Lab Report of Python Programming

Lab 5: Object-Oriented Programming Credit hour: 2

Student Name:徐政辉 Student ID:2017329621139

## ￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣

1. Objective
   1. Know how to describe objects and classes and use classes to model objects;
   2. Understand inheritance and polymorphism.
2. Lab content
   1. (Design a Deque) Design a deque class to simulate how to
      * Append to the left or right of the queue
      * Pop out of from left or right of the queue
      * Modify the size of the queue
      * Display the current number of the element in the queue
      * Clear the queue
      * Test if the queue is empty
      * Test if the queue is full

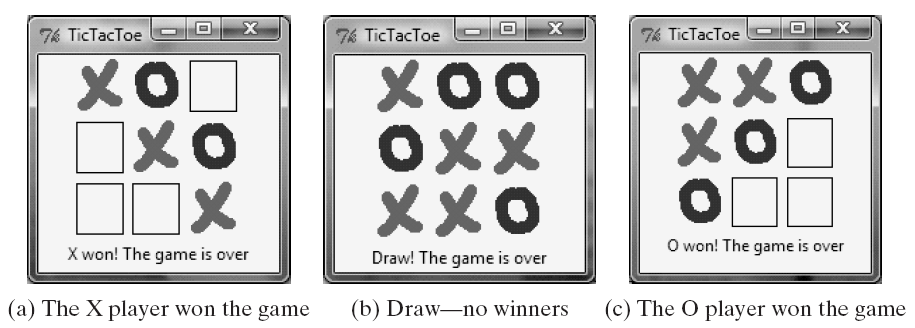
And save the class definition in a file and write another program to testify your class definition.

Tip: Append() method of list object can be used to append element in the end of the list and pop() can be used to return the top element of the list.

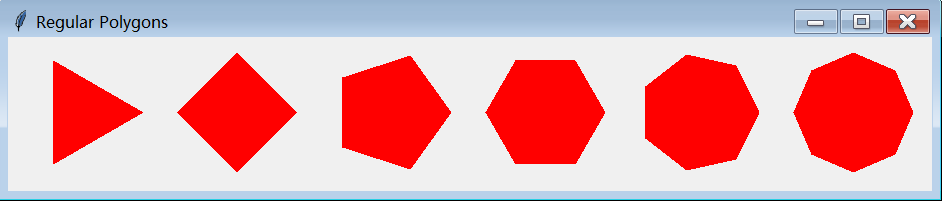
* 1. (Design an Array) Design an Array class to simulate how to support
     + Calculation between array and numeric data
     + Addition and comparison between arrays
     + Access and modification of the element in the array
     + Test the data member in the array

And save the class definition in a file and write another program to testify your class definition.

* 1. (Game: Tic-tac-toe) Write a program that plays the tic-tac-toe game. Two players take turns clicking an available cell in a grid with their respective tokens (either X or O). When one player has placed three tokens in a horizontal, vertical, or diagonal row on the grid, the game is over and that player has won. A draw (no winner) occurs when all the cells in the grid have been filled with tokens and neither player has achieved a win.



* 1. (The RegularPolygonCanvas Class) Define a subclass of Canvas, named RegularPolygonCanvas, to paint an n-sided regular polygon. The class contains a property named numberOfSides, which specifies the number of sides in the polygon. The polygon is centered in the canvas, and the polygon’s size is proportional to the size of the canvas. Create a triangle, square, pentagon, hexagon, heptagon, and octagon from RegularPolygonCanvas and display them.



1. Code list

**2.1**

**Dequeue.py**

class deque:

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

self.items=[]

self.largestSize=0

def addLeft(self,item):

if len(self.items)<self.largestSize:

self.items.insert(0,item)

def addRight(self,item):

if len(self.items)<self.largestSize:

self.items.append(item)

def popLeft(self):

if len(self.items)>0:

self.items.pop(0)

def popRight(self):

if len(self.items)>0:

self.items.pop()

def modeifySize(self,size):

if self.largestSize>size:

for i in range(self.largestSize-size):

self.popRight()

self.largestSize=size

def displayNumber(self):

return len(self.items)

def clearQueue(self):

self.items=[]

def isEmpty(self):

return len(self.items)==0

DequeueMain.py

from dequeue import \*

dq=deque()

dq.modeifySize(10)

dq.addLeft(5)

dq.addRight(4)

print(dq.displayNumber())

dq.popLeft()

print(dq.displayNumber())

dq.modeifySize(20)

**2.2**

**Array.py**

class Array:

def \_\_init\_\_(self,arr=[]):

self.elements=arr

def \_\_mul\_\_(self,a):

elem=self.elements

for i in range(len(self.elements)):

elem[i]\*=a

return Array(elem)

def \_\_truediv\_\_(self,a):

elem=self.elements

for i in range(len(self.elements)):

elem[i]/=a

return Array(elem)

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

return Array(arr.elements+self.elements)

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

re=[]

for i in range(len(self.elements)):

judge=True

for k in range(len(arr.elements)):

if self.elements[i]==arr.elements[k]:

judge=False

break

if judge:

re.append(self.elements[i])

judge=True

return Array(re)

def \_\_cmp\_\_(self,arr):

return self.elements==arr.elements

def modifyElem(self,v,pos):

self.elements[pos]=v

Arraymain.py

from Array import \*

x=[1,2,3]

a=Array(x)

b=a\*3

print(b.elements)

c=a+b

print(c.elements)

d=c-a

print(d.elements)

e=b/2

print(e.elements)

print(a==b)

**2.3**

**Tic\_tac\_toe.py**

from tkinter import \*

class ticTacToe:

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

self.player=0

self.palytimes=0

self.window=Tk()

self.window.title("tic-tac-toe")

self.gameover=0

self.havedraw=[0]\*9

self.empty=PhotoImage(file="empty.gif")

self.o=PhotoImage(file="o.gif")

self.x=PhotoImage(file="x.gif")

self.frame=Frame(self.window)

self.frame.pack()

self.canvas=Canvas(self.frame)

self.canvas["width"]=180

self.canvas["height"]=180

self.canvas.pack(side=LEFT)

self.window.bind("<Button-1>",self.showphoto)

self.canvas.create\_image((40\*1,40\*1),image=self.empty,tags="empty1")

self.canvas.create\_image((40\*2,40\*1),image=self.empty,tags="empty2")

self.canvas.create\_image((40\*3,40\*1),image=self.empty,tags="empty3")

self.canvas.create\_image((40\*1,40\*2),image=self.empty,tags="empty4")

self.canvas.create\_image((40\*2,40\*2),image=self.empty,tags="empty5")

self.canvas.create\_image((40\*3,40\*2),image=self.empty,tags="empty6")

self.canvas.create\_image((40\*1,40\*3),image=self.empty,tags="empty7")

self.canvas.create\_image((40\*2,40\*3),image=self.empty,tags="empty8")

self.canvas.create\_image((40\*3,40\*3),image=self.empty,tags="empty9")

self.window.mainloop()

def drawo(self,pos):

self.canvas.create\_image(pos,image=self.o)

def drawx(self,pos):

self.canvas.create\_image(pos,image=self.x)

def getpos(self,x,y):

if x+20>40 and x+20<80 and y+20>40 and y+20<80:

return (40,40)

elif x+20>80 and x+20<120 and y+20>40 and y+20<80:

return (80,40)

elif x+20>120 and x+20<160 and y+20>40 and y+20<80:

return (120,40)

elif x+20>40 and x+20<80 and y+20>80 and y+20<120:

return (40,80)

elif x+20>80 and x+20<120 and y+20>80 and y+20<120:

return (80,80)

elif x+20>120 and x+20<160 and y+20>80 and y+20<120:

return (120,80)

elif x+20>40 and x+20<80 and y+20>120 and y+20<160:

return (40,120)

elif x+20>80 and x+20<120 and y+20>120 and y+20<160:

return (80,120)

elif x+20>120 and x+20<160 and y+20>120 and y+20<160:

return (120,120)

else:

return (0,0)

def deleteempty(self,number):

if number==0:

self.canvas.delete("empty1")

if number==1:

self.canvas.delete("empty2")

if number==2:

self.canvas.delete("empty3")

if number==3:

self.canvas.delete("empty4")

if number==4:

self.canvas.delete("empty5")

if number==5:

self.canvas.delete("empty6")

if number==6:

self.canvas.delete("empty7")

if number==7:

self.canvas.delete("empty8")

if number==8:

self.canvas.delete("empty9")

def showphoto(self,event):

if self.gameover==1:

return

pos=self.getpos(event.x,event.y)

number=pos[0]//40+3\*(pos[1]//40)-4

if pos!=(0,0) and self.havedraw[number]==0 and self.palytimes<10:

self.deleteempty(number)

if self.player==0 :

self.drawo(pos)

self.player=1

self.havedraw[number]=1

elif self.player==1:

self.drawx(pos)

self.player=0

self.havedraw[number]=2

self.palytimes+=1

win=self.ifsomeonewin()

if win==1:

self.canvas.create\_text(90,160,text="The O player won the game")

self.gameover=1

elif win==2:

self.canvas.create\_text(90,160,text="The X player won the game")

self.gameover=1

elif win==0 and self.palytimes==9:

self.canvas.create\_text(90,160,text="Draw--no winners")

self.gameover=1

def ifsomeonewin(self):

for i in range(1,4):

cnt1=0

cnt2=0

for k in range(1,4):

if self.havedraw[i+3\*k-4]==1:

cnt1+=1

elif self.havedraw[i+3\*k-4]==2:

cnt2+=1

if cnt1==3:

return 1

if cnt2==3:

return 2

for i in range(1,4):

cnt1=0

cnt2=0

for k in range(1,4):

if self.havedraw[i+(k-1)\*3-1]==1:

cnt1+=1

elif self.havedraw[i+(k-1)\*3-1]==2:

cnt2+=1

if cnt1==3:

return 1

if cnt2 ==3:

return 2

if self.havedraw[0]==1 and self.havedraw[4]==1 and self.havedraw[8]==1:

return 1

elif self.havedraw[2]==1 and self.havedraw[4]==1 and self.havedraw[6]==1:

return 1

elif self.havedraw[0]==2 and self.havedraw[4]==2 and self.havedraw[8]==2:

return 2

elif self.havedraw[2]==2 and self.havedraw[4]==2 and self.havedraw[6]==2:

return 2

return 0

a=ticTacToe()

**2.4**

**Regularpolygoncanvas.py**

from tkinter import \*

import math

class RegularPolygonCanvas(Canvas):

def \_\_init\_\_(self,numberOfSides,width=200,height=200):

super().\_\_init\_\_(width=width,height=height)

self.width=width

self.height=height

self.numberOfSides=numberOfSides

self.createPolygon()

self.pack(side=LEFT)

mainloop()

def createPolygon(self):

points=[]

radius=min(self.width/2,self.height/2)

p=(self.width/2,self.height/2)

for i in range(self.numberOfSides):

radian=2/self.numberOfSides\*math.pi\*i

k=math.tan(radian)

x1=p[0]-radius/(k\*k+1)\*\*0.5

x2=p[0]+radius/(k\*k+1)\*\*0.5

if 0<=radian and radian<math.pi/2:

y=p[1]+k\*(x1-p[0])

points.append(x1)

points.append(y)

elif 360/self.numberOfSides\*i-90<0.001:

points.append(self.width/2)

points.append(0)

elif math.pi>radian and radian>math.pi/2:

y=p[1]+k\*(x2-p[0])

points.append(x2)

points.append(y)

elif math.pi<=radian and radian<math.pi/2\*3:

y=p[1]+k\*(x2-p[0])

points.append(x2)

points.append(y)

elif 360/self.numberOfSides\*i-270<0.001:

points.append(self.width/2)

points.append(self.height)

elif math.pi\*2>radian and radian>math.pi/2\*3:

y=p[1]+k\*(x1-p[0])

points.append(x1)

points.append(y)

print(points)

self.create\_polygon(points,fill="red")

RPC1=RegularPolygonCanvas(3)

RPC2=RegularPolygonCanvas(4)

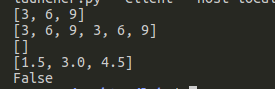
RPC3=RegularPolygonCanvas(5)

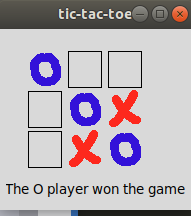
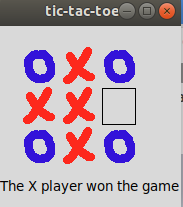
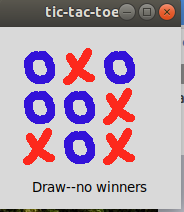
RPC4=RegularPolygonCanvas(6)

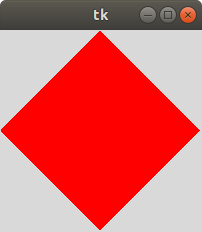
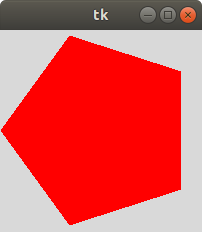
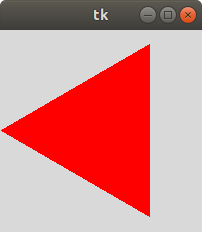
RPC5=RegularPolygonCanvas(7)

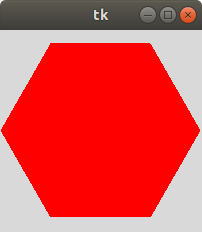
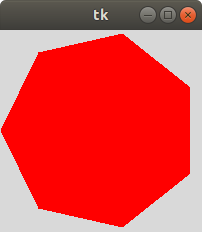
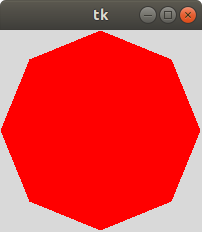
RPC6=RegularPolygonCanvas(8)

1. Output









1. Analysis and conclusions

This lab work let me learn how to use class and how to use inheritence and polymophism so I have learned a lot . Not only how to build a program but how to use math and thought to achieve the program that I want to write. Inheritence is very important it’s from c++ to python and other objects oriented programming language . When you really understand it .You will know how important it is.