Week 3, Examples 1

#1.py

#This program teaches you about functions in a very simple way

#Think of program main() as the boss/head of operations.
#The boss needs to do some work .... and we will use a VERY simple example
#to represent the work

#He wants to add all the odd numbers between 1 and 100 (call this sum s_odd)
#and he also
#wants to add all the even numbers between 101 and 200 (call this sum s_even)

#So he asks his first helper Larry() to get s_odd, and he asks
#his second helper Moe() to get s_even

#Thus, Larry() is a function, and Moe() is another function
#Both are called by the boss (i.e., main()) to do work

#It turns out that Larry() and Moe() will each call yet another
#function called even() to help them. So there is no restriction on
#how many levels you can go with function calls.

def even(n):
    #tells if n is an odd number or an even number

    #n is a "parameter". It comes is an an input that even() can use

    if(n%2 == 0):
        #if there is no remainder when dividing by 2
        return(1) #even() returns the value 1 to caller
    else:
        return(0) #otherwise even() returns 0

def Larry(low,high):

    #Larry's inputs or parameters are low and high
    #Notice that Larry has get help himself from function even()

    sum = 0
    for i in range(low,high+1,1):
        if(even(i)==0): #if it's an odd number, increment sum
            sum = sum+i
#Larry returns sum to his caller. He has done his work
   return(sum)

def Moe(low,high):
#Moe's inputs or parameters are low and high
#Notice that Moe has to get help too from function even()
   sum = 0
   for i in range(low,high+1,1):
      if(even(i)==1): #if it's an even number increment sum
         sum = sum + i

#Moe returns sum to his caller. He has done his work.
   return(sum)

def main():
   s_odd = Larry(1,100)
   s_even = Moe(101,200)
   print(" Answer = ",s_even + s_odd)

#__________________________________________________________________________

#2.py

#Curly() tells the boss that he can do both Larry's job and Moe's job
#and he can give the boss both the values that he wants
#So the boss agrees and just uses Curly. Now we know how to make Curly()
#do the work, but we want to show how Curly returns both values to the
#boss

def even(n): #tells if n is an odd number or an even number
   n is a "parameter". It comes is an an input that even() can use
   if(n%2 == 0): #if there is no remainder when dividing by 2
      return(1) #even() returns the value 1 to caller
   else:
      return(0) #otherwise even() returns 0

def Curly(Larry_low,Larry_high,Moe_low,Moe_high):
# Curly needs both Larry's and Moe's inputs since he is doing all the work

s_odd = 0
for i in range(Larry_low, Larry_high+1, 1):
    if (even(i) == 0):  # if it's an odd number, increment sum
        s_odd = s_odd + i  # You can also say s_odd += i to do this

s_even = 0
for i in range(Moe_low, Moe_high+1, 1):
    if (even(i) == 1):  # if it's an even number increment sum
        s_even = s_even + i  # You can also say s_even += i to do this

return (s_odd, s_even)  # IMPORTANT: observe how Curly() returns more than
# one value to the caller when he is called

def main():

    s_odd, s_even = Curly(1,100,101,200)  # Curly() gives main two values
    print(" Answer = ", s_even + s_odd)

# 3.py

# Graphing stock price using candlesticks. We will only do the axes portion
# now so we can see how to draw axes and make labels. Later will read in input
# data and actually draw prices in the graph

def wait():
    dummy = input(" ")

def getwindow(text, sizex, sizey):

    # this is an example of a "function"
    # think of it as a helper you call to do some work for you

    # You'll see that getwindow does not have anything to work with unless
    # whoever calls it also gives it something to work with. These are
    # "parameters" of the function, or variables that act as placeholders.
win = GraphWin(text, sizex, sizey)

#win is "local" to this function getwindow(). Whoever called getwindow() 
does not know this variable and is asking for its value, so it can 
reach 
#the window that getwindow() creates

#So getwindow() gives this value to the caller via the "return" 
statement 
#It finishes doing its job and RETURNS THE VARIABLE's VALUE

return(win)

def label_yaxis(w, x, start, stop, step):

#Put little lines and labels next to those lines on the y-axis

for i in range(start, stop, step):
    if (i > start):
        lin = Line(Point(x-0.10, i), Point(x+0.10, i))
        lin.draw(w)

    t = Text(Point(x-0.25, i), str(i))  #convert number i to text
    #by asking str() for help
    t.draw(w)

#between two labeled points, add a little midpoint line without a label

#this if-statement prevents a little mid-point line from printing
#above the blue y-axis line

if (i < (stop-25)):
    midpt = Line(Point(x-0.05, i+step/2), Point(x+0.05, i+step/2))
    midpt.draw(w)


from graphics import *

def main():

    #get window()

    w = getwindow("GOOG price-chart", 1600, 800)

    #inside this window our chart has to fit someplace

    #let the bottom left corner of the window be at (xl,yl) and the
    #top right corner be at (xh,yh)
#xl(i.e.,xlow), xh, yl and yh will b our new coordinate system

xl = -3
xh = 15
yl = 450
yh = 600

#inside this window our chart has to fit someplace, and we need
#an x-axis and a y-axis

#let the bottom left corner of the x-y graph be at (0,475) and the
#top right corner be at (11,575). So we will have space for 10 data
#points on the GOOG graph. We can easily make this much larger later.

w.setCoords(xl,yl,xh,yh) #remember (x,y) low point, (x,y) high point

# now draw both axes
# the bottom left corner, i.e., (0,0) will be at (xl+3,yl+25)

horz = Line(Point(xl+3,yl+25),Point(xl+14,yl+25))
horz.setOutline("blue3")
horz.setWidth(3)
vert = Line(Point(xl+3,yl+25),Point(xl+3,yh-25))
vert.setOutline("blue3")
vert.setWidth(3)

vert.draw(w)
horz.draw(w)

#remember that (xl+3,yl+25) is the (0,0) on the axes, and
#at the top right of the axes system is the high value (xl+14,yh-25)

#you don't see the high value, but knowing it's there makes you
#think about the square inside which you will draw your graph

#----------------------------------------------------------------------
#now put labels for 11 days on the x-axis.We'll print a tiny vertical
#line at every unit on the x-axis, but use an "if-statement" to skip the
#origin because we really don't need such a line there

for i in range(0,11,1):
    if(i > 0):
        lin = Line(Point(xl+3+i,yl+25+1),Point(xl+3+i,yl+25-1))
        lin.draw(w)

t = Text(Point(xl+3+i-0.05,yl+25-6),str(i)) #convert number i to text
# by asking str() for 

t.draw(w)

# Now instead of doing the y-axis labeling in main() itself, as for x-axis, 
why
# don't we ask a function to do this for us? Inside the function will
# be code very much like what we did for the x-axis
# We have to give the function parameters to work with

# We'll give it the window w, the x-axis, and the y-axis values at which
# we want little label lines and label values

# Remember that (xl+3,yl+25) is the (0,0) of our axes
# If you look inside the code for this function you will see what it does

label_yaxis(w,xl+3,yl+25,yh,25) # yl+25 because we start label at 475

# Notice how, by calling some function to do a clear piece of work, you
# can farm out such tasks cleanly, and reduce clutter in the main program
#
# 4.py

#This is the same code from last week's lecture. It shows us a way to
#use random numbers to estimate pi. The only thing different here is that we
#will show what is happening graphically, since we are now learning to use
#the graphics library

#We want n random numbers (integers) from the interval [0,100000]

import random

def rand(n):
    for i in range(n):
        r = random.randrange(1,100000)
        print("random number ",i+1," : ",r)

def nrand():  # this is just rand normalized, result is in (0,1)
    # we'll omit the print statement to avoid clutter
    # and we'll just return one random number instead of n
    r = random.randrange(1,100000)/100000
    return(r)
# Let's estimate the value of pi using random numbers

# see http://www.coe.utah.edu/~hodgson/Monte_Carlo.html

# 1. Draw a circle of radius 1, centered at the origin
# 2. Focus only on first quadrant, draw a square of side 1 containing
# part of circle in quadrant 1
# 3. Area of this part of circle is pi/4
# 4. Area of square is 1

# 5. q = Area of this part of circle / area of square = pi/4

# 5. Throw darts (generate random points) at square
# 6. Estimate r = number of darts falling in circle part / total # of darts
# 7. Because r = pi/4, we get pi = 4*r

import math
import time
from graphics import *

def throw_a_dart(w):
    xpt = nrand()  # x coordinate of dart
    ypt = nrand()  # y coordinate of dart

    if ((xpt**2 + ypt**2) <= 1):
        inside = 1
        c = Circle(Point(xpt, ypt), 0.002)
        c.setFill("blue")
        c.draw(w)
    else:
        inside = 0
        c = Circle(Point(xpt, ypt), 0.003)
        c.setFill("red")
        c.draw(w)

    return(inside)

def pi(w, x, y, n, xin, yin, xout, yout, xtot, ytot):
    in_count = 0  # number of darts falling inside circle part
    out_count = 0
    piflag = 0  # to control undraw() for pi

    for i in range(1, n+1, 1):
        success = throw_a_dart(w)

        if success:
            in_count += 1
        else:
            out_count += 1

    pi = 4 * in_count / (in_count + out_count)
    return(pi)
if (success > 0):
    in_count = in_count + 1
    if (in_count > 1):# if we drew a number before, undraw it
        tin.undraw()
    tin = Text(Point(xin+0.20,yin),str(in_count))
    tin.setSize(36)
    tin.setStyle("bold")
    tin.setTextColor("blue")
    tin.draw(w)
    time.sleep(0.01)
else:
    out_count = out_count + 1
    if (out_count > 1):# if we drew a number before, undraw it
        tout.undraw()
    tout = Text(Point(xout+0.20,yout),str(out_count))
    tout.setSize(36)
    tout.setStyle("bold")
    tout.setTextColor("red")
    tout.draw(w)
    time.sleep(0.01)

# the total count and print below is part of the loop but
# not part of the if-else statement (notice the indentation)

# the statements begins at the same place that "where" and "if"
# and "else" begin

total = in_count + out_count
    if (total > 1):# if we drew a number before, undraw it
        ttot.undraw()
    ttot = Text(Point(xtot+0.20,ytot),str(total))
    ttot.setSize(36)
    ttot.setStyle("bold")
    ttot.setTextColor("black")
    ttot.draw(w)
    time.sleep(0.01)

if (piflag == 1):# if we drew a number before, undraw it
    tpi.undraw()

r = in_count/total
pi_est = 4*r
piflag = 1
    tpi = Text(Point(xtot+0.7,ytot),str(pi_est))
    tpi.setSize(36)
    tpi.setStyle("bold")
tpi.setTextColor("blue")
tpi.draw(w)
time.sleep(0.01)

#Note: Later we will learn how to control how many digits
#of pi are printed after the decimal point. Since we are
#not controlling this, Python prints whatever it computes
#and omits the last string of 0's

return(pi_est)

def draw_dartboard(w,x,y):
    center = Point(x,y)
c = Circle(center,1)
c.setOutline("blue")
c.setWidth(3)
c.draw(w)

    #Notice that the graphics lib allowed us to draw a circle even though
    #much of the circle was drawn outside the graphics window w and thus
    #cannot be seen!

    xaxis = Line(Point(x,y),Point(x+1,y))
yaxis = Line(Point(x,y),Point(x,y+1))

    xaxis.setOutline("red")
xaxis.setWidth(3)
xaxis.draw(w)

    yaxis.setOutline("red")
yaxis.setWidth(3)
yaxis.draw(w)

    #Now complete the box, so that we have a rectangle (really a square)
    #to throw darts at
    boxtop = Line(Point(x,y+1),Point(x+1,y+1))
boxrightside = Line(Point(x+1,y+1),Point(x+1,y))

    boxtop.setOutline("red")
boxtop.setWidth(3)
boxtop.draw(w)

    boxrightside.setOutline("red")
boxrightside.setWidth(3)
boxrightside.draw(w)
def main():

    win = GraphWin("Simulation: Estimate pi by throwing darts",800,800)

    # (xl,yl) and (xh,yh) will define our new coordinate system

    xl = -0.2
    xh = 1.2
    yl = -0.2
    yh = 1.2

    win.setCoords(xl,yl,xh,yh)

    # The (0,0) of the (x,y) we will draw is at (xl+0.2,yl+0.2)

    x = xl + 0.2
    y = yl + 0.2

    draw_dartboard(win,x,y)

    # Now we'll modify the original pi program and throw one dart
    # at a time and show where it falls on the dartboard. The dartboard
    # is only the red square

    # Now throw a dart at the dartboard and show where it falls. If it
    # falls inside the circle part, colour it blue
    # or else colour it red

    inlabel = Text(Point(x,y-0.1),"INSIDE = ")
inlabel.setSize(25)
inlabel.setTextColor("blue")
inlabel.draw(win)

    outlabel = Text(Point(x+0.5,y-0.1),"OUTSIDE = ")
outlabel.setSize(25)
outlabel.setTextColor("red")
outlabel.draw(win)

    totlabel = Text(Point(x,yh-0.1),"TOTAL = ")
totlabel.setSize(25)
totlabel.setTextColor("black")
totlabel.draw(win)
n = eval(input("How many darts will you throw? "))

est = pi(win,x,y,n,x,y-0.1,x+0.5,y-0.1,x,yh-0.1)

#Normally, we will not pass the same parameter "x" multiple times, but since we are just focusing on how to print/draw things in specific places in the graphics window, we will overlook this bit of inefficiency

#Our goal here was to give the function pi() places where it needs to put labels and values in the graphics screen

print(" Estimated value of pi: ",est)

#5.py

#Here we show how the graphics window can recognize your mouse clicks and determine where they occur. It can do things once it knows the points.

#We will open a graphics window and click the mouse inside that window

#Python will capture the point at which you clicked the mouse

#Since it now knows that point, you can ask it to do various things, such as drawing with the help of that point

#So let's draw some strange figures that connect lines

# See if you can use this program to accurately draw some clever or famous person such as a university administrator or politician

from graphics import *

def main():

    w = GraphWin("Fooling around with clicks and lines",800,800)

    w.setCoords(0.0,0.0,10.0,10.0) #changing coordinate system

    n = eval(input("How many points do you want to connect?"))

    msg = Text(Point(2.0,0.5),"Click on one point at a time ")
    msg.setSize(20)
    msg.draw(w)
while (n > 0): #this is the first time you are seeing a "while-loop"
    #it simply says to repeat execution of all the
    #statements in here as long as n is greater than 0
    
    p1 = w.getMouse()
    savep1 = p1  #save this first point. We'll need to draw a line from
    #this point
    #to the very last point at the end, to complete the figure.
    c = Circle(p1,0.05)
    c.setFill("red")
    c.setOutline("blue")
    c.draw(w)

    for i in range(1,n):
        p2 = w.getMouse()
        c = Circle(p2,0.05)
        c.setFill("red")
        c.setOutline("red")
        c.draw(w)

    #now draw a line from p1 to p2 while we have p1 and p2
    lin = Line(p1,p2)
    lin.setWidth(3)
    lin.setFill("blue")
    lin.draw(w)
    
    p1 = p2  #after drawing p2, rename p2 as p1. Why? Because on
    #the next iteration of the for-loop we'll get a new
    #point and call that p2. This will repeat until the loop is
    done

# now all the points are drawn and the for-loop is done. All we need
# to do is to connect the last point (now called p1 or p2, since p2 is
# unchanged) to the first point saved, i.e., savep1

    lin = Line(p2,savep1)
    lin.setWidth(3)
    lin.setFill("blue")
    lin.draw(w)

# now we have exhausted the n points that were input. Perhaps you want
# to draw another figure in the same window. Why not? We are inside a
# a while loop, and so can continue as long as we want. Let's input another
# value of n and continue. If you input 0 for n's value, we will exit the
# while loop and the program will terminate
n = eval(input("How many points do you want to connect now?"))

# the above statement is the last statement in the while loop
# if you enter a value of 0 for n, the while loop condition will
# be checked and the next iteration will not start. Instead the
# while-loop will be exited, and since there is no other code
# below, the while loop will terminate
#

#6.py

#Earlier to estimate pi, we used the Python prompt to enter the number of
data points to simulate

#Here will show to to open a graphics window which will catch your mouse
clicks, accept textual input and also return output in the same window.

#We want n random numbers (integers) from the interval [0,100000]

import random

def rand(n):
    for i in range(n):
        r = random.randrange(1,100000)
        print("random number ",i+1,": ",r)

def nrand():  # this is just rand normalized, result is in (0,1)
    # we'll omit the print statement to avoid clutter
    # and we'll just return one random number instead of n
    r = random.randrange(1,100000)/100000
    return(r)

#Let's estimate the value of pi using random numbers

#see http://www.coe.utah.edu/~hodgson/Monte_Carlo.html

# 1. Draw a circle of radius 1, centered at the origin
# 2. Focus only on first quadrant, draw a square of side 1 containing
#    part of circle in quadrant 1
# 3. Area of this part of circle is pi/4
# 4. Area of square is 1

# 5. q = Area of this part of circle / area of square = pi/4
# 5. Throw darts (generate random points) at square
# 6. Estimate \( r = \frac{\text{number of darts falling in circle part}}{\text{total \# of darts}} \)
# 7. Because \( r = \frac{\pi}{4} \), we get \( \pi = 4r \)

```python
import math
import time
from graphics import *

def throw_a_dart(w):
    xpt = nrand()       # x coordinate of dart
    ypt = nrand()       # y coordinate of dart

    if ((xpt**2 + ypt**2) <= 1):
        inside = 1
        c = Circle(Point(xpt, ypt), 0.002)
        c.setFill("blue")
        c.draw(w)
    else:
        inside = 0
        c = Circle(Point(xpt, ypt), 0.003)
        c.setFill("red")
        c.draw(w)

    return(inside)

def pi(w, x, y, n, xin, yin, xout, yout, xtot, ytot):
    in_count = 0     # number of darts falling inside circle part
    out_count = 0
    piflag = 0       # to control undraw() for pi

    for i in range(1, n+1, 1):
        success = throw_a_dart(w)

        if (success > 0):
            in_count = in_count + 1
            if (in_count > 1): # if we drew a number before, undraw it
                tin.undraw()
                tin = Text(Point(xin+0.20, yin), str(in_count))
                tin.setSize(36)
                tin.setStyle("bold")
                tin.setTextColor("blue")
                tin.draw(w)
                time.sleep(0.01)
        else:
            out_count = out_count + 1
```

```python
if (out_count > 1):  # if we drew a number before, undraw it
    tout.undraw()
    tout = Text(Point(xout+0.20, yout), str(out_count))
    tout.setSize(36)
    tout.setStyle("bold")
    tout.setTextColor("red")
    tout.draw(w)
    time.sleep(0.01)

# the total count and print below is part of the loop but
# not part of the if-else statement (notice the indentation)
# the statements begin at the same place that "where" and "if"
# and "else" begin

    total = in_count + out_count
    if (total > 1):  # if we drew a number before, undraw it
        ttot.undraw()
        ttot = Text(Point(xtot+0.20, ytot), str(total))
        ttot.setSize(36)
        ttot.setStyle("bold")
        ttot.setTextColor("black")
        ttot.draw(w)
        time.sleep(0.01)

if (piflag == 1):  # if we drew a number before, undraw it
    tpi.undraw()

    r = in_count/total
    pi_est = 4*r
    piflag = 1
    tpi = Text(Point(xtot+0.7, ytot), str(pi_est))
    tpi.setSize(36)
    tpi.setStyle("bold")
    tpi.setTextColor("blue")
    tpi.draw(w)
    time.sleep(0.01)

    # Note: Later we will learn how to control how many digits
    # of pi are printed after the decimal point. Since we are
    # not controlling this, Python prints whatever it computes
    # and omits the last string of 0's

    return(pi_est)
```

def draw_dartboard(w,x,y):

Computer Science Courses - http://courses.cs.purdue.edu/
center = Point(x,y)
c = Circle(center,1)
c.setOutline("blue")
c.setLineWidth(3)
c.draw(w)

#Notice that the graphics lib allowed us to draw a circle even though
#much of the circle was drawn outside the graphics window w and thus
#cannot be seen!

xaxis = Line(Point(x,y),Point(x+1,y))
yaxis = Line(Point(x,y),Point(x,y+1))

xaxis.setOutline("red")
xaxis.setLineWidth(3)
xaxis.draw(w)

yaxis.setOutline("red")
yaxis.setLineWidth(3)
yaxis.draw(w)

#Now complete the box, so that we have a rectangle (really a square)
#to throw darts at

boxtop = Line(Point(x,y+1),Point(x+1,y+1))
boxrightside = Line(Point(x+1,y+1),Point(x+1,y))

boxtop.setOutline("red")
boxtop.setLineWidth(3)
boxtop.draw(w)

boxrightside.setOutline("red")
boxrightside.setLineWidth(3)
boxrightside.draw(w)

def main():

#same as the previous program for pi. Only the code in between the lines
#is newly added --- for handling textual input

# New code is between the lines below

#__________________ handling textual input in a window____________________

w = GraphWin("Input for pi estimator",400,300)
```python
w.setCoords(0,0,3,4)

Text(Point(1,3), "Number of darts: ").draw(w)
Text(Point(1,1),"Estimated value of pi: ").draw(w)

inp = Entry(Point(2,3),5)
inp.setText("0")
inp.draw(w)

outp = Text(Point(2,1),"")
outp.draw(w)

button = Text(Point(1.5,2.0),"Run simulation!")
button.draw(w)
Rectangle(Point(1,1.5),Point(2,2.5)).draw(w)

# wait for a mouse click before user types input in small box

w.getMouse()

# now get the input and then call pi() to do the work

n = eval(inp.getText())

# now we got the value of n through the graphics window instead
# of through a Python prompt.

# Create the output window and let pi() do the estimation work

#____________________ new code is done

win = GraphWin("Simulation: Estimate pi by throwing darts",800,800)

# (xl,yl) and (xh,yh) will define our new coordinate system

xl = -0.2
xh = 1.2
yl = -0.2
yh = 1.2

win.setCoords(xl,yl,xh,yh)

# The (0,0) of the (x,y) we will draw is at (xl+0.2,yl+0.2)

x = xl + 0.2
y = yl + 0.2

draw_dartboard(win,x,y)
```
# Now we'll modify the original pi program and throw one dart
# at a time and show where it falls on the dartboard. The dartboard
# is only the red square

# Now throw a dart at the dartboard and show where it falls. If it
# falls inside the circle part, colour it blue
# or else colour it red

inlabel = Text(Point(x,y-0.1),"INSIDE = ")
inlabel.setSize(25)
inlabel.setTextColor("blue")
inlabel.draw(win)

outlabel = Text(Point(x+0.5,y-0.1),"OUTSIDE = ")
outlabel.setSize(25)
outlabel.setTextColor("red")
outlabel.draw(win)

totlabel = Text(Point(x,yh-0.1),"TOTAL = ")
totlabel.setSize(25)
totlabel.setTextColor("black")
totlabel.draw(win)

est = pi(win,x,y,n,x,y-0.1,x+0.5,y-0.1,x,yh-0.1)

#Normally, we will not pass the same parameter "x" multiple
#times, but since we are just focusing on how to print/draw
#things in specific places in the graphics window, we will
#overlook this bit of inefficiency

#Our goal here was to give the function pi() places where
#it needs to put labels and values in the graphics screen,
#apart from simply estimating the value of pi

print(" Estimated value of pi: ",est)

#______________the rest of the change is
#below______________________________

# display the output (estimated value) and change the button to
# give the user a chance to exit

# if we do not want to exit, we can put all of the code in a loop
# as we shall see later

outp.setText(est)
button.setText("Quit")

#wait for click and then quit
w.getMouse()
w.close()
lthigh = Line(Point(x+.17,9-0.4),Point(x+0.3,9-0.6))
lthigh.draw(w)
lleg = Line(Point(x+0.3,9-0.6),Point(x+0.3,9-0.8))
lleg.draw(w)

rback = Line(Point(x,9-0.4),Point(x-0.25,9-0.8))
rback.draw(w)

return(lthigh,lleg,rback)

def clearbody(f,n,b):
    n.undraw()
    f.undraw()
    b.undraw()

def clearleftforward(lthigh,lleg,rback):
    lthigh.undraw()
    lleg.undraw()
    rback.undraw()

def clearrightforward(rthigh,rleg,lback):
    rthigh.undraw()
    rleg.undraw()
    lback.undraw()

#IMPORTANT: The functions used here return multiple values at the same time

from graphics import *  #get access to all of graphics.py's functions
import time

def main():
    wait()  #first get a window

t = 1
wait()

delay = 1

while (1):
    #since the condition in the while-loop is always true
    #this loop will run forever, and Clark Gable will walk
    #in circles forever
for x in range(1,12,2):
    if (x>1):
        clearrightforward(rthigh,rleg,lback)
        face,nose,body = clark(w,x) # face is more correctly his head
        lthigh,lleg,rback = leftlegforward(w,x)
        time.sleep(delay)
        clearbody(face,nose,body)
        clearleftforward(lthigh,lleg,rback)
        rthigh,rleg,lback = rightlegforward(w,x+1)
        face,nose,body = clark(w,x+1)
        time.sleep(delay)
        clearbody(face,nose,body)