print(small)

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

****2. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P10 了不起的分支和循环3

语法：

for 目标 in 表达式：

循环体

len()

语法：

range( [strat,] stop [,step=1] )

·这个BIF有三个参数，其中用中括号括起来的两个表示这两个参数是可选的 。

·step=1 表示第三个参数的默认值是1

·range这个BIF的作用是生成一个从start参数的值开始到stop参数的值结束的数字序列

for i in range(10):

if i%2 != 0:

print(i)

continue

i +=2

print(i)

代码解析：

1. i是0，if条件不成立，执行i+=2,打印i的值：2

2. i是1，if条件成立，打印i的值：1，continue

3. i是2，if条件不成立，执行i+=2,打印i的值：4

4. i是3，if条件成立，打印i的值：3，continue

5. i是4，if条件不成立，执行i+=2，打印i的值：6

6. i是5，if条件成立，打印i的值：5，continue

7. i是6，if条件不成立，执行i+=2，打印i的值：8

8. i是7，if条件成立，打印i的值：7，continue

9. i是8，if条件不成立，执行i+=2，打印i的值：10

10. i是9，if条件成立，打印i的值：9，continue

11.i遍历结束

****测试题：****  
    
0. 下面的循环会打印多少次"I Love FishC"？

for i in range(0,10,2):

print(‘I Love FishC’)

五次

5次，因为从0开始，到10结束，步进为2。

1. 下面的循环会打印多少次"I Love FishC"？

for i in 5:

print('I Love FishC')

五次

会报错，in是成员资格运算符，Python的for更像脚本语言的foreach。

2. 回顾一下 break 和 continue 在循环中起到的作用？

break：如程序运行到这一步，跳出整个循环体

continue：如程序运行到这一步，跳出本次循环，执行下一次循环

break：语句的作用是终止当前循环，跳出循环体。

continue：语句的作用是终止本轮循环并开始下一轮循环（这里要注意的是：在开始下一轮循环之前，会先测试循环条件的）  
  
3. 请谈下你对列表的理解？  
    列表：类似与c，c++，PHP里面的数组，但是比数组强大

listname = [element1,element2,element3,...,elementN]

使用list函数创建列表

list1 = list()  
  
4. 请问 range(10) 生成哪些数？  
     0,1,2,3,4,5,6,7,8,9

包头不包尾  
  
5. 目测以下程序会打印什么？

while True:

while True:

break

print(1)

print(2)

break

print(3)

会打印

2

3

因为break只能跳出一层循环，记住咯！

6. 什么情况下我们要使循环永远为真？

我们在使用了break这个BIF，就可以使循环永远为真

while Ture:

循环体  
 同样用于游戏实现，因为游戏只要运行着，就需要时刻接收用户输入，因此使用永远为真确保游戏“在线”。操作系统也是同样的道理，时刻待命，操作系统永远为真的这个循环叫消息循环。另外，许多通讯服务器的客户端/服务器系统也是通过这样的原理进行工作的。  
7. 【学会提高代码的效率】你的觉得以下代码效率方面怎样？有没有办法可以大幅度改进(仍然使用while)？

i = 0

string = 'ILoveFishC.com'

while i < len(string):

print(i)

i += 1

i = 0

string = ‘IloveFishC.com’

num = len(string)

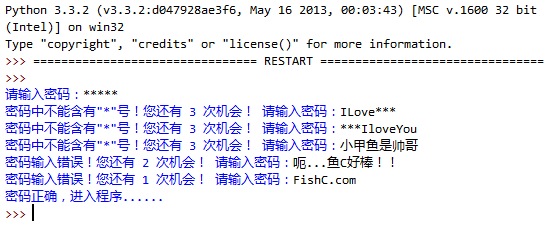
while i < num:

print(i)

i+=1

这段代码之所以“效率比较低”是因为每次循环都需要调用一次 len() 函数（我们还没有学到函数的概念，小甲鱼这里为零基础的朋友形象的解释下：就像你打游戏打得正HIGH的时候，老妈让你去买盐......你有两种选择，一次买一包，一天去买五次，或者一次性买五包回来，老妈要就直接给她。）

****动动手：****  
    
细节决定成败，看答案前记得自己先敲代码！  
    
****0. 设计一个验证用户密码程序，用户只有三次机会输入错误，不过如果用户输入的内容中包含"\*"则不计算在内。****  
    
****程序演示如图：****



****print("验证用户密码程序，用户只有三次机会输入错误，输入'\*'则不计算在内")****

****Cpawd = '123456'****

****pawd = input('请输入密码：')****

****num = 2****

****while num > 0:****

****for i in pawd:****

****if i == '\*':****

****pawd = input('密码中不能含有"\*"！您还有'+str(num)+'次机会！请输入密码：')****

****elif pawd == Cpawd:****

****pirnt('密码正确，进入程序......')****

****else:****

****pawd = input('密码输入错误！您还有'+str(num)+'次机会！请输入密码：')****

****num-=1****

****count = 3****

****password = ‘123455’****

****while count:****

****passwd = input(‘请输入密码：’)****

****if passwd == password:****

****print(‘密码正确，进入程序......’)****

****break****

****elif: ‘\*’ in passwd:****

****print(‘密码中不能含有”\*”号！您还有’,count,’次机会！’,end = ‘ ’)****

****continue****

****else:****

****print(‘密码输入错误！您还有’,count-1,’次机会！’,end = ‘ ’)****

****count -= 1****

****1. 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。

print("编写一个程序，求100-999之间的所有水仙花数")

''' mark: 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。'''

for i in range(100,1000):

bit = i%10)

ten = i//10%10

hun = i//100

if i == (bit\*\*3) + (ten\*\*3) + (hun\*\*3):

print(i)

****2. 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。

for red in range(0,4):

for yellow in range(0,4):

for green in range(2,7):

if red + yellow + green == 8:

print("red:{}".format(red))

print("yellow:{}".format(yellow))

print("green:{}".format(green)+'\n')

print('red\tyellow\tgreen')

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] 这 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)

****3. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

# ● List常用方法说明与示例

1、索引：list[i]—>i: 1,2,3,4,5,6,i: -1,-2,-3,-4,-5

2、切片：list[a:b]

colors = [‘blue’,’red’,’yellow’,’pink’,’white’,’black’]

colors [1:3] # [‘red’,’yellow’]

colors [1:] # [’red’,’yellow’,’pink’,’white’,’black’]

colors [:4] # [‘blue’,’red’,’yellow’,’pink’]

colors[:] # [‘blue’,’red’,’yellow’,’pink’,’white’,’black’]

colors [1:5:2]# [’red’,’pink’]

colors [::2] # [‘blue’,’yellow’,’white’]

3、查询：index() 获取列表元素下标

colors = [‘blue’,’red’,’yellow’,’pink’,’white’,’black’]

colors.index(‘red’) # 1

4、修改：list[i] = ‘value’

colors = [‘blue’,’red’,’yellow’,’pink’,’white’,’black’]

colors[1] = ‘green’ # ['blue', 'green', 'yellow', 'pink', 'white', 'black']

5、增加：追加 (append()) , 插入 (insert())

colors = [‘blue’,’red’,’yellow’,’pink’,’white’,’black’]

colors.append(‘green’) # [‘blue’,’red’,’yellow’,’pink’,’white’,’black’,’green’]

追加的意思：就是在列表的最后，添加元素

colors.insert(1,’green’) # [‘blue’,’green’,’red’,’yellow’,’pink’,’white’,’black’]

插入的意思：就是在列表中所需要的位置之前，添加元素

6、删除：del list[index] , list.remove(element) , list.pop()

colors = ['blue', 'red', 'yellow', 'pink', 'white', 'black']

del colors[0] # ['red', 'yellow', 'pink', 'white', 'black']

指定下标元素，删除

colors.remove['pink'] # ['red', 'yellow', 'white', 'black']

匹配到的第一个元素，进行删除，如有多个元素，只删除第一个元素

colors.pop() # ['red', 'yellow', 'white']

删除列表最后一个元素，pop(i) 可以传index下标，如pop(1)

7、扩展(list.extend(new\_list))

扩展，将一个列表追加到另一个列表后面，组成一个新的列表

colors = ['blue', 'red', 'yellow', 'pink', 'white', 'black']

other\_colors = ['green', 'gray', 'orange']

colors.extend(other\_colors) # ['blue', 'red', 'yellow', 'pink', 'white', 'black', 'green', 'gray', 'orange']

colors = ['blue', 'red', 'yellow', 'pink', 'white', 'black']

colors.extend('other') # colors = ['blue', 'red', 'yellow', 'pink', 'white', 'black', 'o', 't', 'h', 'e', 'r'] extend后面的变量名不能写成字符串，会将字符串遍历出来当成元素添加到列表最后

8、统计(list.count(element))

colors = ['blue', 'red', 'yellow', 'pink', 'white', 'black', 'blue', 'red', 'yellow', 'red', 'yellow']

colors.count(‘blue’) # 2

colors.count(‘red’) # 3

9、排序 (list.sort(self, key = None, reverse = False))

nums = [1, 22, 35, 63, 3, 11, 7]

nums.sort() # 默认数字从小到大排列 [1, 3, 7, 11, 22, 35, 63]

9.1、反转 reverse()方法

nums[1, 3, 7, 11, 22, 35, 63]

nums.reverse() # [63, 35, 22, 11, 7, 3, 1]

nums.reverse() # [1, 3, 7, 11, 22, 35, 63]

10、拷贝 浅拷贝，深拷贝

copy：第一层开辟新地址，第二层共有第一层的地址，第二层中的元素变成元组而不是列表

deepcopy：为每一层都开辟新地址，确保每一层的拷贝都是可修改的列表，而非元组

## ● P11：一个打了激素的数组1

**Built in function:**

append():末尾追加一个内容

extend():末尾追加一个列表，列表内可以包含两个内容

insert():指定位置插入一个内容，例：insert(0,’河蟹’)，在最前面插入一个内容

****测试题：****  
0. 列表都可以存放一些什么东西？  
   整数，小数，字符串，对象

我们说Python的列表是一个打了激素的数组，如果把数组比喻成集装箱，那么Python的列表就是一个大仓库，Ta可以存放我们学习过的任何数据类型

>>> mix = [1,’小甲鱼’,3.14,[1,2,3]]

1. 向列表增加元素有哪些方法？  
   append()

extend()

insert()

这节课我们教过大家三种方法向列表增加元素，分别是：append()、extend()和 insert()

2. append() 方法和 extend() 方法都是向列表的末尾增加元素，请问他们有什么区别？  
   append()方法只能向列表末尾增加一个元素

extend()方法可以向列表追加多个元素，但是要以列表格式进行追加

append()方法是将参数作为一个元素增加到列表的末尾。

extend()方法则是将参数作为一个列表去扩展列表的末尾。

**示例：**

>>> name = ['F', 'i', 's', 'h']

>>> name.append('C')

>>> name

['F', 'i', 's', 'h', 'C']

>>> name.extend(['.', 'c'])

>>> name

['F', 'i', 's', 'h', 'C', '.', 'c']

>>> name.append(['o', 'm'])

>>> name

['F', 'i', 's', 'h', 'C', '.', 'c', ['o', 'm']]  
3. member.append(['竹林小溪', 'Crazy迷恋']) 和 member.extend(['竹林小溪', 'Crazy迷恋']) 实现的效果一样吗？  
   前者：列表末尾增加列表

后者：列表末尾增加内容

不一样，因为怕大家没有仔细看上一题的示例，所以不懂的请看上一题解释。

4. 有列表 name = ['F', 'i', 'h', 'C']，如果小甲鱼想要在元素 'i' 和 'h' 之间插入元素 's'，应该使用什么方法来插入？  
 用insert()

name.insert(2,’s’)  
    
****动动手：****  
****0. 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 = ['小甲鱼', '黑夜', '迷途', '怡静', '秋舞斜阳']  
  
要求将列表修改为：  
  
member = ['小甲鱼', 88, '黑夜', 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。

insert()方法修改列表：

member = ['小甲鱼','黑夜','迷途','怡静','秋舞斜阳']

number = [88,90,85,90,88]

temp = 1

for i in number:

member.insert(temp,i)

temp+=2

print(member)

member.insert(1, 88)

member.insert(3, 90)

member.insert(5, 85)

member.insert(7, 90)

append()方法修改列表：

member.append(88)

方法二：重新创建一个同名字的列表覆盖。

member = ['小甲鱼','黑夜','迷途','怡静','秋舞斜阳']

member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

print(member)  
****1. 利用 for 循环打印上边 member 列表中的每个内容，如图：****



member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

for i in member:

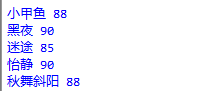
print(i)

member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

for each in member:

print(each)

****2. 上一题打印的样式不是很好，能不能修改一下代码打印成下图的样式呢？【请至少使用两种方法实现】****



第一种：

member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

Line = 1

for i in member:

if Line%2==0:

print(i,end=' '+'\n')

else:

print(i,end=' ')

Line+=1

member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

count = 0

length = len(member)

while count < length:

print(member[count],member[count+1])

count += 2

第二种：

member = ['小甲鱼', 88, '黑夜', 90, '迷途', 85, '怡静', 90, '秋舞斜阳', 88]

for each in range(len(member)):

if each%2 == 0:

print(member[each],member[each+1])

****3. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P12：一个打了激素的数组2

remove()：语法-->listname.remove(‘name’)

del()： 语法--> del listname[x],按照下标删除值，如不带下标(del listname)直接删除列表

pop()：语法-->listname.pop(), 总是删除列表最后一个值，并把该值打印出来

****测试题：****  
****0. 下边的列表分片操作会打印什么内容？****

>>> list1 = [1, 3, 2, 9, 7, 8]

>>> list1[2:5]

[2,9,7]（注意不包含8哦，因为5-2==3，只有三个元素）

****1. 请问 list1[0] 和 list1[0:1] 一样吗？****

不一样，list1[0]返回第0个元素的值，list[0:1]返回一个只含有第0个元素的列表

list1[0]取出的是值

list1[0:1]取出的是列表

****2. 如果你每次想从列表的末尾取出一个元素，并将这个元素插入到列表的最前边，你会怎么做？****   list1 = [1,2,3,4,5,6,7,8]

num = len(list1)

for i in range(len(list1)):

temp = list1.pop()

list1.insert(0,temp)

print(list1)

list.insert(0, list1.pop())

****3. 有些鱼油比较调皮，他说我想试试 list1[-3:-1] 会不会报错，怎么知道一试居然显示 [9, 7]，这是怎么回事呢？****  -1表示是列表最后一个值，-3表示倒数第三个值

list1 = [1,3,2,9,7,8]

-3是9，-1是8，包头不包尾

python 的列表很聪明，竟然支持负数索引，如图：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 正常下标 | 0 | 1 | 2 | 3 | 4 | 5 |
| 列表内容 | 1 | 3 | 2 | 9 | 7 | 8 |
| 负数下标 | -6 | -5 | -4 | -3 | -2 | -1 |

****4. 在进行分片的时候，我们知道分片的开始和结束位置需要进行指定，但其实还有另外一个隐藏的设置：步长。****

在普通的分片操作中，步长默认设置为1，表示逐个遍历元素。其实我们可以人为调整步长以达到不可告人的秘密。

>>> list1[0:6:2]

[1, 2, 7]

那么依你推测，关于步长的知识点还有哪些（很多知识点都是通用的）？  
   从0开始，到第6个值，按照2步长进行取值

1，2，7，步长，包头不包尾

1）之前提到的‘简洁’分片操作在这里有效：

>>> list1[::2]

[1, 2, 7]

2）步长不能为0，要不就走不动了：

>>> list1[::0]

Traceback (most recent call last):

File "<pyshell#11>", line 1, in <module>

list1[::0]

ValueError: slice step cannot be zero

3） 步长可以为负数，改变方向（从尾部开始向左走）：

>>> list1[::-2]

[8,9,3]

4）步子太大容易扯着蛋......

****5. 课堂上小甲鱼说可以利用分片完成列表的拷贝 list2 = list1[:]，那事实上可不可以直接写成 list2 = list1 更加简洁呢？****

事实上可以的，为什么不能这么写呢？

不知道大家还记不记得我们在讲解变量的时候说过，Python的变量就像一个标签，就一个名字而已，贴哪是哪，指哪打哪，呃，我还是给大家画个图好。

现在你对list11列表进行操作，相当于操作了list13，所以不能这样拷贝

1， 3， 2， 9， 7， 8

list11

list12

1， 3， 2， 9， 7， 8

list13

****6. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！****

## ● P13：一个打了激素的数组3

reverse():反转列表里面的元素

index():索引

count():统计出现的字数

sort():排序

****测试题：****  
     
**0. 注意，这道题跟上节课的那道题有点儿不同，回答完请上机实验或参考答案。**

>>> old = [1, 2, 3, 4, 5]

>>> new = old

>>> old = [6]

>>> print(new)

如果不上机操作，你觉得会打印什么内容？

[6]

right key：

会打印：[1,2,3,4,5]

Reason：开始old和new都指向[1,2,3,4,5]，通过old = [6]，old改变指向[6]，new没有改变指向

**1. 请问如何将下边这个列表的'小甲鱼'修改为'小鱿鱼'？**

list1 = [1, [1, 2, ['小甲鱼']], 3, 5, 8, 13, 18]

list1[1][2] = [‘小鱿鱼’] #[1, [1, 2, ['小鱿鱼']], 3, 5, 8, 13, 18]

list1[1][2][0] = ‘小鱿鱼’ #[1, [1, 2, ['小鱿鱼']], 3, 5, 8, 13, 18]

**2. 要对一个列表进行顺序排序，请问使用什么方法？**   list.sort()  
    
**3. 要对一个列表进行逆序排序，请问使用什么方法？**   3.1、list.sort()

3.2、list.reverse()

或者：列表名.sort(reverse = True)  
    
**4. 列表还有两个内置方法没给大家介绍，不过聪明的你应该可以自己摸索使用的门道吧：copy() 和 clear()**   copy()：复制一份列表元素给新列表

copy() 方法跟使用切片拷贝一样的：

>>> list2 = list1.copy()

>>> list2

[1, [1, 2, ['小甲鱼']], 3, 5, 8, 13, 18]clear()：用于清空列表，类似于 del list[:]

 clear() 方法用于清空列表的元素，但要注意，清空完后列表仍然还在哦，只是变成了一个空列表

>>> list2.clear()

>>>list2

[]  
**5. 你有听说过列表推导式或列表解析吗？**   没有  
没听过？！没关系，我们现场来学习一下吧，看表达式：

>>> [ i\*i for i in range(10) ]

你觉得会打印什么内容？

>>> [i\*i for i in range(10)]

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

居然分别打印了0到9各个数的平方，然后还放在列表里边了有木有？！  
    
列表推导式（List comprehensions）也叫列表解析，灵感取自函数式编程语言 Haskell。Ta 是一个非常有用和灵活的工具，可以用来动态的创建列表，语法如：  
    
[有关A的表达式 for A in B]

例如

>>> list1 = [x\*\*2 for x in range(10)]

>>> list1

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

相当于

list1 = []

for x in range(10):

list1.append(x\*\*2)

问题：请先在 IDLE 中获得下边列表的结果，并按照上方例子把列表推导式还原出来。

>>> list1 = [(x, y) for x in range(10) for y in range(10) if x%2==0 if y%2!=0]

相当于：

list1 = []

for x in range(10):

if x % 2 == 0:

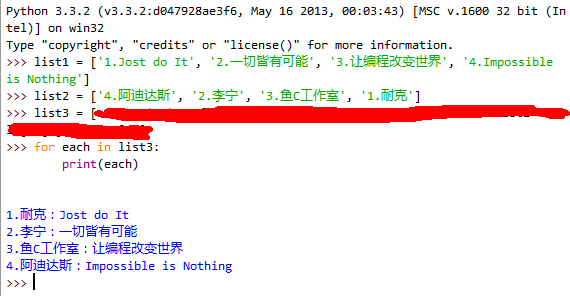
for y in range(10):

if y % 2 != 0:

list1.append((x,y))

print(list1)

**6. 活学活用：请使用列表推导式补充被小甲鱼不小心涂掉的部分**



list3 = [name + ' : ' + slogan[2:] for slogan in list1 for name in list2 if slogan[0] == name[0]]

**7. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P14：戴上枷锁的列表

列表切片语法：列表名[开始元素，结束元素，步长]

开始元素为负数，结束元素只能为负数或者不写

结束元素为负数，开始元素可为正负数或者不写

****测试题：****  
     
**0. 请用一句话描述什么是列表？再用一句话描述什么是元组？**

列表：用[]包括起来的数据，可以是任何类型

元组：用逗号来表示这是一个元组，元组不可直接修改，可以通过拼接来进行修改

列表：一个大仓库，你可以随时往里面添加和删除任何东西

元组：封闭的列表，一旦定义，就不可改变（不能添加、删除和修改）。  
     
**1. 什么情况下你需要使用元组而不是列表？**   数据格式统一，不进行修改的数据

当我们希望内容不被轻易改写的时候，我们使用元组（把权力关进牢笼）。

当我们频繁更改数据，我们使用列表。

**2. 当元组和列表掉下水，你会救谁？**   元组

如果是我，我会救列表，因为列表提供了比元组更丰富的内置方法，这相当大的提高了编程的灵活性

回头看一下元组，元组固然安全，但元组一但创建就无法修改(除非通过新建一个元组来间接修改，但这就带来了消耗)，而我们人是经常摇摆不定的，所以元组只有在特殊的情况才用到，平时还是列表用的多。

**3. 请将下图左边列表的内置方法与右边的注释连线，并圈出元组可以使用的方法。**



元组果然比较悲催，仅有两个方法元组可以使用，其他方法都不行，不信你试试？

**4. 创建一个元组，什么情况下逗号和小括号必须同时存在，缺一不可？**   只有一个元素的时候，必须有逗号和小括号

在拼接只有一个元素的元组的时候，例如：

>>> temp = (‘小甲鱼’, ‘黑夜’, ‘迷途’, ‘小布丁’)

# 如果我想在“黑夜”和“迷途”之间插入“怡静”，我们应该：

>>> temp = temp[:2] + (‘怡静’,) + temp[2:]

**5. x, y, z = 1, 2, 3 请问x, y, z是元组吗？**   不是

所有的多对象的、逗号分隔的、没有明确用符号定义的这些集合默认的类型都是元组，例如：

>>> x,y,z = 1,2,3

>>> type(x)

<class 'int'>

>>> h = x,y,z

>>> type(h)

<class 'tuple'>  
**6. 请写出以下情景中应该使用列表还是元组来保存数据：**

1) 游戏中角色的属性：  
 列表  
2) 你的身份证信息：  
 元组  
3) 论坛的会员：  
 列表  
4) 团队合作开发程序，传递给一个你并不了解具体实现的函数的参数：  
 列表

元组  
5) 航天火箭各个组件的具体配置参数：  
   元组  
6) NASA系统中记录已经发现的行星数据：

元组

列表

**7. 上节课我们通过课后作业的形式学习到了“列表推导式”，那请问如果我把中括号改为小括号，会不会得到“元组推导式”呢？**

不能  
   python 没有“元组推导式”  
**8. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P15：各种奇葩的内置方法

****测试题：****  
     
**0. 还记得如何定义一个跨越多行的字符串吗（请至少写出两种实现的方法）？**

str1 = '''待我长发及腰，将军归来可好？

此身君子意逍遥，怎料山河萧萧。

天光乍破遇，暮雪白头老。

寒剑默听奔雷，长枪独守空壕。

醉卧沙场君莫笑，一夜吹彻画角。

江南晚来客，红绳结发梢。'''

print(str1)

str2 = '待卿长发及腰，我必凯旋回朝。\

昔日纵马任逍遥，俱是少年英豪。\

东都霞色好，西湖烟波渺。\

执枪血战八方，誓守山河多娇。\

应有得胜归来日，与卿共度良宵。\

盼携手终老，愿与子同袍。'

print(str2)

str3 = ('待卿长发及腰，我必凯旋回朝。'

'昔日纵马任逍遥，俱是少年英豪。'

'东都霞色好，西湖烟波渺。'

'执枪血战八方，誓守山河多娇。'

'应有得胜归来日，与卿共度良宵。'

'盼携手终老，愿与子同袍。')

print(str3)  
     
**1. 三引号字符串通常我们用于做什么使用？**   多行注释  
 三引号字符串不赋值的情况下，通常当作跨行注释使用

**2. file1 = open('C:\windows\temp\readme.txt', 'r') 表示以只读方式打开“C:\windows\temp\readme.txt”这个文本文件，但事实上这个语句会报错，知道为什么吗？你会如何修改？**   这里面有\t和\r等转义字符，需要多添加一个\，进行转义

“\t” 横向制表符(TAB)

“\r” 回车符

用原始字符串操作符(R或r)即可

file1 = open(r ‘C:\windows\temp\readme.txt’, ‘r’)  
  
**3. 有字符串：str1 = '<a href="http://www.fishc.com/dvd" target="\_blank">鱼C资源打包</a>'，请问如何提取出子字符串：'www.fishc.com'**

str1 = '<a href="http://www.fishc.com/dvd" target="\_blank">鱼C资源打包</a>'

print(str1.index('w')) #索引值返回：16

print(str1.index('m')) #索引值返回：28

print(str1[16:29]) #索引值加1，才能输出完整，包头不包尾  
  
**4. 如果使用负数作为索引值进行分片操作，按照第三题的要求你能够正确目测出结果吗？** print(str1[-45:-32])  
**5. 还是第三题那个字符串，请问下边语句会显示什么内容？**

>>> str1[20:-36]

print(str1[20:-36]) #显示fishc

**6. 据说只有智商高于150的鱼油才能解开这个字符串（还原为有意义的字符串）：str1 = 'i2sl54ovvvb4e3bferi32s56h;$c43.sfc67o0cm99'**

str[起始位：结束位：步长]

print(str[::3])

输出：ilovefishc.com

****动动手：****  
    
**0. 请写一个密码安全性检查的代码代码：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



passwd = input('请输入需要检查的密码组合：')

length = len(passwd)

strpun = ""

passwd\_strong = 0

if length <= 8:

passwd\_strong += 1

if length > 8:

passwd\_strong += 1

for i in passwd:

if i.isalnum():

strpun += "1"

elif i in string.punctuation:

strpun += "2"

if "1" in strpun:

passwd\_strong += 1

if "2" in strpun:

passwd\_strong += 1

if length >= 16:

passwd\_strong += 1

for i in passwd[:1]:

if i.isalpha():

passwd\_strong += 1

# passwd\_strong == 2 低级

# passwd\_strong == 3 中级

# passwd\_strong == 5 高级

if passwd\_strong == 2:

print("您的密码安全级别评定为：低\n请按以下方式提升您的密码安全级别：")

print("\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")

elif passwd\_strong == 3:

print("您的密码安全级别评定为：中\n请按以下方式提升您的密码安全级别：")

print("\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")

elif passwd\_strong == 5:

print("您的密码安全级别评定为：高\n请继续保持")

## ● P16：字符串：格式化

**测试题：  
0. 根据说明填写相应的字符串格式化符号**

|  |  |
| --- | --- |
| 符   号 | 说     明 |
| %c | 格式化字符及其ASCII码 |
| %s | 格式化字符串 |
| %d | 格式化整数 |
| %o | 格式化无符号八进制数 |
| %x | 格式化无符号十六进制数 |
| %X | 格式化无符号十六进制数（大写） |
| %f | 格式化定点数，可指定小数点后的精度 |
| %e | 用科学计数法格式化定点数 |
| %E | 作用同%e，用科学计数法格式化定点数 |
| %g | 根据值的大小决定使用%f或者%e |
| %G | 作用同%e，根据值的大小决定使用%F或者%E |

**1. 请问以下这行代码会打印什么内容？**

>>> "{{1}}".format("不打印", "打印")

>>>{1}

>>>’{1}’  
  
**2. 以下代码中，a, b, c是什么参数？**

>>> "{a} love {b}.{c}".format(a="I", b="FishC", c="com")

'I love FishC.com'

关键字参数，与format里面的参数一一对应

**3. 以下代码中，{0}, {1}, {2}是什么参数？**

>>> "{0} love {1}.{2}".format("I", "FishC", "com")

'I love FishC.com'

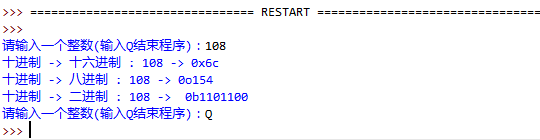
位置参数

**4. 如果想要显示Pi = 3.14，format前边的字符串应该怎么填写呢？**

''.format('Pi = ', 3.1415)

“{0} {1}”.format(‘Pi=’,3.1415)

‘{0}{1:.2f}’.format(‘Pi = ’,3,1415)  
****动动手：****  
**0. 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）：**



代码名称：BinaryConversion.py

代码地址：D:\code\python\yu\_c

**1. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P17：序列！序列

****测试题：****  
**0. 我们根据列表、元组和字符串的共同特点，把它们三统称为什么？**   序列，因为他们有以下共同点：

1) 都可以通过索引得到每一个元素

2) 默认索引值总是从0开始(当然灵活的Python还支持负数索引)

3) 可以通过分片的方法得到一个范围内的元素的集合

4) 有很多共同的操作符(重复操作符，拼接操作符，成员关系操作符)

**1. 请问分别使用什么BIF，可以把一个可迭代对象转换为列表、元组和字符串？** list([iterable]) 把可迭代对象转换为列表

tuple([iterable]) 把可迭代对象转换为元组

str(obj) 把对象转换为字符串

**2. 你还能复述出“迭代”的概念吗？**   迭代：重复反馈过程的活动，其目的接近或达到所需的目标或结果，每一次对过程重复，我们称之为一次迭代，而每一次迭代得到的结果都会被用来为下一次迭代做初始值

迭代：所谓迭代，是重复反馈过程的活动，其目的通常是为了接近并到达所需的目标或结果。每一次对过程的重复被称为一次“迭代”，而每一次迭代得到的结果会被用来作为下一次迭代的初始值。

**3. 你认为调用 max('I love FishC.com') 会返回什么值？为什么？**    v，把字母转换成ASCII码来进行比较

因为字符串在计算机中是以ASCII码的形式存储

**4. 哎呀呀，现在的小屁孩太调皮了，邻居家的孩子淘气，把小甲鱼刚写好的代码画了个图案，麻烦各位鱼油恢复下啊，另外这家伙画的是神马吗？怎么那么眼熟啊！？？**

****动动手：****  
**0. 猜想一下 min() 这个BIF的实现过程**

nums = [1,23,4,57,63,6]

min = nums[0]

for each in nums:

if each < min:

min = each

print(min)

def min(x):

least = x[0]

for each in x:

if each < least:

least = each

return least

print(min(‘123456789’))

**1. 视频中我们说 sum() 这个BIF有个缺陷，就是如果参数里有字符串类型的话就会报错，请写出一个新的实现过程，自动“无视”参数里的字符串并返回正确的计算结果**

D:\code\python\yu\_c

SumStatistics.py     
**2. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P18：函数，Python的乐高积木

****测试题：****  
  
**0. 你有听说过DRY吗？** DRY（Don’t Repeat Yourself）：不做重复的事

降低可管理单元复杂度的一个基本策略就是将他们拆解成更小的单元。

答：DRY是程序员们公认的指导原则

快快武装你的思想吧，拿起函数，不要再去重复拷贝一段代码了！

**1. 都是重复一段代码，为什么我要使用函数（而不使用简单的拷贝黏贴）呢？** 直接调用，使用率高

使用函数有以下好处：

· 可以降低代码量（调用函数只需要一行，而拷贝粘贴需要N倍代码）

· 可以降低维护成本（函数只需要修改def部分内容，而拷贝粘贴则需要每一处出现的地方都作修改）

· 更容易阅读（没有人希望看到一个程序重复一万行）  
  
**2. 函数可以有多个参数吗？** 原则上可以有多个参数，但是建议参数不要过多。

答：可以的，理论上你想要多少个就可以有多少个，只不过函数的参数过多，在调用的时候出错的机率就会大大提高，因而写这个函数的程序员会被相应的问候祖宗，所以，尽量精简吧，在Python的世界里，精简才是王道！  
  
**3. 创建函数使用什么关键字，要注意什么？** def 函数名():

要注意的是括号不能忘记，还有冒号

答：使用“def”关键字，要注意函数名后边要加上小括号“()”，然后小括号后面是冒号“:”，然后缩进部分均属于函数体的内容，例如：

def Myfun():

# 我是函数体

# 我也是函数体

#我们都属于函数Myfun()

# 奥，我不属于Myfun()函数的了  
  
**4. 请问这个函数有多少个参数？**

def MyFun((x, y), (a, b)):

    return x \* y - a \* b

两个参数

答：如果你回答的是两个，那么恭喜你错啦，答案是0，因为类似于这样的写法是错误的！

我们分析一下，函数的参数需要的是变量，而这里你试图用“元组”的形式来传递是不可行的。

我想你如果这么写，你应该要表达这么个意思：

def MyFun(x,y):

return x[0] \* x[1] - y[0] \* y[1]

MyFun((3,4),(1,2))   
  
5. 请问调用以下这个函数会打印什么内容？

>>> def hello():

        print('Hello World!')

        return

        print('Welcome To FishC.com!')

答：不打印，没有调用函数

hell()

Hello world!

因为当Python执行return语句的时候，Python认为函数到此结束，需要返回了（尽管没有任何返回值）。  
  
****动动手：****  
0. 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。  
 def power(x,y):

num = x\*\*y

return num

print(power(11,11))  
  
**1. 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。**

辗转相除法，又名欧几里得算法（Euclidean algorithm），是求最大公约数的一种方法。它的具体做法是：用较小数除较大数，再用出现的余数(第一余数)去除除数，在用出现的余数(第二余数)去除第一余数，如此反复，直到最后余数是0为止。如果是求两个数的最大公约数，那么最后的除数就是这两个数的最大公约数。  
def gcd(x,y):

while y :

t = x % y

x = y

y = t

return x

print(gcd(10,12))

**2. 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”（[补脑链接](https://baike.baidu.com/item/%E5%8D%81%E8%BF%9B%E5%88%B6%E8%BD%AC%E4%BA%8C%E8%BF%9B%E5%88%B6" \t "https://fishc.com.cn/_blank)）的方式，结果与调用 bin() 一样返回字符串形式。** 格式：bin(111)——>0b1101111

def Dec2Bin(dec):

temp = []

result = ''

while dec:

quo = dec % 2

dec = dec // 2 # 返回int型，如果dec/2 会有小数点

temp.append(quo)

while temp: # 这是列表，为空结束循环

result += str(temp.pop()) # 从尾部删除数据，并打印出来

return result

print(Dec2Bin(22))

**3. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P19：函数 灵活即强大

****测试题：****  
        
0. 请问以下哪个是形参哪个是实参？

def MyFun(x):

    return x \*\* 3

y = 3

print(MyFun(y))

x是形式参数(形参)，y是实际参数(实参)。

跟绝大部分编程语言一样，形参指的是函数创建和定义过程中小括号里的参数，而实参指的是函数在调用过程中传递进去的参数。

**1. 函数文档和直接用“#”为函数写注释有什么不同？**     当我们不看源代码，直接使用调用函数文档形式即可知道该函数的作用

给函数写文档是为了让别人可以更好的理解你的函数，所以这是一个好习惯：

def MyFirstFunction(name):

‘函数文档在函数定义的最开头部分，用不记名字符串表示’

print(‘aaa!’)

我们看到在函数开头写下的字符串Ta是不会打印出来的，但Ta会作为函数的一部分存储起来，这个我们称之为函数文档字符串，Ta的功能跟注释是一样的。

函数文档字符串可以按如下方式访问：

>>> MyFirstFunction.\_\_doc\_\_

'函数文档在函数定义的最开头部分，用不记名字符串表示'

另外，我们用help()来访问这个函数也可以看到这个文档字符串：

>>> help(MyFirstFunction)

Help on function MyFirstFunction in module \_\_main\_\_:

MyFirstFunction(name)

函数文档在函数定义的最开头部分，用不记名字符串表示

**2. 使用关键字参数，可以有效避免什么问题的出现呢？** 可以有效避免因为索引的问题给形参赋错值

关键字参数：是指函数在调用的时候，带上参数的名字去指定具体调用的是哪个参数，从而可以不用按照参数的顺序调用函数，例如：

>>>def SaySome(name,words):

print(name + ‘->’ + words)

SaySome(words=’让编程改变世界！’, name=‘河蟹’):

使用关键字参数，可以有效避免因不小心搞乱参数的顺序导致的BUG出现。

**3. 使用help(print)查看print()这个BIF有哪些默认参数？分别起到什么作用？**      print(value, ..., sep=’ ’ , end=’\n’, file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

# 文件类型对象，默认是sys.stdout(标准输出流)

sep：string inserted between values, default a space.

# 第一个参数如果有多个值（第一个参数是收集参数），各个值之间默认用空格（space）隔开

end：string appended after the last value,default a newline.

# 打印最后一个值之后默认参数一个新行标识符(‘/n’)

flush:whether to forcibly flush the stream.

# 是否强制刷新流

**4. 默认参数和关键字参数表面最大的区别是什么？**

**默认参数：在形参关键字设置默认参数，如果实参不传值过来，默认使用默认参数，实参传值，使用实参值参数**

**关键字参数：实参的关键字与形参相匹配，不使用索引取值，而是关键字取值**

       关键字参数是在函数调用的时候，通过参数名制定需要赋值的参数，这样做就不怕搞不清参数的顺序而导致函数调用出错。

默认参数：是在参数定义的过程中，为形参赋初值，当函数调用的时候，不传递实参，则默认使用形参的初始值代替。          
****动动手：****  
       
0. 编写一个符合以下要求的函数：  
     
    a) 计算打印所有参数的和乘以基数（base=3）的结果  
    b) 如果参数中最后一个参数为（base=5），则设定基数为5，基数不参与求和计算。  
     def mFun(\*param, base = 3):

result = 0

for each in param:

result += each

result \*= base

print('结果是：', result)

mFun(1,2,3,4,5,base=5)  
     
**1. 寻找水仙花数  
题目要求：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是一个水仙花数。编写一个程序，找出所有的水仙花数。**

def Narcissus():

for each in range(100,1000):

temp = each

sum = 0

while temp:

sum = sum +(temp%10)\*\*3

temp = temp // 10 # 用地板除，返回int型

if sum == each:

print(each , end='\t')

print("所有的水仙花数分别是：", end='')

Narcissus()  
     
**2. 编写一个函数 findstr()，该函数统计一个长度为 2 的子字符串在另一个字符串中出现的次数。例如：假定输入的字符串为“You cannot improve your past, but you can improve your future. Once time is wasted, life is wasted.”，子字符串为“im”，函数执行后打印“子字母串在目标字符串中共出现 3 次”。**

def findStr(desStr,subStr):

count = 0

length = len(desStr)

if subStr not in desStr:

print('在目标字符串中未找到字符串！')

else:

for each1 in range(length-1):

if desStr[each1] == subStr[0]:

if desStr[each1+1] == subStr[1]:

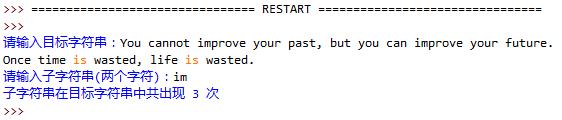
count += 1

print('子字符串在目标字符串中共出现 %d 次'% count)

desStr = input('请输入目标字符串：')

subStr = input('请输入子字符串(两个字符):')

findStr(desStr,subStr)



## ● P20：函数— 我的地盘听我的

****测试题：****来自：[https://fishc.com.cn](https://fishc.com.cn/" \t "https://fishc.com.cn/_blank)  
**0. 下边程序会输入什么？**

def next():

print('我在next()函数里...')

pre()

def pre():

print('我在pre()函数里...')

next()

输出：

我在next()函数里...

我在pre()函数里...

**1. 请问以下这个函数有返回值吗？版权属于：**

>>> def hello():

print('Hello FishC!')

答：有返回值，返回值为 ‘None’

答：如果没有使用return语句指定返回值，Python也不是什么都不返回的，Ta会返回一个None对象，所以我们说Python 所有的函数都有返回值。

**2. 请问 Python 的 return 语句可以返回多个不同类型的值吗？**

答：可以

答：可以，默认用逗号隔开，是以元组的形式返回，你当然也可以用列表包含起来返回：

**3. 目测以下程序会打印什么内容：**

def fun(var):

var = 1314

print(var, end='')

var = 520

fun(var)

print(var)

输出：1314520

**4. 目测以下程序会打印什么内容？**

var = ' Hi '

def fun1():

global var

var = ' Baby '

return fun2(var)

def fun2(var):

var += 'I love you'

fun3(var)

return var

def fun3(var):

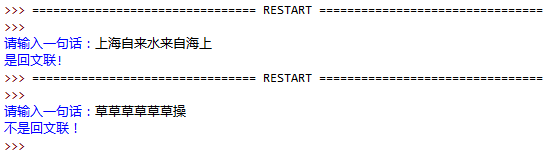
var = ' 小甲鱼 '

print(fun1())

输出：Bady I love you

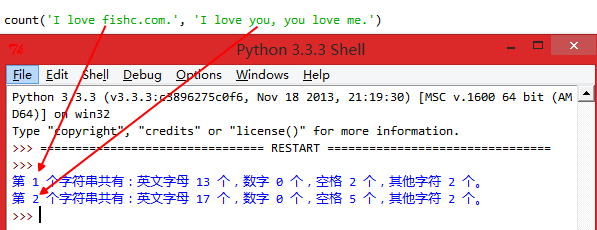
****动动手：****

**0. 编写一个函数，判断传入的字符串参数是否为“回文联”（回文联即用回文形式写成的对联，既可顺读，也可倒读。例如：上海自来水来自海上）**程序执行结果如图



**1. 编写一个函数，分别统计出传入字符串参数（可能不只一个参数）的英文字母、空格、数字和其它字符的个数。**

程序执行结果如图：



**2. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

y

## ● P21：内嵌函数和闭包

****测试题：****Powered by [bbs.fishc.com](http://bbs.fishc.com/" \t "https://fishc.com.cn/_blank)  
        
**0. 如果希望在函数中修改全局变量的值，应该使用什么关键字？**

global 关键字

count = 5

def MyFun():

global count

count = 10

print(count)

>>MyFun()

10

>>count

10  
**1. 在嵌套的函数中，如果希望在内部函数修改外部函数的局部变量，应该使用什么关键字？**

nonlocal 关键字

def Fun1():

x = 5

def Fun2():

nonlocal x

x \*= x

return x

return Fun2()

>>>Fun1()

25  
**2. Python 的函数可以嵌套，但要注意访问的作用域问题哦，请问以下代码存在什么问题呢？**

def outside():

print(‘I am outside!’)

def inside():

print(‘I am inside!’)

inside()

答：不能直接访问inside()，因为它定义在outside()函数里面

使用嵌套函数要注意一点就是作用域的问题，inside()函数是内嵌在outside()函数中的，所以inside()是人妻，除了身为老公的outside()可以碰(调用)，在外边或者别的函数体里是无法对其进行调用的。

正确调用:

def outside():

print(‘I am outside!’)

def inside():

print(‘I am inside!’)

inside()

outside()

**3. 请问为什么代码 A 没有报错，但代码 B 却报错了？应该如何修改？**

代码A：

def outside():

var = 5

def inside():

var = 3

print(var)

inside()

outside()

代码B：

def outside():

var = 5

def inside():

nonlocal var

print(var)

var = 3

inside()

outside()

答：使用nonlocal

仔细一看报错内容是：UUnboundLocalError: local variable ‘var’ referenced before assignment,s说的是变量var没有被定义就拿来使用了，所以出错了。

这里outside()函数里有一个var变量，但要注意的是，内嵌函数inside()也有一个同名的变量，Python为了保护变量的作用域，故将outside()的var变量屏蔽起来，因此此时是无法访问到外层的var变量的。

**4. 请问如何访问 funIn() 呢？**

def funOut():

def funIn():

print(‘宾果！你成功访问到我啦！’)

return funIn()

funOut()

只需要直接调用funOut()即可。

**5. 请问如何访问 funIn() 呢？**

def funOut():

def funIn():

print(‘宾果！你成功访问到我啦！’)

return funIn

i = funOut()

i()

区别于上一题，这里需要用funOut()()访问啦：

funOut()()

**6. 以下是“闭包”的一个例子，请你目测下会打印什么内容？**

def funX():

x = 5

def funY():

nonlocal x

x += 1

return funY

return fun Y

a = funX()

print(a())

print(a())

print(a())

答：666或678

6

7

8

疑问：这怎么跟全局变量一样？局部变量x不是应该在每次调用的时候都重新初始化了吗？

解惑：当a = funX()的时候，只要a变量没有被重新赋值，funX()就没有被释放，也就是说局部变量x就没有被重新初始化。所以当全局变量不适用的时候，可以考虑闭包更稳定和安全。

****动动手：****版权属于：[bbs.fishc.com](http://bbs.fishc.com/" \t "https://fishc.com.cn/_blank)  
**0. 请用已学过的知识编写程序，统计下边这个长字符串中各个字符出现的次数并找到小甲鱼送给大家的一句话。9cJiUE**（由于我们还没有学习到文件读取方法，大家下载后拷贝过去即可)

请下载字符串文件： IMG_256 [string1.zip](https://fishc.com.cn/forum.php?mod=attachment&aid=MjAxNDZ8ZjRjYzhmYzN8MTU4Njc2NDI4NXw4NTUyMjd8NDI2ODU=" \t "https://fishc.com.cn/_blank) (55.49 KB, 下载次数: 26793)

相关代码：CountStr.py

**1. 请用已学过的知识编写程序，找出小甲鱼藏在下边这个长字符串中的密码，密码的埋藏点符合以下规律：**    a) 每位密码为单个小写字母Powered by [bbs.fishc.com](http://bbs.fishc.com/" \t "https://fishc.com.cn/_blank)  
    b) 每位密码的左右两边均有且只有三个大写字母  
（由于我们还没有学习到文件读取方法，大家下载后拷贝过去即可）9zk3R(q{  
  
请下载字符串文件： IMG_257 [string2.zip](https://fishc.com.cn/forum.php?mod=attachment&aid=MjAxNDd8MzM2M2JmMjJ8MTU4Njc2NDI4NXw4NTUyMjd8NDI2ODU=" \t "https://fishc.com.cn/_blank) (6.17 KB, 下载次数: 24458)

相关代码：isupper\_islower.py  
  
**2. 请写下这一节课你学习到的内容：格式不限，回忆并复述是加强记忆的好方式！**

## ● P22：函数-lambda表达式

**单词记录：**

lambda：匿名函数

filter()：过滤器

map()：映射

****测试题：****版权属于：[https://fishc.com.cn](https://fishc.com.cn/" \t "https://fishc.com.cn/_blank)  
**0. 请使用lambda表达式将下边函数转变为匿名函数？**

def fun\_A(x, y=3):

return x \* y

转lambda函数：

g = lambda x,y=3: x \* y

print(g(2))

**1. 请将下边的匿名函数转变为普通的屌丝函数？**

lambda x : x if x % 2 else None

转函数：

def Myfun(x):

if x%2 != 0:

return x

else:

return None

print(Myfun(55))

**2. 感受一下使用匿名函数后给你的编程生活带来的变化？**

答：使用匿名函数后，函数名定义的烦恼没有了，代码减少了，开发效率提高了

**3. 你可以利用 filter() 和 lambda 表达式快速求出 100 以内所有 3 的倍数吗？**

g = list(filter(lambda x: x if x%3==0 else None ,range(100)))

print(g)

**4. 还记得列表推导式吗？完全可以使用列表推导式代替 filter() 和 lambda 组合，你可以做到吗？**

print([i for i in range(3,101,3)])

**5. 还记得 zip 吗？使用 zip 会将两数以元祖的形式绑定在一块，例如：**

>>> list(zip([1, 3, 5, 7, 9], [2, 4, 6, 8, 10]))

[(1, 2), (3, 4), (5, 6), (7, 8), (9, 10)]

但如果我希望打包的形式是灵活多变的列表而不是元祖（希望是 [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]] 这种形式），你能做到吗？（采用 map 和 lambda 表达式）

g = list(map(lambda x,y:[x,y],[1,3,5,7,9],[2,4,6,8,10]))

print(g)

**6. 请目测以下表达式会打印什么？**

def make\_repeat(n):

return lambda s : s \* n

double = make\_repeat(2)

print(double(8)) # 16

print(double('FishC')) # FishCFishC

## ● P23：递归

****单词记录：****

****测试题：****  
        
**0. 递归在编程上的形式是如何表现的呢？** 在编程上，递归表现为函数调用本身这么一个行为。

**1. 递归必须满足哪两个基本条件？** 1、函数调用自身

2、设置了正确的返回条件

**2. 思考一下，按照递归的特性，在编程中有没有不得不使用递归的情况？** 答：例如汉诺塔，目录索引（因为你永远都不知道这个目录里面是否还有目录），快速排序（二十世纪十大算法之一），树结构的定义等如果使用递归，会事半功倍，否则会导致程序无法实现或相当难以理解。  
  
**3. 用递归去计算阶乘问题或斐波那契数列是很糟糕的算法，你知道为什么吗？** 答：小甲鱼在课程的开头说”普通程序员用迭代，天才程序员用递归”这句话不无道理的。

但是你不要理解错了，不是说会使用递归，把所有迭代的东西用递归来代替就是”天才程序员”了，恰好相反，如果你整的这样做的话，那你就是”乌龟程序员”了.

为什么这样说的呢？不要忘记了，递归的实现是自个儿调用自个儿，每次函数的调用都需要进行压栈、弹栈、保存和恢复寄存器的栈操作，所以在这上边是非常消耗时间和空间的。

另外，如果递归一旦忘记了返回，或者错误的设置了返回条件，那么执行这样的递归代码就会变成一个无底洞：只进不出！所以在写递归代码的时候，千万要记住口诀：递归递归，归去来兮！出来混，总有一天是要还的！  
  
**4. 请聊一聊递归的优缺点（无需官方陈词，想到什么写什么就可以）  
 优点：**

1、递归的基本思想是把规模大的问题转变成规模小的问题组合，从而简化问题的难度(例如汉诺塔游戏)。  
 2、有些问题使用递归使得代码简洁易懂(例如你可以很容易的写出前中后序的二叉树遍历的递归算法，但如果要写出相应的非递归算法就不是初学者可以做到的了。)

**缺点：**

1、由于递归的原理是函数调用自个儿，所以一旦大量的调用函数本身空间和时间消耗是’奢侈的’

2、初学者很容易错误的设置了返回条件，导致递归代码无休止调用，最终栈溢出，程序崩溃。  
**5. 拿手机拍一张“递归自拍照片”**



****动动手：****  
       
**0. 使用递归编写一个 power() 函数模拟内建函数 pow()，即 power(x, y) 为计算并返回 x 的 y 次幂的值。  
 相关代码：**PowerFunction.py  
  
**1. 使用递归编写一个函数，利用欧几里得算法求最大公约数，例如 gcd(x, y) 返回值为参数 x 和参数 y 的最大公约数。**

**相关代码：**GcdFunction.py

## ● P24：递归：这帮小兔崽子、汉诺塔

****动动手：****

**0. 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）。**

RecursionDec2Bin.py

**1. 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到列表中。举例：get\_digits(12345) ==> [1, 2, 3, 4, 5]**

get\_digits.py

**2. 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，亲还能骄傲的说我可以吗？**

is\_palindrome.py

**3. 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4个人大2岁。问第4个人岁数，他说比第3个人大2岁。问第三个人，又说比第2人大两岁。问第2个人，说比第一个人大两岁。最后问第一个人，他说是10岁。请问第五个人多大？”

fnFiveYear.py

## ● P25：字典：当索引不好用时

前言：键(key)，值(value)。即是“哈希”，也称“关系数组”，字典是python中唯一的映射类型。

****测试题：****  
  
**0. 当你听到小伙伴们在谈论“映射”、“哈希”、“散列”或者“关系数组”的时候，事实上他们就是在讨论什么呢？** 字典

是的，事实上他们就是在讨论我们这一讲介绍的“字典”，都是一个概念！(切记，装X的本质就是把同一个东西说成各种不同的事物~)  
  
**1. 尝试一下将数据（'F': 70, 'C': 67, 'h': 104, 'i': 105, 's': 115）创建为一个字典并访问键 'C' 对应的值？** dict1 = {'F': 70, 'C': 67, 'h': 104, 'i': 105, 's': 115}

print(dict1['C'])  
  
**2. 用方括号（“[]”）括起来的数据我们叫列表，那么使用大括号（“{}”）括起来的数据我们就叫字典，对吗？** 不一定，大括号括起来，还需要有key，value，需要映射关系

不对。

NotADict = {1,2,3,4,5}

type(NotADict)

<class ‘set’>  
 不难发现，虽然我们用大括号(“{}”)把一些数据括起来了，但由于没有反映出这些数据有映射的关系，所以创建出来的不是字典，而是叫‘set’的东西，那‘set’到底又是啥玩意儿呢? 请看第27讲|集合：在我的世界里，你就是唯一！  
**3. 你如何理解有些东西字典做得到，但“万能的”列表却难以实现（臣妾做不到T\_T）？** 代码在书92页

列表brand、slogan的索引和相对的值是没有任何关系的，我们可以看出唯一有联系的就是两个列表间，索引号相同的元素是有关系的（品牌对应口号），所以这里我们通过brand.index(‘鱼C工作室’)这样的语句，间接的实现了通过品牌查找对应的口号的功能。

太过麻烦，需要有字典这种映射类型的出现。  
  
**4. 下边这些代码，他们都在执行一样的操作吗？你看得出差别吗？**

>>> a = dict(one=1, two=2, three=3)

>>> b = {'one': 1, 'two': 2, 'three': 3}

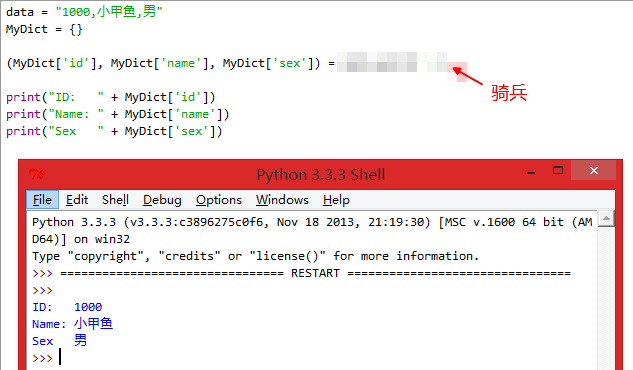
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))

>>> d = dict([('two', 2), ('one', 1), ('three', 3)])

>>> e = dict({'three': 3, 'one': 1, 'two': 2})

**他们的键一样，对应的值也一样。没差别**

**5. 如图，你可以推测出打了马赛克部分的代码吗？**



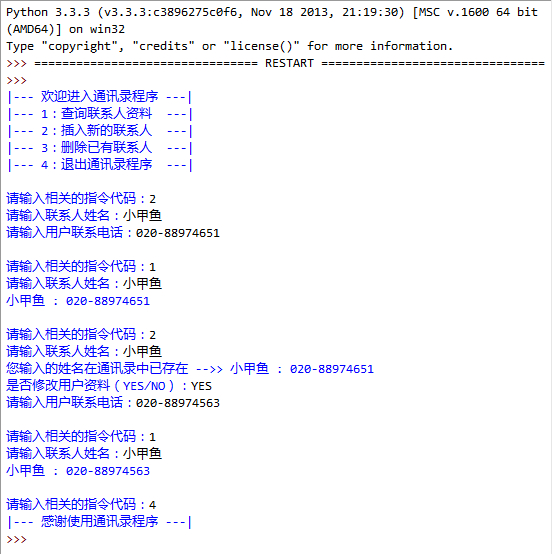
****(Mydict['id'], Mydict['name'], Mydict['sex']) = ('1000','小甲鱼','男')****

****# 还记得字符串的分割方法吧，别学过就忘了****

****(Mydict['id'], Mydict['name'], Mydict['sex']) = data.split(‘,’)****

****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：**

Mail\_list.py



## ● P26：字典：当索引不好用时2

****测试题：****

**0. Python的字典是否支持一键（Key）多值（Value）？** 不支持，键值对，一个键只能对应一个值

不支持，对相同的键再次赋值会将上一次的值直接覆盖

dict1 = {1:’one’, 1:’yi’}

print(dict1[1])

‘yi’  
  
**1. 在字典中，如果试图为一个不存在的键（Key）赋值会怎样？** 创建该键值对

会自动创建对应的键(key)并添加相应的的值(value)进去  
  
**2. 成员资格操作符（in和not in）可以检查一个元素是否存在序列中，当然也可以用来检查一个键（Key）是否存在字典中，那么请问哪种的检查效率更高些？为什么？** 检查键更快一点；

在字典中检查键(key)是否存在比在序列中检查指定元素是否存在更高效。因为字典的原理是使用了哈希算法存储，一步到位，不需要使用查找算法进行匹配，因此时间复杂度是0(1)，效率非常高。  
  
**3. Python对键（Key）和值（Value）有没有类型限制？** 有的，键必须为不可更改的数值，类型，list就不能做键

python对键的要求相对要严格一些，要求它们必须是可哈希(Hash)的对象，不能是可变类型(包括变量、列表、字典本身等)。

但是python对值是没有任何限制的，它们可以是任意的python对象。

如果不清楚哈希原理以及字典的存放原理的童鞋，推荐阅读下小甲鱼帮你整理的这篇文章：你知道Python的字典（Dict）是如何存储的吗？（[http://bbs.fishc.com/thread-45016-1-1.html](http://bbs.fishc.com/thread-45016-1-1.html" \t "https://fishc.com.cn/_blank)）  
  
**4. 请目测下边代码执行后，字典dict1的内容是什么？**

>>> dict1.fromkeys((1, 2, 3), ('one', 'two', 'three'))

>>> dict1.fromkeys((1, 3), '数字')

报错，不能更改个别键里面的值

执行完成后，字典dict1的内容是：{1:’数字’,3:‘数字’}

这里要注意的是，fromkeys方法是直接创建一个新的字典，不要试图使用它来修改一个原有的字典，因为它会直接无情的把整个字典给覆盖掉。

**5. 如果你需要将字典dict1 = {1: 'one', 2: 'two', 3: 'three'}拷贝到dict2，你应该怎么做？**

dict2 = dict1.copy()

可以利用字典的copy()方法：dict2 = dict1.copy()，如果用直接赋值的方法（dict2 = dict1）,当我们改变dict2的值，dict1也会改变。

****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：**



## ● P27：集合：在我的世界里，你就是唯一

0. 请问集合的唯一作用是什么呢？  
 唯一

标准答案：集合几乎所有的作用就是确保里面包含的元素的唯一性，就像世界上没有两片完全相同的树叶一样，集合内不可能存在两个相同的元素！  
1. 如果你希望创建的集合是不变的，应该怎么做？  
 使用frozenset()函数，就不能add()方法添加集合的元素了。  
2. 请问如何确定一个集合里边有多少个元素？  
 用迭代方式进行统计

标准答案：len()函数正好可以满足你此刻的需求

num\_set = set([1,2,3,4,5])

len(num\_set)

5  
3. 请目测以下代码会打印什么内容？

>>> num\_set = set([1, 2, 3, 4, 5])

>>> num\_set[0]

报错误：TypeError: 'set' object does not support indexing

标准答案：报错，因为集合是无序的

4. 请问 set1 = {[1, 2]} 和 set1 = set([1, 2]) 执行的结果一样吗？  
 标准答案：不一样，set1 = set([1,2])会生成一个集合{1,2}，但set1 = {[1,2]}却会报错

>>> set1 = {[1, 2]}

Traceback (most recent call last):

File "<pyshell#17>", line 1, in <module>

set1 = {[1, 2]}

TypeError: unhashable type: 'list'

从报错信息上我们看到“列表不是可哈希类型”，没错，列表是可变的，它怎么可以哈希呢？！咦，等等，这句话好像在那听过……呃，敢情集合跟字典的存储方式一样的丫！  
其实你再想想就会觉得很有道理，利用哈希函数计算，相同的元素得到的哈希值（存放地址）是相同的，所以在集合中所有相同的元素都会覆盖掉，因此有了集合的唯一性。  
然后你继续接着想就觉得更有道理了，通过哈希函数计算的地址不可能是按顺序排放的，所以集合才强调是无序的！

5. 打开你的IDLE，输入set1 = {1, 1.0}，你发现了什么？  
 标准答案：没错， 集合内容是{1.0}，其实你弄懂了上一题，这一题一样容易：因为在Python的哈希函数会将相同的值的元素计算得到相同的地址，所以1和1.0是等值的^\_^  
6. 请问如何给集合添加和删除元素？

添加元素：set1.add(6)

删除元素：set1.remove(5)

标准答案：>>> num1.add(6)

>>> num1

{0, 1, 2, 3, 4, 5, 6}

>>> num1.remove(6)

>>> num1

{0, 1, 2, 3, 4, 5}

## ● P28：文件：因为懂你，所以永恒

****测试题：****  
  
**0. 下边只有一种方式不能打开文件，请问是哪一种，为什么？**

>>> f = open('E:/test.txt', 'w') # A

>>> f = open('E:\test.txt', 'w') # B

>>> f = open('E://test.txt', 'w') # C

>>> f = open('E:\\test.txt', 'w') # D

C

标准答案：答：B不能打开文件。  
Windows在路径名中既可以接受斜线（/）也可以接受反斜线（\），不过如果使用反斜线作为路径名的分隔符的话，要注意使用双反斜线（\\）进行转义，否则Python会将反斜线进行转义，例如（\n）看成一个换行符，（\t）看作一个制表符等。

**1. 打开一个文件我们使用open()函数，通过设置文件的打开模式，决定打开的文件具有那些性质，请问默认的打开模式是什么呢？** r：只读

标准答案：open()函数默认的打开模式是’rt’，即可读、文本的模式打开。  
  
**2. 请问 >>> open('E:\\Test.bin', 'xb') 是以什么样的模式打开文件的？** 二进制模式打开文件

标准答案：以”可写入以及二进制模式”打开文件”E:\\Test.bin”。

这里注意的是’x’和’w’均是以”可写入”的模式打开文件，但以’x’模式打开的时候，如果路径下已经存在存在相同的文件名，会抛出异常，而’w’模式的话会直接覆盖同名文件。因此，’w’模式打开文件会比较危险，容易导致此前的内容遗失，因此使用’w’模式打开文件前先检查该文件是否存在显得非常重要！  
  
**3. 尽管Python有所谓的“垃圾回收机制”，但对于打开了的文件，在不需要用到的时候我们仍然需要使用f.close()将文件对象“关闭”，这是为什么呢？**

1、断电依然会丢失数据

2、养成一个好习惯

标准答案：Python拥有垃圾收集机制，会在文件对象的引用计数降至零的时候自动关闭文件，所以在Python编程里，如果忘记关闭文件并不会造成内存泄漏那么危险。

但并不是说就可以不要关闭文件，如果你对文件进行了写入操作，那么你应该在完成写入之后进行关闭文件。因为Python可能会缓存你写入的数据，如果这中间断电了什么的，那些缓存的数据根本就不会写入到文件中。所以，为了安全起见，要养成使用完文件后立刻关闭的优雅习惯。  
  
**4. 如何将一个文件对象（f）中的数据存放进列表中？** f = open(‘D:\code\python\pythontest\yu\_c\test\_data\test.txt’)

print(list(f))

标准答案：list(f)，是不是非常的方便！  
  
**5. 如何迭代打印出文件对象（f）中的每一行数据？** 标准答案：

for each\_line in f:

print(each\_line)

**6. 文件对象的内置方法f.read([size=-1])作用是读取文件对象内容，size参数是可选的，那如果设置了size=10，例如f.read(10)，将返回什么内容呢？** 返回前10个字符的数据

标准答案：将返回从文件指针开始（注意这里不是文件头哦）的连续10字符。

**7. 如何获得文件对象（f）当前文件指针的位置？**

tell()方法

标准答案：f.tell()会告诉你  
  
**8. 还是视频中的那个演示文件（record.txt），请问为何f.seek(45, 0)不会出错，但f.seek(46)就出错了呢？**

>>> f.seek(46)

46

>>> f.readline()

Traceback (most recent call last):

File "<pyshell#18>", line 1, in <module>

f.readline()

UnicodeDecodeError: 'gbk' codec can't decode byte 0xe3 in position 4: illegal multibyte sequence

答：程序不知道文件从哪里开始移动指针

标准答案：因为使用f.seek()定位的文件指针是按字节为单位进行计算的，演示文件(record.txt)是以GBK进行编码的，按照规则，一个汉字需要占用两个字节，f.seek(45)的位置位于字符”小”的开始位置，因此可以正常答应，而f.seek(46)的位置刚好位于字符”小”的中间位置，因此按照GBK编码的形式无法将其解码！

****动动手：****

**0. 尝试将文件（ IMG_256 [OpenMe.mp3](https://fishc.com.cn/forum.php?mod=attachment&aid=MjI5Mzh8NTlhZDM4YWN8MTU5OTEwMjkwMnw4NTUyMjd8NDUzNjQ=" \t "https://fishc.com.cn/_blank) (700 Bytes, 下载次数: 15716) ）打印到屏幕上** f = open('OpenMe.mp3')

for each\_line in f:

print(each\_line, end='')

f.close()

**1. 编写代码，将上一题中的文件（OpenMe.mp3）保存为新文件（OpenMe.txt）**

f1 = open('OpenMe.mp3')

f2 = open('OpenMe.txt', 'x') # 使用”x”打开更安全

f2.write(f1.read())

f2.close()

f1.close()

## ● P31：永久存储：腌制一缸美味的泡菜

****测试题：****

0. pickle的实质是什么？

把python的对象都转化为二进制的形式存放

答：pickle的实质就是利用一些算法将你的数据对象“腌制”成二进制文件，存储在磁盘上，当然也可以放在数据库或者通过网络传输到另一台计算机上。

1. 使用pickle的什么方法存储数据？

答：pickle.dump(data,file)方法 # 第一个参数是待存储的数据对象，第二个参数是目标存储的文件对象，注意要先使用’wb’的模式open文件。

2. 使用pickle的什么方法读取数据？

答：pickle.load(file)方法 # 参数是目标存储的文件对象，注意要先使用’rb’的模式open文件。

3. 使用pickle能不能保存为”\*.txt”类型的文件？

能保存，但是最好使用.pkl或.pickle保存

答：可以，不过打开后是乱码，因为是以二进制的模式写入的。

****动动手：****

0. 编写一个程序，这次要求使用pickle将文件（ IMG_256 [record.txt](https://fishc.com.cn/forum.php?mod=attachment&aid=MjM1NDh8NDAyYWUxMDl8MTYwMDIyNjc1NHw4NTUyMjd8NDU4OTA=" \t "https://fishc.com.cn/_blank) (1.1 KB, 下载次数: 9037) ）里的对话按照以下要求腌制成不同文件（没错，是第29讲的内容小改，考考你自己能写出来吗？）：

● 小甲鱼的对话单独保存为boy\_\*.txt的文件（去掉“小甲鱼:”）

● 小客服的对话单独保存为girl\_\*.txt的文件（去掉“小客服:”）

● 文件中总共有三段对话，分别保存为boy\_1.txt, girl\_1.txt，boy\_2.txt, girl\_2.txt, boy\_3.txt, gril\_3.txt共6个文件（提示：文件中不同的对话间已经使用“==========”分割）

代码：P31\_0.py

## ● P32：异常处理，你不可能总是对的

0. 结合你自身的编程经验，总结下异常处理机制的重要性？

1. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

>>> my\_list = [1, 2, 3, 4,,]

ValueError

答：语法错误

SyntaxError: invalid syntax

2. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

>>> my\_list = [1, 2, 3, 4, 5]

>>> print(my\_list[len(my\_list)])

IndexError

答：len(my\_list)是获得列表的长度，这里长度为5，该列表各个元素的访问索引号分别是:[0,1,2,3,4]，因此试图访问my\_list(5)会引发IndexError：list index out of range异常

3. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

>>> my\_list = [3, 5, 1, 4, 2]

>>> my\_list.sorted()

答：初学者容易疏忽的错误，列表的排序方法叫list.sort()，sorted()是BIF。因此会引发AttributeError: ‘list’ object has no attribute ‘sorted’异常。

4. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

>>> my\_dict = {'host': 'http://bbs.fishc.com', 'port': '80'}

>>> print(my\_dict['server'])

KeyError

答：尝试访问字典中一个不存在的“键”引发KeyError：‘server’异常，为了避免这个异常发生，可以使用dict.get()方法：

if not my\_dict.get(‘server’):

print(‘您所访问的键【server】不存在！’)

5. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

def my\_fun(x, y):

print(x, y)

my\_fun(x=1, 2)

TypeError

syntaxError:语法错误

答：如果使用关键字参数的话，需要两个参数均使用关键字my\_fun(x=1,y=2)

6. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

f = open('C:\\test.txt', wb)

f.write('I love FishC.com!\n')

f.close()

TypeError

NameError：变量异常

答：注意open()第二参数是字符串，应该f = open(‘C:\\test.txt’,’wb’)。wb不加双引号Python还以为是变量名呢，往上一找，没找到......引发NameError异常。由于打开文件失败，接着下面一连串与f相关的均会报同样的异常。

7. 请问以下代码是否会产生异常，如果会的话，请写出异常的名称：

def my\_fun1():

x = 5

def my\_fun2():

x \*= x

return x

return my\_fun2()

my\_fun1()

SyntaxError: 语法错误 invalid syntax 局部变量x找不到

'''代码改造：

在python3z之前没有直接的解决方案，只能间接地通过容器类型来存放，因为容器类型不是放在栈里，所以不会被“屏蔽掉”。

容器类型这个词就是字符串、列表、元组，这些都是容器类型。

'''

# def my\_fun1():

# x = [5]

# def my\_fun2():

# x[0] \*= x[0]

# return x[0]

#

# my\_fun1()

'''但是到了Python3的世界里，又有了不少的改进，如果我们希望在内部函数里可以修改外部函数里的局部变量的值，

那么也有一个关键字可以使用，就是nonlocal：

'''

# def my\_fun1():

# x = 5

# def my\_fun2():

# nonlocal x

# x \*= x

# return x

# return my\_fun2()

#

# f = my\_fun1()

# print(f)

## ● P33：异常处理，你不可能总是对的2

****测试题：****

0. 我们使用什么方法来处理程序中出现的异常？

答：try-except语句、try-finally语句、raise语句、else语句、with语句方法

1. 一个 try 语句可以和多个 except 语句搭配吗？为什么？

答：可以，因为异常的原因有很多种，只要分别设置不同的异常，就可以捕获这些异常

2. 你知道如何统一处理多类异常吗？

答：知道，except(Error\_one,Error\_two,......)就可以对多类异常统一处理

3. except 后边如果不带任何异常类，Python 会捕获所有（try 语句块内）的异常并统一处理，但小甲鱼却不建议这么做，你知道为什么吗？

答：知道，因为用户按下Ctrl+C终止程序，也会被捕获，用户体验不好

4. 如果异常发生在成功打开文件后，Python 跳到 except 语句执行，并没有执行关闭文件的命令（用户写入文件的数据就可能没有保存起来），因此我们需要确保无论如何（就算出了异常退出）文件也要被关闭，我们应该怎么做呢？

答：finally：不管try语句块有没有错误，都执行该语句块。

5. 请恢复以下代码中马赛克挡住的内容，使得程序执行后可以按要求输出。

代码：Chapter9\_Error/P33\_TryVarious.py

****动动手：****

0. 还记得我们第一个小游戏吗？只要用户输入非整型数据，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。

1. input() 函数有可能产生两类异常：EOFError（文件末尾endoffile，当用户按下组合键 Ctrl+d 产生）和 KeyboardInterrupt（取消输入，当用户按下组合键 Ctrl+c 产生），再次修改上边代码，捕获处理 input() 的两类异常，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。

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？

## ● P34：丰富的else语句及简洁的with语句

****测试题：****

0. 在 Python 中，else 语句能跟哪些语句进行搭配？

1. 请问以下例子中，循环中的 break 语句会跳过 else 语句吗？

2. 请目测以下代码会打印什么内容？

3. 使用什么语句可以使你不比再担心文件打开后却忘了关闭的尴尬？

4. 使用 with 语句固然方便，但如果出现异常的话，文件还会自动正常关闭吗？

5. 你可以换一种形式写出下边的伪代码吗？

****动动手：****

0. 使用 with 语句改写以下代码，让 Python 去关心文件的打开与关闭吧。

1. 你可以利用异常的原理，修改下面的代码使得更高效率的实现吗？

## ● P35：图形用户界面入门：EasyGui

****动动手：****

0. 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？

pythontest/yu\_c/Chapter10\_GUI/P35\_0.py

1. 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）。

pythontest/yu\_c/Chapter10\_GUI/P35\_1.py

2. 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。

3. 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，因为你有可能会被一个小问题卡住，但请坚持，自己想办法找到这个小问题所在并解决它！）

pythontest/yu\_c/Chapter10\_GUI/P35\_3.py

4. 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数量
* 要求三：显示总行数与百分比

pythontest/yu\_c/Chapter10\_GUI/P35\_4.py

## ● P36：类和对象：给大家介绍对象

****测试题：****

0. 对象中的属性和方法，在编程中实际是什么？

属性：特征

方法：行为

**答：变量（属性）和函数（方法）。**

1. 类和对象是什么关系呢？

类：定义了属性和方法，等于画好了施工图

对象：按照施工图进行施工，最终变成房子

关系：一个类，可以被多个对象调用，一张施工图可以造很多房子

**答：类和对象的关系就如同模具制作出的物品之间的关系。一个类为它的全部对象给出了一个统一的定义，而他的每个对象则是符合这种定义的一个实体，因此类和对象的关系就是抽象和具体的关系。**

2. 如果我们定义了一个猫类，那你能想象出由“猫”类实例化的对象有哪些？

抓老鼠，吃饭，睡觉，晒太阳

**答：叮当猫，咖啡猫，Tom(Tom&Jerry)，Kitty(Hello Kitty)**

3. 类的定义有些时候或许不那么“拟物”，有时候会抽象一些，例如我们定义一个矩形类，那你会为此添加哪些属性和方法呢？

属性：长、宽

方法：面积，周长

4. 类的属性定义应该尽可能抽象还是尽可能具体？

**答：正确的做法是应该尽可能的抽象，因为这样更符合面向对象的思维。**

5. 请用一句话概括面向对象的几个特征？

|  |  |
| --- | --- |
| 封装：**对外部隐藏对象的工作细节** |  |
| 继承：**子类自动共享父类之间的数据和方法的机制** |  |
| 多态：**可以对不同类的对象调用相同的方法，产生不同的效果** |  |

6. 函数和方法有什么区别？

函数：在代码里面把要处理的数据进行函数调用，显得规范

方法：都可以来调用，方便

**答：方法和函数其实几乎完全一样，但有一点区别是方法默认有一个self参数**

****动动手：****

****0. 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“小甲鱼”）

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

yu\_c/Chapter11\_ClassandObjects/P36\_0.py

****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

yu\_c/Chapter11\_ClassandObjects/P36\_1.py

## ● P37：类和对象：面向对象编程

**测试题：**

**0. 以下代码体现了面向对象编程的什么特征？**

>>> "FishC.com".count('o')

1

>>> [1, 1, 2, 3, 5, 8].count(1)

2

>>> (0, 2, 4, 8, 12, 18).count(1)

0

**答：体现了面向对象编程的多态特征**

**1. 当程序员不想把同一段代码写几次，他们发明了函数解决了这种情况。当程序员已经有了一个类，而又想建立一个非常相近的新类，他们会怎么做呢？**

答：使用继承特征

**答：他们会定义一个新类继承已有的这个类，这样子就只需要简单添加和重写需要的方法即可。**

**例如已有龟类，那么如果要新定义一个甲鱼类，我们只需要让甲鱼类继承已有的龟类，然后重写壳的属性为“软的”即可**

**2. self参数的作用是什么？**

答：定位调用者作用

**答：绑定方法，据说有了这个参数，Python再也傻傻分不清是哪个对象在调用方法了，你可以认为方法中的self其实就是实例对象的唯一标志。**

**3. 如果我们不希望对象的属性或方法被外部直接引用，我们可以怎么做？**

答：在属性或方法前面加上双下划线”\_\_”，变成伪私有

**答：我们可以在属性或方法名字前边加上双下划线，这样子从外部是无法直接访问到，会显示AttributeError错误。**

演示代码：yu\_c/Chapter11\_ClassandObjects/P11\_2.py

**我们把getName方法称之为“访问器”。Python事实上是采用一种叫“name mangling”技术，将以双下划线开头的变量名巧妙的改个名字而已，我们仍然可以在外部通过“\_类名\_\_变量名”的方式访问**

**不提倡这种抬杠较真粗暴不文明的访问形式......**

**4. 类在实例化后哪个方法会被自动调用？**

答：\_\_init\_\_()方法

**答：\_\_init\_\_()方法会在类实例化时被自动调用，我们称之为魔法方法。你可以重写这个方法，为对象定制初始化方案。**

**5. 请解释下边代码错误的原因：**

1. class MyClass:
2. name = 'FishC'
3. def myFun(self):
4. print("Hello FishC!")
6. >>> MyClass.name
7. 'FishC'
8. >>> MyClass.myFun()
9. Traceback (most recent call last):
10. File "<pyshell#6>", line 1, in <module>
11. MyClass.myFun()
12. TypeError: myFun() missing 1 required positional argument: 'self'
13. >>>

答：没有实例化对象就直接调用了，调用了不存在的方法，触发类型错误

**动动手：**

**0. 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个小孩平日票价。**

* 平日票价100元
* 周末票价为平日的120%
* 儿童半票

**1. 游戏编程：按以下要求定义一个乌龟类和鱼类并尝试编写游戏。（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<=x<=10，0<=y<=10**
* **游戏生成1只乌龟和10条鱼**
* **它们的移动方向均随机**
* **乌龟的最大移动能力是2（Ta可以随机选择1还是2移动），鱼儿的最大移动能力是1**
* **当移动到场景边缘，自动向反方向移动**
* **乌龟初始化体力为100（上限）**
* **乌龟每移动一次，体力消耗1**
* **当乌龟和鱼坐标重叠，乌龟吃掉鱼，乌龟体力增加20**
* **鱼暂不计算体力**
* **当乌龟体力值为0（挂掉）或者鱼儿的数量为0游戏结束**