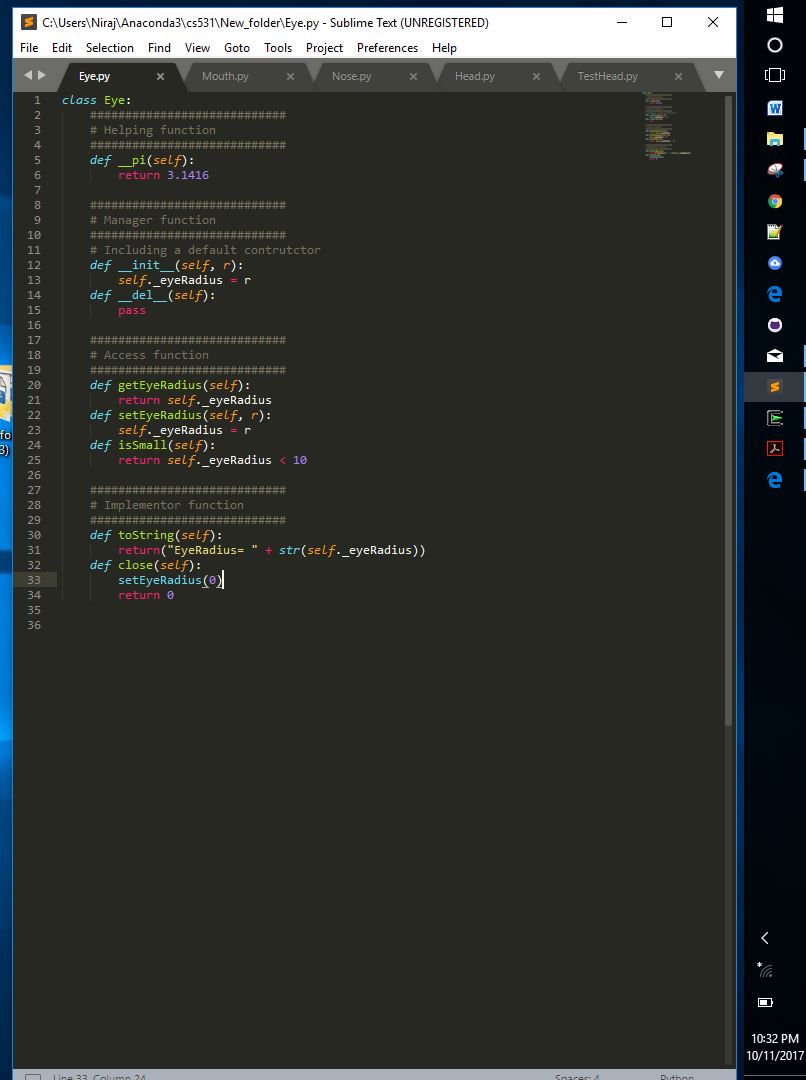
**Name : Niraj Thanki SID : 19376 CLASS : CS531**

**Eye.py**



**Eye.py Soure Code**

class Eye:

############################

# Helping function

############################

def \_\_pi(self):

return 3.1416

############################

# Manager function

############################

# Including a default contrutctor

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

self.\_eyeRadius = r

def \_\_del\_\_(self):

pass

############################

# Access function

############################

def getEyeRadius(self):

return self.\_eyeRadius

def setEyeRadius(self, r):

self.\_eyeRadius = r

def isSmall(self):

return self.\_eyeRadius < 10

############################

# Implementor function

############################

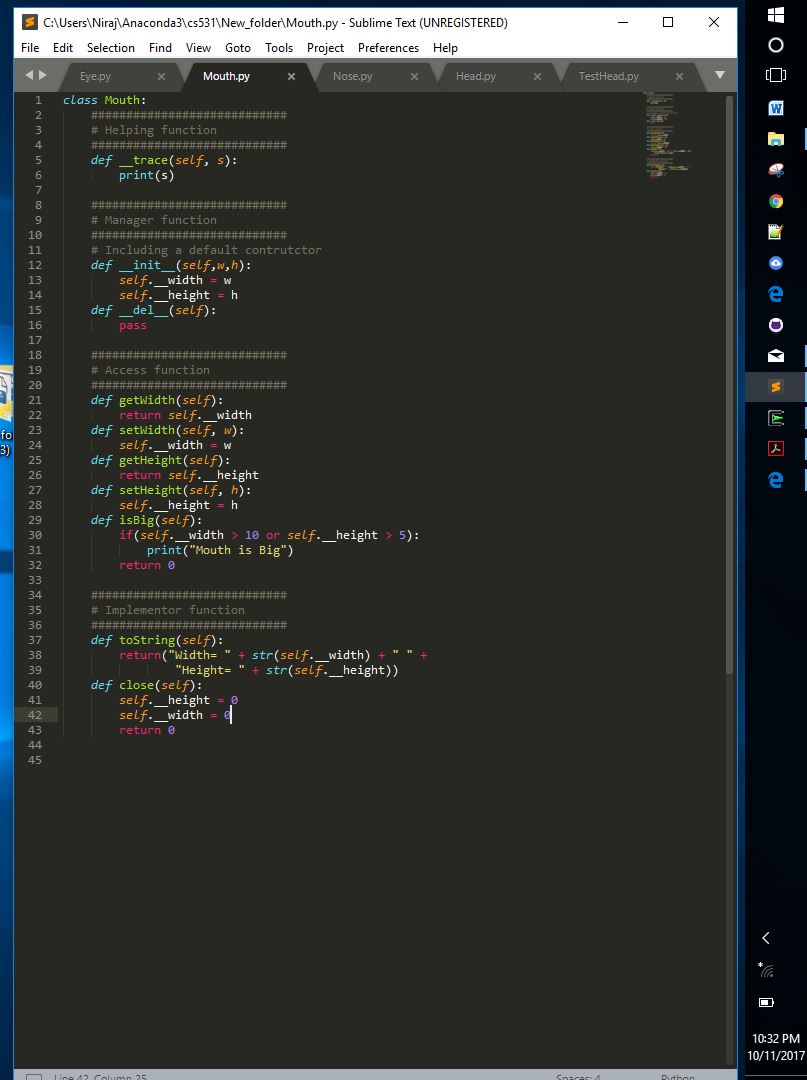
def toString(self):

return("EyeRadius= " + str(self.\_eyeRadius))

def close(self):

setEyeRadius(0)

return 0

**Mouth.py**

**Mouth.py Soure Code**

class Mouth:

############################

# Helping function

############################

def \_\_trace(self, s):

print(s)

############################

# Manager function

############################

# Including a default contrutctor

def \_\_init\_\_(self,w,h):

self.\_\_width = w

self.\_\_height = h

def \_\_del\_\_(self):

pass

############################

# Access function

############################

def getWidth(self):

return self.\_\_width

def setWidth(self, w):

self.\_\_width = w

def getHeight(self):

return self.\_\_height

def setHeight(self, h):

self.\_\_height = h

def isBig(self):

if(self.\_\_width > 10 or self.\_\_height > 5):

print("Mouth is Big")

return 0

############################

# Implementor function

############################

def toString(self):

return("Width= " + str(self.\_\_width) + " " +

"Height= " + str(self.\_\_height))

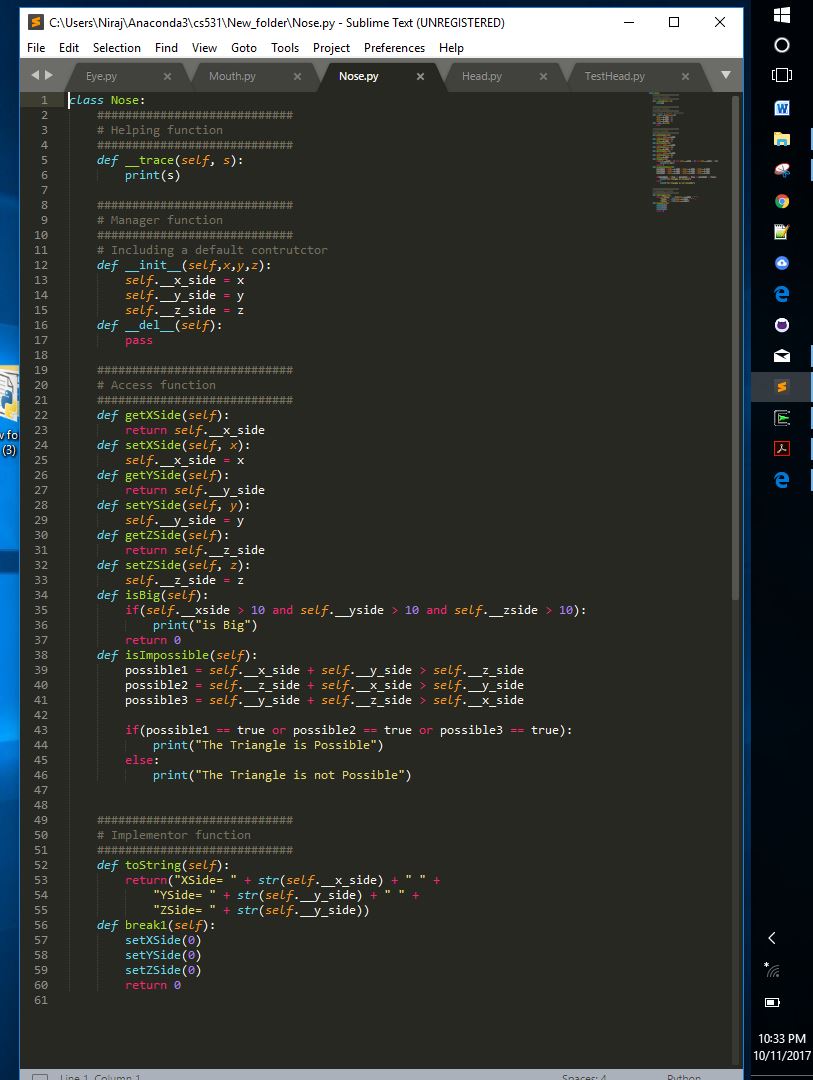
def close(self):

self.\_\_height = 0

self.\_\_width = 0

return 0

**Nose.py**



**Nose.py Soure Code**

class Nose:

############################

# Helping function

############################

def \_\_trace(self, s):

print(s)

############################

# Manager function

############################

# Including a default contrutctor

def \_\_init\_\_(self,x,y,z):

self.\_\_x\_side = x

self.\_\_y\_side = y

self.\_\_z\_side = z

def \_\_del\_\_(self):

pass

############################

# Access function

############################

def getXSide(self):

return self.\_\_x\_side

def setXSide(self, x):

self.\_\_x\_side = x

def getYSide(self):

return self.\_\_y\_side

def setYSide(self, y):

self.\_\_y\_side = y

def getZSide(self):

return self.\_\_z\_side

def setZSide(self, z):

self.\_\_z\_side = z

def isBig(self):

if(self.\_\_xside > 10 and self.\_\_yside > 10 and self.\_\_zside > 10):

print("is Big")

return 0

def isImpossible(self):

possible1 = self.\_\_x\_side + self.\_\_y\_side > self.\_\_z\_side

possible2 = self.\_\_z\_side + self.\_\_x\_side > self.\_\_y\_side

possible3 = self.\_\_y\_side + self.\_\_z\_side > self.\_\_x\_side

if(possible1 == true or possible2 == true or possible3 == true):

print("The Triangle is Possible")

else:

print("The Triangle is not Possible")

############################

# Implementor function

############################

def toString(self):

return("XSide= " + str(self.\_\_x\_side) + " " +

"YSide= " + str(self.\_\_y\_side) + " " +

"ZSide= " + str(self.\_\_y\_side))

def break1(self):

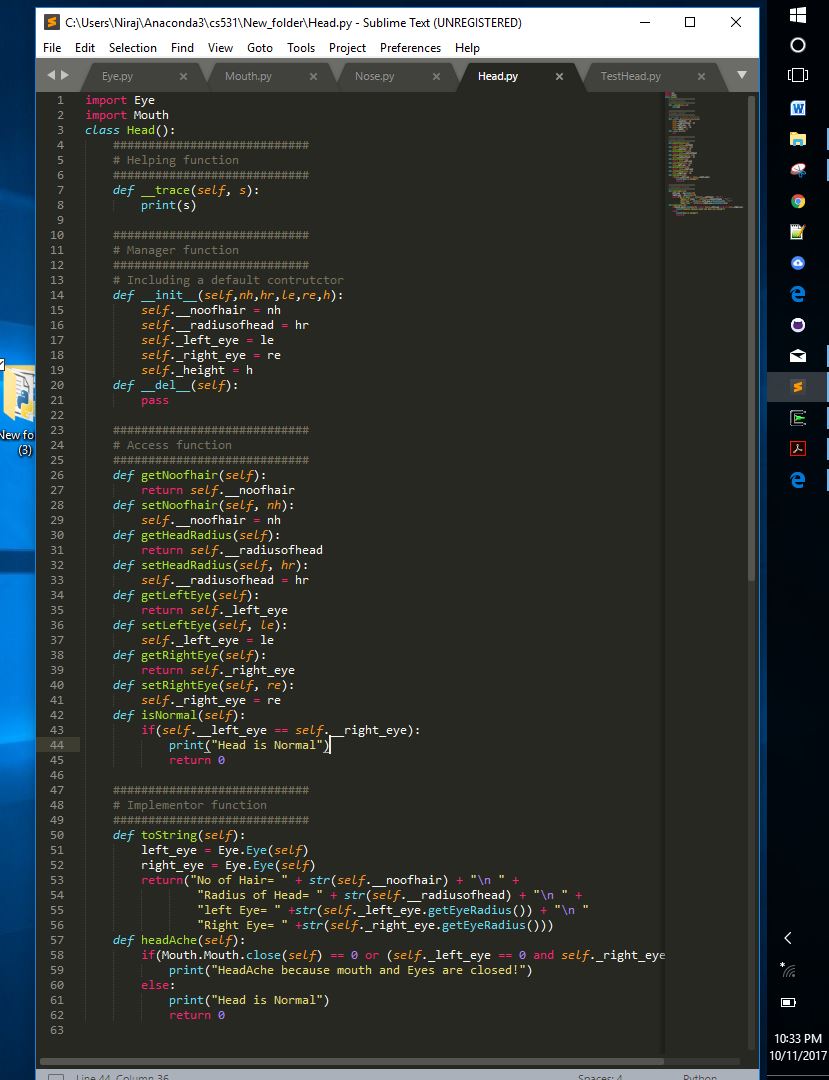
setXSide(0)

setYSide(0)

setZSide(0)

return 0

**Head.py**



**Head.py Source Code**

import Eye

import Mouth

class Head():

############################

# Helping function

############################

def \_\_trace(self, s):

print(s)

############################

# Manager function

############################

# Including a default contrutctor

def \_\_init\_\_(self,nh,hr,le,re,h):

self.\_\_noofhair = nh

self.\_\_radiusofhead = hr

self.\_left\_eye = le

self.\_right\_eye = re

self.\_height = h

def \_\_del\_\_(self):

pass

############################

# Access function

############################

def getNoofhair(self):

return self.\_\_noofhair

def setNoofhair(self, nh):

self.\_\_noofhair = nh

def getHeadRadius(self):

return self.\_\_radiusofhead

def setHeadRadius(self, hr):

self.\_\_radiusofhead = hr

def getLeftEye(self):

return self.\_left\_eye

def setLeftEye(self, le):

self.\_left\_eye = le

def getRightEye(self):

return self.\_right\_eye

def setRightEye(self, re):

self.\_right\_eye = re

def isNormal(self):

if(self.\_\_left\_eye == self.\_\_right\_eye):

print("Head is Normal")

return 0

############################

# Implementor function

############################

def toString(self):

left\_eye = Eye.Eye(self)

right\_eye = Eye.Eye(self)

return("No of Hair= " + str(self.\_\_noofhair) + "\n " +

"Radius of Head= " + str(self.\_\_radiusofhead) + "\n " +

"left Eye= " +str(self.\_left\_eye.getEyeRadius()) + "\n "

"Right Eye= " +str(self.\_right\_eye.getEyeRadius()))

def headAche(self):

if(Mouth.Mouth.close(self) == 0 or (self.\_left\_eye == 0 and self.\_right\_eye == 0)):

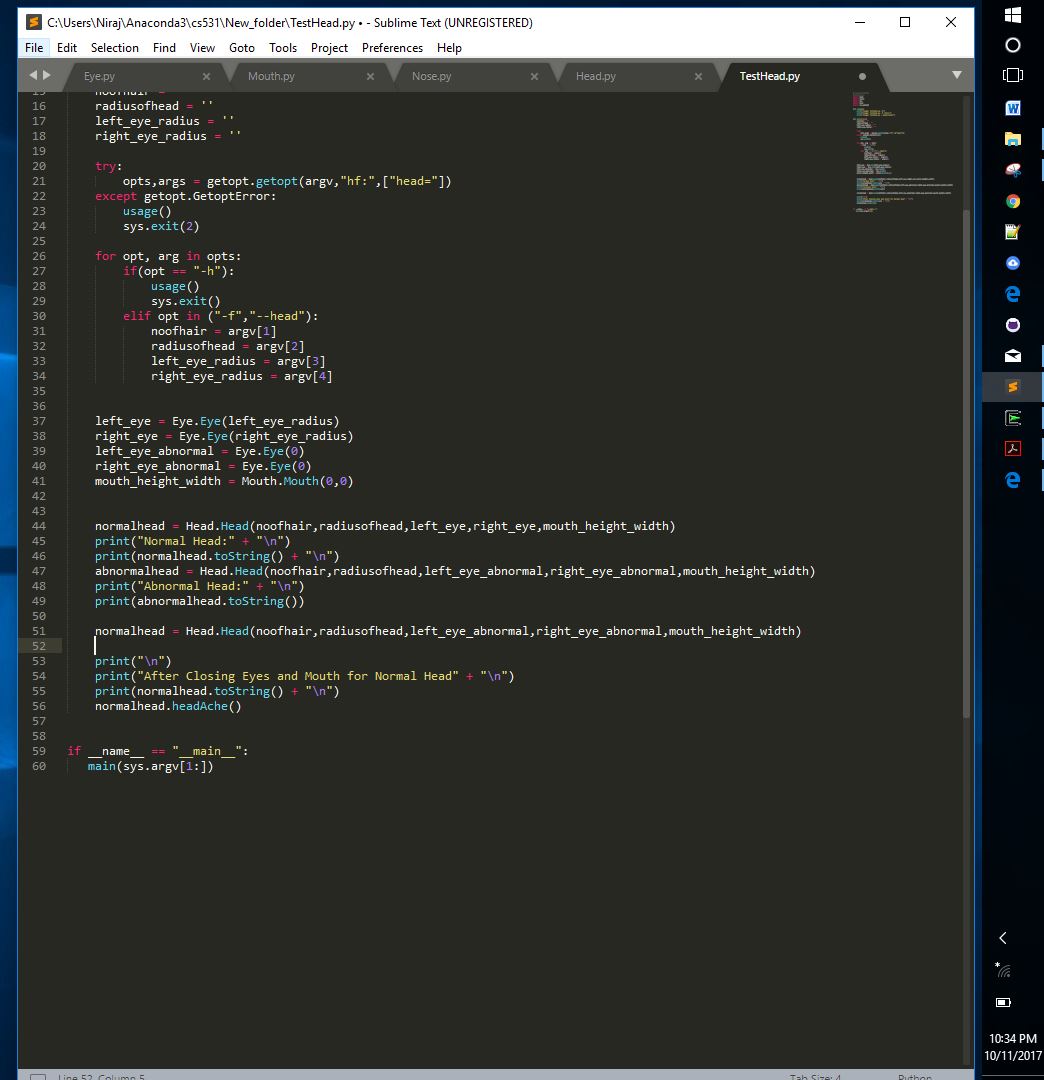
print("HeadAche because mouth and Eyes are closed!")

else:

print("Head is Normal")

return 0

**TestHead.py**



**TestHead.py Source Code**

#!/usr/bin/python

#TestFace.py

import Head

import Mouth

import Eye

import Nose

import sys,getopt

def usage():

print("Usage: TestHead.py -h")

print("Usage: TestHead.py -f <face>")

print("Usage: TestHead.py --head=<head>")

def main(argv):

noofhair = ''

radiusofhead = ''

left\_eye\_radius = ''

right\_eye\_radius = ''

try:

opts,args = getopt.getopt(argv,"hf:",["head="])

except getopt.GetoptError:

usage()

sys.exit(2)

for opt, arg in opts:

if(opt == "-h"):

usage()

sys.exit()

elif opt in ("-f","--head"):

noofhair = argv[1]

radiusofhead = argv[2]

left\_eye\_radius = argv[3]

right\_eye\_radius = argv[4]

left\_eye = Eye.Eye(left\_eye\_radius)

right\_eye = Eye.Eye(right\_eye\_radius)

left\_eye\_abnormal = Eye.Eye(0)

right\_eye\_abnormal = Eye.Eye(0)

mouth\_height\_width = Mouth.Mouth(0,0)

normalhead = Head.Head(noofhair,radiusofhead,left\_eye,right\_eye,mouth\_height\_width)

print("Normal Head:" + "\n")

print(normalhead.toString() + "\n")

abnormalhead = Head.Head(noofhair,radiusofhead,left\_eye\_abnormal,right\_eye\_abnormal,mouth\_height\_width)

print("Abnormal Head:" + "\n")

print(abnormalhead.toString())

normalhead = Head.Head(noofhair,radiusofhead,left\_eye\_abnormal,right\_eye\_abnormal,mouth\_height\_width)

print("\n")

print("After Closing Eyes and Mouth for Normal Head" + "\n")

print(normalhead.toString() + "\n")

normalhead.headAche()

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

main(sys.argv[1:])

**OUTPUT :**