CS65: Introduction to Computer Science

Graphics

Md Alimoor Reza
Assistant Professor of Computer Science

Graphics library

- A simple library (containing other python codes) that makes it easy to experiment with graphics components
- You will learn how to draw stuffs (shapes, text, etc) on a window using Python programming
- Graphics library: https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html

Graphics library

- The graphics library might not be installed in your Thonny
  - ERROR!

```
  6 from graphics import *
  7

>>> %Run lec5_demo1.py
  Traceback (most recent call last):
    File "/Users/reza/Class_and_Research/drake_teaching/CS65/cs
  6, in <module>
      from graphics import *
  ModuleNotFoundError: No module named 'graphics'

>>>
```

Graphics

Credit John Zelle

There is a library (a bunch of code) that we will import into Thonny so we can make simple Python code that will draw items on the screen.

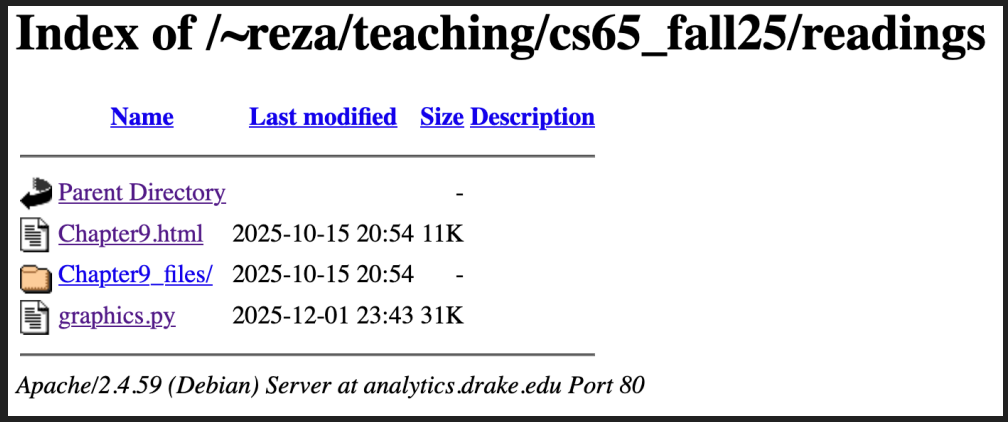
But, we need to download a file and put it into a specific folder first.

Download the graphics.py file somewhere on your computer.

Link here

right-click on “graphics.py” and select “Save As”
Save it anywhere on your computer (e.g. Desktop)

Graphics



Credit John Zelle

There is a library (a bunch of code) that we will import into Thonny so we can make simple Python code that will draw items on the screen.

But, we need to download a file and put it into a specific folder first.

Download the graphics.py file somewhere on your computer.

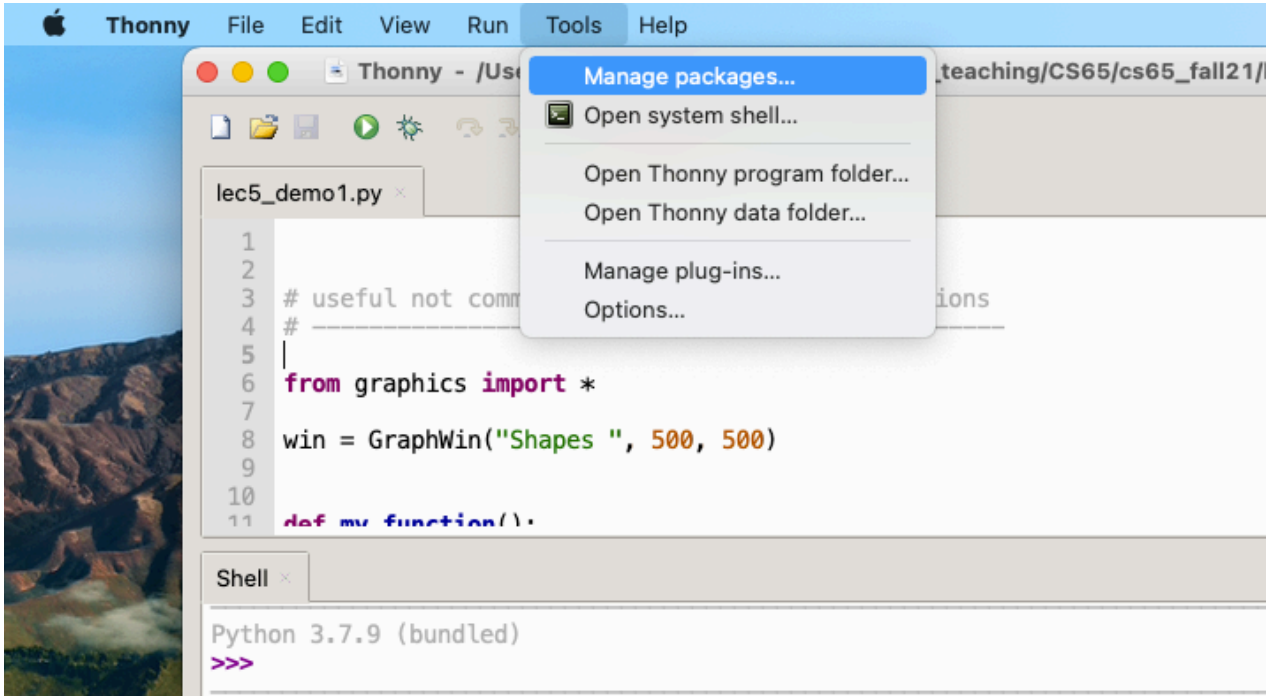
Link here

right-click on “graphics.py” and select “Save As”

Save it anywhere on your computer (e.g. Desktop)

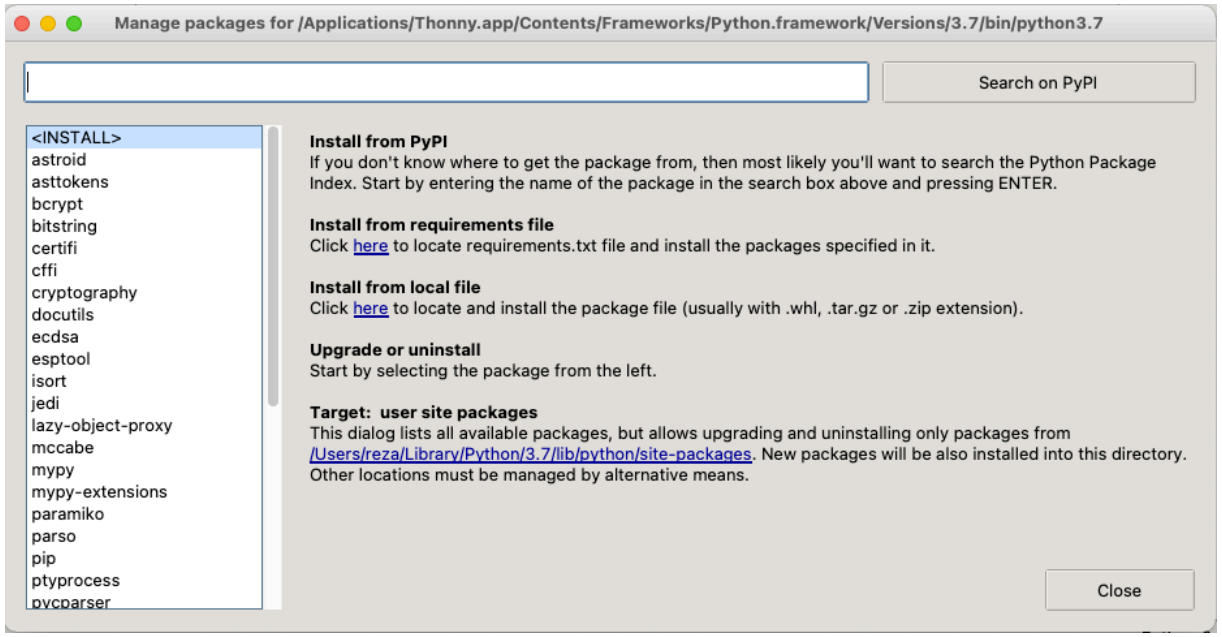
Alternatively: Quick installation of graphics in Thonny

- Find the **Tools** option from the list of menus



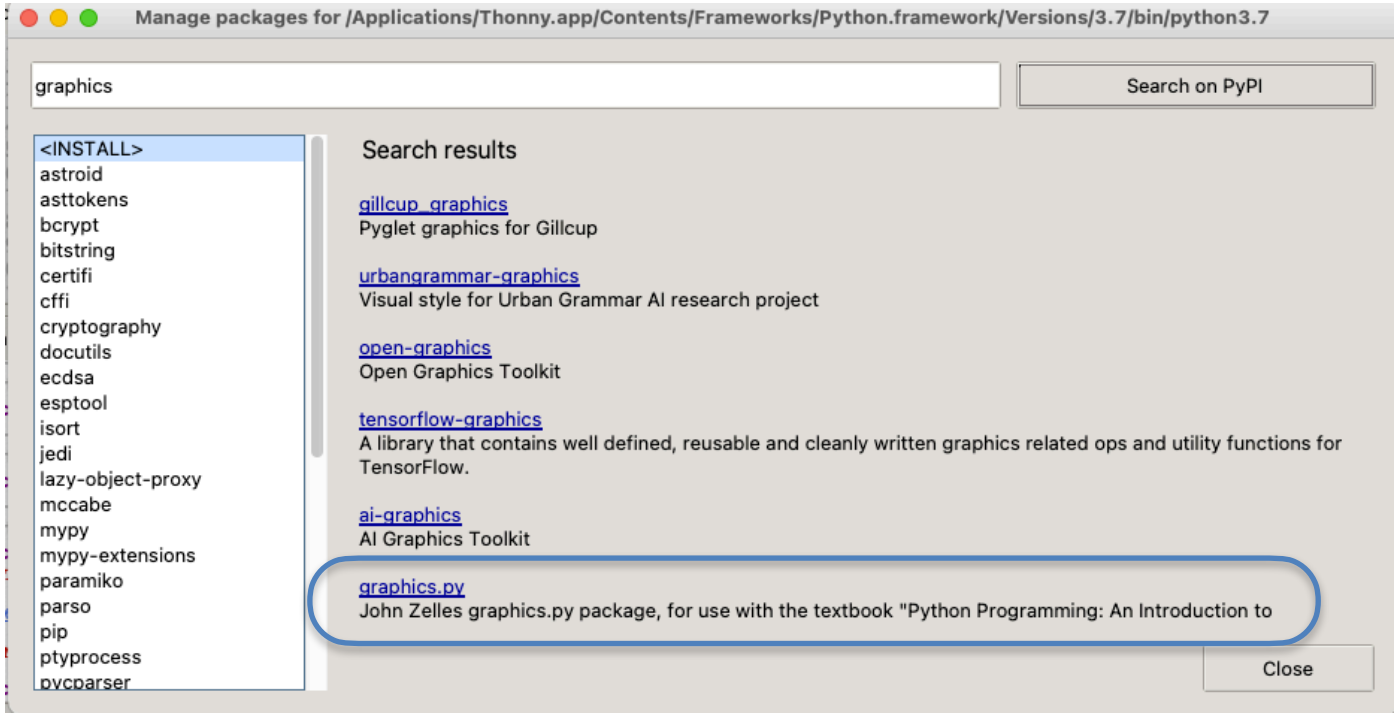
Alternatively: Quick installation of graphics in Thonny

- Type in '<u>graphics</u>' in the empty textbox and then **hit** '<u>Search on PyPI</u>'



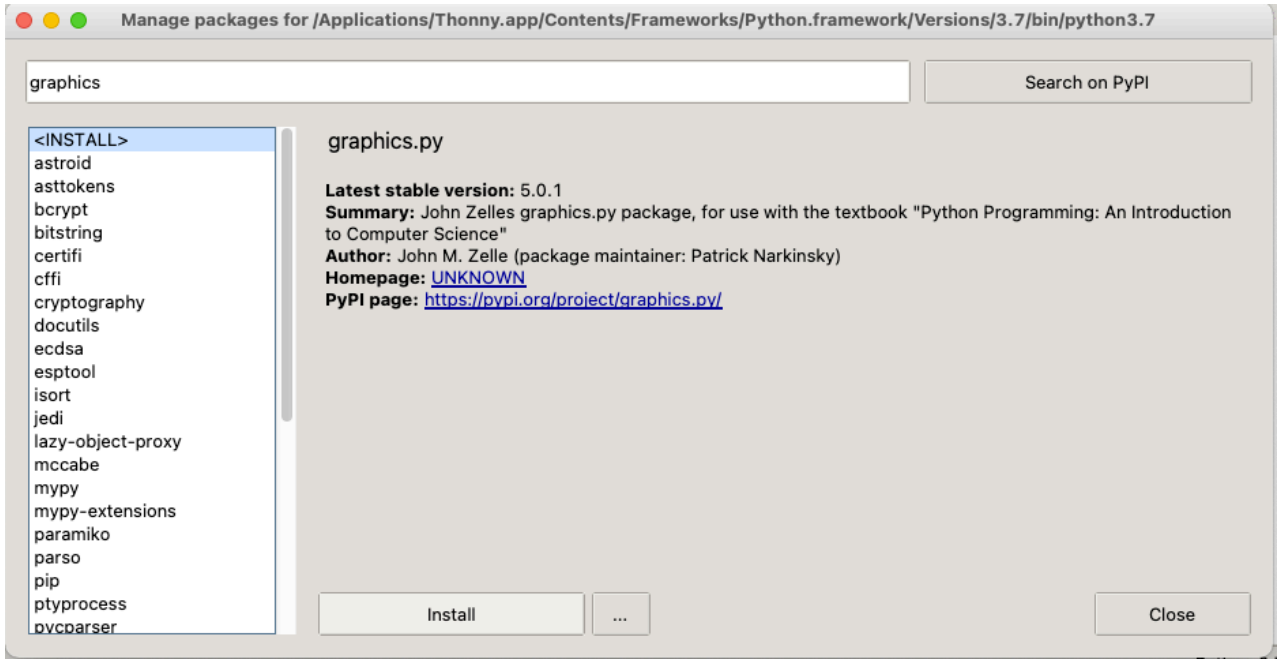
Alternatively: Quick installation of graphics in Thonny

- Select the graphics.py from the bottom



Alternatively: Quick installation of graphics in Thonny

- Finish the installation! Now you are ready to access graphics library components



Topics

- Creating a graphical window

- Drawing shapes inside the window
  - Circle
  - Rectangle
  - Line, Text, and combinations of these shapes

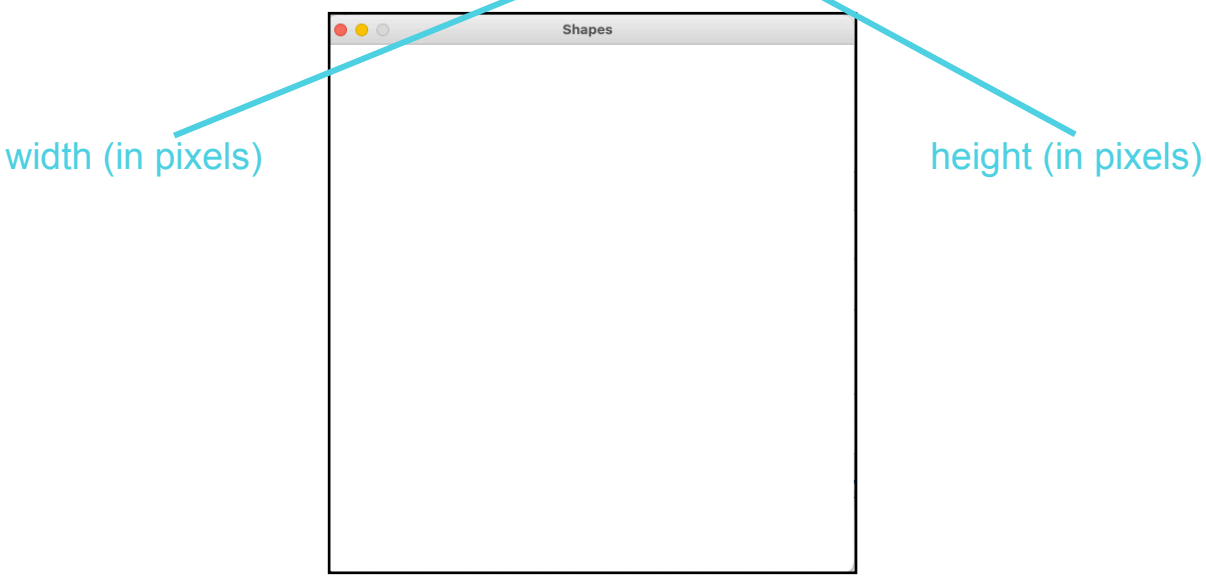
- Changing coordinate system

- Mouse interaction inside graphics window

A simple program using graphics

- *GraphWin(…)*: creates the **canvas** or **panel** where everything will be drawn

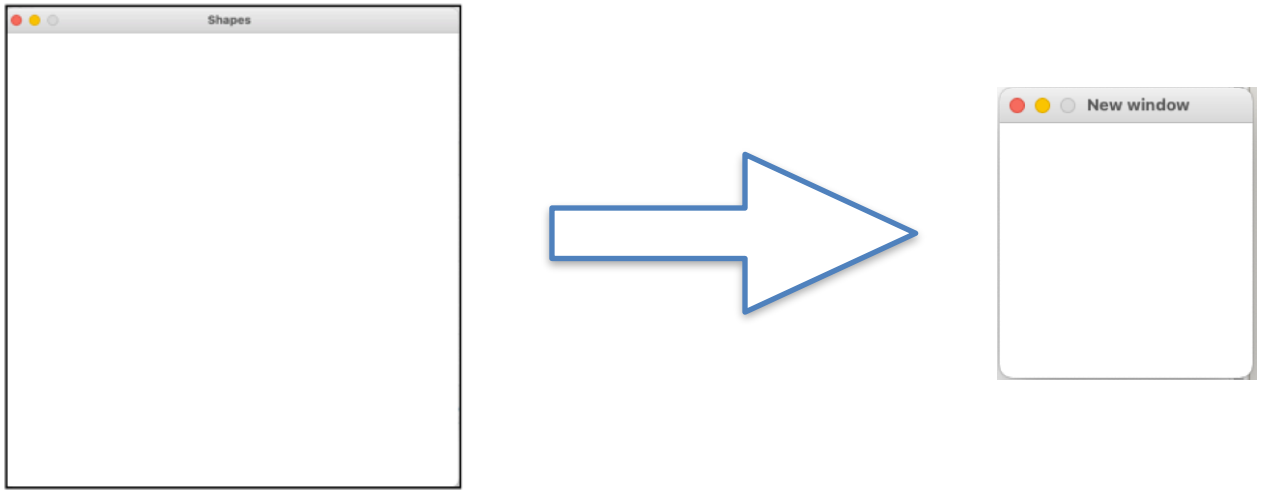
```
6  from graphics import *
7
8  win = GraphWin("Shapes ", 500, 500)
```



Changing window size

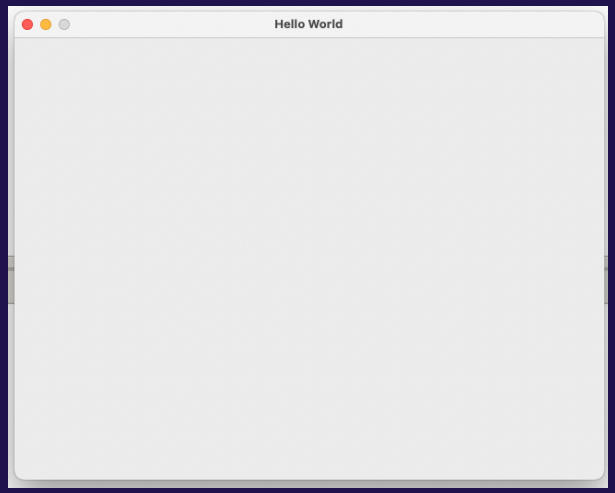
- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*

```
6  from graphics import *
7
8  win = GraphWin("Shapes ", 500, 500)
```



Exercise#1

Try it and run it. When you run your program, you should get a window that pops up:



Try making windows of different sizes

Try making windows with different titles.

Graphics library

- A simple library (containing other python codes) that makes it easy to experiment with graphics components
- Graphics library: https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html
- Graphics library provides different graphical objects
  - **Point**, Line, **Circle**
  - Oval, Rectangle, Polygon
  - Text, Image
- You can manipulate properties of these shapes/objects
  - change color and sizes

Topics

- Creating a graphical window
- Drawing shapes inside the window
  - Circle
  - Rectangle
  - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

Graphical objects from graphics library

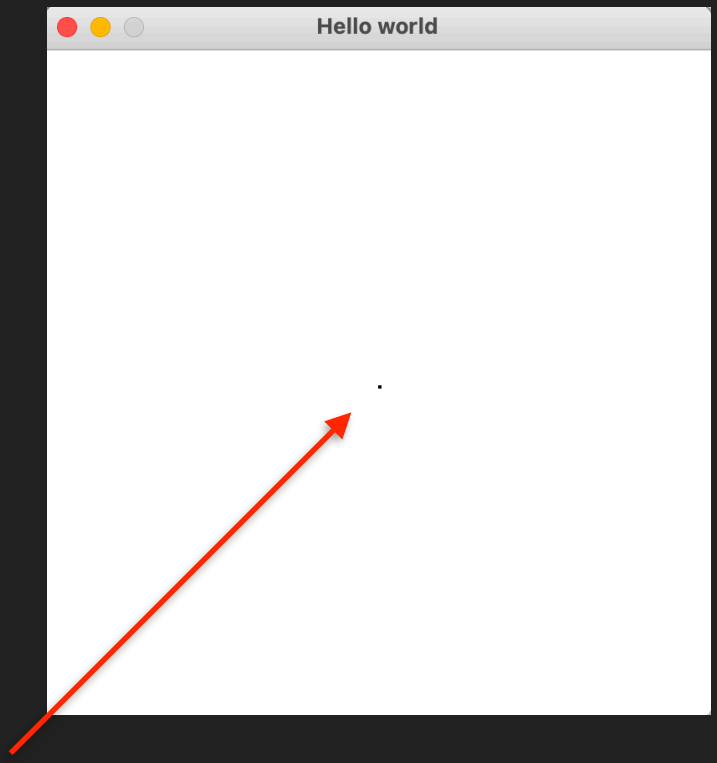
- Graphics library provides different shapes (graphical objects):
  - **Point**, Line, **Circle**
  - Oval, **Rectangle**, Polygon
  - Text, Image

- You can manipulate properties of these shapes/objects
  - change color and sizes

- You can also move them around inside the window

Drawing a Point

```
1  from graphics import *
2
3  # create a window
4  win = GraphWin("Hello world", 400, 400)
5
6  # create a point
7  p = Point(200, 200)
8  p.draw(win)
```



Moving it…

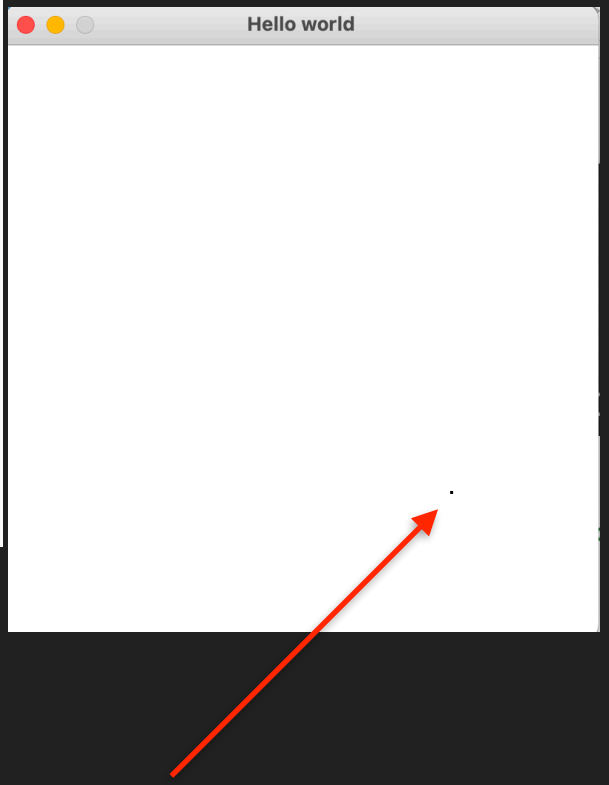
Adding p.move(100,100)

```
1  from graphics import *
2
3  # create a window
4  win = GraphWin("Hello world", 400, 400)
5
6  # create a point
7  p = Point(200, 200)
8  p.draw(win)
9
10 # moving it
11 p.move(100, 100)
```

Moving it…

Adding p.move(100,100)

```
1  from graphics import *
2
3  # create a window
4  win = GraphWin("Hello world", 400, 400)
5
6  # create a point
7  p = Point(200, 200)
8  p.draw(win)
9
10 # moving it
11 p.move(100, 100)
```



Wait, what?

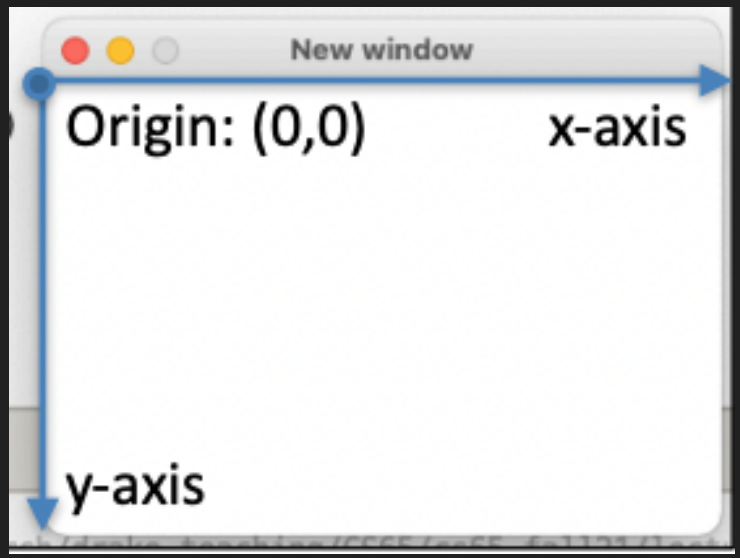
Why did moving the point (100,100) move it in the direction of lower right?

In most window-based graphics systems, the origin (0,0) is the upper-left corner of the window

Wait, what?

Why did moving the point (100,100) move it in the direction of lower right?

In most window-based graphics systems, the origin (0,0) is the upper-left corner of the window



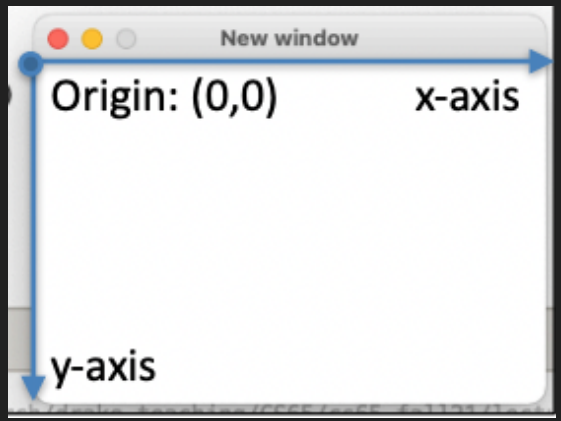
Wait, what?

Why did moving the point (100,100) move it in the direction of lower right?

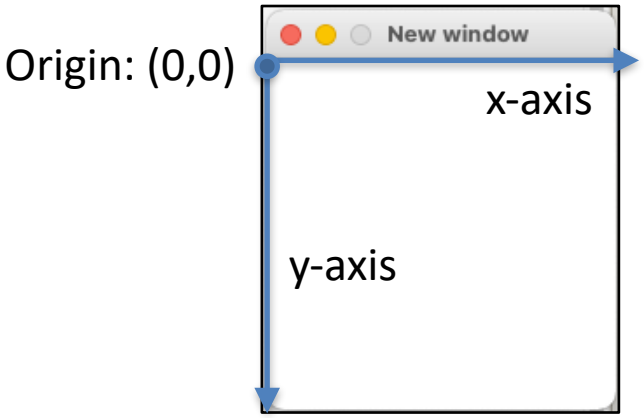
In most window-based graphics systems, the origin (0,0) is the upper-left corner of the window

Moving things in the +x direction => moves to the right

Moving things in the +y direction => moves down



- *GraphWin(…)*: creates the **canvas** or **panel** where everything will be drawn

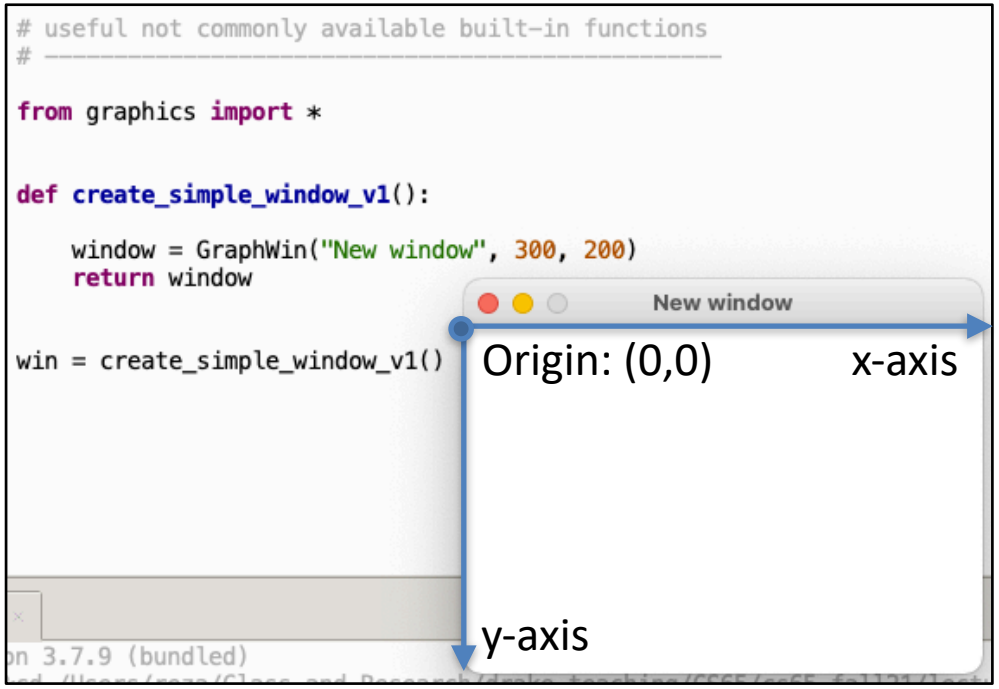


- Coordinate system
  - x: top-left —> top-right
  - y: top-left —> bottom-left

- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis — — —> width
  - y-axis — — —> height

Drawing rectangular window

- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis — — —> width
  - y-axis — — —> height

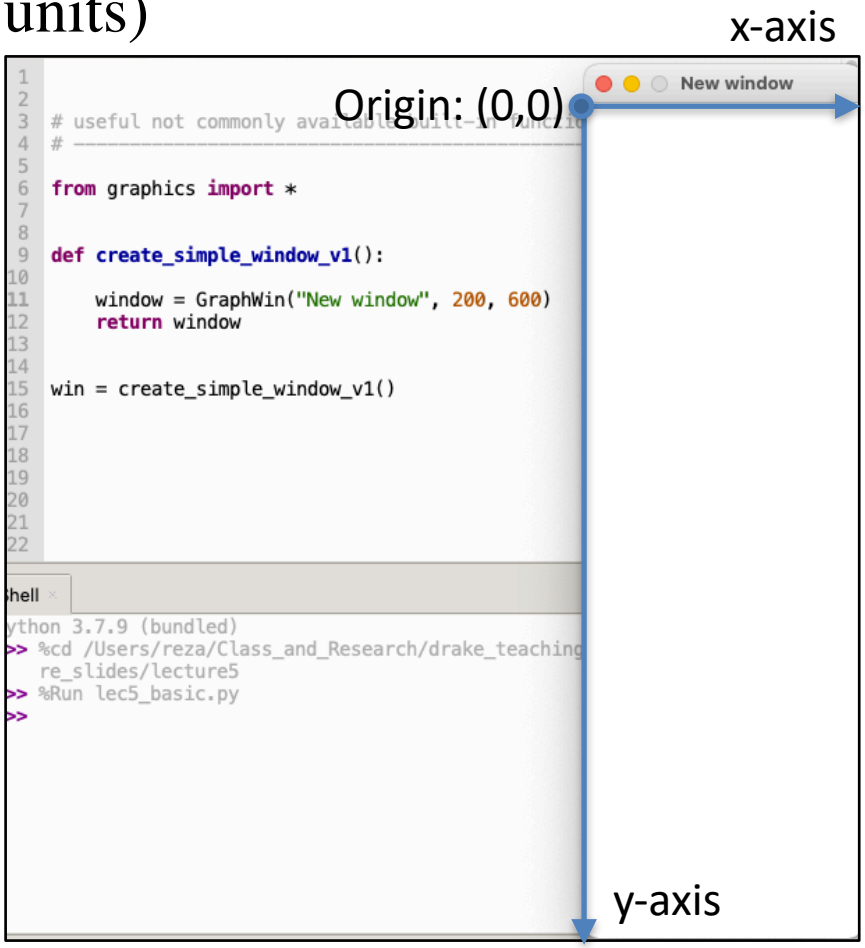



```
# useful not commonly available built-in functions
# ------------------------------------------------

from graphics import *


def create_simple_window_v1():

    window = GraphWin("New window", 300, 200)
    return window


win = create_simple_window_v1()
```

Drawing a rectangular window

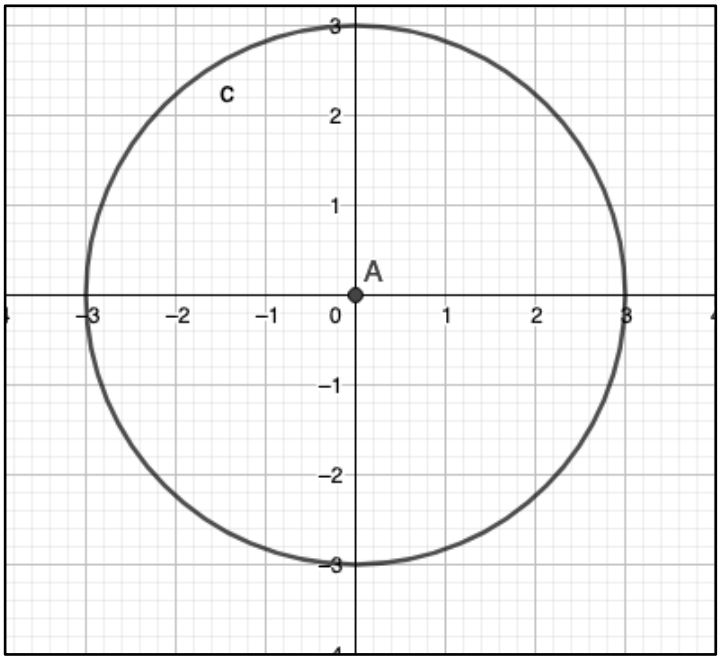
- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis — — —> width
  - y-axis — — —> height

- Coordinate system
  - x: top-left —> top-right
  - y: top-left —> bottom-left



Drawing inside the window

- You can draw inside the window
- Drawing a circle inside
  - how many variables do we need for a circle?



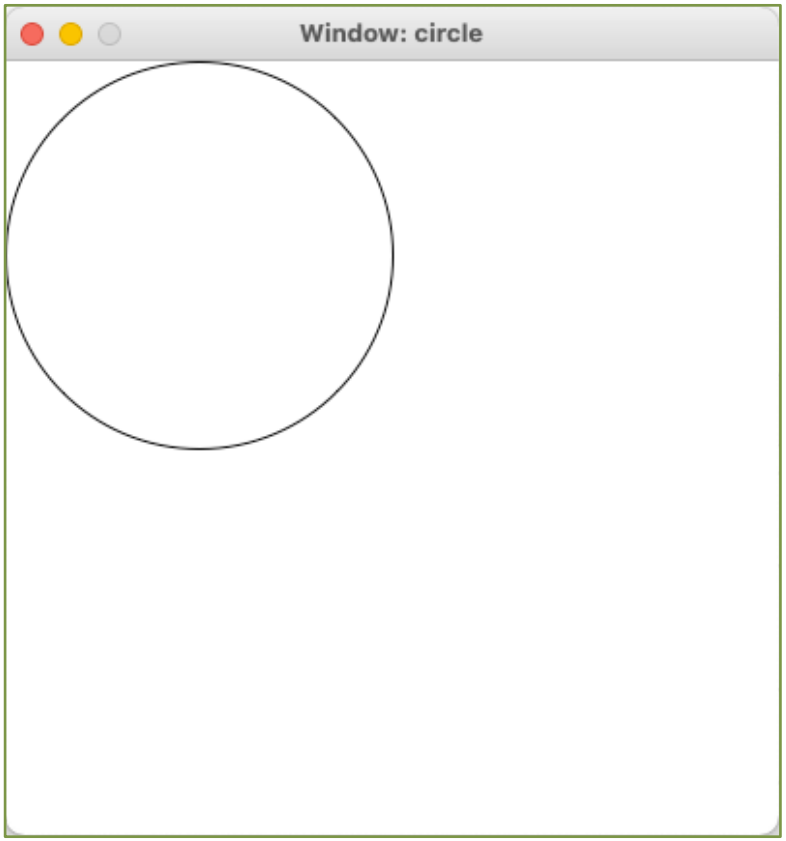
Drawing inside the window

- Step 1: Construct a circle
  - Step 1.1: construct a point —> the center of the circle
  - Step 1.2: fix the radius
  - Step 1.3: put them together

- Step 2: **Draw** the newly constructed circle inside the window

```
1  from graphics import *
2
3  win = GraphWin("circle", 400, 400)
4  center_point = Point(100, 100)
5  radius = 100
6  circle = Circle(center_point, radius)
7  circle.draw(win)
```

Coding demo



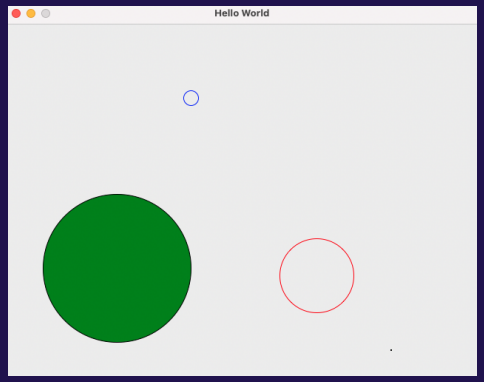
Exercise #2: Play with Circles

1. What happens if you call
```
# move the point again
p.move(100,100)
```
after drawing the circle? Does the circle move with the point? Or does just the point move?
2. Make more circle variables and draw them! What kinds of colors can you use?

https://www.wikipython.com/tkinter-ttk-tix/summary-information/colors/

1. You can fill your circles with a call to **`setFill(color)`**

```
circ3 = Circle(Point(150, 330), 100)
circ3.setOutline("black")
circ3.setFill("green")
circ3.draw(win)
```



Text

You can put text anywhere on the window you would like

```
from graphics import *

# create a window
win = GraphWin("Hello World", 640, 480)

# text
my_text= Text(Point(100,100), "Hello, world!")
my_text.draw(win)
```

Text

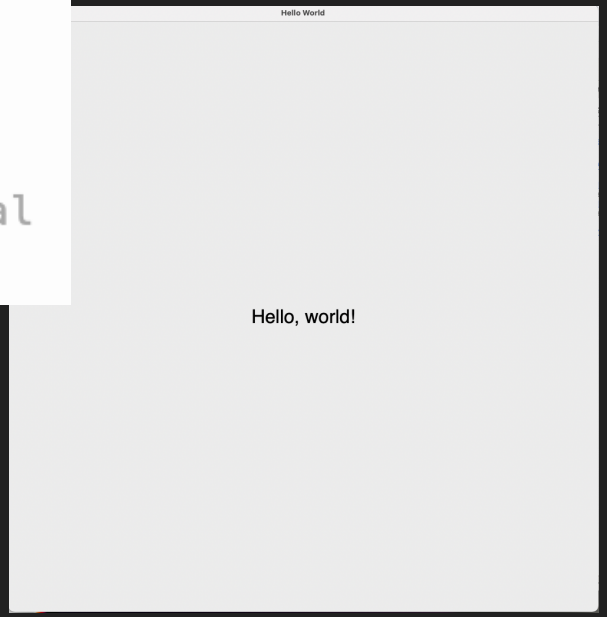
You can put text anywhere on the window you would like

```
from graphics import *

# create a window
win = GraphWin("Hello World", 640, 480)

# text
my_text= Text(Point(100,100), "Hello, world!")
my_text.draw(win)
```

Notice how I make the point *within* the call to Text

Increased Window Size
Increased Font Size

```
from graphics import *

# create a window
win = GraphWin("Hello World", 1000, 1000)

# text
my_text= Text(Point(500,500), "Hello, world!")
my_text.setSize(32) #note: sizes from 5 to 36 are legal
my_text.draw(win)
```



More Options

The lab includes instructions on how to change the font, text color, move text, set the text to something different.

```
1  from graphics import *
2
3  win = GraphWin("Hello world", 640, 480)
4  my_text = Text(Point(100, 100), "hello")
5  my_text.setSize(32) # note: sizes from 5 to 36 are legal
6  my_text.draw(win)
7
8
9  my_text2 = Text(Point(200, 200), "world")
10 my_text2.setSize(32) # note: sizes from 5 to 36 are legal
11 my_text2.setFace("courier")
12 my_text2.setTextColor("blue")
13 my_text2.move(20, 50)
14 my_text2.draw(win)
```

Exercise #3

Play around with creating more text objects, drawing them to the window, and experimenting with font sizes, font styles, and font families. Make two different text variables and display in different colors, sizes, and different fonts.

Application Programming Interface

An API (Application Programming Interface) is documentation on how to use some software

The Graphics API is available here:

https://mcsp.wartburg.edu/zelle/python/graphics/graphics/graphref.html

Drawing rectangle

- Step 1: Construct a rectangle
  - Step 1.1: construct a point —> one corner
  - Step 1.2: construct a point —> opposite corner
  - Step 1.3: put them together

- Step 2: **Draw** the newly constructed rectangle inside the window

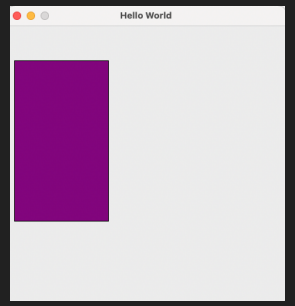
```
from graphics import *


def create_simple_window_w_rect():
    window = GraphWin("Window: Rectangle", 400, 400)

    point1 = Point(50,50)
    point2 = Point(250, 350)

    rect   = Rectangle(point1, point2)
    rect.setFill("blue")
    rect.draw(window)

    return window


w1 = create_simple_window_w_rect()
```

```
on 3.7.9 (bundled)
cd /Users/reza/Class_and_Research/drake_teaching/CS65
Run lec6_rect.py
Run lec6_rect.py
```

The world needs more rectangles

Rectangles can be drawn (and filled with a color) by specifying the coordinates of the upper left and lower right corners

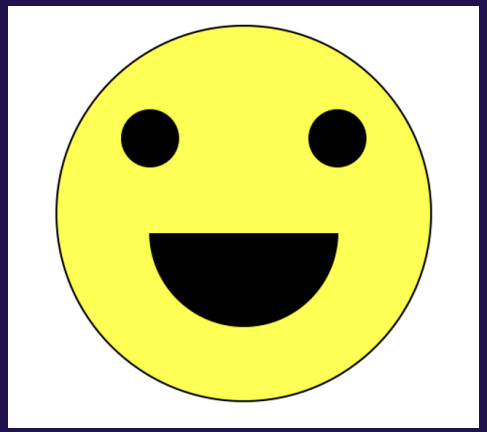
```
# create a window
win = GraphWin("Hello World", 400, 400)

# draw a rectangle with opposite corners at (10,50) and (145, 280)
rect = Rectangle(Point(10,50), Point(145,280))
rect.setFill("purple")
rect.draw(win)
```



Exercise #4

Write a program to draw a smiley face. Note that it doesn't have to be perfect! Just get the hang of using graphics objects.

Note that there is no "semi circle" drawing function, so you'll have to draw a complete circle, and draw a rectangle after the circle so it covers part of the circle. Make something like this:



Random Numbers

We can incorporate random numbers into our program by

```
import random
```

At the top of our code

Then, we can use a function such randint as follows:

```
# generate a random integer between 1 and 100 (including 1 and 100)
number = random.randint(1,100)
```

Computer Colors

In computer monitors, colors are made up of 3 different numbers between 0 and 255 that correspond to the amount of red, green, and blue light.

We can create a “random” color by selecting three different numbers to make the color like this:

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in _____

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in **black**

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in **black**
  - red = 255, green=255, blue = 255 results in _____

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in **black**
  - red = 255, green=255, blue = 255 results in **white**

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in **black**
  - red = 255, green=255, blue = 255 results in **white**
  - red = 255, green=0, blue = 255 results in _____

Computer Colors

Each pixel (picture element) has three components

- Red
- Green
- Blue
  - Values for each component range between 0 and 255
  - red = 0, green=0, blue = 0 results in **black**
  - red = 255, green=255, blue = 255 results in **white**
  - red = 255, green=0, blue = 255 results in **purple**

Random Colors

We can create a “random” color by selecting three different numbers to make the color like this:

```
# a random color
r = random.randint(0,255)
g = random.randint(0,255)
b = random.randint(0,255)

color = color_rgb(r,g,b)
```

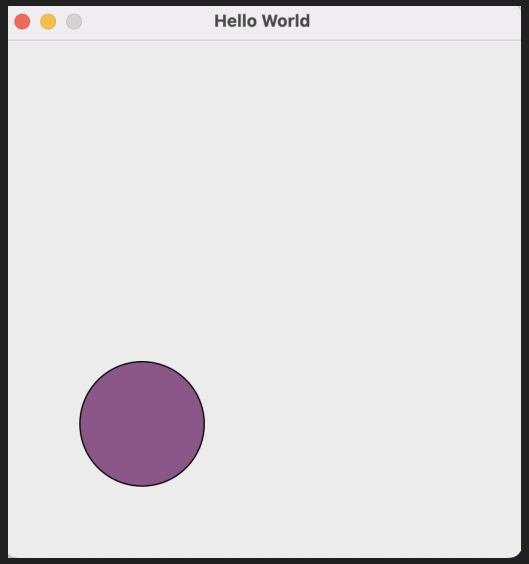
Random Locations

Similarly, we can select a random x and y values and a random radius for a circle by using **`randint`**:

```
# a random location
x = random.randint(0,350)
y = random.randint(0,350)

# random radius
radius = random.randint(10,50)
```

The world needs more random circles

Putting it all together, the following program will generate a random circle:

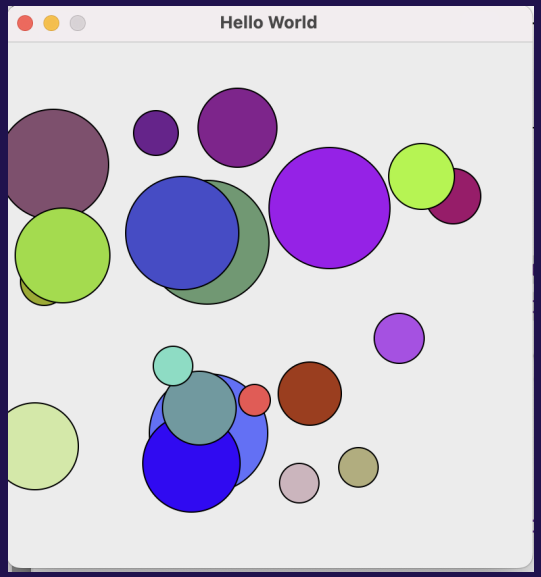



```
from graphics import *

import random

# create a window
win = GraphWin("Hello World", 400, 400)

# a random color
r = random.randint(0,255)
g = random.randint(0,255)
b = random.randint(0,255)

color = color_rgb(r,g,b)

# a random location
x = random.randint(0,350)
y = random.randint(0,350)

# random radius
radius = random.randint(10,50)

# a random circle
circ = Circle(Point(x,y),radius)
circ.setFill(color)
```

Exercise #5

Prompt the user for an integer. Use a for loop to run the random circle drawing algorithm the number of times the user entered to draw that number of random circles. For example:

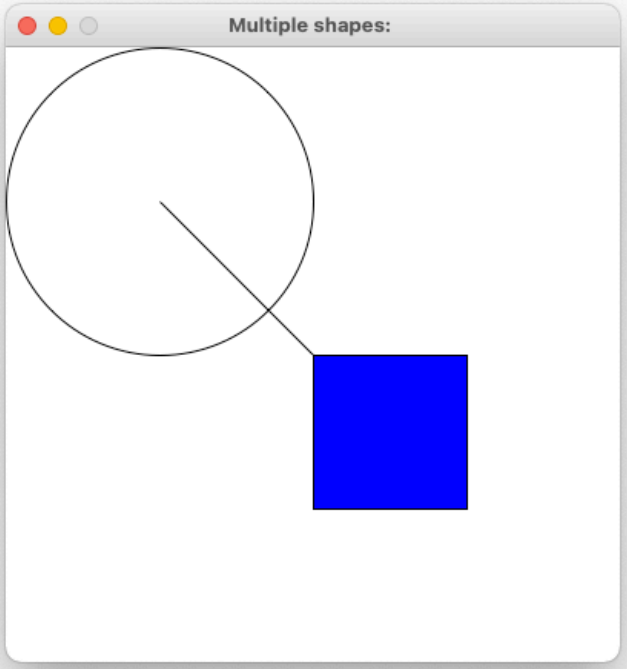
```
How many circles would you like?20
```



Draw multiple shapes

- Drawing circle, rectangle, and a line connecting them
- Demo

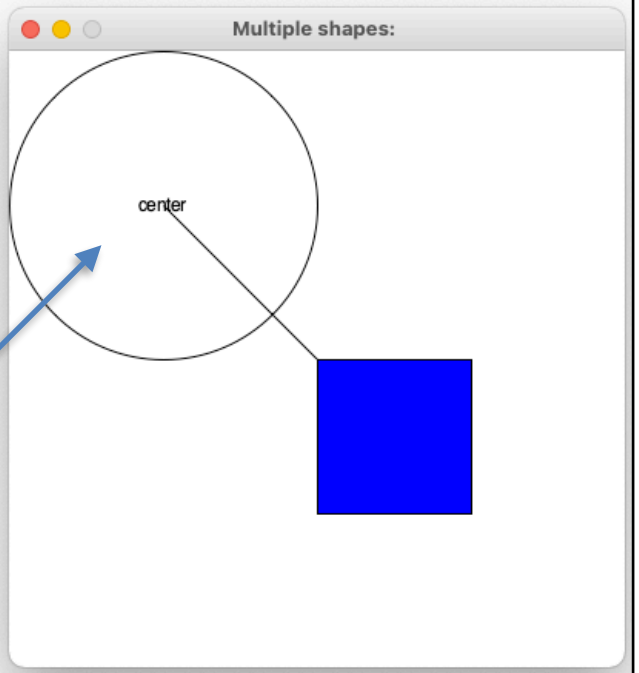
```
from graphics import *

def create_random_shapes():

    win = GraphWin("Multiple shapes: ", 400, 400)

    # create circle
    cir_center = Point(100, 100)
    cir_radius = 100
    my_cir     = Circle(cir_center, cir_radius)


    # create rectangle
    rect_point_a = Point(200, 200)
    rect_point_b = Point(300, 300)
    my_rect      = Rectangle(rect_point_a, rect_point_b)
    my_rect.setFill("blue")


    # line connecting circle and rectangle
    my_line      = Line(cir_center, rect_point_a)

    # draw all the components
    my_cir.draw(win)
    my_rect.draw(win)
    my_line.draw(win)

    return win


new_window = create_random_shapes()
```



Draw multiple shapes

- Drawing **Text** inside the window
- Demo

```
from graphics import *

def create_random_shapes():

    win = GraphWin("Multiple shapes: ", 400, 400)

    # create circle
    cir_center = Point(100, 100)
    cir_radius = 100
    my_cir     = Circle(cir_center, cir_radius)


    # create rectangle
    rect_point_a = Point(200, 200)
    rect_point_b = Point(300, 300)
    my_rect      = Rectangle(rect_point_a, rect_point_b)
    my_rect.setFill("blue")


    # line connecting circle and rectangle
    my_line      = Line(cir_center, rect_point_a)

    # text
    text_point   = Point(100, 100)
    my_text      = Text(text_point, "center")

    # draw all the components
    my_cir.draw(win)
    my_rect.draw(win)
    my_line.draw(win)
    my_text.draw(win)

    return win
```



Extra Challenges

try out the Polygon method--can you draw a triangle?

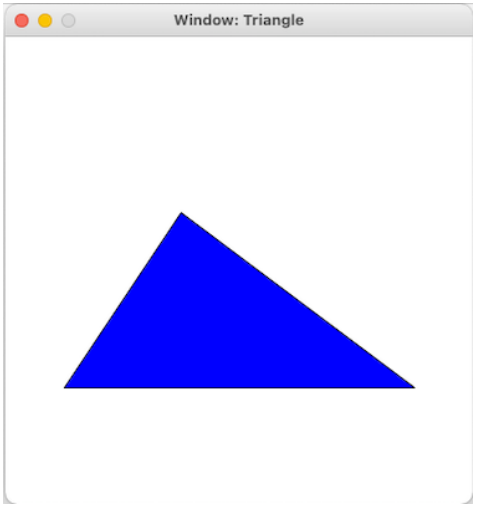
Draw something else cool--can you draw a scene? Maybe a dog, robot, beach, house, etc?

Extra Challenges

- Write a function that draws a **Triangle** based on
  - challenge 1: find out what how many variables do you need to draw it?
  - challenge 2: then receive that many user inputs

- Hints: read the specification below and try to figure out what might be a useful graphical object for this task

  https://mcsp.wartburg.edu/zelle/python/graphics/graphics/graphref.html

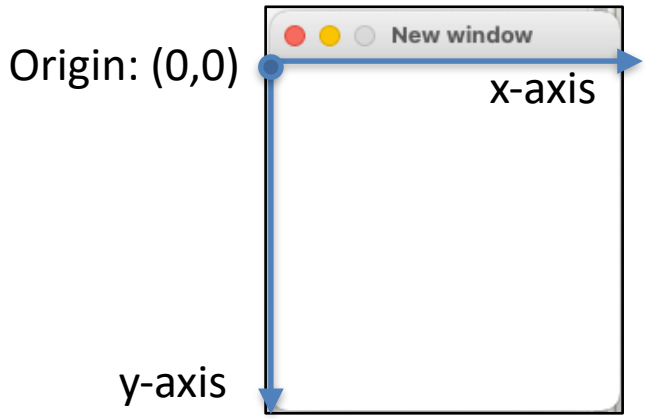


Topics

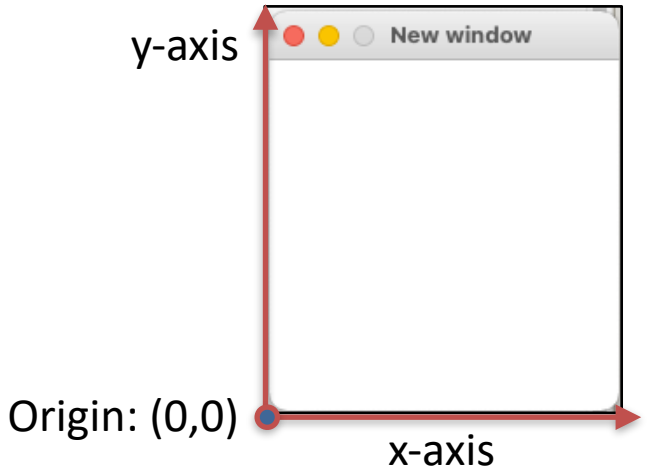
- Creating a graphical window
- Drawing shapes inside the window
  - Circle
  - Rectangle
  - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

Changing coordinate system

- Default coordinate system:
  - x: top-left —> top-right
  - y: top-left —> bottom-left

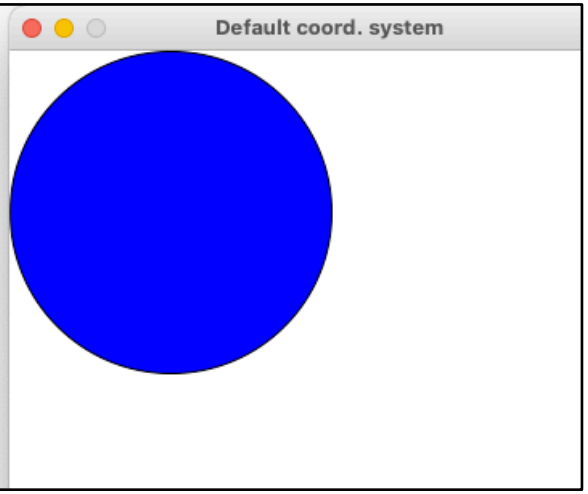


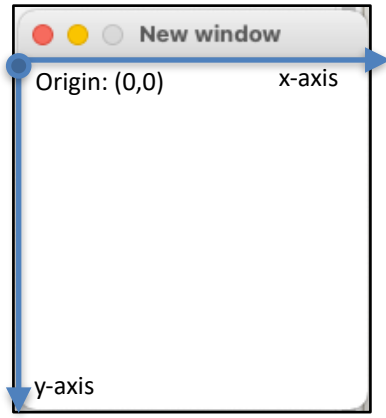
- Transforming into traditional coordinate system
  - x: bottom-left —> bottom-right
  - y: bottom-left —> bottom-up



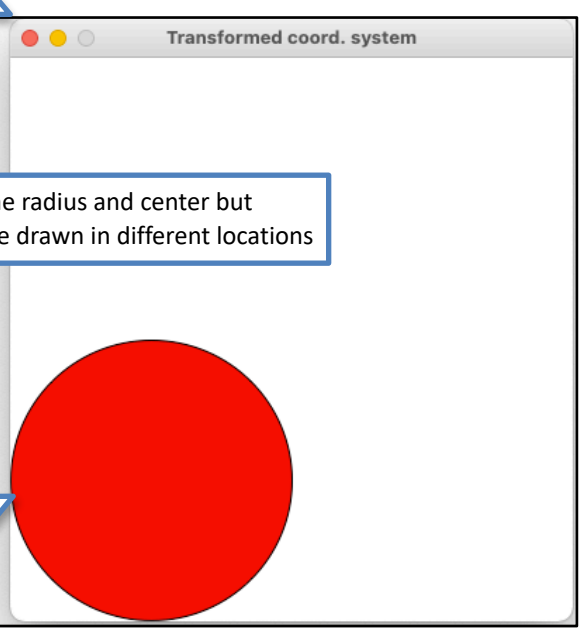
Changing coordinate system

```
from graphics import *


def create_circle_default_coord(px, py, radius):

    window = GraphWin("Default coord. system", 400, 400)

    # create circle
    cir_center = Point(px, py)
    cir_radius = radius
    my_cir     = Circle(cir_center, cir_radius)
    my_cir.setFill("blue")

    my_cir.draw(window)

    return window

win_default = create_circle_default_coord(100, 100, 100)
```

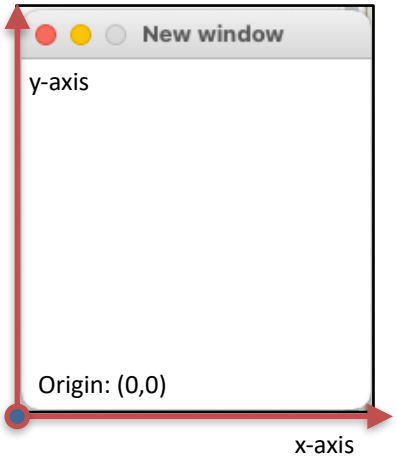





```
from graphics import *

def create_circle_transformed_coord(px, py, radius):


    window = GraphWin("Transformed coord. system", 400, 400)

    #---------- changing the coordinate system ----------
    # lower left  corner becomes 0,0
    # upper right corner becomes 400, 400
    window.setCoords(0,0, 400, 400)
    #----------------------------------------------------

    # create circle
    cir_center = Point(px, py)
    cir_radius = radius
    my_cir     = Circle(cir_center, cir_radius)
    my_cir.setFill("red")

    my_cir.draw(window)

    return window


# ------ lower left corner (0,0) and upper left corner (400, 400) ------
win_trans = create_circle_transformed_coord(100, 100, 100)
```

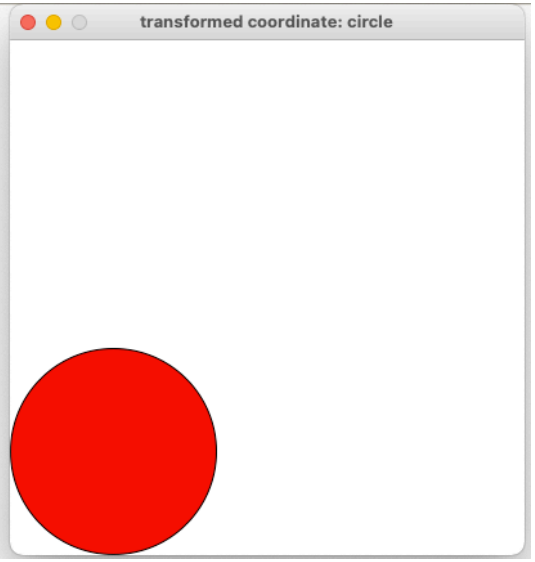


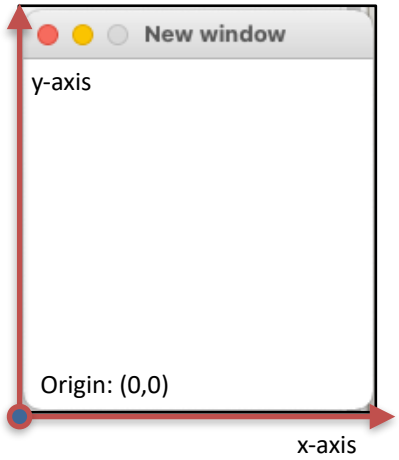
Same radius and center but
were drawn in different locations



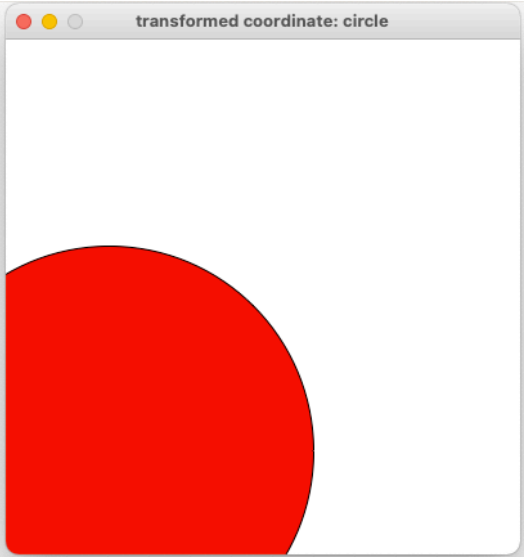
Changing coordinate system

```
from graphics import *

def create_transformed_window_v1(x, y, radius):

    window = GraphWin("transformed coordinate: circle", 400, 400)

    window.setCoords(-100, -100, 400, 400)

    circle_center = Point(x, y)
    circle_radius = radius
    tr_circle = Circle(circle_center, circle_radius)
    tr_circle.setFill("red")
    tr_circle.draw(window)

    return window

my_win = create_transformed_window_v1(0, 0, 100)
```





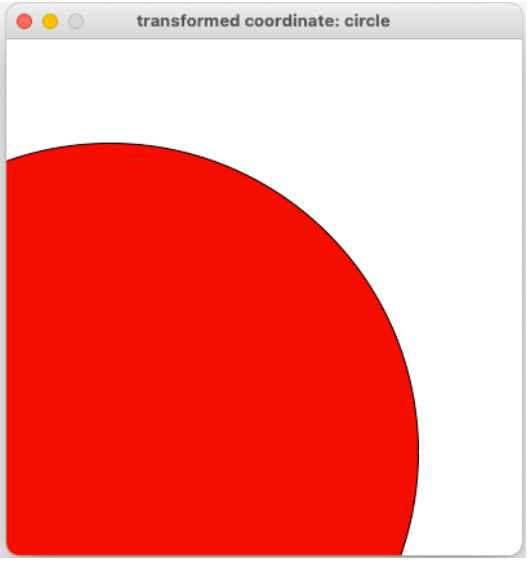

```
from graphics import *

def create_transformed_window_v1(x, y, radius):

    window = GraphWin("transformed coordinate: circle", 400, 400)

    window.setCoords(-100, -100, 400, 400)

    circle_center = Point(x, y)
    circle_radius = radius
    tr_circle = Circle(circle_center, circle_radius)
    tr_circle.setFill("red")
    tr_circle.draw(window)

    return window

my_win = create_transformed_window_v1(0, 0, 200)
```

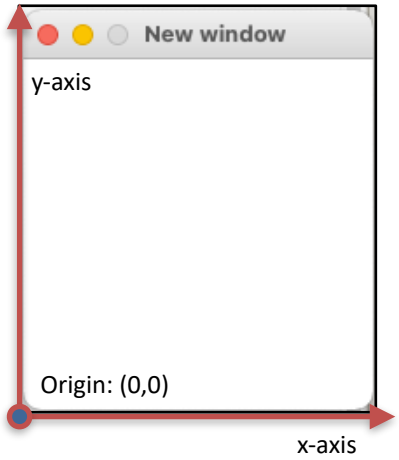


The red circle has been clipped

Changing coordinate system

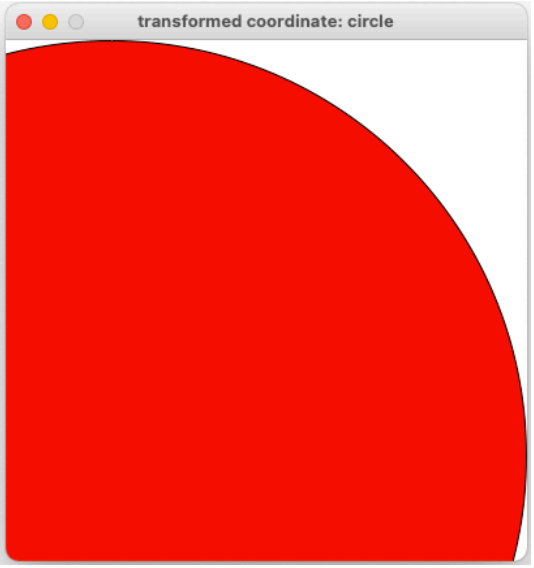
```
from graphics import *

def create_transformed_window_v1(x, y, radius):

    window = GraphWin("transformed coordinate: circle", 400, 400)

    window.setCoords(-100, -100, 400, 400)

    circle_center = Point(x, y)
    circle_radius = radius
    tr_circle = Circle(circle_center, circle_radius)
    tr_circle.setFill("red")
    tr_circle.draw(window)

    return window

my_win = create_transformed_window_v1(0, 0, 300)
```





The red circle has been clipped

```
from graphics import *

def create_transformed_window_v1(x, y, radius):

    window = GraphWin("transformed coordinate: circle", 400, 400)

    window.setCoords(-100, -100, 400, 400)

    circle_center = Point(x, y)
    circle_radius = radius
    tr_circle = Circle(circle_center, circle_radius)
    tr_circle.setFill("red")
    tr_circle.draw(window)

    return window

my_win = create_transformed_window_v1(0, 0, 400)
```



The red circle has been clipped

Topics

- Creating a graphical window
- Drawing shapes inside the window
  - Circle
  - Rectangle
  - Line, Text, and combinations of these shapes
- Changing coordinate system
- Mouse interaction inside graphics window

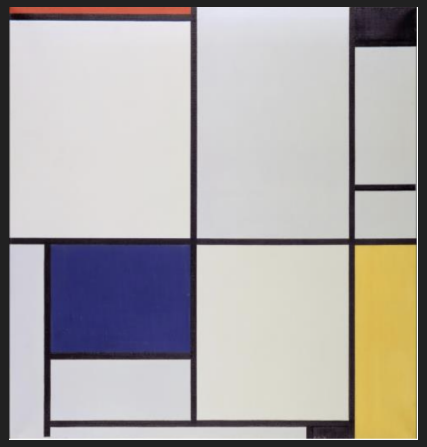
Interaction with the user using Mouse

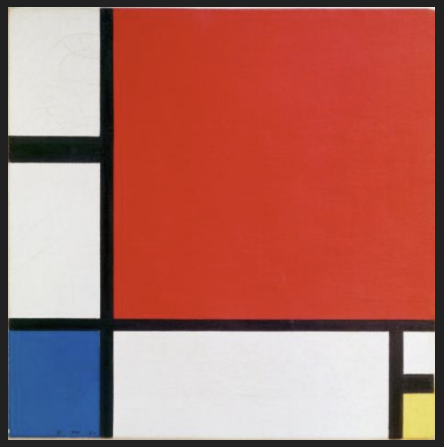
- GraphWin() has a function that allows us to identify the location of your mouse-click
  - 2D coordinate
  - represented by Point object

```
from graphics import *


window = GraphWin("Mouse interaction", 400, 400)

mouse_point1 = window.getMouse()
mouse_point2 = window.getMouse()

print(mouse_point1)
print(mouse_point2)
```

Lab #9

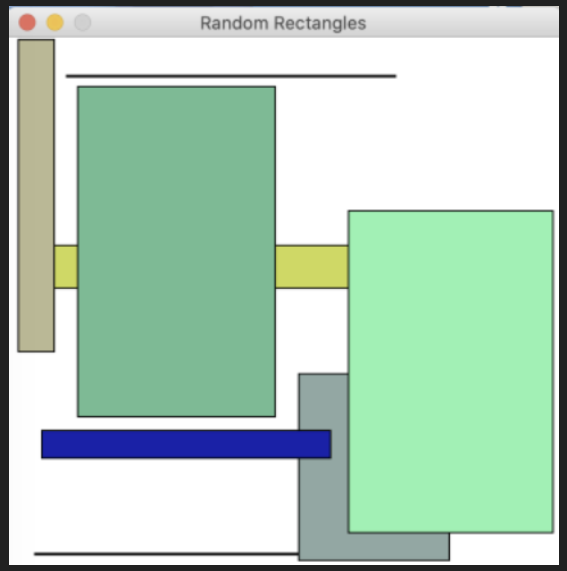
Piet Mondrian is an artist known as one of the pioneers of 20th-century abstract art. Here are a few of his paintings:

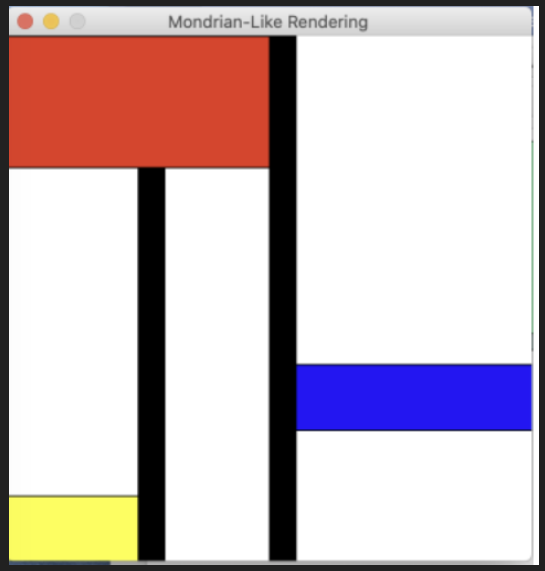




Lab #9

Create a Mondrian-inspired rendering. It can be a bunch of randomly-generated rectangles, or something that you make more intentional. Each of the following would be acceptable:





Lab #9

You are welcome to include all of the code you've developed for Lab 6, but I will be looking primarily for the random rectangles or Mondrian-inspired graphics that you have created.

Submit your code via CodePost before the deadline.