Welcome !

2024 Fall CS101 Introduction to Programming

KAIST School of Computing

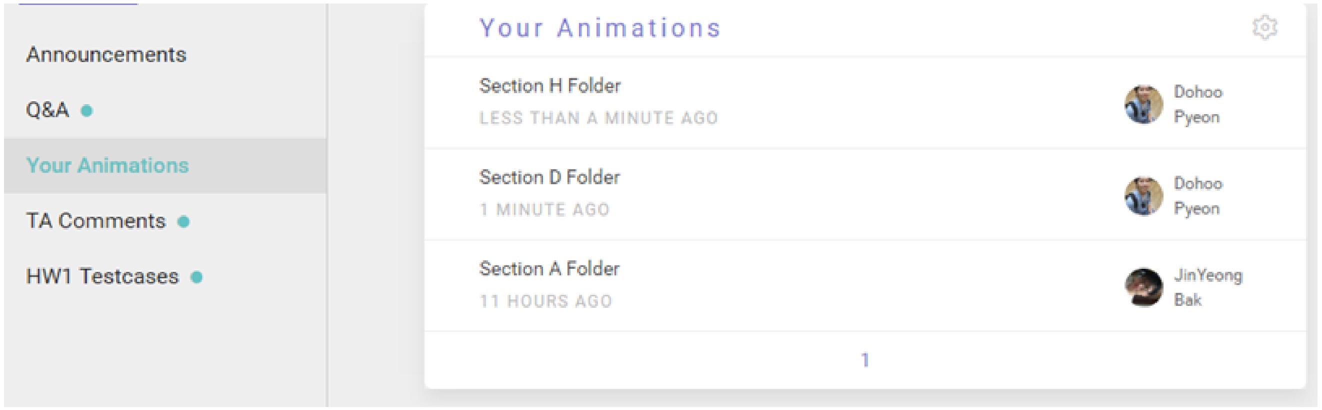
Week 5

Create graphical display
(Practice objects and basic data types)

Week 5
Today’s Tasks

Tasks for Today!

- Bank
- Help!
  - `>> help(‘print’)`
- Animation
  - After you finish, upload your file through the links on the “Your Animation” in ***elice***



Your Animations

Section H Folder LESS THAN A MINUTE AGO	Dohoo Pyeon
Section D Folder 1 MINUTE AGO	Dohoo Pyeon
Section A Folder 11 HOURS AGO	JinYeong Bak

1

Task 1 | Bank

- Complete 'bank.py'
- Implement functions ***`deposit`*** and ***`withdrawal`*** that **change the global variable** **`balance`**
  - def deposit:
    - balance = balance + money
  - def withdrawal
    - balance = balance - money
  - If you don't have sufficient money, then print the amount of money that can withdraw
- Implement a function bank
  - It first asks
  - "Deposit(d) or withdrawal(w) or balance check(c)??"
  - If user input is empty string, '', then quit this function using 'return'
  - If user input is 'w', then ask the amount of money to be withdrawn and withdraw it
  - If user input is 'd', then ask the amount of money to be deposited and deposit it
  - If user input is 'c', then check the current balance

Task 1 | Bank – Example

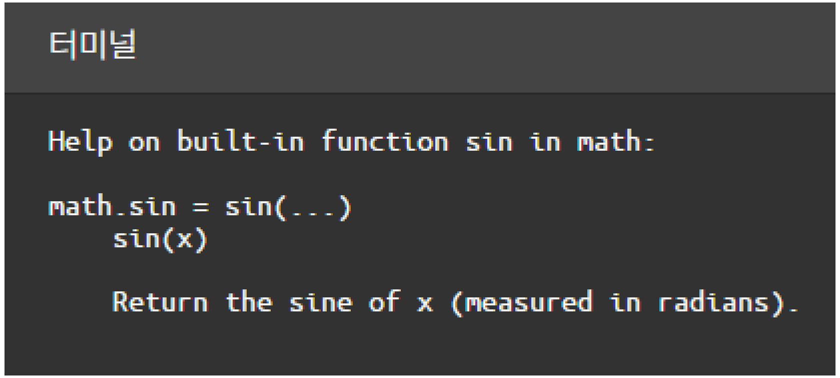
```
Deposit(d) or withdrawal(w) or balance check(c)?? c
Your current balance is 0 won
Deposit(d) or withdrawal(w) or balance check(c)?? d
How much do you want to deposit? 10000
You deposited 10000 won
Deposit(d) or withdrawal(w) or balance check(c)?? t
Please, press d or w or return
Deposit(d) or withdrawal(w) or balance check(c)?? w
How much do you want to withdraw? 9000
You've withdraw 9000 won
Deposit(d) or withdrawal(w) or balance check(c)?? w
How much do you want to withdraw? 5000
You've withdrawn 5000 won
But you only have 1000 won
Deposit(d) or withdrawal(w) or balance check(c)??
```

Task 2 | Help

- There are hundreds of pre-defined functions.

- How can programmer remember everything?
- It’s impossible. We can ask for help!

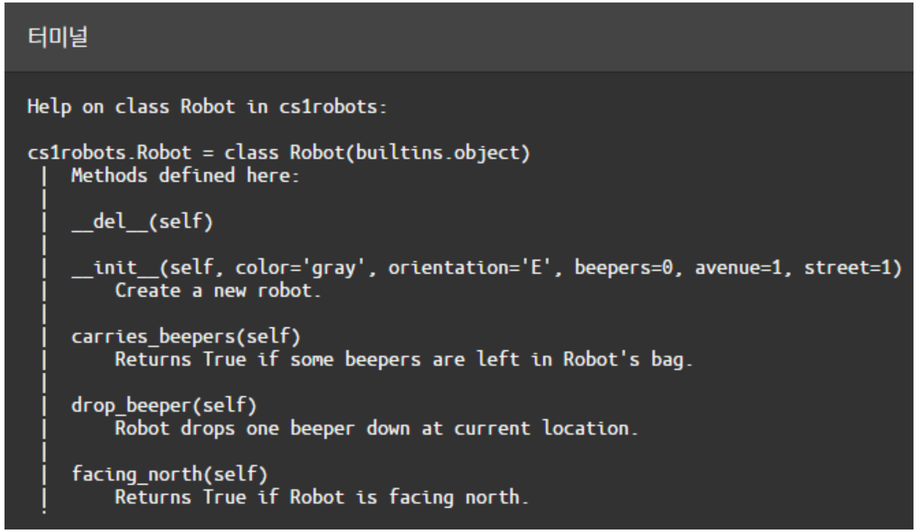
- Function help()
    - Try help(‘print’)
    - Try help(‘math.sin’)



```
Help on built-in function sin in math:

math.sin = sin(...)
    sin(x)

    Return the sine of x (measured in radians).
```

Task 2 | Help

- We can also use 'help function' for special modules.
- Try!
  - **cs1robots**
    - help('cs1robots')
    - help('cs1robots.Robot')
    - help('cs1robots.Robot.turn_left')
    - help('cs1robots.create_world')
  - **cs1graphics**
    - help('cs1graphics.Ellipse')
    - help('cs1graphics.Color')
    - help('cs1graphics.Text')
    - help('cs1graphics.Square.rotate')



```
Help on class Robot in cs1robots:

cs1robots.Robot = class Robot(builtins.object)
 |  Methods defined here:
 |
 |  __del__(self)
 |
 |  __init__(self, color='gray', orientation='E', beepers=0, avenue=1, street=1)
 |      Create a new robot.
 |
 |  carries_beepers(self)
 |      Returns True if some beepers are left in Robot's bag.
 |
 |  drop_beeper(self)
 |      Robot drops one beeper down at current location.
 |
 |  facing_north(self)
 |      Returns True if Robot is facing north.
 |
```

Task 3 | Animation

- Implement a function '**draw_animal**' that draws an animal of your choice.
  - Your animal should be drawn on a layer (layer will be explained in later slides)
  - 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.

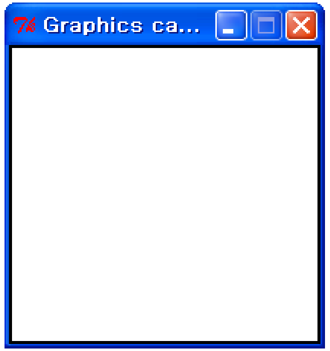
- You can choose others if it has some moving parts.
  - Ex) Cartoon character, Car, Airplane

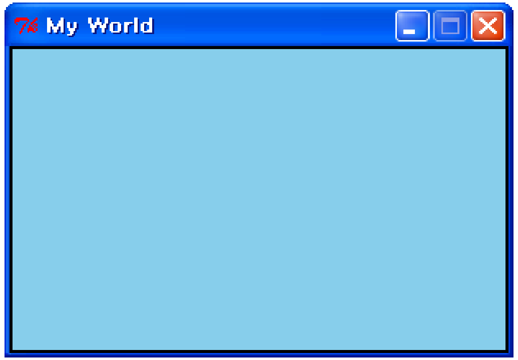
Task 3 | Graphical Display (1/4) – Animation

- Canvas
  - A window upon which we draw

```
from cs1graphics import*
from time import*

paper = Canvas()
```

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



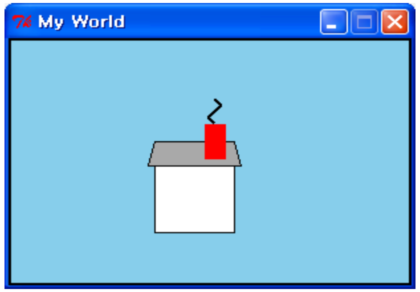


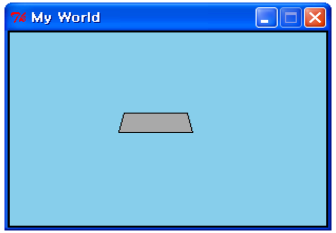
paper = Canvas(300, 200, 'skyBlue', 'My World')

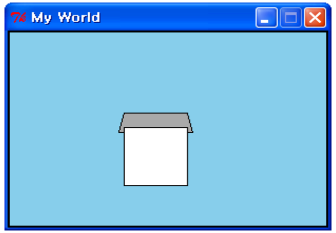
Task 3 | Graphical Display (2/4) – Animation

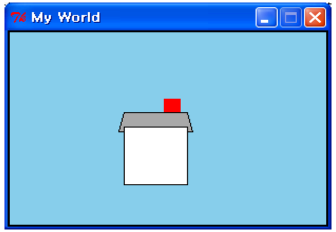
- Drawable objects
  - Polygon, Square, Rectangle, Path
  - Depth b/w drawable objects (Default. 50)

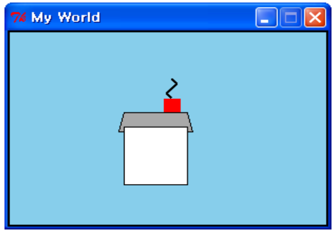
```
roof = Polygon(Point(105, 105), Point(175, 105), Point(170,
85), Point(110, 85))
roof.setFillColor('darkgray')
roof.setDepth(30) # in front of façade
paper.add(roof)

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

chimney = Rectangle(15, 28, Point(155, 85))
chimney.setFillColor('red')
chimney.setBorderColor('red‘)
chimney.setDepth(20) # in front of roof
paper.add(chimney)

smoke = Path(Point(155, 70), Point(150, 65),
             Point(160, 55), Point(155, 50))
smoke.setBorderWidth(2)
paper.add(smoke)
```







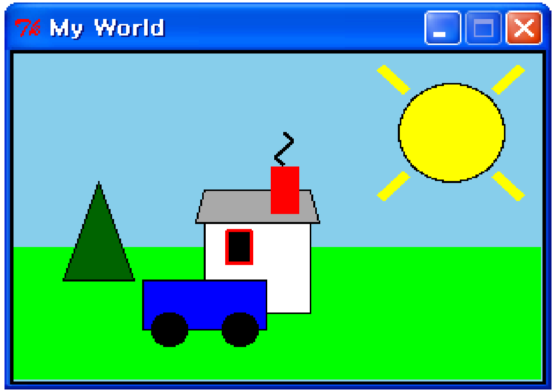




Task 3 | Graphical Display (3/4) – Animation

- Layer
  - Group a collection of other elements as a single composite object
  - (e.g.) A car in the world

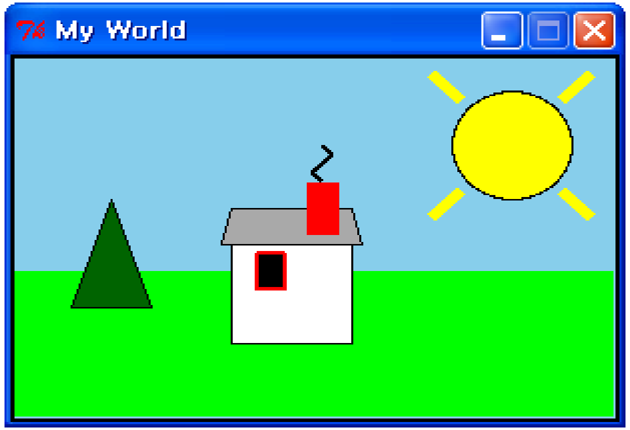
```
car = Layer()

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)
```



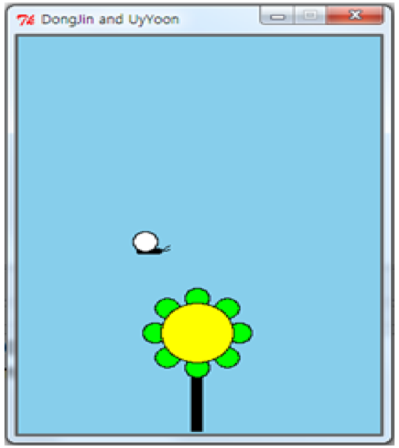
Task 3 | Graphical Display (4/4) – Animation

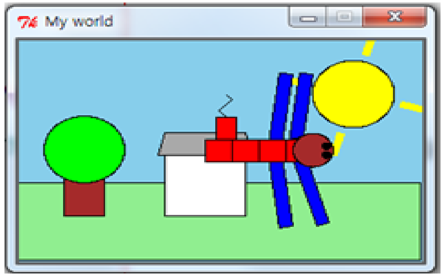
- Animation
  - Give some moves to objects
  - (e.g.) Running car in the world

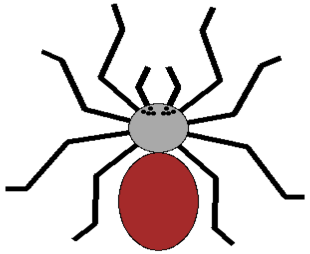
```
paper.add(car)

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)
```

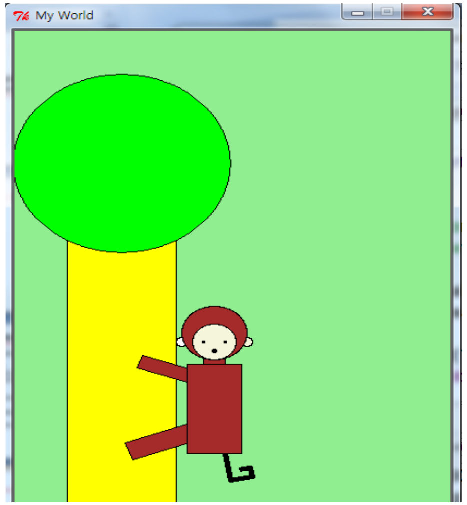


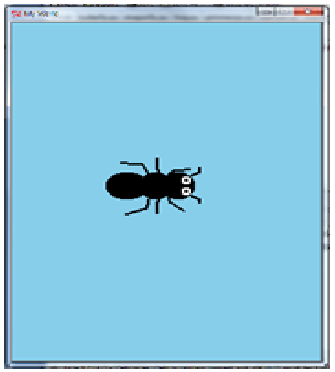
Task 3 | Example – Animation

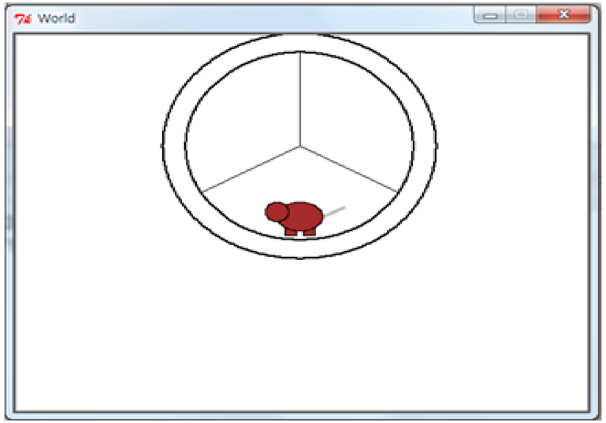






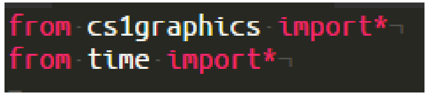






Tips

- Use ‘cs1graphics’ module and ‘time’ module.



```
from cs1graphics import*
from time import*
```

- After you choose an animal, simplify it and decide moving parts.
- When you decide moving parts, think about the functions you can use.

- Not Recommended:
  - Make more than 2 animals (If you have a lot of time, it will be okay.)
  - Choose an animal which it is hard to simplify
    - Ex) Hedgehog (고슴도치), Specific person or job (a figure skater)
  - Implement too simple thing.

Useful *cs1grphics* functions

- **Objects**
  - Canvas, Layer
  - Circle, Ellipse, Square, Rectangle, Polygon, Path, Text, …

- **Object methods**
  - Color
    - setBorderColor, setFillColor
  - Move
    - move, moveTo
  - Depth
    - setDepth
  - Others
    - rotate, scale, flip
  - Reference Point
    - getReferencePoint, adjustReference

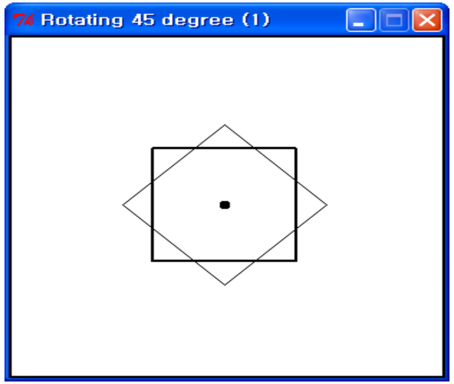
Be Creative!

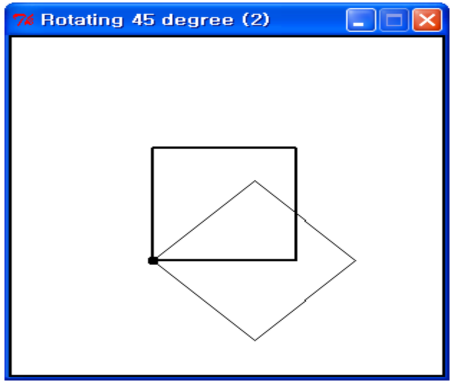
Additional Graphical Display (1/5)

- Operations on drawable objects (1)
  - Rotating

```
from cs1graphics import *

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)
```





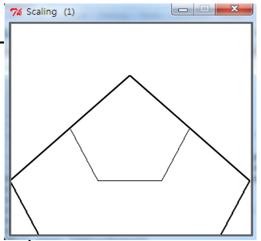
The default reference point for a square(rectangle, circle) is its center.

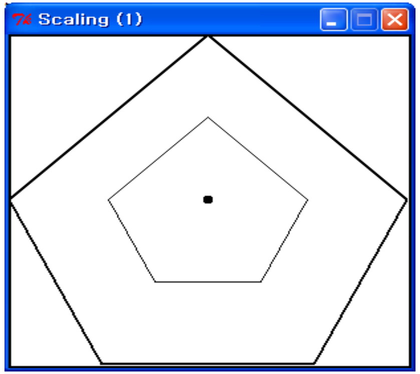
Additional Graphical Display (2/5)

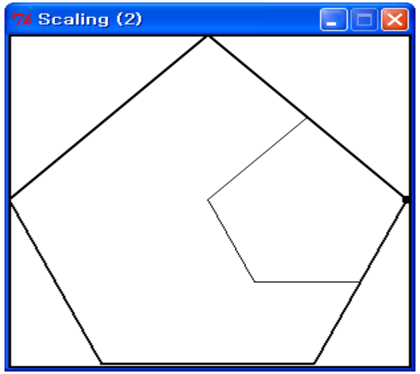
- Operations on drawable objects (2)
  - Scaling

```
from cs1graphics import *

width = 300
height = 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.adjustReference(width/4, 0)

pentagon2 =  pentagon1.clone()
pentagon2.scale(2)
pentagon1.move(width/4, 0)
pentagon2.move(width/4, 0)
```







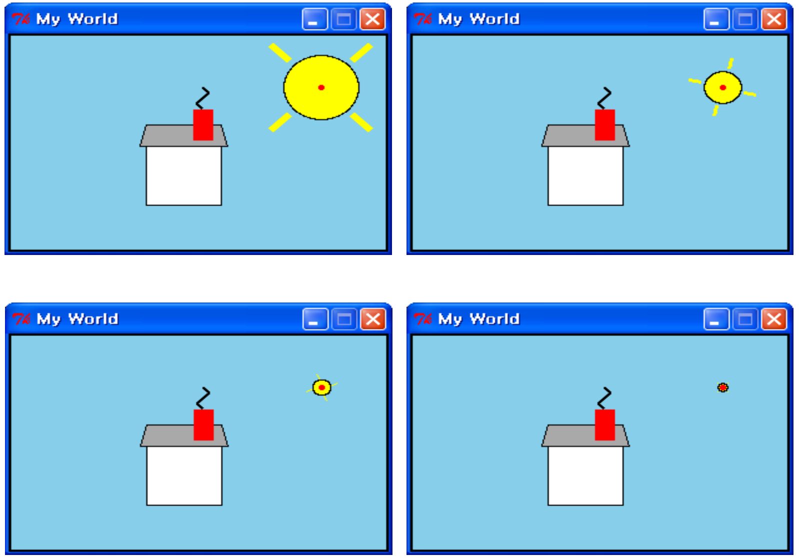
The default reference point for a polygon is initially aligned with the first point of the polygon.

Additional Graphical Display (3/5)

- How to make rotating and shrinking sun?
  - Rotating and Scaling

```
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)
```

$\rightarrow$ **from time import sleep**

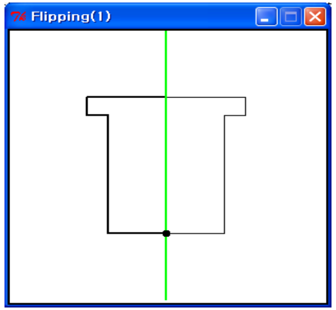


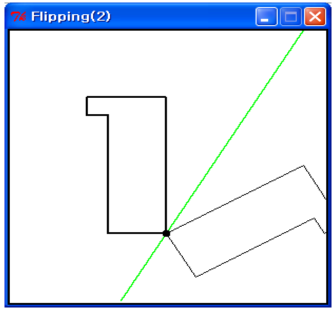
Additional Graphical Display (4/5)

- Operations on drawable objects (3)
  - Flipping

```
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)
```



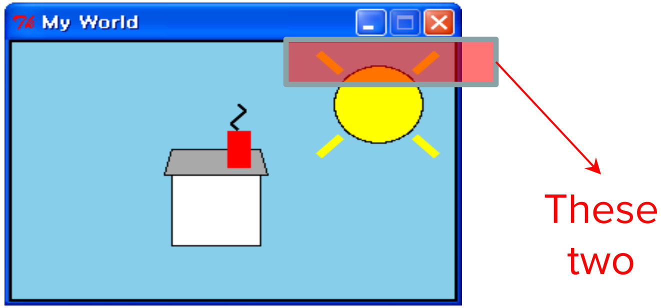


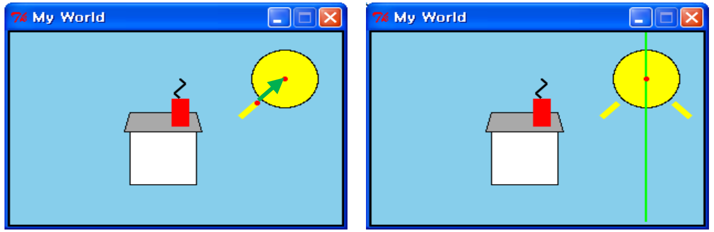
```
cs1graphics.Polygon.flip = flip(self, angle=0) unbound cs1graphics.Polygon method
    Flip the object reflected about its current reference point.

    By default the flip is a left-to-right flip with a vertical axis of symmetry.

    angle     a clockwise rotation of the axis of symmetry away from vertical
```

Additional Graphical Display (5/5)

- How to avoid finding the exact geometry of each ray?
  - Cloning and Flipping

```
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)

questions?