WHAT WE HAVE LEARNED

Files, Sets and Dictionaries
Playing with **graphics objects**

- Canvas
- Drawable objects
- Operations on drawable objects
- Layer
- Animation
An object has **state** and can perform **actions**

We have already met some types of objects - tuples, lists, strings, files, etc.

For example, elements in a list represents a **state** of the list, and `append()` and `remove()` are **actions** of list objects

Circle: its state consists of its radius, position, depth, border, and fill color. It supports various actions to change its color, size, and position, and to perform transformation.
Object whose state can never change are called immutable. string and tuple are typical examples of immutable object in Python

Object whose state can change are called mutable. list and graphics objects are mutable

Remember that we can have more than one name for the same object. Be careful for a mutable object

sun = Circle(30)
sun.setFillColor('dark orange')
moon = sun
moon.setFillColor('wheat')
print sun.getFillColor()
from cs1graphics import *
world = Canvas(400, 400)
world.setBackgroundColor("light blue")

sun = Circle(30)
world.add(sun)

sun.setFillColor("yellow")
sun.move(10,10)

#Animation is graphics with loops:
for i in range(400):
    sun.move(1,1)
We first need to create a canvas to draw on

```
from cs1graphics import *
```

The coordinate system: x goes from 0 to 399 left-to-right, y from 0 to 299 top-to-bottom

```
canvas = Canvas(400, 300)
```
Canvas - a window upon which we draw

```python
paper = Canvas()
paper.setBackgroundColor('skyBlue')
paper.setWidth(300)
paper.setHeight(200)
paper.setTitle('My World')
```

```
paper = Canvas(300, 200, 'skyBlue', 'My World')
```
To create a drawing, we add drawable objects to the canvas:

1. Circle(radius)
2. Square(side)
3. Rectangle(width, height)
4. Polygon
5. Path
6. Text(message, font_size)
7. Image(image_filename)

*Color* is a string or an (r, g, b)-tuple

Border color:
- `obj.setBorderColor(color)`
- `obj.getBorderColor()`

Fill color:
- `obj.setFillColor(color)`
- `obj.getFillColor()`
Every object has a reference point. The location of the reference point on the canvas is set using `move(dx, dy)` and `moveTo(x, y)`.

```python
sq = Square(100)
canvas.add(sq)
sq.setFillColor('blue')
sq.setBorderColor('red')
sq.setBorderWidth(5)
sq.moveTo(200, 200)
```

**animation**
for i in range(100):
  sq.move(1, 0)
GRAPHICAL DISPLAY: DEPTH

\[ r = \text{Rectangle}(150, 75) \]
\[ \text{canvas.add}(r) \]
\[ r.\text{setFillColor}(\text{‘yellow’}) \]
\[ r.\text{moveTo}(280, 150) \]

# Changing the depth:
\[ \text{sq.} \text{.setDepth}(10) \]
\[ r.\text{setDepth}(20) \]

# Objects with smaller depth appear in foreground.
Graphical Display: Rotate, Scale, Flip

We can rotate an object around its reference point:
\[ \text{sq.rotate(45)} \]

Scaling makes an object smaller or larger:
\[ \text{sq.scale(1.5)} \]
\[ r_.scale(0.5) \]

Fade-out:
\[
\text{for } i \text{ in range(80):}
\quad \text{sq.scale(0.95)}
\quad \text{canvas.remove(sq)}
\]

Flipping mirrors around an axis
\[ \text{sq.flip(angle=0)} \]
GRAPHICAL DISPLAY: DRAWABLE OBJECTS

Polygon, Square, Rectangle, Path

Default depth of drawable object is 50

roof = Polygon(Point(105, 105), Point(175, 105), Point(170, 85), Point(110, 85))
roof.setFillColor('darkgray')
roof.setDepth(30) # in front of facade
paper.add(roof)

facade = Square(60, Point(140, 130))
facade.setFillColor('white')
paper.add(facade)

chimney = Rectangle(15, 28, Point(155, 85))
chimney.setFillColor('red')
chimney.setBorderColor('red')
chimney.setDepth(40) # behind roof
paper.add(chimney)

smoke = Path(Point(155, 70), Point(150, 65), Point(160, 55), Point(155, 50))
smoke.setBorderWidth(2)
paper.add(smoke)
GRAPHICAL DISPLAY : DRAWABLE OBJECTS

Circle and 4 path objects for sun in the world

```
sun = Circle(30, Point(250, 50))
sun.setFillColor('yellow')
paper.add(sun)

sunraySW = Path(Point(225, 75), Point(210, 90))
sunraySW.setBorderColor('yellow')
sunraySW.setBorderWidth(6)
paper.add(sunraySW)

sunraySE = Path(Point(275, 75), Point(290, 90))
sunraySE.setBorderColor('yellow')
sunraySE.setBorderWidth(6)
paper.add(sunraySE)

sunrayNE = Path(Point(275, 25), Point(290, 10))
sunrayNE.setBorderColor('yellow')
sunrayNE.setBorderWidth(6)
paper.add(sunrayNE)

sunrayNW = Path(Point(225, 25), Point(210, 10))
sunrayNW.setBorderColor('yellow')
sunrayNW.setBorderWidth(6)
paper.add(sunrayNW)
```
def animate_sunrise(sun):
    w = canvas.getWidth()
    h = canvas.getHeight()
    r = sun.getRadius()
    x0 = w / 2.0
    y0 = h + r
    xradius = w / 2.0 - r
    yradius = h
    for angle in range(181):
        rad = (angle/180.0) * math.pi
        x = x0 - xradius * math.cos(rad)
        y = y0 - yradius * math.sin(rad)
        sun.moveTo(x, y)
def interpolate_colors(t, color1, color2):
    """Interpolate between color1 (for t == 0.0) and color2 (for t == 1.0)."""
    r1, g1, b1 = color1
    r2, g2, b2 = color2
    return (int((1-t) * r1 + t * r2), int((1-t) * g1 + t * g2), int((1-t) * b1 + t * b2))

def color_value(color):
    """Convert a color name to an (r,g,b) tuple."""
    return Color(color).getColorValue()
def animate_sunrise(sun, morning_sun, noon_sun, morning_sky, noon_sky):
    morning_color = color_value(morning_sun)
    noon_color = color_value(noon_sun)
    dark_sky = color_value(morning_sky)
    bright_sky = color_value(noon_sky)
    w = canvas.getWidth()
    h = canvas.getHeight()
    r = sun.getRadius()
    x0 = w / 2.0
    y0 = h + r
xradius = w / 2.0 – r
yradius = h
for angle in range(181):
    rad = (angle/180.0) * math.pi
    t = math.sin(rad)
    col = interpolate_colors(t, morning_color, noon_color)
sun.setFillColor(col)
    col = interpolate_colors(t, dark_sky, bright_sky)
canvas.setBackgroundColor(col)
x = x0 - xradius * math.cos(rad)
y = y0 - yradius * math.sin(rad)
sun.moveTo(x, y)
width = 300
height = 300

paper = Canvas(width, height, 'white', 'Rotating')

square1 = Square(100, Point(width/2, height/2))
square1.setFillColor('transparent')
square1.setBorderWidth(2)
paper.add(square1)

square2 = square1.clone()
square2.rotate(45)
square2.setDepth(40)
square2.setBorderWidth(1)
paper.add(square2)

square1.adjustReference(-50, 50)
square2 = square1.clone()
square2.rotate(45)
GRAPHICAL DISPLAY: SCALING

width = 300
hhight = 300
	paper = Canvas(width, height, 'white', 'Scaling (1)')
	pentagon1 = Polygon(Point(width/2, height/4),
Point(width/4, height/2), Point(width/2-40, height*3/4), Point(width/2+40, height*3/4),
Point(width*3/4, height/2))

pentagon1.adjustReference(0, height/4)
paper.add(pentagon1)

pentagon2 = pentagon1.clone()
pentagon2.scale(2)
paper.add(pentagon2)

pentagon1.move(width/4,0)
pentagon1.adjustReference(width/4, 0)
pentagon2 = pentagon1.clone()
pentagon2.scale(2)
i = 0
while 0 < sun.getRadius():
    if (i % 2) == 0:
        sunraySW.scale(1.1)
        sunraySE.scale(1.1)
        sunrayNE.scale(1.1)
        sunrayNW.scale(1.1)
        sun.scale(1.1)
    else:
        sunraySW.scale(0.9)
        sunraySE.scale(0.9)
        sunrayNE.scale(0.9)
        sunrayNW.scale(0.9)
        sun.scale(0.9)
    sunraySW.rotate(30)
    sunraySE.rotate(30)
    sunrayNE.rotate(30)
    sunrayNW.rotate(30)
    i += 1
sleep(.05) from time import sleep
width = 300
height = 300

paper = Canvas(width, height, 'white', 'Flipping(1)')

flag1 = Polygon(Point(width/2, height*3/4),
                 Point(width/2, height/4), Point(width/4, height/4),
                 Point(width/4, height/4+20), Point(width/4+20, height/4+20),
                 Point(width/4+20, height*3/4))

paper.add(flag1)

flag2 = flag1.clone()
flag2.flip()

paper.add(flag2)

flag2.flip(30)
sunraySW = Path(Point(225, 75), Point(210, 90))
sunraySW.setBorderColor('yellow')
sunraySW.setBorderWidth(6)
paper.add(sunraySW)

# Add the sunraySE by using Cloning and Flipping
sunRefPt = sun.getReferencePoint()
sunraySWRefPt = sunraySW.getReferencePoint()

diffX = sunRefPt.getX() - sunraySWRefPt.getX()
diffY = sunRefPt.getY() - sunraySWRefPt.getY()

sunraySW.adjustReference(diffX, diffY)

sunraySE = sunraySW.clone()
sunraySE.flip()
paper.add(sunraySE)

Let's finish the rest of two !!!
(sunrayNE and sunrayNW)

(Hint1) Clone the sunraySE rather than sunraySW
(Hint2) Use flip function with degree
(e.g.) flip(90)
GRAPHICAL DISPLAY : LAYER

Group a collection of other elements as a single composite object

That is, groups together several graphics object so that they can be moved and transformed as a whole:

- E.g) a car in the world

```python
car = Layer()
tire1 = Circle(10, Point(-20, -10))
tire1.setFillColor('black')
car.add(tire1)
tire2 = Circle(10, Point(20, -10))
tire2.setFillColor('black')
car.add(tire2)
body = Rectangle(70, 30, Point(0, -25))
body.setFillColor('blue')
body.setDepth(60)  # behind the tires
car.add(body)
car.moveTo(110, 180)
car.setDepth(20)  # in front of the house
paper.add(car)
```
The whole layer can be transformed as a single object

```python
for i in range(50):
    car.move(2, 0)
for i in range(22):
    car.rotate(-1)
for i in range(50):
    car.move(2,-1)
for i in range(22):
    car.rotate(1)
for i in range(50):
    car.move(2,0)
for i in range(10):
    car.scale(1.05)
car.flip(90)
```
GRAPHICAL DISPLAY : ANIMATION

Give some moves to objects
- E.g) running car in the world

```
paper.add(car)
timeDelay = 5
sleep(timeDelay)
car.move(-10, 0)
sleep(timeDelay)
car.move(-30, 0)
sleep(timeDelay)
car.move(-60, 0)
sleep(timeDelay)
car.move(-100, 0)
sleep(timeDelay)
```
PRACTICE : ANIMAL ANIMATION

Implement a function ‘draw_animal’ that draws an animal of your choice.
- Your animal should be drawn on a layer
- You must be able to move the entire animal by only moving the layer.
- The animal must also have some moving parts, such as legs, wings, or flippers.

Write functions to change the position of these moving parts.

Write a function ‘show_animation’ that shows an animation of your animal.
- It should move around and its moving parts should be moving.
EXAMPLE : SUN ANIMATION

‘draw()’ function
- Draw a sun on a layer
- Moving parts are sun and rays

‘move()’ function
- Rotate 30 degree of the position of each ray

‘show()’ function
- Show an animation of the sum
- It moves within the boundary and sun rays are also rotating.
EXAMPLE: SUN ANIMATION

```python
from cs1graphics import *
from time import *
from math import *

width=400
height=300
sunR=20
sun = Layer()
moveWidthDirection=1
moveHeightDirection=1
moveStep=30

#definition of draw, move, show....

paper = Canvas(width, height, 'white', 'Sun Animation with Layer')
draw(sun, sunR)

sun.moveTo(100, 100)
paper.add(sun)
show(sun)
```
def draw(sunLayer, sunR) :
    sun = Circle(sunR, Point(0, 0))
    sun.setFillColor('yellow')
    sunLayer.add(sun)

    sunraySW = Path(Point(-sunR, sunR), Point(-(sunR+ sunR/2), sunR + sunR/2))
    sunraySW.setBorderColor('yellow')
    sunraySW.setBorderWidth(6)
    sunraySW.adjustReference(sunR, -sunR)
    sunLayer.add(sunraySW)

    sunraySE = sunraySW.clone()
    sunraySE.flip()
    sunLayer.add(sunraySE)

    sunrayNW = sunraySW.clone()
    sunrayNW.flip(90)
    sunLayer.add(sunrayNW)

    sunrayNE = sunrayNW.clone()
    sunrayNE.flip()
    sunLayer.add(sunrayNE)

def move(sunLayer) :
    sunContents = sunLayer.getContents()
    for i in range(len(sunContents)) :
        sunContents[i].rotate(30)
def show(sunLayer):
    while True:
        global moveWidthDirection
        global moveHeightDirection
        global width
        global height

        sunCenterPt = sunLayer.getReferencePoint()
        if (moveWidthDirection > 0) and (sunCenterPt.getX() > (width - (sunR + sunR/2 + moveStep*2))):
            moveWidthDirection = -1

        if (moveWidthDirection < 0) and (sunCenterPt.getX() < (sunR + sunR/2 + moveStep*2)):
            moveWidthDirection = 1

        if (moveHeightDirection > 0) and (sunCenterPt.getY() > (height - (sunR + sunR/2 + moveStep*2))):
            moveHeightDirection = -1

        if (moveHeightDirection < 0) and (sunCenterPt.getY() < (sunR + sunR/2 + moveStep*2)):
            moveHeightDirection = 1

        sunLayer.move(moveStep * moveWidthDirection, moveStep * moveHeightDirection)
        move(sunLayer)
        sleep(.5)