Python Programming

Turtle graphics II

Mariusz Dzieńkowski

Institute of Computer Science
Lublin University of Technology
m.dzienkowski@pollub.pl

Pseudorandom numbers

- The `randint(start,end)` function – returns a pseudorandom integer values from a given range
- `randint(1,5)` returns numbers from 1 to 5
- Function `randrange(5,25,3)` returns a multiple of 3 between 5 and 23, inclusive
- The `random` function is used to generate pseudo random floating point values
- The `random` takes no parameters and returns values uniformly distributed between 0 (including) and 1 (excluding)
- Each call to `randint`, `random` or `randrange` functions generates a new pseudorandom number

Drawing with Color

- `pencolor(acolor)` command – `acolor` is a colorstring or an rgb tuple
- `fillcolor(acolor)` - `acolor` is either a colorstring or an rgb tuple
- `color(pen_color,fill_color)` –set both with one function
- `begin_fill()` – mark where region to color in begins
- `begin_fill()` – mark where region to color in begins
- `end_fill()` – mark where region to color in ends

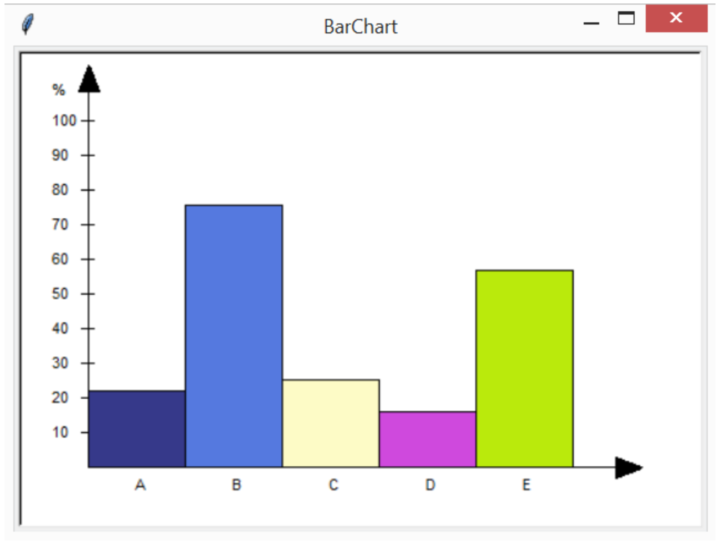
Problem 1

Bar chart - app that draws bars with random height and random color

- Outlines:
  - Import all turtle commands from the turtle library
  - Import two functions: randint and random from the random library
  - Write functions to draw X and Y axes with the scale and labels
  - Write a function to draw a bar with two parameters:

    xpos – position of a bar on X axis

    height – height of a bar
  - Write a function to draw a bar chart

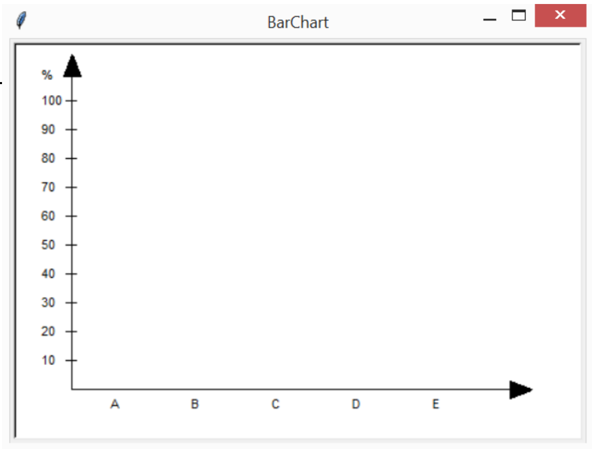


Problem 1 - solution

- Source code – drawing axes

```
from turtle import *
from random import randint,random

def axisX():
    up()
    goto(-200,-125)
    down()
    fd(400)
    lt(160)
    fillcolor("black")
    begin_fill()
    fd(20)
    lt(110)
    fd(15)
    lt(110)
    fd(20)
    end_fill()
    x=-165
    y=-125
    for i in range(5):
        up()
        goto(x,y-20)
        down()
        write(chr(65+i))
        x+=70
```

```
def axisY():
    up()
    goto(-200,-125)
    down()
    lt(70)
    fd(290)
    lt(160)
    fillcolor("black")
    begin_fill()
    fd(20)
    lt(110)
    fd(15)
    lt(110)
    fd(20)
    end_fill()
    rt(20)
    x=-200
    y=-125
    for i in range(10):
        up()
        y+=25
        goto(x-5,y)
        down()
        goto(x+5,y)
        up()
        goto(x-25,y-7)
        write((i+1)*10)
    goto(x-25,y+15)
    write('%')
```

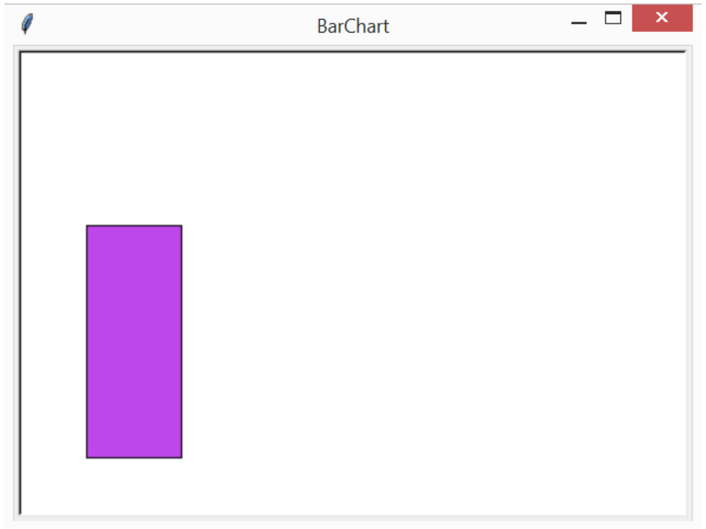


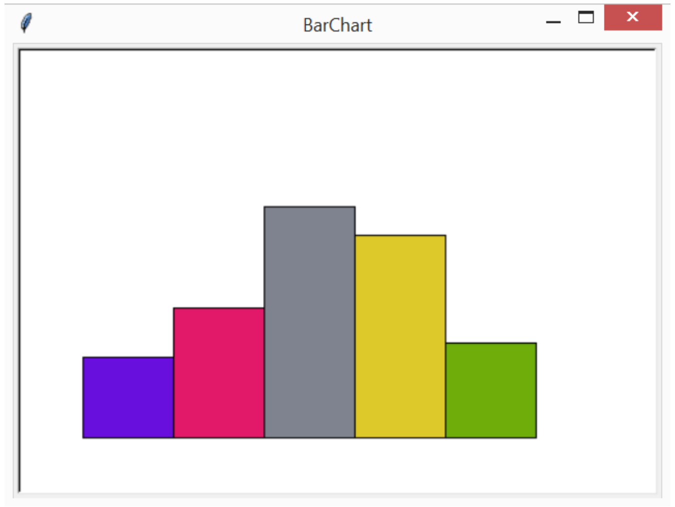
Problem 1 - solution

- Source code – drawing a bar and a barchart

```
def bar(xpos,height):
    up()
    goto(-200+xpos,-125)
    down()
    fillcolor((random(), random(), random()))
    begin_fill()
    for i in range(2):
        fd(height)
        rt(90)
        fd(70)
        rt(90)
    end_fill()
```

```
def barchart():
    setup(500,350)
    title('BarChart')
    ht()
    x=0
    axisX()
    axisY()
    for i in range(5):
        y=randint(0,250)
        bar(x,y)
        x+=70

barchart()
```





Motion turtle commands

- `forward(distance)` – move forward `distance` in current direction
- `backward(distance )` – move backward `distance` in the opposite direction
- `right(angle)` –turn right by angle units
- `left(angle)` –turn left by angle units
- `goto(x,y)` – move turtle to absolute screen position (x,y)
- `home()` – move turtle to origin (0,0), facing the default direction
- `speed(`*`speed`*`)` – set turtle drawing *`speed`* as int in range 1-10

Objects

- Everything in Python is an object (ints, strs, lists, functions, turtles, screen, etc.)
- an object can have attributes - data associated with an object
- an object can have methods - which are basically things that the object can do
- When drawing with the turtle module
  - a **Screen** object - represents the window for drawing on
  - one or more **Turtle** objects - the turtle object represents a pen

Methods

- A method is essentially a function that is associated with a particular object
- Calling a method is like calling a function - additionally before a method an object and a dot operator is required: `object.method()`

e.g.

```
import turtle            # importing the turtle module
wn = turtle.Screen()     # creating a Screen object
t1 = turtle.Turtle()     # creating at least one Turtle object
t1.forward(100)          # calling the forward() method on
                           object named t1
wn.mainloop()            # preventing the window from closing
```

Screen object methods

- Methods for calling on the Screen object:
  - `setup(width, height)` - window dimensions (default is 50% and 75% of screen)
  - `bgcolor(colorstring)` – changing the background color of a window to `colorstring`

  e.g.

```
    wn.setup(500, 500)
    wn.bgcolor("lightblue")
```

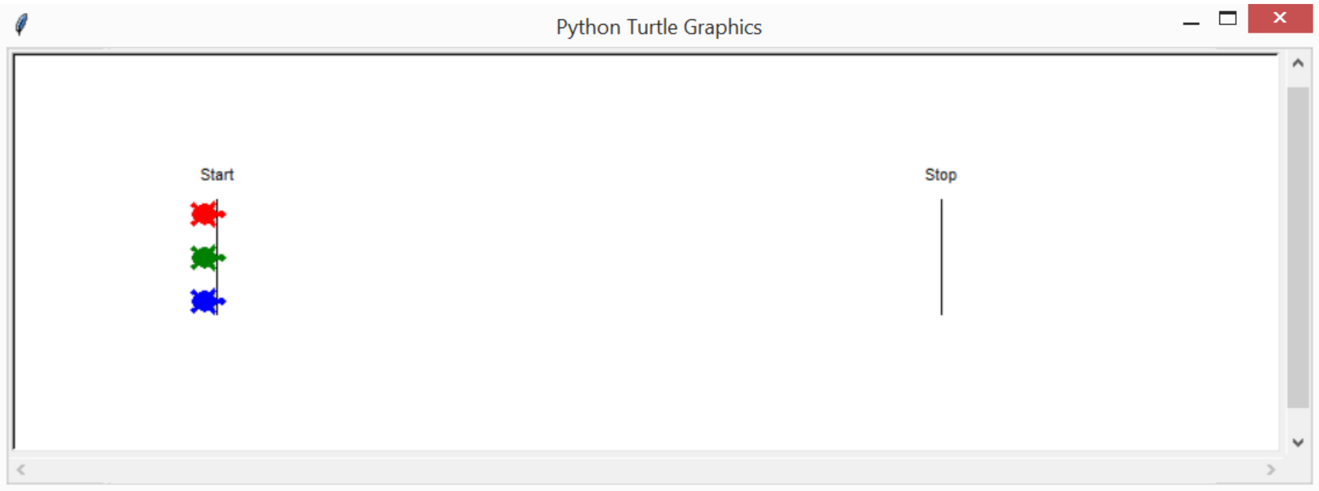
Turtle object as a parameter

```
def draw_square(t, side, c):
      t.pencolor(c)
      for i in range(4):
            t.forward(side)
            t.right(90)

t1 = Turtle()
t2 = Turtle()
draw_square(t1,100,"red")
draw_square(t2,200,'#225599')
```

Problem 2

Racing turtle game - app as a game in which 3 turtles are racing

- Step 1
  - Import the Turtle class: `from turtle import Turtle`
  - Create an instance of a Turtle object t0 for creating a race track: `t0 = Turtle()` and draw a race track consisting of start and stop lines
  - Create 3 instances of Turtle objects: t1, t2, t3 for racing
  - Use the color and shape methods to customise Turtle objects attributes
  - Send turtles t1, t2, t3 to the starting line

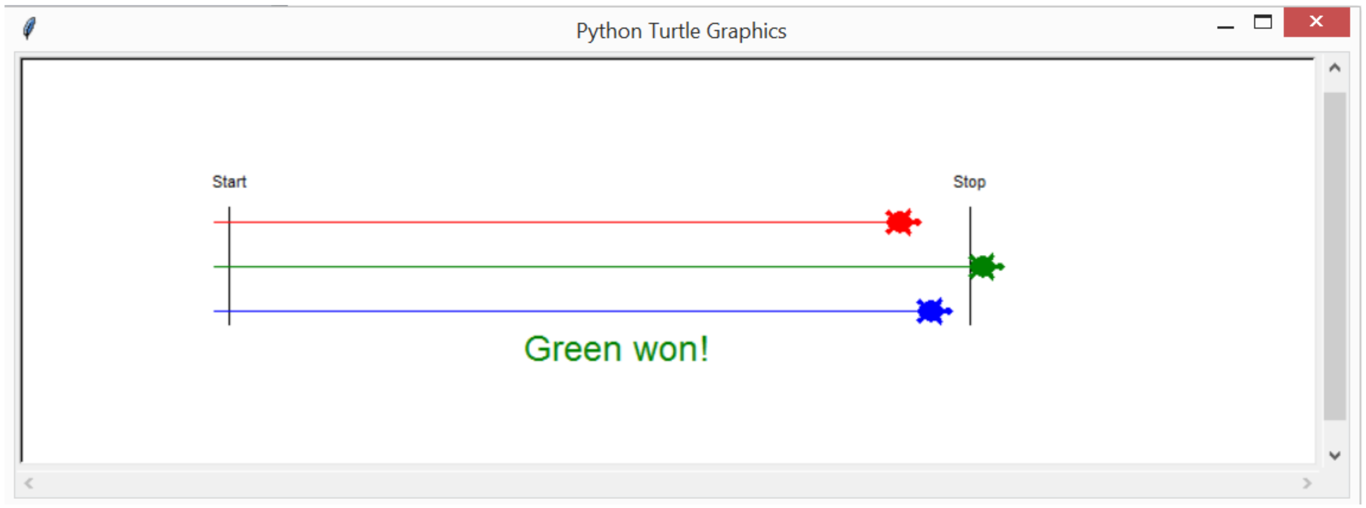


Problem 2

Racing turtle game

- Step 2
    - Use a while-loop to create a racing turtle game
    - Make the turtle race by moving a random number of steps at a time

      - use the randint function to randomize a number of steps for each turtle
    - Show information which turtle wins (use write(text, font) function)



Problem 2 - solution

- Source code

```
from turtle import *
from turtle import Turtle
from random import randint

setup(900,300)
t0=Turtle()
t1=Turtle()
t2=Turtle()
t3=Turtle()
t0.ht()
t1.ht()
t2.ht()
t3.ht()
```

```
t0.pu()
t0.goto(-310,50)
t0.write('Start')
t0.goto(-300,40)
t0.down()
t0.goto(-300,-40)

t0.pu()
t0.goto(190,50)
t0.write('Stop')
t0.goto(200,40)
t0.down()
t0.goto(200,-40)
```

```
t1.color("red")
t2.color("green")
t3.color("blue")

t1.shape('turtle')
t2.shape('turtle')
t3.shape('turtle')

t1.pu()
t1.goto(-310,30)
t1.down()
t1.st()

t2.pu()
t2.goto(-310,0)
t2.down()
t2.st()

t3.pu()
t3.goto(-310,-30)
t3.down()
t3.st()
```

Problem 2 - solution

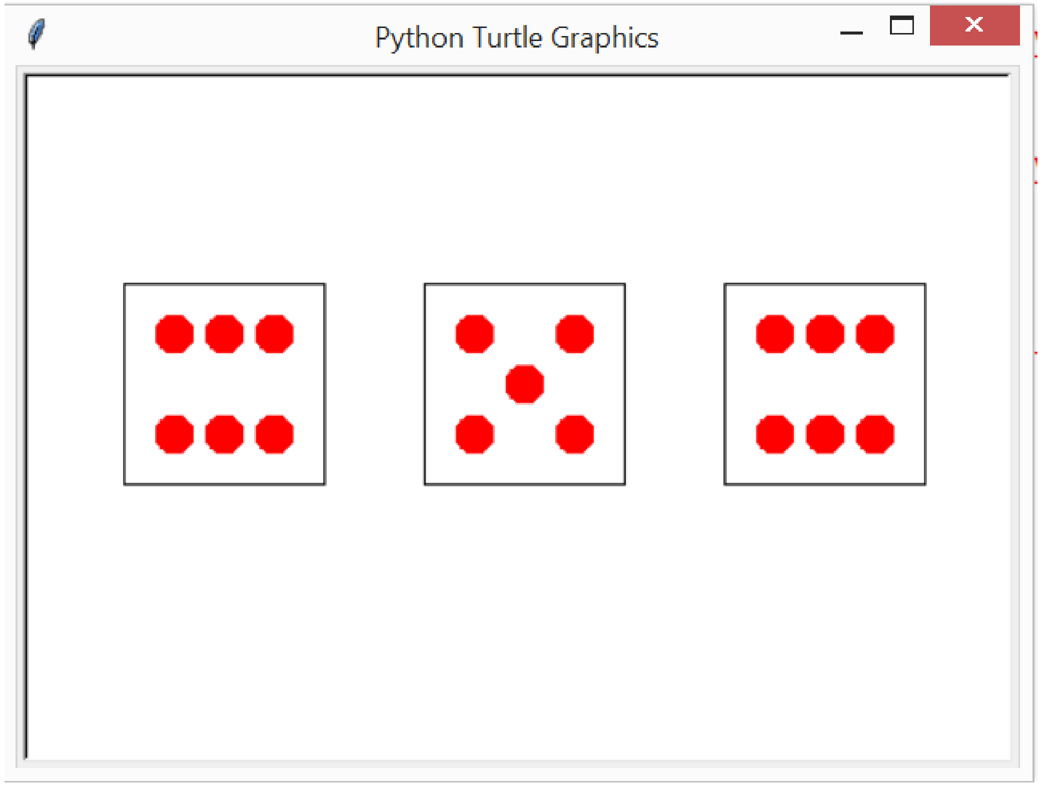
- Source code

```python
def game():
    x1=x2=x3=step=0
    while True:
        a = randint(1,30)
        b = randint(1,30)
        c = randint(1,30)
        x1 += a
        x2 += b
        x3 += c
        step += 1
        t1.fd(a)
        t2.fd(b)
        t3.fd(c)
        if x1>=500 or x2>=500 or x3>=500:
            t0.ht()
            t0.up()
            t0.goto(-100,-70)
            if x1>x2 and x1>x3:
                t0.pencolor("red")
                t0.write("Red won!",font=("Arial", 18, "normal"))
            if x2>x1 and x2>x3:
                t0.pencolor("green")
                t0.write("Green won!",font=("Arial", 18, "normal"))
            if x3>x2 and x3>x1:
                t0.pencolor("blue")
                t0.write("Blue won!",font=("Arial", 18, "normal"))
            break

game()
```

Problem 3

Throwing dices

- Write a program that simulates throwing of three dices



Problem 3

Throwing dices

- Hints
  - Use turtle’s commands for drawing dices
  - Write functions drawing faces of a dice containing spots from 1 to 6 – each of six functions should have x and y parameters as a position of bottom left corner of a square
  - Write a roll() function that returns a pseudorandom number in the range 1...6, inclusive - use the randrange function
  - Write a function for drawing a square
  - Write a main() function to simulate throwing of three dices

Problem 3 - solution

- Source code

```
from turtle import *
from random import randrange

setup(500,350)

def square(x,y):
    up()
    goto(x,y)
    down()
    ht()
    for i in range(4):
        fd(100)
        lt(90)

def r1(x,y):
    square(x,y)
    up()
    goto(x+50,y+50)
    dot(20,"red")

def r2(x,y):
    square(x,y)
    j=25
    for d in range(0,3,2):
        up()
        goto(x+j*(d+1),y+j*(3-d))
        dot(20,"red")
```

```
def r3(x,y):
    square(x,y)
    j=25
    for d in range(3):
        up()
        goto(x+j*(d+1),y+j*(3-d))
        dot(20,"red")

def r4(x,y):
    square(x,y)
    j=25
    for d in range(3,0,-2):
        for z in range(1,4,2):
            up()
            goto(x+j*d,y+j*z)
            dot(20,"red")

def r6(x,y):
    square(x,y)
    j=25
    for d in range(1,4,2):
        for z in range(3,0,-1):
            up()
            goto(x+j*z,y+j*d)
            dot(20,"red")
```

Problem 3 - solution

- Source code

```
def r5(x,y):
    square(x,y)
    j=25
    for d in range(3,0,-2):
        for z in range(1,4,2):
            up()
            goto(x+j*d,y+j*z)
            dot(20,"red")
    up()
    goto(x+50,y+50)
    dot(20,"red")

def roll():
    return randrange(1,7)
```

```
def main():
    x=-200
    y=-30
    for i in range(0,3):
        r = roll()
        if r==1:
            r1(x,y)
        elif r==2:
            r2(x,y)
        elif r==3:
            r3(x,y)
        elif r==4:
            r4(x,y)
        elif r==5:
            r5(x,y)
        else:
            r6(x,y)
        x+=150

main()
```

Ontimer

- ontimer – the Screen object function allows a function to be executed some specified time (in milliseconds) later
  `wn.ontimer(mydraw, 700)` – mydraw function is calling 700 ms later

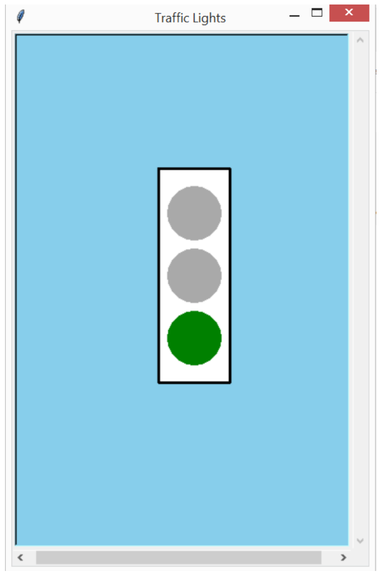
e.g.

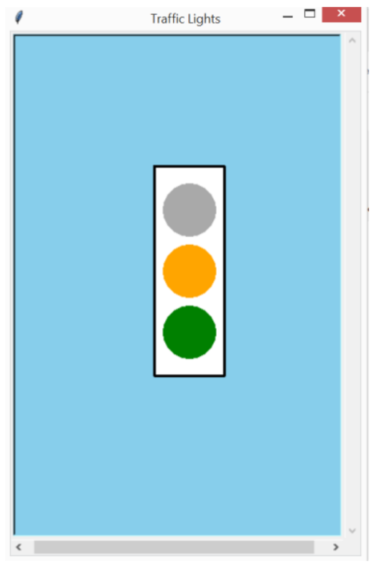
```
def fun():
      print('called')

wn.ontimer(fun, 500)
wn.ontimer(fun, 1000)
wn.ontimer(fun, 3000)
```

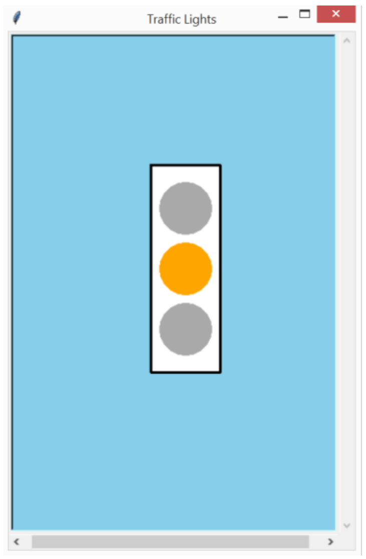
Problem 4

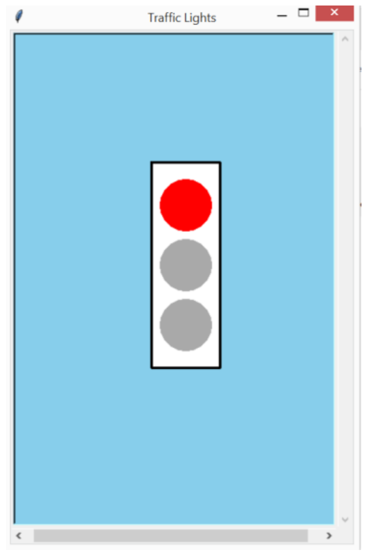
Traffic lights simulation

- Build a program that uses a turtle to simulate the traffic lights
- Hints:
  - There will be four states in a traffic light: Green (3 s), Green and Orange (1 s), Orange (1 s), and Red (2 s)
  - Determine the window size, create a playground for turtles, set the window title and the window background color





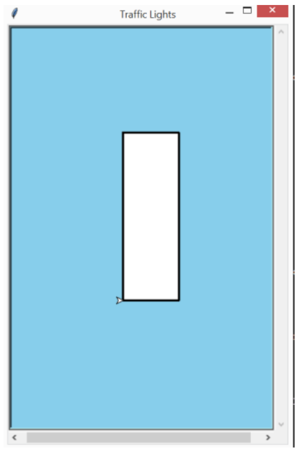
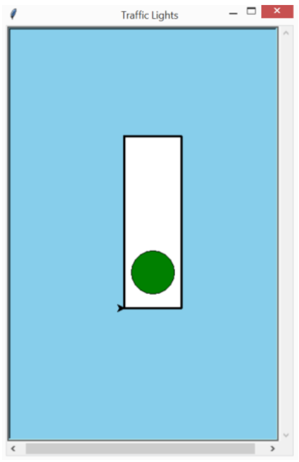
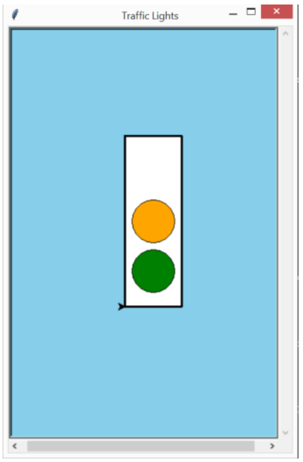
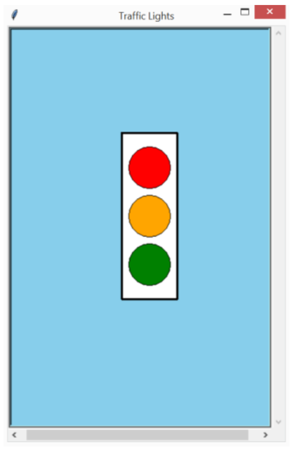




Problem 4

Traffic lights simulation

- Hints:
  - Create 3 instances of Turtle objects
  - Draw a housing to hold the traffic lights
  - Write a function to draw a circle with three parameters: t – instance of a turtle object, ht – position of a turtle where the light should be placed, col – color of the circle
  - Write a function to simulate traffic lights - set the time to change the state by using the ontimer event


Traffic Lights

Traffic Lights

Traffic Lights

Traffic Lights