《Python游戏编程入门》

这些文章负责整理在这本书中的知识点、注意事项和课后习题的尝试实现。  
并且对每一个章节给出的最终实例进行分析和注释。  
  
初识pygame：pie游戏  
pygame游戏库使得如下功能成为可能：绘制图形、获取用户输入、执行动画  
以及使用定时器让游戏按照稳定的帧速率运行。  
使用pygame库；  
以一定字体打印文本；  
使用循环来重复动作；  
绘制圆、矩形、线条和户型；  
创建pie游戏；  
  
从哪里获得pygame库：http://www.pygame.org/download.shtml  
我现在使用的Python2.7和pygame1.9  
书中使用的环境是Python3.2和pygame1.9  
现在不在Python3的环境下安装上pip工具导致环境无法一致  
  
pygame库的初始化工作：  
    import pygame  
    from pygame.locals import \*  
    pygame.init()  
  
创建一个（600,500）大小的屏幕  
    screen=pygame.display.set\_mode((600,500))  
    screen同时被赋值为<Surface(600x500x32 SW)>  
    这是一个有用的值，所以用screen变量存储。  
  
打印文本  
    1、创建字体对象  
    myfont=pygame.font.Font(None,60)  
    None：使用默认字体  
    60：字体大小  
    2、创建一个可以使用screen.blit()绘制的平面  
    textimage=myfont.render("Hello Python",True,(255,255,255))  
    render需要三个参数，需要被显示的字符串、是否抗锯齿True/False、颜色  
    3、将textimage交给screen.blit()进行绘制  
    screen.blit(textimage,(100,100))  
    screen.blit()需要两个参数，绘制的对象及其（左上角顶点）坐标  
  
背景填充  
    screen.fill((0,0,0))  
    screen.fill()需要给出背景颜色  
  
刷新显示  
    screen.display.update()  
    一般配合while循环使用  
  
while循环  
    通过while循环可以进行事件处理和持续的屏幕刷新  
    while True:  
        for event in pygame.event.get():  
            if event.type in (QUIT,KEYDOWN):  
                sys.exit()  
        screen.display.update()  
  
绘制圆形  
    pygame.draw.circle(screen,color,position,radius,width)  
    color    (0,0,0)给定颜色  
    radius圆半径  
    position     (0,0)给定圆心坐标  
    width线条宽度  
  
绘制矩形  
    pygame.draw.rect(screen,color,position,width)  
    position      (pos\_x,pos\_y,100,100)给定左上角顶点的坐标、长和宽  
  
绘制线条  
    pygame.draw.line(screen,color,(0,0),(100,100),width)  
    (0,0)(100,100)负责给定线段的两个端点  
  
绘制弧形  
    start\_angle=math.radians(0)  
    end\_angle=math.radians(90)  
    position=x-radius,y-radius,radius\*2,radius\*2  
    #x,y表示弧形所在的圆的圆心坐标，radius表示半径  
    pygame.draw.arc(screen,color,position,start\_angle,end\_angle,width)  
    start\_angle起始角度 指向正右侧的半径开始逆时针旋转就是0到360  
    end\_angle结束角度  
  
      
  
  
两段值得学习的示例  
1、绘制移动矩形  
#!/usr/bin/python  
  
import sys  
import random  
from random import randint  
import pygame  
from pygame import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Drawing Rectangles")  
pos\_x=300  
pos\_y=250  
vel\_x=2  
vel\_y=1  
color=100,100,100              
while True:  
    for event in pygame.event.get():  
        if event.type in (QUIT,KEYDOWN):  
            sys.exit()  
  
    screen.fill((0,0,200))  
      
    pos\_x +=vel\_x  
    pos\_y +=vel\_y  
    if pos\_x>500 or pos\_x<0:  
        vel\_x=-vel\_x  
        rand1=randint(0,255)  
        rand2=randint(0,255)  
        rand3=randint(0,255)  
        color=rand1,rand2,rand3  
    if pos\_y>400 or pos\_y<0:  
        vel\_y=-vel\_y  
        rand1=randint(0,255)  
        rand2=randint(0,255)  
        rand3=randint(0,255)  
        color=rand1,rand2,rand3  
    width=0  
    pos=pos\_x,pos\_y,100,100  
    pygame.draw.rect(screen,color,pos,width)  
  
    pygame.display.update()  
  
  
2、pie游戏  
#!/usr/bin/python  
  
#init  
import sys  
import math  
import pygame  
from pygame.locals import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("The Pie Game-Press 1,2,3,4")  
myfont=pygame.font.Font(None,60)  
  
color=200,80,60  
width=4  
x=300  
y=250  
radius=200  
position=x-radius,y-radius,radius\*2,radius\*2  
  
piece1=False  
piece2=False  
piece3=False  
piece4=False  
  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
        elif event.type==KEYUP:  
            if event.key==pygame.K\_ESCAPE:  
                sys.exit()  
            elif event.key==pygame.K\_1:  
                piece1=True  
            elif event.key==pygame.K\_2:  
                piece2=True  
            elif event.key==pygame.K\_3:  
                piece3=True  
            elif event.key==pygame.K\_4:  
                piece4=True  
  
    screen.fill((0,0,200))  
  
    #draw the four numbers  
    textimage1=myfont.render("1",True,color)  
    screen.blit(textimage1,(x+radius/2-20,y-radius/2))      
    textimage2=myfont.render("2",True,color)  
    screen.blit(textimage2,(x-radius/2,y-radius/2))  
    textimage3=myfont.render("3",True,color)  
    screen.blit(textimage3,(x-radius/2,y+radius/2-20))  
    textimage4=myfont.render("4",True,color)  
    screen.blit(textimage4,(x+radius/2-20,y+radius/2-20))  
  
    #draw arc,line  
    if piece1:  
        start\_angle=math.radians(0)  
        end\_angle=math.radians(90)  
        pygame.draw.arc(screen,color,position,start\_angle,end\_angle,width)  
        pygame.draw.line(screen,color,(x,y),(x+radius,y),width)  
        pygame.draw.line(screen,color,(x,y),(x,y-radius),width)  
  
    if piece2:      
        start\_angle=math.radians(90)  
        end\_angle=math.radians(180)  
        pygame.draw.arc(screen,color,position,start\_angle,end\_angle,width)  
        pygame.draw.line(screen,color,(x,y),(x,y-radius),width)  
        pygame.draw.line(screen,color,(x,y),(x-radius,y),width)  
  
    if piece3:  
        start\_angle=math.radians(180)  
        end\_angle=math.radians(270)  
        pygame.draw.arc(screen,color,position,start\_angle,end\_angle,width)  
        pygame.draw.line(screen,color,(x,y),(x-radius,y),width)  
        pygame.draw.line(screen,color,(x,y),(x,y+radius),width)  
  
    if piece4:  
        start\_angle=math.radians(270)  
        end\_angle=math.radians(360)  
        pygame.draw.arc(screen,color,position,start\_angle,end\_angle,width)  
        pygame.draw.line(screen,color,(x,y),(x,y+radius),width)  
        pygame.draw.line(screen,color,(x,y),(x+radius,y),width)  
          
    #if success,display green  
    if piece1 and piece2 and piece3 and piece4:  
        color=0,255,0  
  
    pygame.display.update()  
  
  
挑战  
1、绘制椭圆  
#!/usr/bin/python  
  
import sys  
import pygame  
from pygame.locals import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Drawing Ellipse")  
  
while True:  
        for event in pygame.event.get():  
                if event.type in (QUIT,KEYDOWN):  
                        sys.exit()  
  
        screen.fill((0,255,0))  
  
        color=255,0,255  
        position=100,100,400,300  
        width=8  
        pygame.draw.ellipse(screen,color,position,width)  
  
        pygame.display.update()  
这个题目就是让你认识一下pygame.draw.ellipse()函数的相关使用。  
该函数和pygame.draw.rect()函数使用方式十分相似。  
  
2、随机的绘制1000个线条  
#!/usr/bin/python  
import random  
from random import randint  
import sys  
import pygame  
from pygame import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((800,600))  
pygame.display.set\_caption("Drawing Line")  
  
screen.fill((0,80,0))  
color=100,255,200  
width=2  
for i in range(1,1001):  
        pygame.draw.line(screen,color,(randint(0,800),randint(0,600)),(randint(0,800),randint(0,600)),width)  
  
while True:  
        for event in pygame.event.get():  
                if event.type in (QUIT,KEYDOWN):  
                        sys.exit()  
  
  
  
        pygame.display.update()  
  
通过这个题目理解了如果绘制图形和刷新显示都在循环中时，while True循环每次都会绘  
制图形并刷新显示。  
调用pygame模块中的randint()函数。  
而在while True循环外绘制图形，则图形绘制完成之后保持不变。刷新显示的是一个已经绘制好  
的图形。  
  
3、修改矩形程序，使矩形碰到屏幕边界是，矩形会改变颜色  
  
#!/usr/bin/python  
  
import sys  
import random  
from random import randint  
import pygame  
from pygame import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Drawing Rectangles")  
pos\_x=300  
pos\_y=250  
vel\_x=2  
vel\_y=1  
rand1=randint(0,255)  
rand2=randint(0,255)  
rand3=randint(0,255)  
color=rand1,rand2,rand3              
while True:  
    for event in pygame.event.get():  
        if event.type in (QUIT,KEYDOWN):  
            sys.exit()  
  
    screen.fill((0,0,200))  
      
    pos\_x +=vel\_x  
    pos\_y +=vel\_y  
    if pos\_x>500 or pos\_x<0:  
        vel\_x=-vel\_x  
        rand1=randint(0,255)  
        rand2=randint(0,255)  
        rand3=randint(0,255)  
        color=rand1,rand2,rand3  
    if pos\_y>400 or pos\_y<0:  
        vel\_y=-vel\_y  
        rand1=randint(0,255)  
        rand2=randint(0,255)  
        rand3=randint(0,255)  
        color=rand1,rand2,rand3  
    width=0  
    pos=pos\_x,pos\_y,100,100  
    pygame.draw.rect(screen,color,pos,width)  
  
    pygame.display.update()  
  
这里需要用到random模块。在每次碰到屏幕边界时，不仅改变矩形的运动方向，而且使用随机数改变矩形的颜色。  
也可以先将color设置为定值，可以少写三行代码。  
  
  
底层（嵌套的层数较多）代码块初次使用的变量在顶层代码块中依然生效。  
以上问题属于变量的作用域问题。说明我在这一方面认识不够清晰。

I/O、数据和字体：Trivia游戏  
  
本章包括如下内容：  
Python数据类型  
获取用户输入  
处理异常  
Mad Lib游戏  
操作文本文件  
操作二进制文件  
Trivia游戏  
  
其他的不说，我先去自己学习文件类型和字符串类型去了。这部分讲道理换行符还  
是有点丑陋的，Python还需要想着各种地方加换行符，狗屎。  
  
不去想书中示例丑陋的换行符，只要不影响其中几个重要的游戏效果就行。  
  
  
  
  
  
实验空行会不会对于trivia游戏的结果产生影响  
  
Python文件类型  
f=open("filename","r")  
  
读写模式r w a r+ w+ a+  
  
r只读模式  
w只写模式，已有文件内容会被清空，文件不存在则会创建新文件  
a添加模式，不清空文件，在文件末尾加入内容  
其他的说不了，很复杂。  
  
怎么对文件指针位置进行调整  
  
string.strip()函数删除末尾的换行符  
  
madlib游戏简介：  
    Mad Lib游戏相当简单。它要求某个人填入一些名称、事情、地点，然后使用这些单词和短语来组成一个  
故事，往往会得到出人意料的、幽默的结果。这个小程序的有趣之处在于，故事是如何构建出来的（确实如此）。  
  
这个程序是经过修改的，和书中story部分不同，没有许多令人心烦的换行符。而是直接用三个引号来完成多  
行的输入。  
  
论三个引号会怎么影响sublime的Python输出格式（所有的代码一篇黄）。  
  
  
    #!/usr/bin/python  
    print("MAD LIB GAME")  
    print("Enter answers to the following prompts")  
    print  
    guy=raw\_input("Name of a famous man:")  
    girl=raw\_input("Name of a famous woman:")  
    food=raw\_input("Your favorite food:")  
    ship=raw\_input("Name of a space ship:")  
    job=raw\_input("Name of a profession:")  
    planet=raw\_input("Name of a planet:")  
    drink=raw\_input("Your favorite drink:")  
    number=raw\_input("A number from 1 to 10:")  
  
    story="""      
    A famous married couple,GUY and GIRL,went on  
    vacation to the planet PLANET.It took NUMBER  
    weeks to get there travelling by SHIP.They  
    enjoyed a luxurious candlelight dinner overlooking  
    a DRINK ocean while eating FOOD.But,since  
    they were both JOB,they had to cut their  
    vacation short."""  
  
    story=story.replace("GUY",guy)  
    story=story.replace("GIRL",girl)  
    story=story.replace("FOOD",food)  
    story=story.replace("SHIP",ship)  
    story=story.replace("JOB",job)  
    story=story.replace("PLANET",planet)  
    story=story.replace("DRINK",drink)  
    story=story.replace("NUMBER",number)  
    print(story)  
  
这个程序值得学习的部分：  
    对于字符串类型，可以使用str.replace将给定字符替换成相应的变量。  
    嗯，小故事也不错。  
  
  
  
trivia游戏简介：  
    Trivia游戏，从一个文件中读取出一些问题，并且要求用户从多个可选的答案中做出选择。  
这个游戏涉及数据类型的设计，对于现在我面向对象的编程水平来说有点困难。需要一步一步将  
整个程序分块分析，看看面向对象的编程设计思路。也可以帮助理解为什么面向对象是一种更高  
级的编程方式。  
  
  
  
    #!/usr/bin/python  
    import sys,pygame  
    from pygame.locals import \*  
    pygame.init()  
  
    #定义数据类型  
    #Trivia包括\_\_init\_\_,show\_question,handle\_input,next\_question等。一个初始化和三个自带函数。  
    #\_\_init\_\_定义了Trivia所应包含的常量。data,current,total,correct,score,scored,failed,  
    #wronganswer和colors。其中data是从文件中读取的字符串列表。  
    #show\_question负责显示当前问题，它会调用到全局的函数print\_text  
    #handle\_input负责判断对错以及对score,scored,failed,wronganswer的进行相应的修改  
    #next\_question负责重置scored,failed,current,correct  
    class Trivia():  
        def \_\_init\_\_(self,filename):  
            self.data=[]  
            self.current=0  
            self.total=0  
            self.correct=0  
            self.score=0  
            self.scored=False  
            self.failed=False  
            self.wronganswer=0  
            self.colors=[white,white,white,white]  
  
            #read trivia data from file  
            f=open("trivia\_data.txt","r")  
            trivia\_data=f.readlines()  
            f.close()  
  
            #count and clean up trivia data  
            for text\_line in trivia\_data:  
                self.data.append(text\_line.strip())  
                self.total+=1  
  
        def show\_question(self):  
            print\_text(font1,210,5,"TRIVIA GAME")  
            print\_text(font2,190,500-20,"Press Keys (1-4) To Answer",purple)  
            print\_text(font2,530,5,"SCORE",purple)  
            print\_text(font2,550,25,str(self.score),purple)  
              
            #get correct answer out of data(first)  
            if (self.current+5)<self.total:  
                self.correct=int(self.data[self.current+5])  
            else:  
                sys.exit()  
            #display question  
            question=self.current  
            print\_text(font1,5,80,"QUESTION "+str(question))  
            print\_text(font2,20,120,self.data[self.current],yellow)  
  
            #respond to correct answer  
            if self.scored:  
                self.colors=[white,white,white,white]  
                self.colors[self.correct-1]=green  
                print\_text(font1,230,380,"CORRECT!",green)  
                print\_text(font2,170,420,"Press Enter For Next Quetion",green)  
            elif self.failed:  
                self.colors=[white,white,white,white]  
                self.colors[self.wronganswer-1]=red      
                self.colors[self.correct-1]=green  
                print\_text(font1,230,380,"INCORRECT!",red)  
                print\_text(font2,170,420,"Press Enter For Next Quetion",red)  
  
            #display answers  
            print\_text(font1,5,170,"ANSWERS")  
            print\_text(font2,20,210,"1-"+self.data[self.current+1],self.colors[0])  
            print\_text(font2,20,240,"2-"+self.data[self.current+2],self.colors[1])  
            print\_text(font2,20,270,"3-"+self.data[self.current+3],self.colors[2])  
            print\_text(font2,20,300,"4-"+self.data[self.current+4],self.colors[3])  
          
        def handle\_input(self,number):  
            if not self.scored and not self.failed:  
                if number==self.correct:  
                    self.scored=True  
                    self.score+=1  
                else:  
                    self.failed=True  
                    self.wronganswer=number  
  
        def next\_question(self):  
            if self.scored or self.failed:  
                self.scored=False  
                self.failed=False  
                self.correct=0  
                self.colors=[white,white,white,white]  
                self.current+=6  
                if self.current>=self.total:  
                    self.current=0  
  
    def print\_text(font,x,y,text,color=(255,255,255),shadow=True):  
        if shadow:  
            imgtext=font.render(text,True,(0,0,0))  
            screen.blit(imgtext,(x-2,y-2))  
        imgtext=font.render(text,True,color)  
        screen.blit(imgtext,(x,y))  
  
    #main program begins  
    pygame.init()  
    screen=pygame.display.set\_mode((600,500))  
    pygame.display.set\_caption("The Trivia Game")  
    font1=pygame.font.Font(None,40)  
    font2=pygame.font.Font(None,24)  
    white=255,255,255  
    cyan=0,255,255  
    yellow=255,255,0  
    purple=255,0,255  
    green=0,255,0  
    red=255,0,0  
  
    #load the trivia data file  
    trivia=Trivia("trivia\_data.txt")  
  
    #repeating loop  
    while True:  
        for event in pygame.event.get():  
            if event.type==QUIT:  
                sys.exit()  
            elif event.type==KEYUP:  
                if event.key==pygame.K\_ESCAPE:  
                    sys.exit()  
                elif event.key==pygame.K\_1:  
                    trivia.handle\_input(1)  
                elif event.key==pygame.K\_2:  
                    trivia.handle\_input(2)  
                elif event.key==pygame.K\_3:  
                    trivia.handle\_input(3)  
                elif event.key==pygame.K\_4:  
                    trivia.handle\_input(4)  
                elif event.key==pygame.K\_RETURN:  
                    trivia.next\_question()  
  
        #clear the screen  
        screen.fill((0,0,200))  
  
        #display trivia data  
        trivia.show\_question()  
          
        #update the display  
        pygame.display.update()  
  
    这段代码感觉就是面向对象语言的入门范例。首先设计好数据结构，主函数相对来说简单明了。这个跨度我感觉有一点大，  
还需要一定程度的积累才行。

用户输入：Bomb Catcher游戏  
本章介绍使用键盘和鼠标获得用户输入。包括如下主题：  
学习pygame事件  
学习实时循环  
学习键盘和鼠标事件  
学习轮询键盘和鼠标的状态  
编写Bomb Catcher游戏  
  
1本章所涉及pygame事件  
QUIT  
KEYDOWN  
KEYUP  
MOUSEMOTION  
MOUSEBUTTONUP  
MOUSEBUTTONDOWN  
  
    1.1实时事件循环  
    while True:  
        for event in pygame.event.get():  
            if event.type==QUIT:  
                sys.exit()  
    在循环中不断地对事件队列中的特定事件进行相应的处理。  
  
    1.2键盘事件  
    KEYUP  KEYDOWN  
      
    1.3鼠标事件  
    MOUSEBUTTONDOWN  MOUSEBUTTONUP  MOUSEMOTION  
    具体属性见《pygame几个重要模块》  
  
2设备轮询  
pygame中的事件系统并非我们可以用来检测用户输入的唯一方法。我们可以轮询输入设备，看看用户是否与我们的  
程序交互。  
  
    2.1轮询键盘  
    在pygame中，使用pygame.key.get\_pressed()来轮询键盘接口。该方法返回布尔值的一个列表，对应于键盘  
    上的按键，每个键一个标志。使用相同的键常量值来索引布尔值数组。一次轮询所有的键的好处是不必遍历事件  
    系统就可以检测多个键的按下。  
    以下示例检测escape键。  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
  
    2.2轮询鼠标  
    pygame.mouse.get\_pos()  
    pygame.mouse.get\_rel()  
    pygame.mouse.get\_pressed()  
    具体使用参考《pygame几个重要模块》  
import sys,pygame  
from pygame.locals import \*  
pygame.init()  
  
def print\_text(font,x,y,text,color=(255,255,255)):  
    imgtext=font.render(text,True,color)  
    screen.blit(imgtext,(x,y))  
  
#main program begins  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Mouse Demo")  
font1=pygame.font.Font(None,24)  
white=255,255,255  
  
mouse\_x=mouse\_y=0  
move\_x=move\_y=0  
mouse\_down=mouse\_up=0  
mouse\_down\_x=mouse\_down\_y=0  
mouse\_up\_x=mouse\_up\_y=0  
  
#repeating loop  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
        elif event.type==MOUSEMOTION:  
            mouse\_x,mouse\_y=event.pos  
            move\_x,move\_y=event.rel  
        elif event.type==MOUSEBUTTONDOWN:  
            mouse\_down=event.button  
            mouse\_down\_x,mouse\_down\_y=event.pos  
        elif event.type==MOUSEBUTTONUP:  
            mouse\_up=event.button  
            mouse\_up\_x,mouse\_up\_y=event.pos  
  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
      
    screen.fill((0,100,0))  
    print\_text(font1,0,0,"Mouse Events")  
    print\_text(font1,0,20,"Mouse position: "+str(mouse\_x)+","+str(mouse\_y))  
    print\_text(font1,0,40,"Mouse relative: "+str(move\_x)+","+str(move\_y))  
    print\_text(font1,0,60,"Mouse button down: "+str(mouse\_down)+" at "+str(mouse\_down\_x)+","+str(mouse\_down\_y))  
    print\_text(font1,0,80,"Mouse button up: "+str(mouse\_up)+" at "+str(mouse\_up\_x)+","+str(mouse\_up\_y))  
  
    print\_text(font1,0,160,"Mouse Polling")  
    x,y=pygame.mouse.get\_pos()  
        print\_text(font1,0,180,"Mouse Position: "+str(x)+str(y))  
        b1,b2,b3=pygame.mouse.get\_pressed()  
        print\_text(font1,0,200,"Mouse buttons: "+str(b1)+","+str(b2)+","+str(b3))  
  
    pygame.display.update()  
  
  
  
3游戏简介  
bomb Catcher游戏综合了鼠标输入、一些基本图形绘制和少量冲突检测逻辑。  
炸弹是不断重复地从屏幕顶端落下的黄色圆圈。  
当炸弹到达屏幕底部的时候，玩家未接住炸弹就会丢掉一条生命。  
当炸弹撞击到挡板，算作玩家接住炸弹，另一个炸弹还会继续落下。  
  
import sys,random,time,pygame  
from pygame.locals import \*  
pygame.init()  
  
def print\_text(font,x,y,text,color=(255,255,255)):  
    imgtext=font.render(text,True,color)  
    screen.blit(imgtext,(x,y))  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Bomb Catching Game")  
font1=pygame.font.Font(None,24)  
pygame.mouse.set\_visible(False)  
white=255,255,255  
red=220,50,50  
yellow=230,230,50  
black=0,0,0  
  
lives=3  
score=0  
game\_over=True  
mouse\_x=mouse\_y=0  
pos\_x=300  
pos\_y=460  
bomb\_x=random.randint(0,500)  
bomb\_y=-50  
vel\_y=3  
  
#repeating loop  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
        elif event.type==MOUSEMOTION:  
            mouse\_x,mouse\_y=event.pos  
            move\_x,move\_y=event.rel  
        elif event.type==MOUSEBUTTONUP:  
            if game\_over:  
                game\_over=False  
                lives=3  
                score=0  
  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
    screen.fill((0,0,100))  
      
    if game\_over:  
        print\_text(font1,100,200,"<CLICK TO PLAY>")  
    else:  
        #move the bomb  
        bomb\_y+=vel\_y  
      
    #has player missed the bomb?  
    if bomb\_y>500:  
        bomb\_x=random.randint(0,500)  
        bomb\_y=-50  
        lives-=1  
        if lives==0:  
            game\_over=True  
      
    #see if player has caught the bomb  
    elif bomb\_y>pos\_y:  
        if bomb\_x>pos\_x and bomb\_x<pos\_x+120:  
            score+=10  
            bomb\_x=random.randint(0,500)  
            bomb\_y=-50  
      
    #draw the bomb  
    pygame.draw.circle(screen,black,(bomb\_x-4,int(bomb\_y)-4),30,0)  
    pygame.draw.circle(screen,yellow,(bomb\_x,int(bomb\_y)),30,0)  
      
    #set basket position  
    pos\_x=mouse\_x  
    if pos\_x<0:  
        pos\_x=0  
    elif pos\_x>480:  
        pos\_x=500  
    #draw basket  
    pygame.draw.rect(screen,black,(pos\_x-4,pos\_y-4,120,40),0)  
    pygame.draw.rect(screen,red,(pos\_x,pos\_y,120,40),0)  
  
    #print # of lives  
    print\_text(font1,0,0,"Lives: "+str(lives))  
      
    #print score  
    print\_text(font1,500,0,"Score: "+str(score))  
  
    pygame.display.update()  
  
可以改进的地方很多，比如说难度的提升，炸弹的显示效果等等。  
现在我会改进这个程序，但是并不能独立写出这个程序来。

Math和Graphics:Analog Clock示例程序  
本章介绍Python的math模块，该模块可以执行计算，如常见的三角正弦函数、余弦函数、正切函数等。  
  
使用正弦和余弦函数绘制圆  
创建Anlog Clock示例程序  
  
关于math模块  
https://docs.python.org/3/library/math.html  
https://docs.python.org/2.7/library/math.html  
  
  
math.cos(x)  
    Return the cosine of x radians.  
  
math.sin(x)  
    Return the sine of x radians.  
  
math.tan(x)  
    Return the tangent of x radians.  
  
math.degrees(x)  
    Convert angle x from radians to degrees.  
  
math.radians(x)  
    Convert angle x from degrees to radians.  
  
math.pi  
    The mathematical constant π = 3.141592..., to available precision.  
  
math.e  
    The mathematical constant e = 2.718281..., to available precision.  
  
注意：  
    对于负数的取模  
    余数应该是大于等于0小于该数绝对值的那个数,比方说（-5）%3=1,认为-5=3\*(-2)+1  
  
书中给出的示例可以优化。  
  
  
Draw circle in hard way  
  
import sys,pygame,math,random  
from pygame.locals import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Circle Demo")  
angle=0  
color=random.randint(0,255),random.randint(0,255),random.randint(0,255)  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
      
    screen.fill((255,255,255))      
  
      
    x=300  
    y=250  
    radius=200  
      
      
    circle\_x=x+radius\*math.cos(math.radians(angle))  
    circle\_y=y+radius\*math.sin(math.radians(angle))  
  
    pygame.draw.circle(screen,color,(int(circle\_x),int(circle\_y)),10,0)  
    angle+=1  
    if angle>359:  
        color=random.randint(0,255),random.randint(0,255),random.randint(0,255)  
        angle=0      
      
    pygame.display.update()      
  
  
Anlog Clock示例程序  
  
import sys,datetime,pygame,math  
from pygame.locals import \*  
from datetime import datetime,date,time  
pygame.init()  
def print\_text(font,x,y,text,color=(255,255,255)):  
    imgtext=font.render(text,True,color)  
    screen.blit(imgtext,(x,y))  
  
def wrap\_angle(angle):  
    return angle%360  
  
#main program begins  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Anlog Clock Demo")  
font=pygame.font.Font(None,36)  
orange=220,180,0  
white=255,255,255  
yellow=255,255,0  
pink=255,100,100  
  
pos\_x=300  
pos\_y=250  
radius=250  
angle=360  
  
#repeating loop  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
      
    screen.fill((0,0,100))  
    #draw one step around the circle  
    pygame.draw.circle(screen,white,(pos\_x,pos\_y),radius,6)  
      
    #draw the clock numbers1-12  
    for n in range(1,13):  
        angle=math.radians(n\*30-90)  
        x=(radius-20)\*math.cos(angle)+pos\_x-10  
        y=(radius-20)\*math.sin(angle)+pos\_y-10  
        print\_text(font,int(x),int(y),str(n))  
      
    #get the time of day  
    today=datetime.today()  
    hours=today.hour%12  
    minutes=today.minute  
    seconds=today.second  
      
    #draw the hours hand  
    hour\_angle=wrap\_angle(hours\*30-90)  
    hour\_angle=math.radians(hour\_angle)  
    hour\_x=pos\_x+math.cos(hour\_angle)\*(radius-80)  
    hour\_y=pos\_y+math.sin(hour\_angle)\*(radius-80)  
    target\_h=(int(hour\_x),int(hour\_y))  
    pygame.draw.line(screen,pink,(pos\_x,pos\_y),target\_h,25)  
      
    #draw the minutes hand  
    minute\_angle=wrap\_angle(minutes\*6-90)  
    minute\_angle=math.radians(minute\_angle)  
    minute\_x=pos\_x+math.cos(minute\_angle)\*(radius-60)  
    minute\_y=pos\_y+math.sin(minute\_angle)\*(radius-60)  
    target\_m=(int(minute\_x),int(minute\_y))  
    pygame.draw.line(screen,orange,(pos\_x,pos\_y),target\_m,12)  
      
    #draw the seconds hand  
    second\_angle=wrap\_angle(seconds\*6-90)  
    second\_angle=math.radians(second\_angle)  
    second\_x=pos\_x+math.cos(second\_angle)\*(radius-40)  
    second\_y=pos\_y+math.sin(second\_angle)\*(radius-40)  
    target\_s=(int(second\_x),int(second\_y))  
    pygame.draw.line(screen,yellow,(pos\_x,pos\_y),target\_s,6)  
      
    #cover the center  
    pygame.draw.circle(screen,white,(pos\_x,pos\_y),20)  
  
    print\_text(font,0,0,str(hours)+":"+str(minutes)+":"+str(seconds))  
  
    pygame.display.update()  
  
  
wrap\_angle函数是多余的，因为在三角函数的运算中+-360°对于结果没有影响。  
去除所有带有 wrap\_angle()的部分，程序运行正常。  
  
1、修改circle程序，以使得在每个角度绘制不同的形状，而不是绘制一个小的填充的圆。  
import sys,pygame,math,random  
from pygame.locals import \*  
pygame.init()  
  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Circle Demo")  
angle=0  
color=random.randint(0,255),random.randint(0,255),random.randint(0,255)  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
      
    screen.fill((0,0,0))      
  
      
    x=300  
    y=250  
    radius=200  
      
      
    circle\_x=int(x+radius\*math.cos(math.radians(angle)))  
    circle\_y=int(y+radius\*math.sin(math.radians(angle)))  
      
    if angle%3==0:  
        pygame.draw.circle(screen,color,(circle\_x,circle\_y),12)  
    elif angle%3==1:  
        pygame.draw.rect(screen,color,(circle\_x-10,circle\_y-10,20,20))  
    elif angle%3==2:  
        pygame.draw.ellipse(screen,color,(circle\_x-15,circle\_y-10,30,20))  
  
    angle+=1  
    if angle>359:  
        color=random.randint(0,255),random.randint(0,255),random.randint(0,255)  
        angle=0      
      
    pygame.display.update()      
  
感觉画了一个会蠕动的小东西，有一点意思。  
  
2、Analog Circle变得更好看些。这个我没去实现，我把时针的角度做了微调，感觉更符合习惯。现在  
时针不会一直指向整点的位置了，而是会根据分钟数指向两个整点之间。  
import sys,datetime,pygame,math  
from pygame.locals import \*  
from datetime import datetime,date,time  
pygame.init()  
def print\_text(font,x,y,text,color=(255,255,255)):  
    imgtext=font.render(text,True,color)  
    screen.blit(imgtext,(x,y))  
  
def wrap\_angle(angle):  
    return angle%360  
  
#main program begins  
screen=pygame.display.set\_mode((600,500))  
pygame.display.set\_caption("Anlog Clock Demo")  
font=pygame.font.Font(None,36)  
orange=220,180,0  
white=255,255,255  
yellow=255,255,0  
pink=255,100,100  
  
pos\_x=300  
pos\_y=250  
radius=250  
angle=360  
  
#repeating loop  
while True:  
    for event in pygame.event.get():  
        if event.type==QUIT:  
            sys.exit()  
    keys=pygame.key.get\_pressed()  
    if keys[K\_ESCAPE]:  
        sys.exit()  
      
    screen.fill((0,0,100))  
    #draw one step around the circle  
    pygame.draw.circle(screen,white,(pos\_x,pos\_y),radius,6)  
      
    #draw the clock numbers1-12  
    for n in range(1,13):  
        angle=math.radians(n\*30-90)  
        x=(radius-20)\*math.cos(angle)+pos\_x-10  
        y=(radius-20)\*math.sin(angle)+pos\_y-10  
        print\_text(font,int(x),int(y),str(n))  
      
    #get the time of day  
    today=datetime.today()  
    hours=today.hour%12  
    minutes=today.minute  
    seconds=today.second  
      
    #draw the hours hand  
    hour\_angle=wrap\_angle(hours\*30-90+minutes\*0.5)  #对比原来的程序修改了这一行。  
    hour\_angle=math.radians(hour\_angle)  
    hour\_x=pos\_x+math.cos(hour\_angle)\*(radius-80)  
    hour\_y=pos\_y+math.sin(hour\_angle)\*(radius-80)  
    target\_h=(int(hour\_x),int(hour\_y))  
    pygame.draw.line(screen,pink,(pos\_x,pos\_y),target\_h,25)  
      
    #draw the minutes hand  
    minute\_angle=wrap\_angle(minutes\*6-90)  
    minute\_angle=math.radians(minute\_angle)  
    minute\_x=pos\_x+math.cos(minute\_angle)\*(radius-60)  
    minute\_y=pos\_y+math.sin(minute\_angle)\*(radius-60)  
    target\_m=(int(minute\_x),int(minute\_y))  
    pygame.draw.line(screen,orange,(pos\_x,pos\_y),target\_m,12)  
      
    #draw the seconds hand  
    second\_angle=wrap\_angle(seconds\*6-90)  
    second\_angle=math.radians(second\_angle)  
    second\_x=pos\_x+math.cos(second\_angle)\*(radius-40)  
    second\_y=pos\_y+math.sin(second\_angle)\*(radius-40)  
    target\_s=(int(second\_x),int(second\_y))  
    pygame.draw.line(screen,yellow,(pos\_x,pos\_y),target\_s,6)  
      
    #cover the center  
    pygame.draw.circle(screen,white,(pos\_x,pos\_y),20)  
  
    print\_text(font,0,0,str(hours)+":"+str(minutes)+":"+str(seconds))  
  
    pygame.display.update()  
  
看起来更符合习惯了。至于美观的要求就算了。

1.寻找最小的k个数

有n个整数，请找出其中最小的k个数，要求时间复杂度尽可能低。

全部排序。使用快速排序算法排序之后并且输出最小的k个数，时间复杂度为O（nlogn）

[IMG_256](javascript:void(0);)

def quick\_sort(array, left, right):

if left >= right:

return

low = left

high = right

key = array[low]

while left < right:

while left < right and array[right] > key:

right -= 1

array[left] = array[right]

while left < right and array[left] <= key:

left += 1

array[right] = array[left]

array[left] = key

quick\_sort(array, low, left-1)

quick\_sort(array, left+1, high)

[IMG_257](javascript:void(0);)

以上为一般的快速排序算法。由于是对数组直接进行操作，不需要返回值。

线性选择算法

在选择第k个最小元素时其平均时间复杂度为O（n）。

算法思想很简单，类似于快速排序。

我写的快速排序暂时有bug，以后再修改上传。

2.寻找和为定值的两个数

输入一个整数数组和一个整数，在数组中查找一对数，满足它们的和正好是输入的那个整数。

如果有多对数的和等于输入的整数，输出任意一对即可。

穷举法

def two\_sum(array, n):

len\_a = len(array)

for i in range(len\_a):

for j in range(len\_a):

if i != j and array[i] + array[j] == n:

return array[i], array[j]

排序夹逼

先利用快速排序将数组变为有序数组，然后利用二分查找。可以将时间复杂度降为O（nlogn）。

或者是利用快速排序将数组变为有序数组后，使用两个指针从两段往中间扫描,这样查找的时间复杂度为O（n）。

[IMG_258](javascript:void(0);)

def binary\_search(list\_a, n):

low = 0

high = len(list\_a)-1

while low <= high:

mid = (low+high)/2

if list\_a[mid] == n:

return True

elif list\_a[mid] < n:

low = mid+1

else:

high = mid-1

return False

def two\_sum(array, n):

array.sort()

for i in array:

if binary\_search(array, n-i):  
　　　　　　　if array.index(i)!=array.index(n-i):

　　return i, n-i

[IMG_259](javascript:void(0);)

如果n是其中一个数的两倍，则会出现bug。即数组中的一个数的出现了两次，2i == n。因此需要附加额外的判断条件。

[IMG_260](javascript:void(0);)

def two\_sum(array, n):

begin = 0

end = len(array)-1

while begin < end:

if array[begin]+array[end] == n:

return array[begin], array[end]

elif array[begin]+array[end] < n:

begin += 1

else:

end -= 1

[IMG_261](javascript:void(0);)

与上面的相比，这个算法需要数组本身为有序。

不需要额外的判断条件防止i和n-i为同一个数的这种特殊情况。

3.寻找和为定值的多个数

输入两个整数n和sum，要求从数列1,2,3，...，n中取任意个数，使其和等于sum，请将其中所有可能的组合列出来。

n问题转化为n-1问题

思路是如果取第n个数，问题转化为n-1和sum-n的问题

如果不取第n个数，问题转化为n-1和sum的问题

实现部分还有一点看不懂，所以暂时跳过。以后有机会补上。

4.最大连续子数组和

给定一个整数数组，数组里可能有整数，负数和零。数组中连续的一个或多个整数组成一个子数组，每个子数组都有一个和。

求所有子数组的和的最大值。

蛮力枚举

[IMG_262](javascript:void(0);)

def max\_substring(array, length):

max\_sum = array[0]

cur\_sum = 0

for i in range(length):

for j in range(i, length):

for k in range(i, j+1):

cur\_sum += array[k]

if cur\_sum > max\_sum:

max\_sum = cur\_sum

cur\_sum = 0

return max\_sum

[IMG_263](javascript:void(0);)

动态规划

[IMG_264](javascript:void(0);)

def max\_substring(array, length):

max\_sum = array[0]

cur\_sum = 0

for i in range(length):

if cur\_sum >= 0:

cur\_sum += array[i]

else:

cur\_sum = array[i]

if cur\_sum > max\_sum:

max\_sum = cur\_sum

return max\_sum

[IMG_265](javascript:void(0);)

此方法从前往后扫描一遍数组，即完成求最大连续子数组和的需求，所以时间复杂度为O（n）。

5.跳台阶问题

一个台阶总共有n级，如果一次可以跳一级，也可以跳两级，求总共有多少种跳法。

递归

[IMG_266](javascript:void(0);)

def fib(n):

if n == 1:

return 1

if n == 2:

return 2

if n > 2:

return fib(n-1)+fib(n-2)

[IMG_267](javascript:void(0);)

递推

def fib(n):

a = 0

b = 1

for i in range(n):

a, b = b, a+b

return b

6.奇偶数排序

给定一个整数数组，请调整数组中数的顺序，使得所有奇数位于数组的前半部分，所有偶数位于数组的后半部分。

要求时间复杂度为O（n）。

若把奇数看成小的数，偶数看成大的数，那么按照题目要求的奇数放在前面偶数放在后面，

就相当于小数放在前面大数放在后面，联想一下快速排序的划分过程，就是通过一个主元把整个数组分成大小两部分。

划分过程有以下两种实现。

一头一尾两个指针同时往中间扫描

[IMG_268](javascript:void(0);)

def is\_odd(n):

return (n & 1) == 1

def odd\_even\_sort(array, length):

if array == [] or length <= 0:

return

low = 0

high = length-1

while low < high:

if is\_odd(array[low]):

low += 1

elif not is\_odd(array[high]):

high -= 1

else:

array[low], array[high] = array[high], array[low]

[IMG_269](javascript:void(0);)

一前一后两个指针同时从左往右扫  如果前指针遇到的数比主元小，则后指针右移一位，然后交换各自指向的数。

[IMG_270](javascript:void(0);)

def odd\_even\_sort(array, length):  
 if array == [] or length <= 0:  
 return

i = -1

for j in range(length):

if is\_odd(array[j]):

i += 1

array[i], array[j] = array[j], array[i]

1.Python下多线程的限制以及多进程中传递参数的方式

Python多线程有个全局解释器锁，这个锁的意思是任一时间只能有一个线程运用解释器。并发不是并行。

多进程间同享数据，能够运用multiprocession.Value和multiprocessing.Array

认真学习下multiprocessing模块

2.Python是怎样进行内存管理的

内存池的概念

3.什么是lambda函数？他有什么好处？

lambda函数是一个能接纳任意多个参数并且返回单个表达式值的函数。

lambda函数不能包括指令，他们所包括的表达式不能超过一个。

不要试图想lambda函数中塞入太多东西，如果需要更复杂的东西，就定义一个一般函数。

4.怎么用Python输出一个Fibonacci数列？

基础。

5.介绍下Python中webbrowser的用法？

熟悉webbrowser模块。

6.解释下Python的and-or语法

这里需要大致知道and和or用于列表和数字之间的区别。

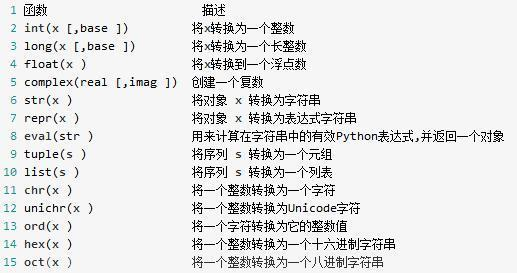
7.如何倒序迭代一个数组

# 通用方法for i in range(len(l)-1, -1,-1):

print(l[i])# 内置函数reversedfor i in reversed(l):

print(i)

8.Python是怎样进行类型变换的



9.Python里面如何实现tuple和list的转换

使用tuple,list方法进行转换即可。

10.请写出一段Python代码实现删除一个list里面的重复元素

[IMG_257](javascript:void(0);)

l=[1,1,33,33,5,6]# 方法1

mylist = list(set(l))# 方法2

d = {}for i in l:

d[i]=1

mylist = list(d.keys())print(mylist)

[IMG_258](javascript:void(0);)

面试技巧

不要给自己挖坑

必被问到redis，高并发解决办法

# **[list,tuple,dict,set的增删改查](https://www.cnblogs.com/tsxh/p/9641218.html)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据结构 | list | tuple | dict | set |
| 增 | append  insert |  | d['key']=value | add |
| 删 | pop  pop(0) |  | d.pop('name') | pop  remove  clear  清空整个集合  del  删除整个集合 |
| 改 | a[2]= |  | d['key']=value2 | update |
| 查 | a.count(1)  查询值出现的次数  a.index('bb')  查询元素位置 | a.count(1)  查询值出现的次数  a.index(1)  查询值出现的位置 | a.get('name')  查询键对应的值  a.has\_key('name')  查询是否存在该键 |  |

附：

可变对象和不可变对象

* 不可变对象，该对象所指向的内存中的值不能被改变。
* 可变对象，该对象所指向的内存中的值可以被改变。

Python中，数值类型（int和float）、字符串str、元组tuple都是不可变类型。

列表，字典，集合是可变类型。

# **[用蓝图实现模块化应用](https://www.cnblogs.com/tsxh/p/9649107.html)**

官方文档笔记

http://www.pythondoc.com/flask/blueprints.html

一个blueprint对象和flask应用对象的工作方式很像，但他确实不是一个应用。

而是一个描述如何构建或扩展应用的蓝图。

1.为什么用蓝图？

flask中蓝图旨在针对以下情况：

* 把一个应用分解成一系列蓝图。
* 以一个URL前缀在一个应用上注册蓝图。
* 在一个应用中用不同的URL规则多次注册一个蓝图。（不理解）
* 通过蓝图提供模板过滤器、静态文件、模板和其他功能。一个蓝图不一定要实现应用或视图函数。

flask蓝图不是即插应用，因为他实际上并不是一个应用。

蓝图作为flask层提供分割的替代，共享应用配置，（如果使用多个应用对象，则应用会有分开的配置）并且可以更改所注册的应用对象。

其短板是你不能再应用创建后撤销一个蓝图而不销毁整个应用对象。

2.蓝图的概念

可以初步理解为单独路由起始路径的子模块。（但是实际上并不是这样）

3.第一个蓝图

[IMG_256](javascript:void(0);)

from flask import Blueprint, render\_template, abortfrom jinja2 import TemplateNotFound

simple\_page=Blueprint('simple\_page',\_\_name\_\_',template\_folder='templates')

@simple\_page.route('/', defaults={'page': 'index'})

@simple\_page.route('/<page>')def show(page):

try:

return render\_template('pages/%s.html' % page)

except TemplateNotFound:

abort(404)

[IMG_257](javascript:void(0);)

当以后在应用中注册蓝图时，这个函数会被注册到应用中。

此外，他会给函数名加上由blueprint的构造函数中给出的 蓝图的名称 作为前缀。

4.注册蓝图

app.register\_blueprint(simple\_page)

# 将蓝图挂载到不同的位置上

app.register\_blueprint(simple\_page, url\_prefix='/pages')

注：可以使用app对象的url\_map方法查看路由规则。

5.蓝图资源

　　1.蓝图资源文件夹

　　　　与常规应用一样，蓝图被认为是包含在一个文件夹中。

　　　　虽然多个蓝图可以源自相同的文件夹中，它并不必须是这种情况并且通常不建议这样做。

　　2.静态文件

　　　　一个蓝图可以通过static\_folder关键字参数提供一个指向文件系统上文件夹的路径，来公开一个带有静态文件的文件夹。

　　　　这可以是一个绝对路径，也可以是相对于蓝图文件夹的路径。

admin = Blueprint('admin', \_\_name\_\_, static\_folder='static')

　　　　蓝图为/admin把静态文件夹注册到/admin/static。

　　　　生成蓝图静态文件的url，类似于应用的静态文件夹所做的那样：

url\_for('admin.static', filename='style.css')

　　3.模板

　　　　如果你想要蓝图公开模板，你可以提供blueprint构造函数中template\_folder参数来实现

admin=Blueprint('admin',\_\_main\_\_, template\_folder='templates')

　　　　像对待静态文件一样，路径可以是绝对的或是相对蓝图资源文件夹的。

　　　　模板文件夹会被加入到模板的搜索路径中，但是比实际的应用模板优先级低。

　　　　那么当你有一个 yourapplication/admin 文件夹中的蓝图并且你想要渲染 'admin/index.html' 模板， 且你已经提供了 templates 作为 template\_folder ，你需要这样创建文件: yourapplication/admin/templates/admin/index.html。

　　　　总感觉有点别扭。

6.构建URLs

推荐使用全称

# 蓝图名称和一个点作为前缀，可以构建到蓝图中视图函数的链接。

url\_for('admin.index')# 单独一个点作为前缀，只适用于在同一个蓝图内的视图函数。

url\_for('.index')

最后我考虑加上一个能运行的最小化附带蓝图实例。

对于templates\_folder参数这部分一定要注意。

如果不赋值，模板应该统一放在应用templates目录下，即yourapplication/templates目录下。

如果赋值为templates/admin，则应该在该蓝图下创建文件: yourapplication/admin/templates/admin/index.html。

# **[flask-login模块](https://www.cnblogs.com/tsxh/p/9696027.html)**

flask-login为flask提供了用户会话管理。他处理了日常的登入，登出并且长时间记住用户的会话。

1.在会话中存储当前活跃的用户ID，让你能够自由地登入和登出。

2.让你限制登入或登出用户可以访问的视图。

3.处理让人棘手的记住我功能。

4.帮助你保护用户会话免遭cookie被盗的牵连。

5.可以与以后可能使用的flask-principal或其他认证扩展集成。

一、配置你的应用

对一个使用flask-login的应用最重要的一部分是loginmanager类。

login\_manager = LoginManager()

login\_manager.init\_app(app)

二、他是如何工作的

必须提供一个user\_loader回调。

@login\_manager.user\_loaderdef load\_user(userid):

return User.get(userid)

接受一个用户的unicodeID作为参数，并且返回响应的用户对象。

如果ID无效的话，他应该返回none。

三、你的用户类

你用来标识用户的类需要实现这些属性和方法：

四个方法：

is\_authenticated、is\_active、is\_anonymous、get\_id

is\_authenticated 当用户通过验证时，也即提供有效证明是返回true。

is\_active 可以直接返回true。

is\_anonymous 如果是匿名用户，返回true。真实用户返回False。

get\_id 返回一个能唯一识别用户的，并能用于user\_loader回调中加载用户的Unicode ID。

要简便的实现用户类，可以从UserMixin继承，他提供了对所有这些方法的默认实现。

四、login示例

一旦用户通过验证，使用login\_user函数让用户登录。

[IMG_256](javascript:void(0);)

@app.route('/login',methods=['GET','POST'])def login():

form = LoginForm()

if form.validate\_on\_submit():

login\_user(user)

flask.flash('Logged in successfully.')

next = flask.request.args.get('next')

if not next\_is\_valid(next):

return flask.abort(400)

return flask.redirct(next or flask.url\_for('index'))

return flask.render\_template('login.html', form=form)

[IMG_257](javascript:void(0);)

这里必须验证next参数，如果不验证，你的应用会收到重定向攻击。

需要用户登入的视图可以用login\_required装饰器来装饰。

@app.route('/settings')

@login\_requireddef settings():

pass

当用户需要登出时，使用logout\_user方法。

@app.route('/logout')

@login\_requireddef logout():

logout\_user()

return redirect(somewhere)

他们会被登出，且他们会话产生的任何cookie都会被清理干净。

五、定制登入过程

默认情况下，当未登录的用户尝试访问一个login\_required装饰的视图，flask-login会闪现

一条消息并且重定向到登录视图。

如果未设置登录视图，他将会以401错误退出。

登录视图的名称可以设置成LoginManager.login\_view。

login\_manager.login\_view = 'users.login'

默认的闪现消息是Please log in to access this page.

要自定义该信息，请设置login\_message。

login\_manager.login\_message = u'please log in'

要自定义消息分类，请设置login\_message\_category。

login\_manager.login\_message\_category = 'info'

当重定向到登入视图，他的请求字符串中会有一个next变量，其值为用户之前访问的页面。

六、使用request loader定制登录

有时你想要不使用cookies情况下登录用户，比如使用http头或者一个作为查询参数的api密钥。

这种情况下应该是用request\_loader回调。

七、匿名用户

默认情况下，当一个用户没有真正的登录，current\_user被设置成一个AnonymousUserMixin对象。

is\_active is\_authenticated的值为False

is\_anonymous的值为true

get\_id返回None

八、记住我

记住我的功能很难实现。但是flask-login几乎透明的实现它。

只要把remember=True传递给login\_user。

一个cookie将会存储在用户计算机中，如果用户会话中没有用户ID的话，flask-login会自动地从cookie中恢复用户ID。

这里cookie是防篡改的。

该层功能是被自动实现的。

九、会话保护

当上述特性保护记住我令牌免遭cookie窃取时，会话cookie依然是脆弱的。

session\_protection的值为basic或者strong。禁用时应设置为None。

默认时被激活为basic模式。

这里需要知道字符串或者数字如何转换为Unicode。非常重要。

计划在下一篇博客中讨论。

# **[effective python-编写高质量python代码的59个有效方法-读书笔记 36-38](https://www.cnblogs.com/tsxh/p/10781303.html)**

并发 计算机似乎是在同一时间做着很多不同的事情。这种交错执行程序的方式，造成了一种假象，使我们以为这些程序可以同时运行。

并行 计算机确实是在同一时间作者很多不同的事。具备多个CPU核心的计算机，能够同时执行多个程序。各程序中的指令，都分别运行在每个CPU内核上面，这些程序就能够在同一时刻向前推进。

在同一个程序内部，并发是一种工具，它使程序员可以更加方便地解决特定类型的问题。

并行与并发的关键区别，就在于能不能提速。

用python语言编写并发程序，是比较容易的。通过系统调用、子进程和C语言扩展等机制，也可以用python平行地处理一些事务。

但是，要想使并发式的python代码以真正并行的方式来运行，却相当困难。

所以我们一定要明白：如何才能在这些有着微妙差别的情境之中，最为恰当地利用Python所提供的特性。

第三十六条 用subprocess模块来管理子进程

python提供了一些非常健壮的程序库，用来运行并管理子进程，这使得python能够很好地将命令行实用程序等工具黏合起来。

由python所启动的多个子进程，是可以平行运作的，这使得我们能够在python程序里充分利用电脑中的全部CPU核心，从而尽量提升程序的处理能力。

用subprocess模块运行子进程，是比较简单的。

煞笔window运行不了这一小节的程序，算了。

要点：

1.可以用subprocess模块运行子进程，并管理其输入流与输出流。

2.python解释器能够平行地运行多条子进程，这使得开发者可以充分利用CPU的处理能力。

3.可以给communicate方法传入timeout参数，以避免子进程死锁或失去响应。

第三十七条 可以用线程来执行阻塞式io，但不要用它做平行计算

标准的python实现叫做cpython。cpython分两步来运行python程序。

首先，把文本形式的源代码解析并编译成字节码。然后，用一种基于栈的解释器来运行这份字节码。

执行python程序时，字节码解释器必须保持协调一致的状态。

python采用GIL机制来确保这种协调性。

GIL实际上就是一把互斥锁，用以防止cpython受到占先式多线程切换操作的干扰。

所谓占先式多线程切换，是指某个线程可以通过打断另外一个线程的方式，来获取程序控制权。

假如这种干扰操作的执行时机不恰当，那就会破坏解释器状态。

GIL有一种非常显著的负面影响。

python程序尽管也支持多线程，但由于受到GIL保护，所以同一时刻，只有一条线程可以向前执行。

这就意味着，如果我们想利用多线程做平行计算，并希望借此为python提速，那么结果会非常令人失望。

[IMG_256](javascript:void(0);)

from time import timedef factorize(number):

for i in range(1, number + 1):

if number % i == 0:

yield i

numbers = [2139079,1214759,151673,1852285]

start = time()for number in numbers:

list(factorize(number))

end = time()print('took %.3f seconds'%(end-start))

took 0.659 seconds

[IMG_257](javascript:void(0);)

[IMG_258](javascript:void(0);)

from threading import Thread from time import time

class FactorizeThread(Thread):

def \_\_init\_\_(self, number):

super().\_\_init\_\_()

self.number = number

def run(self):

self.factors = list(factorize(self.number))

start = time()

threads = []for number in numbers:

thread = FactorizeThread(number)

thread.start()

threads.append(thread)

for thread in threads:

thread.join()

end = time()print('took %.3f seconds'%(end-start))

took 0.645 seconds

[IMG_259](javascript:void(0);)

这样的结果说明，标准cpython解释器中的多线程受到了GIL的影响。

既然如此，python为什么还要支持多线程呢？

首先，多线程使得程序看上去好像能够在同一时间做许多事情。

第二个理由，是处理阻塞式的I/O操作，python在执行某些系统调用时，会触发此类操作。

执行系统调用，是指python程序请求计算机的操作系统与外界环境相交互，以满足程序的需求。（读写文件、在网络间通信，以及与显示器设备相交互）

为了响应这种阻塞式的请求，操作系统必须花一些时间，而开发者可以借助线程，把python程序与这些耗时的I/O操作隔离开。

window又没法正常执行书上的代码。select.select()执行报错。

GIL虽然使得python代码无法并行，但它对系统调用却没有任何负面影响。

由于python线程在执行系统调用的时候会释放GIL，并且一直要等到执行完毕才会重新获取它，所以GIL是不会影响系统调用的。

除了线程，还有其他一些方式，也能处理阻塞式I/O操作，例如内置的asyncio模块等。

虽然那些方式都有着非常显著的优点，但它们要求开发者必须花些功夫，将代码重构成另外一种执行模型。

如果既不想大幅度地修改程序，又要平行地执行多个阻塞式I/O操作，那么使用多线程来实现，会比较简单一些。

要点：

1.因为受到全局解释器锁的限制，所以多条python线程不能在多个CPU核心上面平行地执行字节码。

2.尽管受制于GIL，但是python的多线程功能依然很有用，它可以轻松地模拟出同一时刻执行多项任务的效果。

3.通过python线程，我们可以平行地执行多个系统调用，这使得程序能够在执行阻塞式I/O操作的同时，执行一些运算操作。

第三十八条 在线程中使用Lock来防止数据竞争

明白了全局解释器锁机制之后，许多python编程新手可能会认为：自己在编写python代码时，也不需要再使用互斥锁了。

请注意，真相并非如此。

实际上，GIL并不会保护开发者自己所编写的代码。

同一时刻固然只能有一个python线程得以执行，但是，当这个线程正在操作某个数据结构，其他线程可能会打断它，

也就是说，python解释器在执行两个连续的字节码指令时，其他线程可能会在中途突然插进来。

如果开发者尝试从多个线程中同时访问某个对象，那么上述情形就会引发危险的结果。

这种中断现象随时都可能发生，一旦发生，就会破坏程序的状态，从而使相关的数据结构无法保持其一致性。

[IMG_260](javascript:void(0);)

from threading import Thread, Lock  
from time import time  
  
  
class Counter:  
    def \_\_init\_\_(self):  
        self.count = 0  
    def increment(self, offset):  
        self.count += offset  
  
def worker(sernsor\_index, how\_many, counter):  
    for \_ in range(how\_many):  
        counter.increment(1)  
  
def run\_threads(func, how\_many, counter):  
    threads = []  
    for i in range(5):  
        args = (i, how\_many, counter)  
        thread = Thread(target=func, args=args)  
        threads.append(thread)  
        thread.start()  
    for thread in threads:  
        thread.join()  
  
how\_many = 10\*\*5  
counter = Counter()  
start = time()  
run\_threads(worker, how\_many, counter)  
end = time()  
print('Counter should be %d, found %d' %(5\*how\_many, counter.count))  
print('took %.3f seconds'%(end-start))

Counter should be 500000, found 449003

took 0.133 seconds

[IMG_261](javascript:void(0);)

为了保证所有的线程都能够公平地执行，python解释器会给每个线程分配大致相等的处理器时间。

为了达成这样的分配策略，python系统可能当某个线程正在执行的时候，将其暂停，然后使另外一个线程继续往下执行。

问题就在于，开发者无法准确地获知python系统会在何时暂停这些线程。

有一些操作，看上去好像是原子操作，但python系统依然有可能在线程执行到一半的时候将其暂停。于是就发生了上面那种情况。

为了防止诸如此类的数据竞争行为，python在内置的threading模块里提供了一套健壮的工具，使得开发者可以保护自己的数据结构不受破坏。

其中，最简单、最有用的工具，就是Lock类，该类相当于互斥锁。

我们可以用互斥锁来保护Counter对象，使得多个线程同时访问value值的时候，不会将该值破坏。

同一时刻只有一个线程能够获得这把锁。

范例代码中使用with语句来获取并释放互斥锁，这样写，能够使阅读代码的人更容易看出：线程在拥有互斥锁时，执行的究竟是那一部分代码。

[IMG_262](javascript:void(0);)

from threading import Thread, Lockfrom time import time

class LockingCounter:

def \_\_init\_\_(self):

self.lock = Lock()

self.count = 0

def increment(self, offset):

with self.lock:

self.count += offset

def worker(sernsor\_index, how\_many, counter):

for \_ in range(how\_many):

counter.increment(1)

def run\_threads(func, how\_many, counter):

threads = []

for i in range(5):

args = (i, how\_many, counter)

thread = Thread(target=func, args=args)

threads.append(thread)

thread.start()

for thread in threads:

thread.join()

how\_many = 10\*\*5

counter = LockingCounter()

start = time()

run\_threads(worker, how\_many, counter)

end = time()print('Counter should be %d, found %d' %(5\*how\_many, counter.count))print('took %.3f seconds'%(end-start))

Counter should be 500000, found 500000

took 2.697 seconds

[IMG_263](javascript:void(0);)

这样的运行结果才是我们想要的答案。

由此可见，Lock对象解决了数据竞争的问题。

（但是，加了时间统计的功能，发现差了二十倍的速度，本来我以为只会差五倍左右）

要点：

1.python确实有全局解释器锁，但是在编写自己的程序时，依然要设法防止多个线程争用同一份数据。

2.如果在不加锁的前提下，允许多条线程修改同一个对象，那么程序的数据结构可能会遭到破坏。

3.在python内置的threading模块中，有个名叫Lock的类，它用标准的方式实现了互斥锁。