**Python strftime() to get the current time**

Python strftime() function can be used along with the datetime module to fetch the current timestamp in a proper form according to the format codes.

#Eg.Time

from datetime import datetime

current\_timestamp = datetime.now()

tym = current\_timestamp.strftime("%H:%M:%S")

date = current\_timestamp.strftime("%d-%m-%Y")

print("Current Time:",tym)

print("Current Date:",date)

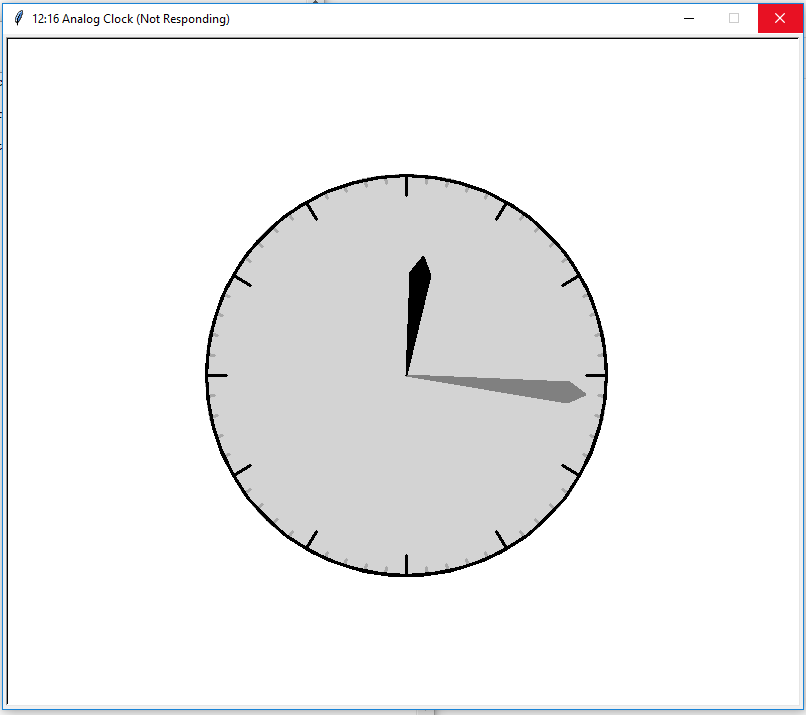


Figure.1

import turtle

import datetime

import time

def drawclock(radius,hours,minutes,length,color):

turtle.pensize(3)

turtle.penup()

turtle.goto(0,radius)

turtle.pendown()

turtle.fillcolor(color)

turtle.pencolor("darkgray")

turtle.begin\_fill()

for degree in range(0,360,360//minutes):

turtle.circle(-radius,360//minutes)

turtle.right(90)

turtle.forward(length/3)

turtle.back(length/3)

turtle.left(90)

turtle.end\_fill()

turtle.pencolor("black")

for degree in range(0,360,360//hours):

turtle.circle(-radius,360//hours)

turtle.right(90)

turtle.forward(length)

turtle.back(length)

turtle.left(90)

turtle.circle(-radius)

def drawhand(angle, radius, width, color, outline=False):

if outline:

turtle.pencolor("black")

else:

turtle.pencolor(color)

turtle.pensize(2)

turtle.penup()

turtle.home()

turtle.fillcolor(color)

turtle.begin\_fill()

turtle.left(90)

turtle.right(angle)

turtle.forward(radius)

turtle.pendown()

turtle.left(150)

turtle.forward(width)

turtle.home()

turtle.left(90)

turtle.right(angle)

turtle.penup()

turtle.forward(radius)

turtle.pendown()

turtle.right(150)

turtle.forward(width)

turtle.home()

turtle.end\_fill()

radius=200

turtle.speed(0)

turtle.hideturtle()

turtle.title("Analog Clock")

drawclock(radius,12,60,radius\*.1,"lightgray")

current\_time=datetime.datetime.now().time()

turtle.title(current\_time.strftime("%H:%M Analog Clock"))

drawhand(current\_time.minute \* 6, radius \* .9, radius // 10, "gray")

drawhand(((current\_time.hour + current\_time.minute / 60) % 12) \* 30, radius \* .6, radius // 10, "black")

while True:

new\_time=datetime.datetime.now().time()

if current\_time.minute is not new\_time.minute:

turtle.title(new\_time.strftime("%H:%M Analog Clock"))

drawhand(current\_time.minute \* 6, radius \* .9, radius // 10, "lightgray")

drawhand(((current\_time.hour + current\_time.minute / 60) % 12) \* 30, radius \* .6, radius // 10, "lightgray")

drawhand(new\_time.minute \* 6, radius \* .9, radius // 10, "gray")

drawhand(((new\_time.hour + new\_time.minute / 60) % 12) \* 30, radius \* .6, radius // 10, "black")

current\_time=new\_time

time.sleep(60)

**Old Question**

#Q4\_1.

from turtle import \*

class Clock\_face:

def \_\_init\_\_(self) -> None:

self.radius = 100

self.hourmaker\_len = 10

self.hour\_hand\_len = 45

self.min\_hand\_len = 60

self.sec\_hand\_len = 65

self.dot\_size = 7

def draw\_hand(self, length, angle):

pu()

goto(0, 0)

pd()

seth(90)

rt(angle)

fd(length)

bk(length + 10)

def draw(self, hour, minute, second):

pu()

goto(0, -self.radius)

pd()

pensize(5)

circle(self.radius)

# draw the hour makers

pensize(3)

self.inner\_radius = 85

for \_ in range(12):

pu()

goto(0, 0)

fd(self.inner\_radius - self.hourmaker\_len)

pd()

# draw a small hourmaker line

fd(self.hourmaker\_len)

lt(30)

# draw the hour hand

# The hour hand of a normal analogue clock turns 360° in 12 hours (720 minutes) or 0.5° per minute

self.hour\_hand\_angle = 0.5 \* ((60 \* hour) + minute) # formula

self.draw\_hand(self.hour\_hand\_len, self.hour\_hand\_angle)

# draw the minute hand

# The minute hand rotates through 360° in 60 minutes or 6° per minute.

self.min\_hand\_angle = 6 \* minute # formula

self.draw\_hand(self.min\_hand\_len, self.min\_hand\_angle)

# draw the second hand

# The second hand also rotates through 360° in 60 seconds or 6° per second.

pencolor('red')

pensize(2)

self.sec\_hand\_angle = 6 \* second # formula

self.draw\_hand(self.sec\_hand\_len, self.sec\_hand\_angle)

# draw a dot at the center

pu()

goto(0, 0)

pd()

dot(self.dot\_size)

hideturtle()

done()

def main():

clock = Clock\_face()

clock.draw(hour=1, minute=53, second=34)

if \_\_name\_\_ == '\_\_main\_\_':

main()

#Q4\_2.py

from Q4\_1 import Clock\_face

from time import time

def performance(func):

def wrapper():

t1 = time()

func()

t2 = time()

print(f'It took {t2-t1} s')

return wrapper

@performance

def main():

clock = Clock\_face()

clock.draw(hour=1, minute=53, second=34)

if \_\_name\_\_ == '\_\_main\_\_':

main()

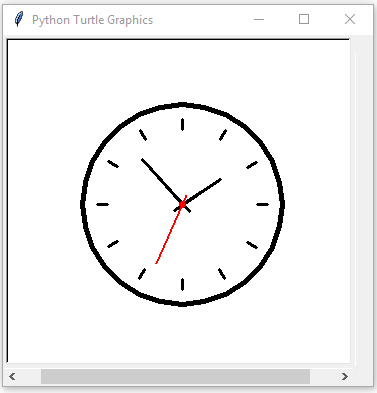


Figure.2

#Eg.ClockEg

import turtle

import time

wndw = turtle.Screen()

wndw.bgcolor("black")

wndw.setup(width=600, height=600)

wndw.title("Analogue Clock")

wndw.tracer(0)

# Create the drawing pen

pen = turtle.Turtle()

pen.hideturtle()

pen.speed(0)

pen.pensize(3)

def draw\_clock(hr, mn, sec, pen):

# Draw clock face

pen.up()

pen.goto(0, 210)

pen.setheading(180)

pen.color("green")

pen.pendown()

pen.circle(210)

# Draw hour hashes

pen.up()

pen.goto(0, 0)

pen.setheading(90)

for \_ in range(12):

pen.fd(190)

pen.pendown()

pen.fd(20)

pen.penup()

pen.goto(0, 0)

pen.rt(30)

# Draw the hands

# Each tuple in list hands describes the color, the length

# and the divisor for the angle

hands = [("white", 80, 12), ("blue", 150, 60), ("red", 110, 60)]

time\_set = (hr, mn, sec)

for hand in hands:

time\_part = time\_set[hands.index(hand)]

angle = (time\_part/hand[2])\*360

pen.penup()

pen.goto(0, 0)

pen.color(hand[0])

pen.setheading(90)

pen.rt(angle)

pen.pendown()

pen.fd(hand[1])

while True:

hr = int(time.strftime("%I"))

mn = int(time.strftime("%M"))

sec = int(time.strftime("%S"))

draw\_clock(hr, mn, sec, pen)

wndw.update()

time.sleep(1)

pen.clear()

wndw.mainloop()

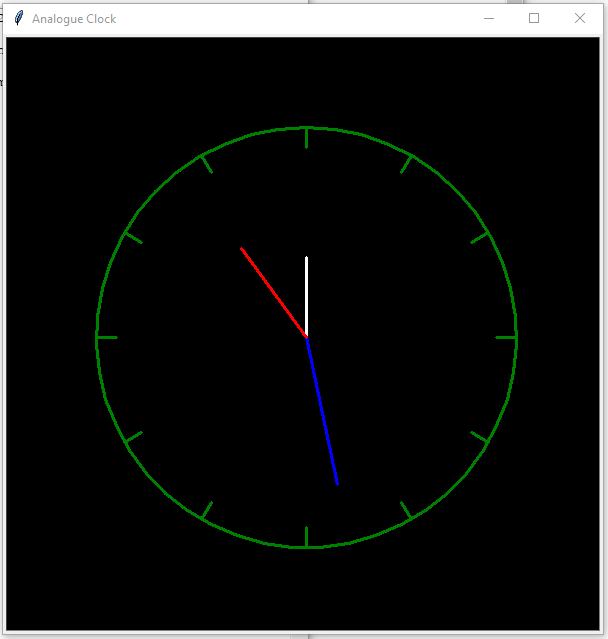


Figure.3