**西南大学**

**商贸学院**

**《Python程序设计与算法基础教程》**

**课程作业**

**年级**：2018级

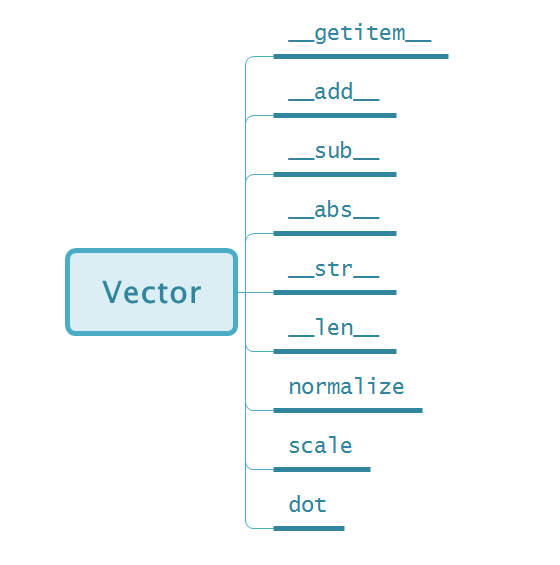
**班级**：计算机科学与技术3班

**学号**：222018603193119

**姓名**：莫健行

# **CS9 文本相似度比较分析**

1. **Vector类的构建与测试**
   1. **Vector类结构：**



* 1. **Vector类源码**

import random, math

class Vector:

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

self.\_\_value = l

self.\_\_length = len(l)

def \_\_getitem\_\_(self, i):

return self.\_\_value[i]

def length(self):

return self.length

def scale(self, n):

return Vector([i \* n for i in self.\_\_value])

def \_\_add\_\_(self, b):

result = [self.\_\_value[i] + b[i] for i in range(self.\_\_length)]

return Vector(result)

def \_\_sub\_\_(self, b):

c = [-i for i in b]

return self.\_\_add\_\_(c)

def \_\_abs\_\_(self):

return math.sqrt(self.dot(self))

def dot(self, b):

result = [self.\_\_value[i] \* b[i] for i in range(self.\_\_length)]

return sum(result)

def normalize(self):

return Vector([i / self.\_\_abs\_\_() for i in self.\_\_value])

def \_\_len\_\_(self):

return self.\_\_length

def \_\_str\_\_(self):

return str(self.\_\_value)

* 1. **Vector类测试**

**测试代码：**

if \_\_name\_\_ == "\_\_main\_\_":

v1 = Vector((1,2))

v2 = Vector((2,3))

print("length of v1 is %s" % len(v1))

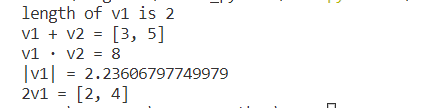
print("v1 + v2 = %s" % (v1+v2))

print("v1 · v2 = %s" % v1.dot(v2))

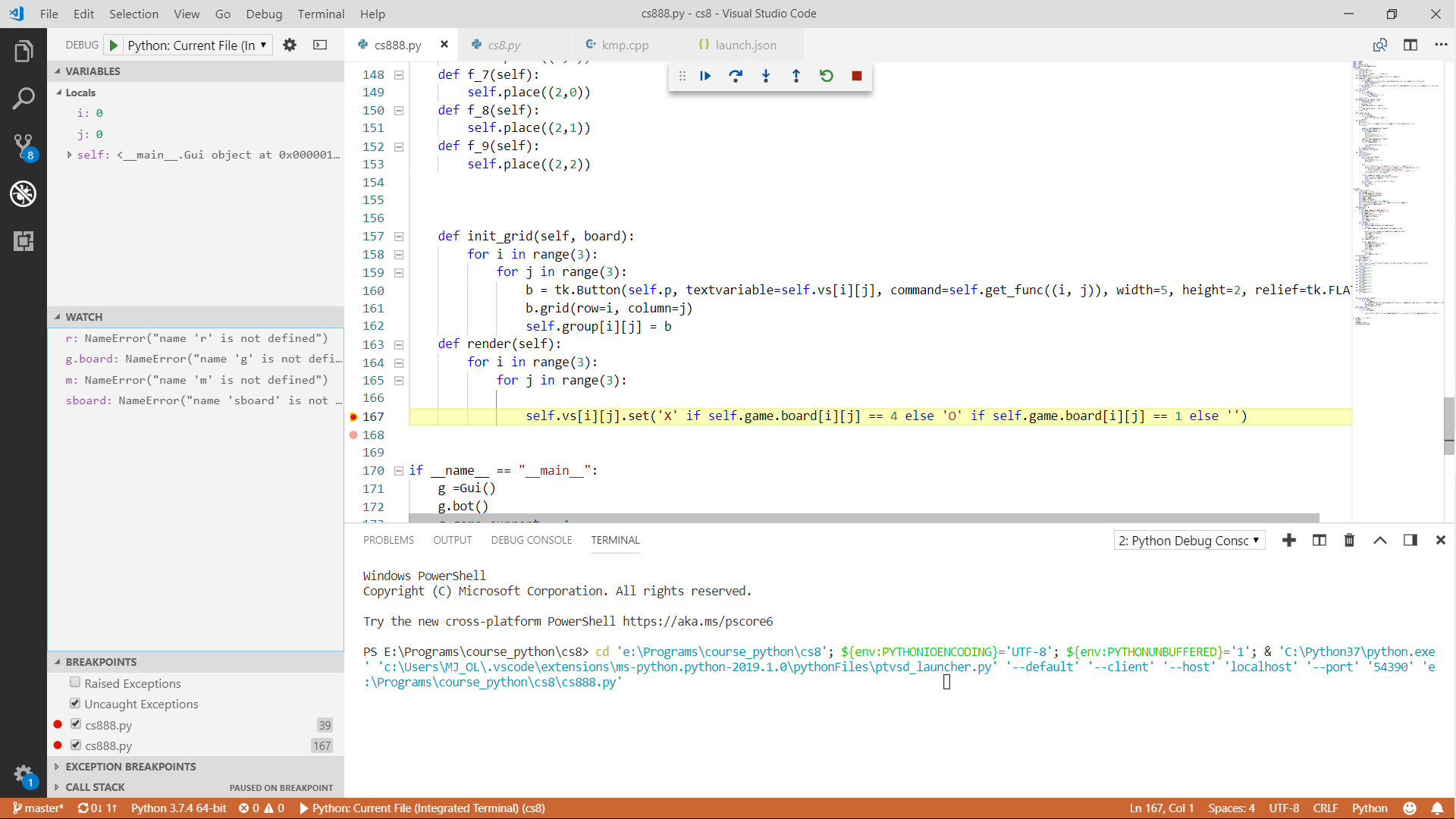
print("|v1| = %s" %abs(v1))

print("2v1 = %s" % v1.scale(2))

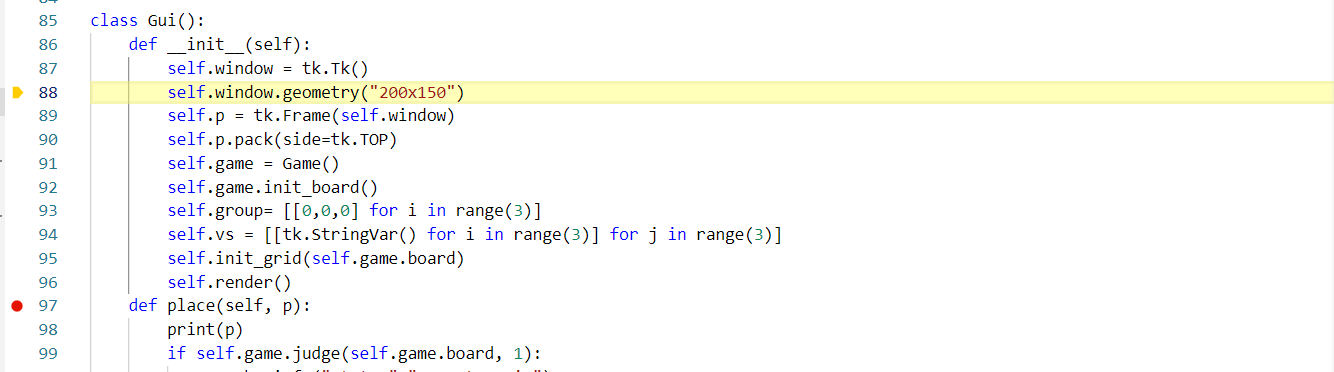
**测试结果**

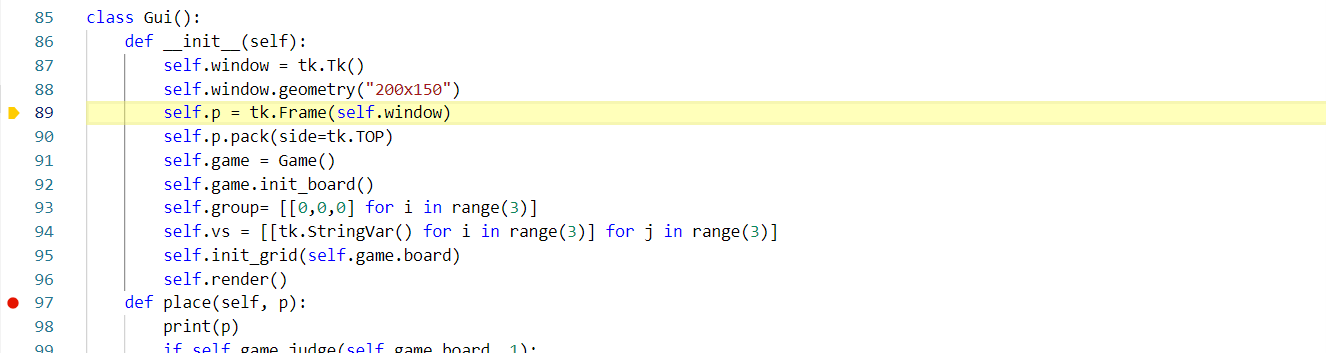
****

1. 断点设置

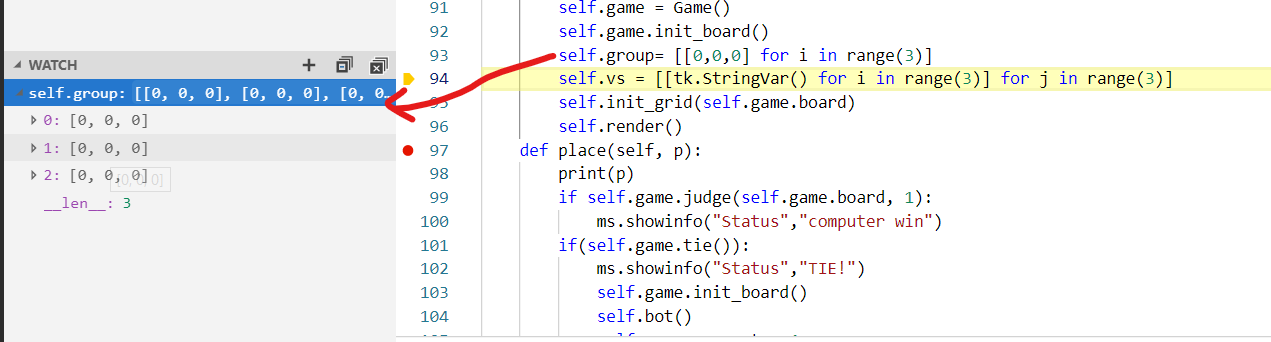


1. 逐步调试（F11）

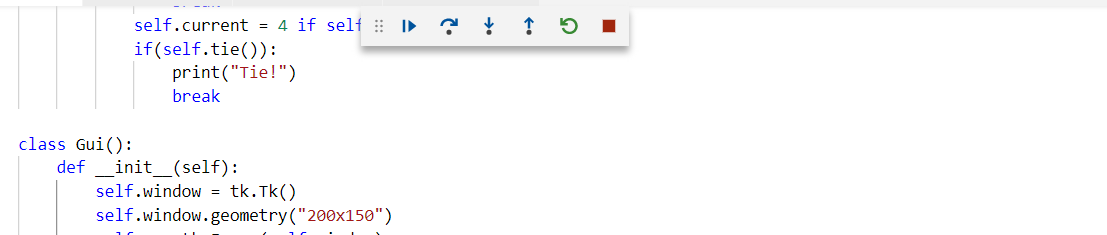




1. 查看变量值



1. 其他工具

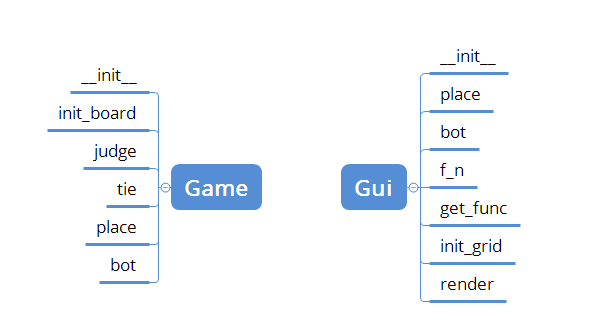


1. 下一断点（F5）
2. 单步执行（F10）
3. 进入函数（F11）
4. 跳出函数（Shift + F11）

# **CS8 图形化界面的井字棋程序**

1. **程序特色**：
2. 使用类对游戏进行包装。
3. 同时设立图形界面类，使用tkinter进行包装

1. **程序结构**



* 1. Game类
     1. \_\_init\_\_：初始化游戏变量
     2. init\_board：初始化棋盘
     3. tie：平局的判断
     4. place：棋子的放置
     5. bot：机器人下棋
  2. Gui类
     1. \_\_init\_\_：初始化界面
     2. init\_grid：初始化九个棋盘位置（按钮）
     3. place：放置棋子
     4. bot：机器人放置棋子的显示
     5. f\_n：实现9个位置被点击之后的位置信息传递（n=1，2，3…9）
     6. get\_func：与按钮绑定，使用lambda表达式，返回对应的f\_n
     7. render：在棋盘上画出位置

1. **代码实现**
   1. 游戏类

class Game:

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

self.board = [[]]

self.current = 1

self.d = {1: 'computer', 4: 'player'}

# 初始化游戏棋盘

def init\_board(self):

self.board = [[0 for j in range(3)] for i in range(3)]

def judge(self, board, current):

for i in range(3):

if sum(board[i]) == 3\*current or sum([board[j][i] for j in range(3)]) == 3\*current:

print(sum(board[i]))

return True

if sum([board[i][i] for i in range(3)])==current\*3 or sum([board[2-i][i] for i in range(3)]) == 3\*current:

return True

return False

# 平局的判断

def tie(self):

for i in range(3):

for j in range(3):

if self.board[i][j] == 0:

return False

return True

# 棋子的放置

def place(self, p, sboard, \*args):

if sboard[p[0]][p[1]] != 0:

return False

if len(args) != 0:

sboard[p[0]][p[1]] = args[0]

else:

sboard[p[0]][p[1]] = self.current

return True

# AI的实现

def bot(self):

print("bot")

m = [[i, j] for i in range(3) for j in range(3) if self.board[i][j] == 0]

for i in m:

# 深复制

sboard = copy.deepcopy(self.board)

self.place(i, sboard, 4)

if self.judge(sboard, 4):

print(i)

# print(i[0],i[1])

self.board[i[0]][i[1]] = 1

return

sboard = copy.deepcopy(self.board)

self.place(i, sboard, 1)

if self.judge(sboard, 1):

self.board[i[0]][i[1]] = 1

return

m = random.choice(m)

self.place(m, self.board)

return

* 1. 图形界面类：

class Gui():

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

self.window = tk.Tk()

self.window.geometry("200x200")

self.p = tk.Frame(self.window)

self.p.pack(side=tk.TOP)

self.game = Game()

self.game.init\_board()

self.group= [[0,0,0] for i in range(3)]

self.vs = [[tk.StringVar() for i in range(3)] for j in range(3)]

self.init\_grid(self.game.board)

self.render()

def place(self, p):

print(p)

if(self.game.tie()):

print("Tie!")

self.game.init\_board()

self.game.current = 1

self.render()

self.render()

if self.game.current == 4:

if not self.game.place(p, self.game.board):

return

if self.game.judge(self.game.board, self.game.current):

print("{} win".format(self.game.d[self.game.current]))

self.game.init\_board()

self.bot()

self.render()

self.game.current = 4

self.game.current = 1

if(self.game.tie()):

print("Tie!")

self.game.init\_board()

self.game.current = 4

self.bot()

self.render()

else:

self.bot()

self.game.current = 4

def bot(self):

self.game.bot()

self.render()

def get\_func(self, p):

i, j = p

funcs = [self.f\_1,self.f\_2,self.f\_3,self.f\_4,self.f\_5,self.f\_6,self.f\_7,self.f\_8,self.f\_9]

return funcs[i\*3 + j]

def f\_1(self):

self.place((0,0))

def f\_2(self):

self.place((0,1))

def f\_3(self):

self.place((0,2))

def f\_4(self):

self.place((1,0))

def f\_5(self):

self.place((1,1))

def f\_6(self):

self.place((1,2))

def f\_7(self):

self.place((2,0))

def f\_8(self):

self.place((2,1))

def f\_9(self):

self.place((2,2))

def init\_grid(self, board):

for i in range(3):

for j in range(3):

b = tk.Button(self.p, textvariable=self.vs[i][j], command=self.get\_func((i, j)), width=5, height=2, relief=tk.FLAT)

b.grid(row=i, column=j)

self.group[i][j] = b

def render(self):

for i in range(3):

for j in range(3):

self.vs[i][j].set('X' if self.game.board[i][j] == 4 else 'O' if self.game.board[i][j] == 1 else '')

* 1. 主函数

if \_\_name\_\_ == "\_\_main\_\_":

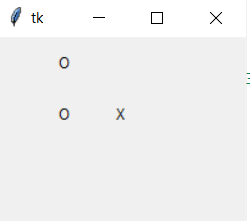
g = Gui()

g.bot()

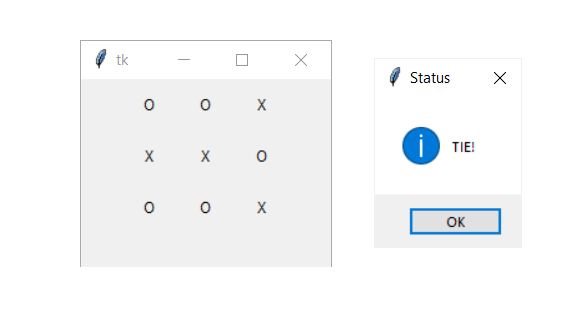
g.game.current = 4

g.window.mainloop()

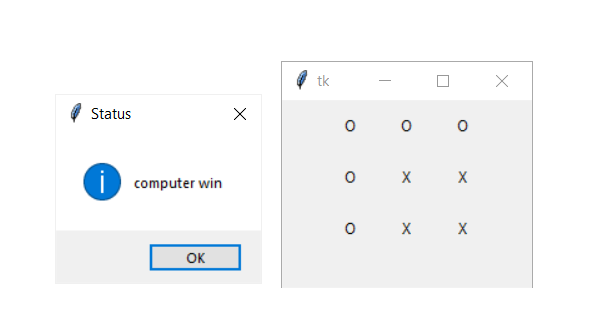
1. **效果演示**
   1. **下棋过程**

****

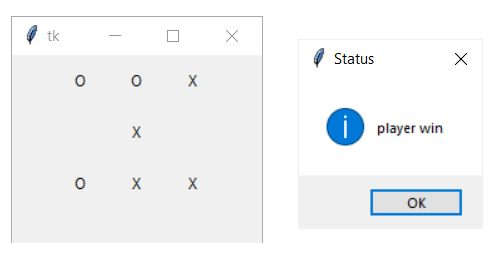
* 1. **平的情况**

****

* 1. **输的情况**

****

* 1. **赢的情况**

****