**4조: 한국인의 눈 건강을 책임지는 팀**

소스코드

import turtle

import random

from tkinter import\*

######노안검사 시작#######

def test3():

turtle.st()

image = "C:/시력검사/file.gif"

screen = turtle.Screen()

screen.addshape(image)

turtle.shape(image)

test3select()

def test3select():

answer = int(input("왼쪽이 더 잘 보이면 1, 오른쪽이 더 잘 보이면 2, 비슷하다면 3을 입력하세요.:"))

if answer == 1:

print("당신은 노안이 아닙니다.")

elif answer == 2:

print("당신은 노안입니다.")

print("전문의에게 정밀 검사를 받아보세요.")

elif answer == 3:

print("당신은 노안이 아닙니다.")

else:

print("다시 입력해주세요.")

test3select()

image2 = "C:/시력검사/white.gif"

screen = turtle.Screen()

screen.addshape(image2)

turtle.shape(image2)

turtle.ht()

select()

######노안검사 마감#######

######근시 및 난시검사 시작#######

def test2():

turtle.st()

image = "C:/시력검사/a.gif"

image2 = "C:/시력검사/white.gif"

screen = turtle.Screen()

screen.addshape(image)

screen.addshape(image2)

turtle.shape(image)

#UI영역

answer = int(input("그림에서 보이는 숫자를 입력하세요:"))

if answer == 3246:

print("난시와 근시가 의심됩니다.")

print("전문의에게 정밀 검사를 받아보세요.")

elif answer == 3240:

print("난시가 의심됩니다.")

print("전문의에게 정밀 검사를 받아보세요.")

elif answer == 1246:

print("근시가 의심됩니다.")

print("전문의에게 정밀 검사를 받아보세요.")

elif answer == 1240:

print("건강한 눈으로 판별 되었습니다")

else:

print("측정이 불가능 한 값입니다.")

turtle.shape(image2)

turtle.ht()

select()

######근시 및 난시검사 마감#######

######도수검사 시작#######

class DrawBoard:

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

self.t = turtle.Pen()

self.t.ht()

self.t.speed("fastest")

def clear(self):

self.t.clear()

#숫자 0~9출력함수모음()

def number0(self, x): #비율

self.t.speed("fastest")

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.up()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(x)

self.t.rt(90)

self.t.fd(x)

self.t.lt(90)

self.t.fillcolor("white")

self.t.begin\_fill()

self.t.fd(x)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x)

self.t.rt(90)

self.t.fd(x\*3)

self.t.end\_fill()

self.t.penup()

self.t.fd(x)

self.t.rt(90)

self.t.fd(-x)

def number1(self, x):

self.t.speed("fastest")

self.t.penup()

self.t.fd(x)

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(-x)

def number2(self, x):

self.t.speed("fastest")

self.t.up()

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*1)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

def number3(self, x):

self.t.speed("fastest")

self.t.fillcolor("black")

self.t.up()

self.t.begin\_fill()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

def number4(self, x):

self.t.speed("fastest")

self.t.pencolor("black")

self.t.up()

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*3)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

def number5(self, x):

self.t.speed("fastest")

self.t.fillcolor("black")

self.t.up()

self.t.begin\_fill()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*3)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

def number6(self, x):

self.t.speed("fastest")

self.t.pencolor("black")

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.up()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(x)

self.t.rt(90)

self.t.fd(x\*3)

self.t.lt(90)

self.t.fillcolor("white")

self.t.begin\_fill()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.end\_fill()

self.t.penup()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(-x)

def number7(self, x):

self.t.speed("fastest")

self.t.fillcolor("black")

self.t.up()

self.t.begin\_fill()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*4)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*1)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

def number8(self, x):

self.t.speed("fastest")

self.t.up()

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fillcolor("white")

self.t.begin\_fill()

self.t.up()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fillcolor("white")

self.t.begin\_fill()

self.t.penup()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.lt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(-x)

def number9(self, x):

self.t.speed("fastest")

self.t.fillcolor("black")

self.t.begin\_fill()

self.t.up()

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*5)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*2)

self.t.lt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fd(x\*2)

self.t.rt(90)

self.t.fd(x\*3)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.lt(90)

self.t.fillcolor("white")

self.t.begin\_fill()

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.fd(x\*1)

self.t.rt(90)

self.t.end\_fill()

self.t.penup()

self.t.fd(x)

self.t.rt(90)

self.t.fd(-x)

#도수검사비율

def ratio(a, standardnumber):#return = 비율을 반환

if a >= -0.05 and a <= 0.05: #0.1초과

return 1

elif a > 0.05 and a <= 0.15: #0.1

return 35.4330+0.296\*standardnumber

elif a > 0.15 and a <= 0.25: #0.2

return 17.7165+0.148\*standardnumber

elif a > 0.25 and a <= 0.35: #0.3

return 11.8110+0.099\*standardnumber

elif a > 0.35 and a <= 0.45: #0.4

return 8.85826+0.074\*standardnumber

elif a > 0.45 and a <= 0.55: #0.5

return 7.08661+0.059\*standardnumber

elif a > 0.55 and a <= 0.65: #0.6

return 5.90551+0.049\*standardnumber

elif a > 0.65 and a <= 0.75: #0.7

return 5.06186+0.042\*standardnumber

elif a > 0.75 and a <= 0.85: #0.8

return 4.42913+0.037\*standardnumber

elif a > 0.85 and a <= 0.95: #0.9

return 3.93700+0.033\*standardnumber

elif a > 0.95 and a <= 1.05: #1.0

return 3.54330+0.030\*standardnumber

elif a > 1.05 and a <= 1.15: #1.1

return 3.22118+0.027\*standardnumber

elif a > 1.15 and a <= 1.25: #1.2

return 2.95275+0.025\*standardnumber

elif a > 1.25 and a <= 1.35: #1.3

return 2.72562+0.023\*standardnumber

elif a > 1.35 and a <= 1.45: #1.4

return 2.53093+0.021\*standardnumber

elif a > 1.45 and a <= 1.55: #1.5

return 2.36220+0.020\*standardnumber

elif a > 1.55 and a <= 1.65: #1.6

return 2.214566+0.018\*standardnumber

elif a > 1.65 and a <= 1.75: #1.7

return 2.084298+0.017\*standardnumber

elif a > 1.75 and a <= 1.85: #1.8

return 1.968503+0.016\*standardnumber

elif a > 1.85 and a <= 1.95: #1.9

return 1.864898+0.015\*standardnumber

elif a > 1.95 and a <= 2.05: #2.0

return 1.771653+0.014\*standardnumber

else: #2.0초과

return 1

board = DrawBoard()

Y = 0.1 #도수

count = 1 #틀린 횟수

standardnumber = 0 #표준도수

#도수검사랜덤숫자 출력

def randomnumber():#Y = 현재 도수

global count

global Y

count = 1 #[\*]도수검사를 재시작했을 때를 위해 초기화

Y = 0.1

while count != 4:

i = random.randrange(0,9)

j = random.randrange(0,9)

h = i\*10 + j

X = Y

X = ratio(Y, standardnumber) #도수를 비율함수에 넣어 1차 비율 값을 받음

if X == 0 or X == 1 :

print("\n\n[#] 시력이 검사 범위를 초과하였습니다. \n다시 검사선택으로 넘어갑니다.")

select()

board.t.ht()

board.clear()

board.t.speed('fastest')

board.t.penup()

board.t.goto(-X\*3.5,X\*2.5)

getattr(board, 'number{num}'.format(num=i))(X)

board.t.ht()

board.t.penup()

board.t.fd(X\*4)

getattr(board, 'number{num}'.format(num=j))(X)

board.t.ht()

board.t.penup()

board.t.fd(-X)

print("\n\n\n\n\n\n(\*)기회:"+str(4-count)+"번 \n(\*)현재 시험 도수 %.1f:" % Y)

answer = int(input("보이는 숫자를 입력해주시기 바랍니다."))

if count == 3:

print("당신의 도수는 %.1f입니다." % Y)

if(Y<=0.5):

print("시력 교정이 필요합니다. 안과에 방문하여 의사에게 재대로 된 진단을 받아보세요")

select()

elif count != 3 and h == answer:

Y += 0.10000000

print("[O] 『정답』")

board.t.reset()

else:

Y -= 0.10000000

count += 1

print("[X] 『오답』")

board.t.reset()

def CardStandard(): #초기 비율 상수 값 정하기

StandardNumber=0

board.t.speed('fastest')

while StandardNumber != True:

board.t.clear()

board.t.up()

board.t.goto(-190-((3.17\*StandardNumber)/2),120+StandardNumber)

board.t.down()

board.t.fillcolor("violet")

board.t.begin\_fill()

board.t.fd(380+(3.17\*StandardNumber))

board.t.rt(90)

board.t.fd(240+(2\*StandardNumber))

board.t.rt(90)

board.t.fd(380+(3.17\*StandardNumber))

board.t.rt(90)

board.t.fd(240+(2\*StandardNumber))

board.t.rt(90)

board.t.end\_fill()

numper=int(input("신용카드 증가=1, 축소=2, 기준점 완료=3 : "))

if numper==1:

StandardNumber=StandardNumber+3

elif numper==2:

StandardNumber=StandardNumber-3

elif numper==3:

board.t.clear()

board.t.reset()

break

else:

print("다시 입력해주세요")

return StandardNumber

######도수검사 마감#######

def select():

print("\n[C] 원하시는 검사를 선택해 주세요(\*^^\*).")

while True:

select = int(input("[\*] 1.도수검사 2.근시난시검사 3.노안 검사 \n검사를 끝내시려면 0번을 적어주세요: "))

if select == 1:

print("\n[\*]신용카드를 화면의 크기와 동일하게 맞춰주세요")

global standardnumber

standardnumber = CardStandard()

randomnumber()

elif select == 2:

test2()

elif select == 3:

test3()

else:

turtle.bye()

break

#if \_name\_ == '\_main\_': #처음 UI

print("안녕하세요! 시력검사프로그램입니다.\n")

print("[#] 프로그램 사용시 주의사항을 알려드리겠습니다")

print(" \*거리는 화면에서 3M 떨어져 주시기 바랍니다.")

print(" \*화면에서 나오는 숫자를 입력해주시면 됩니다.")

print(" \*도수검사의 기회는 3번이 주어지며, 3번이상 틀리실 경우, 도수가 결정됩니다.")

print(" \*눈 교정을 하셨으면 인경이나 렌즈등을 착용하신 상태에서 검사하셔도 됩니다.")

print(" \*왼쪽 -> 오른쪽 또는 오른쪽 -> 왼쪽순으로 검사를 진행하셔도 됩니다.")

print(" \*무선 키보드를 이용하시면 보다 편리하게 시험을 진행하실 수 있습니다.\n")

print("[#] 시작 전 기본적인 정보들을 적어주세요.")

name = input("[\*] 이름 : ")

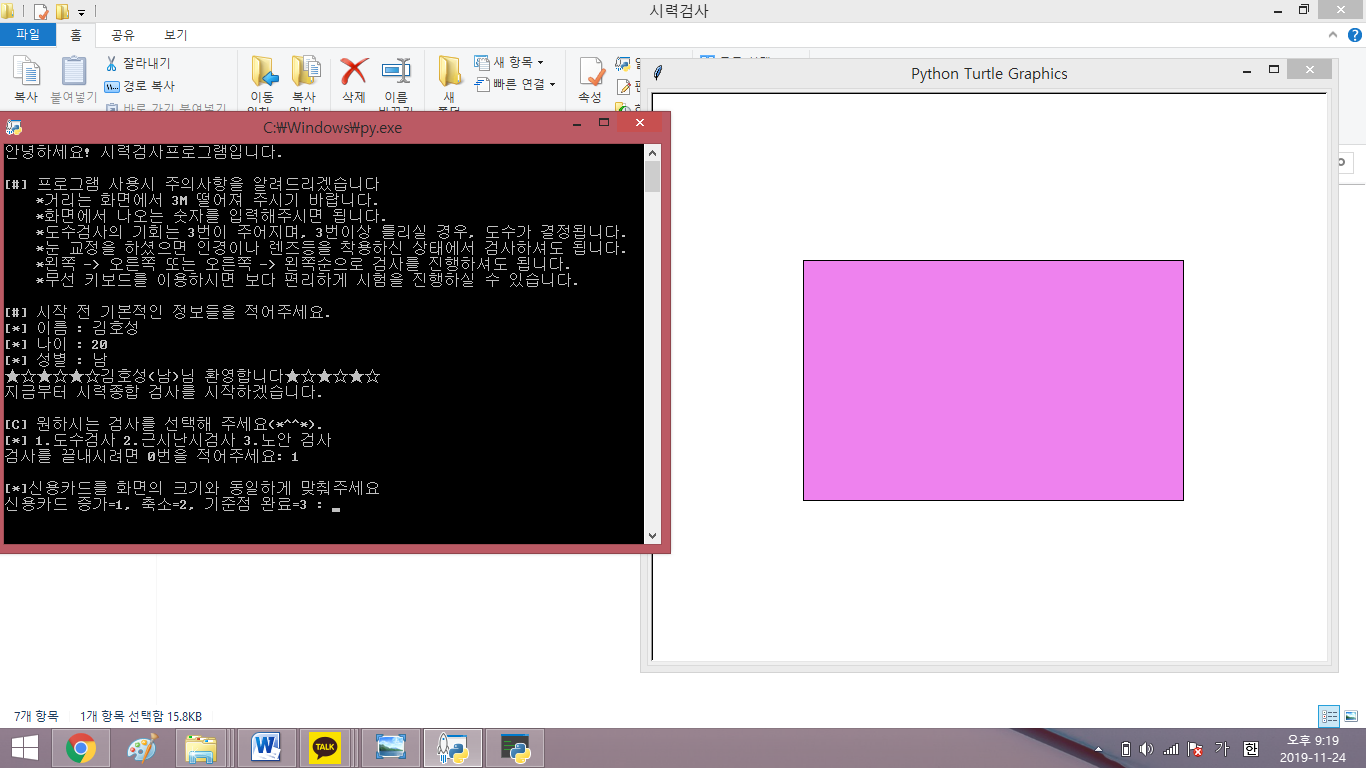
age = input("[\*] 나이 : ")

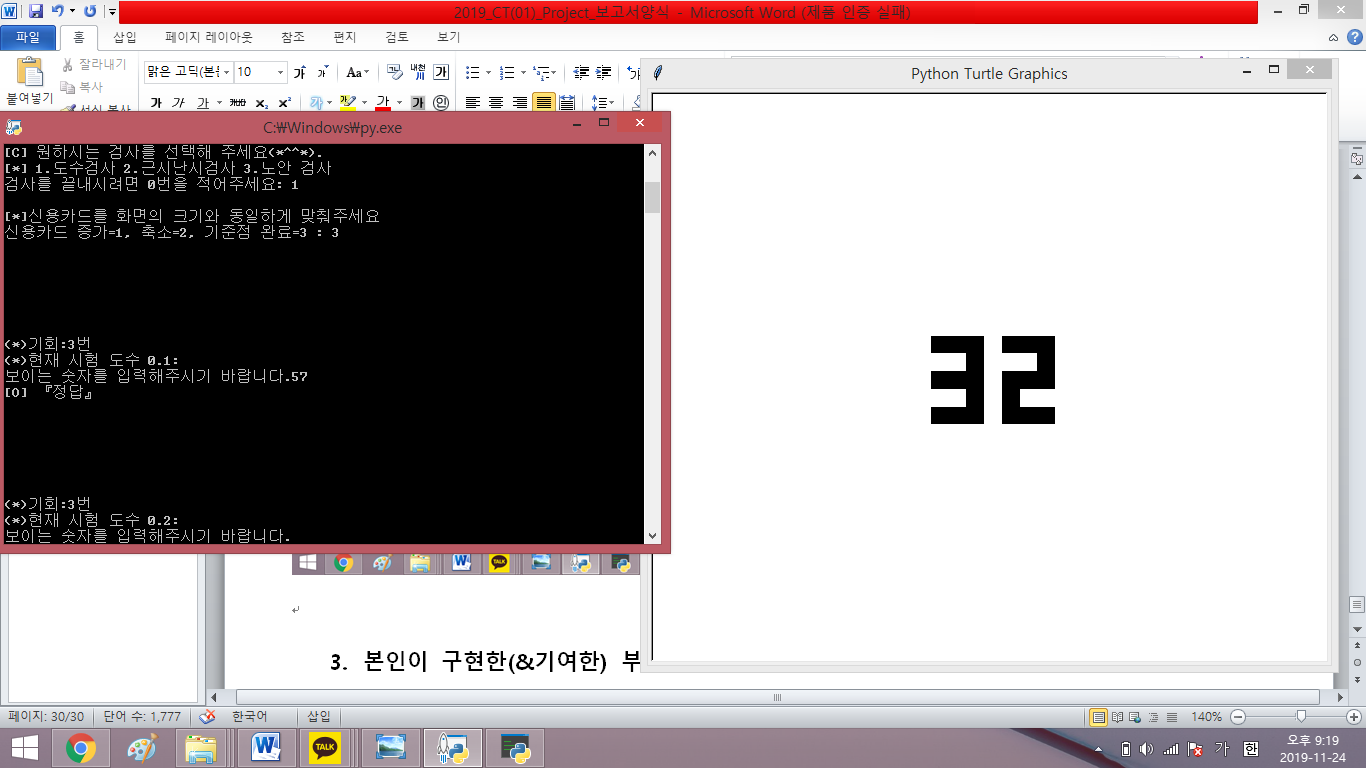
gender = input("[\*] 성별 : ")

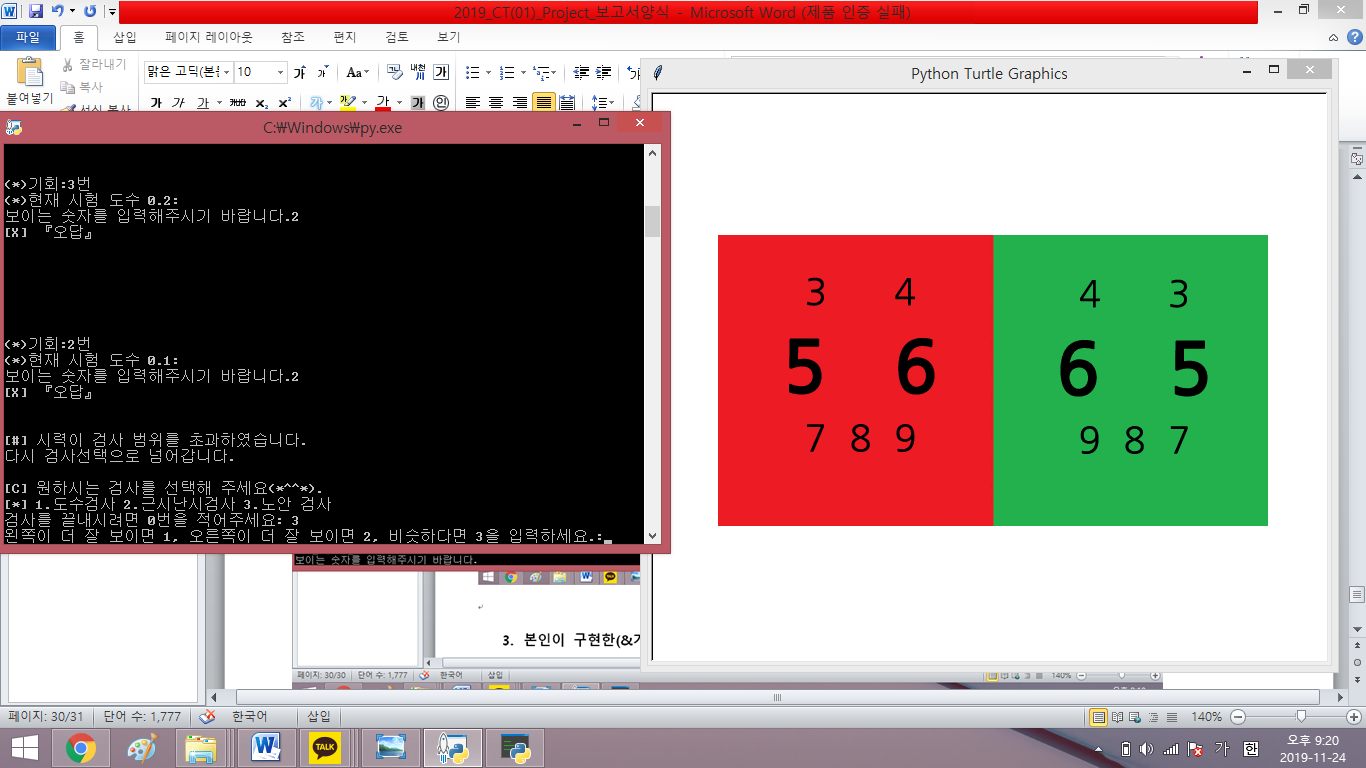
print("★☆★☆★☆{name}({gender})님 환영합니다★☆★☆★☆\n지금부터 시력종합 검사를 시작하겠습니다.".format(name=name, gender=gender))

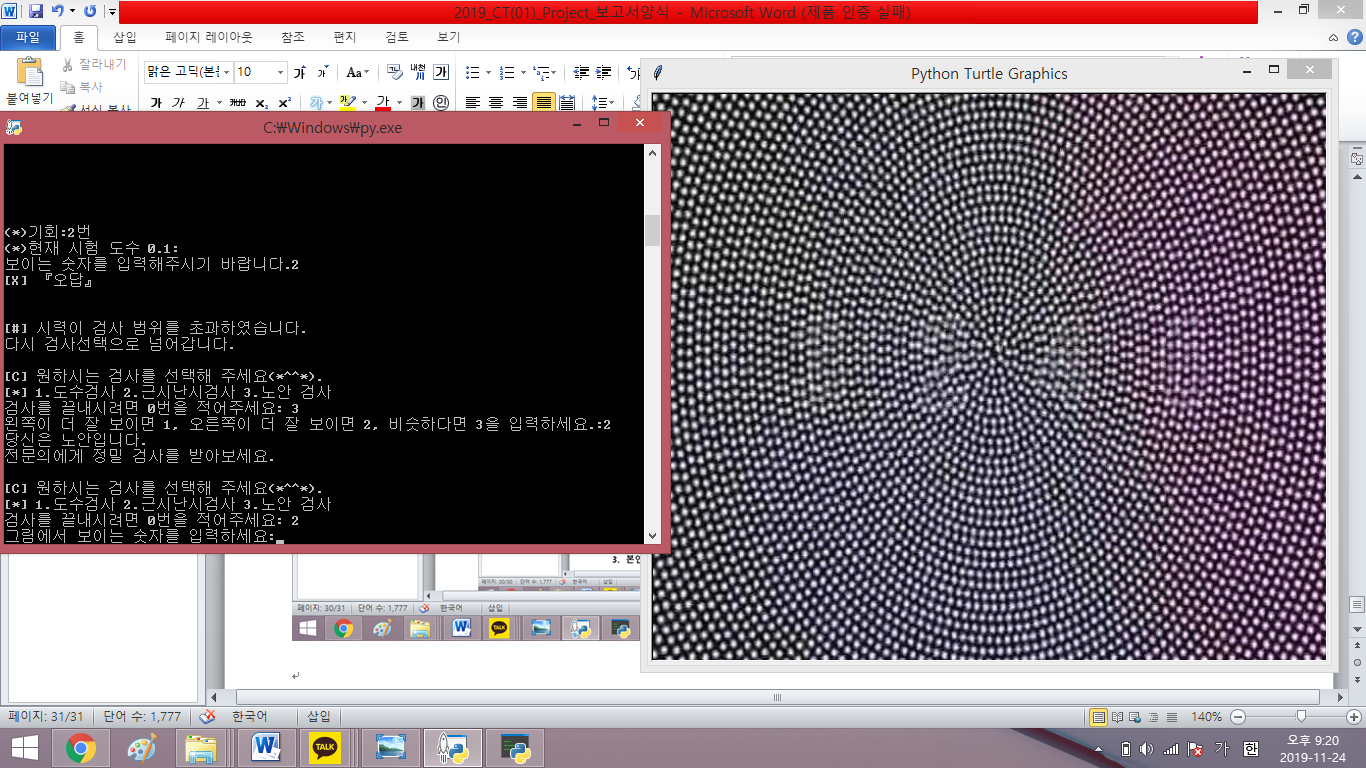
select()

결과

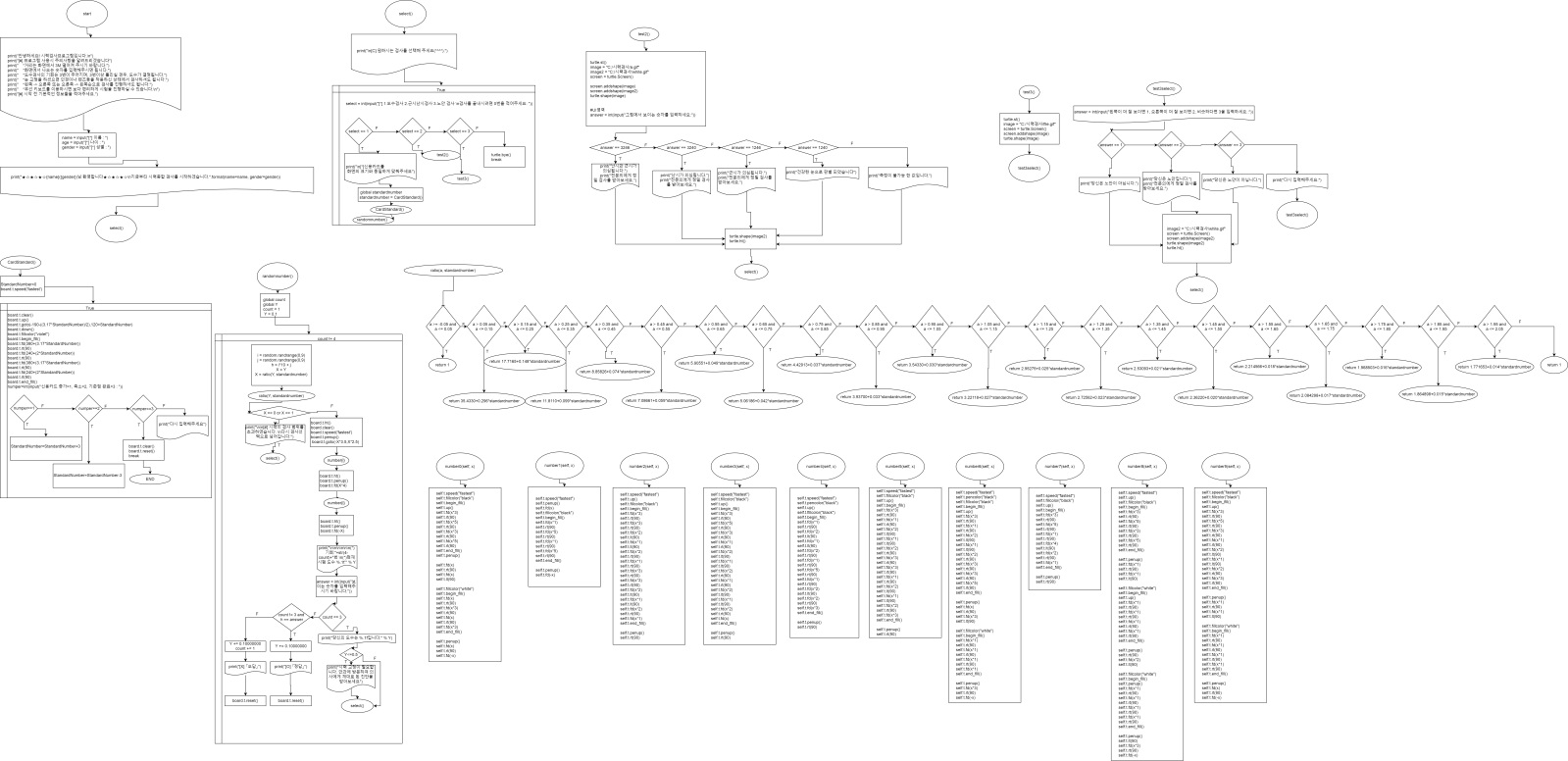








프로그램에 대한 설명(+플로우차트)



**-처음 UI를 시작으로 기본정보를 입력 받은 후 select함수로 넘어감.**

**-select함수에서 1을 받으면 도수 검사, 2를 받으면 난시 및 근시 검사 3을 받으면 노안 검사 그 외의 값을 받으면 프로그램을 종료하게끔 코딩함.**

**-도수 검사 실행시 CardStandard함수가 먼저 실행 됨. CardStandard함수에서 모니터에 카드를 댄 후 비율을 맞추어 해상도 마다 크기의 오차를 해결 함.**

**-해상도의 문제점을 해결한 후 randomnumber함수가 실행됨. 우선 전역변수인 count와 Y(도수)를 불러오며 지역변수 i와 j에 0~9사이의 난수 값을 넣고, h값에 10\*i+j라는 값을 저장함.**

**-그 후, randomnumber함수에서 ratio라는 현재 도수에 맞는 출력되는 숫자의 크기를 정해주는 함수를 불러와 출력해야 하는 숫자의 크기를 정해주고, number0~9함수에 인자 값으로 비율과 i 또는 j값을 넣어주어 무작위 숫자를 출력하게 함.**

**-사용자가 입력한 값인 answer값과 h값을 비교하여 정답일 경우 전역변수 Y값에 +0.1 오답일경우 -0.1을 대입하고 반복하여 count가 3이 되었을 때 만약 도수가 0.5이하일 경우 전문의 상담이 필요하다고 출력 후 도수 출력, 아니라면 도수만 출력하게 함.**

**-또, 만약 검사가능 범위인 0.1~2.0을 넘어갈 경우 검사범위를 넘어갔다는 문구를 출력하게 함.**

**-검사범위를 넘어가거나 count가 3이 되었을 때 select함수가 실행되게 함.**

**-select함수에 2라는 값이 들어왔을 경우 test2를 실행해 난시 및 근시검사에 맞는 그림파일을 출력하게 하였으며, 사용자가 입력한 값에 따라 3246일 경우 난시 및 근시 3240일 경우 난시 1246일 경우 근시 1240일 경우 정상인 눈 그 외의 값을 입력했을 경우 측정이 불가능하다고 출력하게 하고, select함수가 실행되게 함.**

**-select함수에 3이라는 값이 들어왔을 경우 test3을 실행해 노안 검사에 맞는 그림파일을 출력하게 함.**

**-test3에서 그림 출력 후 test3select함수를 실행 해 붉은 배경이 더 잘 보이거나 비슷하다면 노안이 아니라고 출력을 초록 배경이 더 잘 보인다고 했을 경우에는 노안이라고 출력하고 그 외의 값을 입력했을 경우 재입력 하게 함.**

**-재입력 하는 경우가 아닐 때는 다시 select함수가 실행되게 함.**