**作品名称：小游戏“2048”**

1. **成员介绍**

组长：张成 组员：何志颖、邓晨、马崇达、朱明阳、李佳玉。

1. **作品简介**
   1. 应用场景

适用于所有Python2.x用户，但不支持Python3.x用户使用。

* 1. 游戏玩法

通过键盘按键“↑”、“↓“、”←“、”→“操控，移动数字块，若两个数字相同，则他们将合二为一，变为一个更大的数，当所有网格内都填满数字且无法进行移动时，游戏结束。

* 1. 游戏界面

游戏界面为4x4网格，其中会随机出现数字块2或者4，不同的数字有不同的背景颜色。

1. **总体设计**
   1. 基本思路

首先，需要确定一个窗口，在窗口上绘制表格，在表格中随机生成数字。

其次，使数字块可以按照期望的样子进行移动，要判断是否可以移动，移动后产生的后果等。

最后，要判断什么时候会结束游戏，游戏结束后要显示什么才能使玩家明白游戏已经结束。

* 1. 主要技术难点和解决方案

之前从未接触过Tkinter。解决方案：小组成员一起努力，从零开始，一步一步学习，逐步理解和学会简单的运用Tkinter进行编程，最终克服难关。

关于数字块移动的问题。解决方案：设置三个函数以实现，分别为DealSpace()，DealEqualNum()，Move()三个函数，这三个函数经过合理调用，成功实现数字块的移动。

* 1. 构思设计方法

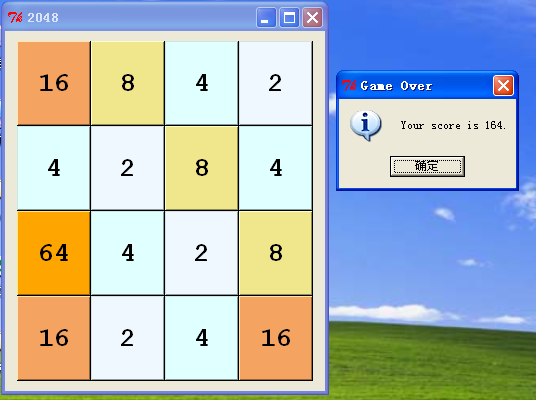
采用自顶向下的方法对项目进行分析构思，设置好所需模块，然后从最底层开始编程，一步步实现设置的模块，最后合并代码，实现整个游戏的制作。

* 1. 项目意义

通过编写程序实现此游戏，可以促进我们对Python和Tkinter相关知识的理解和掌握；可以增加学习的趣味性，让我们更加喜欢编程，热爱编程；通过对项目的开发，也可以使我们了解项目开发的基本过程，提升我们的动手实践能力。

1. **特色和创新点**
   1. 特色：尽最大努力还原原版的2048小游戏，基本达到仿制要求，实现相关环节。原版游戏网址为：<http://2048game.com/> 。
   2. 创新点：项目使用了Tkinter这个Python里的GUI框架。Tkinter模块("Tk 接口")是[Python](https://baike.baidu.com/item/Python/407313)的标准Tk GUI工具包的接口。Tk和Tkinter可以在大多数的[Unix](https://baike.baidu.com/item/Unix/219943)平台下使用，同样可以应用在Windows和Macintosh系统里。Tk8.0的后续版本可以实现本地窗口风格，并良好地运行在绝大多数平台中。因为在课程补充资料里提及Tkinter，所以我们进行了大胆尝试，最终完成了本项目。
2. **运行截图**





1. **项目组成员的工作心得：**

成员一：张成，2016级，项目负责人

学习能力方面：通过完成这个项目，我学习到了关于Tkinter的相关知识，同时也巩固了我之前学习过的python基础知识，提升了自己的动手编程能力，受益匪浅。

独立思考能力方面：作为项目的负责人，我率先提出项目主题，对项目进行了必要的构思设计，独立思考能力得到提升，也与有时候自己思考问题并解决问题会比直接向他人寻求帮助要更好一点。

发现问题能力方面：最初我们计划用pygame来进行项目的开发，但是后来我们发现这样会使我们的工作变得极为困难，因此我们及时改变计划，采用Tkinter进行项目开发。

其它方面：通过大家一起合作，各司其职，我们顺利完成了这个项目，在这个过程中，我的团队协作能力也得到了锻炼，我意识到团队的力量大于个人，虽然我更喜欢一个人完成工作，但是这次团队合作的经历也对我意义重大。

成员二：邓晨，2015级

学习能力方面：通过这次大作业以及一个学期以来的python学习，我对python从一开始的一窍不通，到现在的能写一些简单的小游戏，感觉收获还是很大的。在完成大作业的过程中，我锻炼了自己的自学能力，知道了如何从各种渠道获取资料，并在完成大作业后，对Tkinter有所了解。

独立思考能力方面：锻炼了自己独立处理问题的能力，在调试程序屡次出错的过程中，磨炼了自己的意志，并掌握了几种调试程序的技巧，学会了如何逐步分析程序。

发现问题能力方面：在发现问题方面，基本依靠运行后出现的报错信息入手，对报错的地方进行分析，如果不懂报错的内容指的是哪方面的错误，就去百度。了解错误类型后，在出错的地方，联系上文进行修改

其他方面：在完成这次大作业的过程中，我深刻体会到团队合作的重要性。很多地方，你一个人做的完全没有一个团队去做考虑的周到。将每个人的点子加以分析将会得到各种巧妙的实现方法，遇到错误大家一起克服，这种与他人协同工作的安全感是不能被替代的

成员三：何志颖，2015级

通过本次大作业，我熟悉了python语法，锻炼了编程思维，了解了2048游戏的一种实现方式，对以后编程的学习积攒了经验。

成员四：朱明阳，2016级

学习能力方面：对于陌生的事物，我学会了从不同的层面去了解和深入，总会有一些能看懂的，在不断的接触之后，学到很多新的东西。还有就是模仿，刚开始就处于一种完全下不了手的情况，通过不断地模仿例子后，渐渐熟悉了起来。我算是一个计算机文盲吧，成功地被扫盲了一波。

独立思考能力方面：对于第一次接触编程的我来说，看到几行代码变成可移动的、3D的，而且还能由自己控制，感觉真是很美妙，所以与其说是独立思考的能力的话，还不如说开发了我的想象力。对于三思而后行也有了更多的理解。

发现问题能力方面：问题很多，好像已经不用去发现了，每每只要一操作，就会跳出很多不认识的窗口，然后就卡。大体就这样。

其他方面：零基础学编程，成功地为我打开了一扇门。虽然我还在外面，里面也看不大清，但总有一种神秘的、强大的力量在吸引我。学习是个长时间的过程，需要时间的积淀，通过一学期的学习，为我以后了解计算机指明了方向。

成员五：李佳玉

学习能力方面：通过本次小组作业，我提高了自己的自我学习能力，比如通过网站，视频，等网络学习方式，获得知识，总结经验。

独立思考能力方面：有一定独立思考能力，在本次作业中，独立学习的能力也得到了锻炼。不过大多时候需要寻求帮助，更快更高效得解决问题。

发现问题能力方面：发现问题方面有待提高，希望在今后的不断学习巩固中，不断积累实践能力，能发现自己感兴趣的问题，并解决它。

其他方面：本次小组合作中，每个组员都很用心，但是对于我来说，我发现知识素养更重要，是解决问题的关键，所以在活动中贡献不大，有待改进。

成员六：马崇达，2016级

通过本次python的大作业，我巩固了这个学期所学的python知识，在实际操作中，感受到python强大的库函数所带来的便利，这位以后的编程有非常大的帮助。

1. **存在的问题、建议及其他需要说明的情况**

问题：由于Python2.x与Python3.x的不兼容问题，本项目代码仅供Python2.x用户使用，不支持Python3.x用户使用。具体原因为：在Python3.x中Tkinter变为tkinter，tkMessageBox变为tkinter.messagebox等，正是由于这些问题，导致此程序仅供Python2.x用户使用，若造成不便敬请谅解。

建议：希望本项目以后可以继续完善，如果可能的话，希望加入背景音乐和音效，这样会增加玩家的体验。同时，我们也可以开发适用于Python3.x玩家的2048，毕竟Python3.x也在不断的完善与进步。

说明：小游戏“2048”不会实时的显示玩家的成绩，当且仅当玩家因为没有任何可以移动的数字块，导致游戏结束时，方可得知自己的分数。因为这个游戏并非分数竞技，只有能够成功达到2048的人才是最终的胜利者。

1. **附件：代码**

# -\*- coding: utf8 -\*-

# 经典游戏2048来自第三组的小伙伴

# 调用本程序所需要的模块、包

from Tkinter import \*

import random

import tkMessageBox # 用于显示应用程序的消息框

# 基本窗口建立

root = Tk(className = "2048") # 创建一个根窗口，其余的控件都要在这个窗口上面

root.geometry("320x360") # 改变root大小为320x360

# 相关初始化

str\_data=[]

B\_list = []

frame2 = Frame(root,borderwidth = 10) # 确定框架位置，使不偏离

dic\_color = {0:'GhostWhite', 2:'AliceBlue', 4:'LightCyan', 8:'Khaki',16:'SandyBrown', 32:'Goldenrod',64:'Orange',128:'Maroon',256:'Tomato',512:'OrangeRed',1024:'FireBrick',2048:'Red'} # 为各个值设置不同的颜色

list\_data=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] # 定义数据列表

left=0

up=1

right=2

down=3

derection\_index=[ [[0,1,2,3],[4,5,6,7],[8,9,10,11],[12,13,14,15]],

[[0,4,8,12],[1,5,9,13],[2,6,10,14],[3,7,11,15]],

[[3,2,1,0],[7,6,5,4],[11,10,9,8],[15,14,13,12]],

[[12,8,4,0],[13,9,5,1],[14,10,6,2],[15,11,7,3]] ] # 方向索引

# 4x4网格建立

for i in range(16):

str\_data.append(StringVar()) # StringVar()跟踪变量的值的变化，以保证值的变更随时可以显示在界面上。

B\_list.append(Button(frame2, width=4, height=2, textvariable = str\_data[i] ,font = ("Courier 20 bold roman"),bg=dic\_color[0])) # Button()按钮控件,在程序中显示按钮

B\_list[i].grid(row=i/4,column=i%4) # 行、列网格

# 判定可移动条件函数

def CanMove(derection, index):

for i in range(1, 4):

if list\_data[derection\_index[derection][index][i]] == list\_data[derection\_index[derection][index][i-1]]: # 相同数字可移动

return True

if list\_data[derection\_index[derection][index][i]] != 0 and list\_data[derection\_index[derection][index][i-1]] == 0: # 有空位可移动

return True

return False

# 分配空的区域

def DealSpace(derection, index):

for i in [3,2,1]:

if list\_data[derection\_index[derection][index][i]] == 0:

continue

if list\_data[derection\_index[derection][index][i-1]] == 0:

j = i

while (j < 4):

list\_data[derection\_index[derection][index][j-1]] = list\_data[derection\_index[derection][index][j]]

list\_data[derection\_index[derection][index][j]] = 0

j=j+1

# 消除合并数字函数

def DealEqualNum(derection, index):

for i in [0,1,2]:

if list\_data[derection\_index[derection][index][i]] == list\_data[derection\_index[derection][index][i+1]]:

list\_data[derection\_index[derection][index][i]] = list\_data[derection\_index[derection][index][i]] \* 2

list\_data[derection\_index[derection][index][i+1]] = 0

# 移动函数，通过调用上述两个函数实现

def Move(derection, index):

DealSpace(derection, index)

DealEqualNum(derection, index)

DealSpace(derection, index)

# 刷新新界面函数

def Refresh():

for i in range(16):

B\_list[i].configure(bg=dic\_color[list\_data[i]])

if list\_data[i] == 0:

str\_data[i].set("")

continue

str\_data[i].set(str(list\_data[i]))

# 按键事件函数

def key\_event(event):

canmovelist = []

derection\_code = 9

global list\_data

if event.keycode == 37: # 键盘按键的Unicode值

derection\_code = left

if event.keycode == 39:

derection\_code = right

if event.keycode == 38:

derection\_code = up

if event.keycode == 40:

derection\_code = down

for i in range(4):

if CanMove(derection\_code, i):

Move(derection\_code, i)

canmovelist.append(i)

if len(canmovelist) == 0:

return

list\_data[derection\_index[derection\_code][canmovelist[random.randint(0,len(canmovelist)-1)]][3]] = 2

Refresh()

for i\_derection in range(4): # 判断上下左右

for i\_index in range(4):

if CanMove(i\_derection, i\_index): # 判断是否可继续移动

return

tkMessageBox.showinfo('Game Over','Your score is %u.\n' % sum(list\_data)) # 输出游戏结束

list\_data=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] # 清除所有数字

Refresh() # 刷新新界面

frame2.bind('<KeyRelease>', key\_event) # 事件绑定，捕获键盘事件

frame2.pack() # 布局管理

frame2.focus\_set() # 获取焦点，即直接对框架进行操作而不是根窗口

root.mainloop() #让根窗口进入事件循环