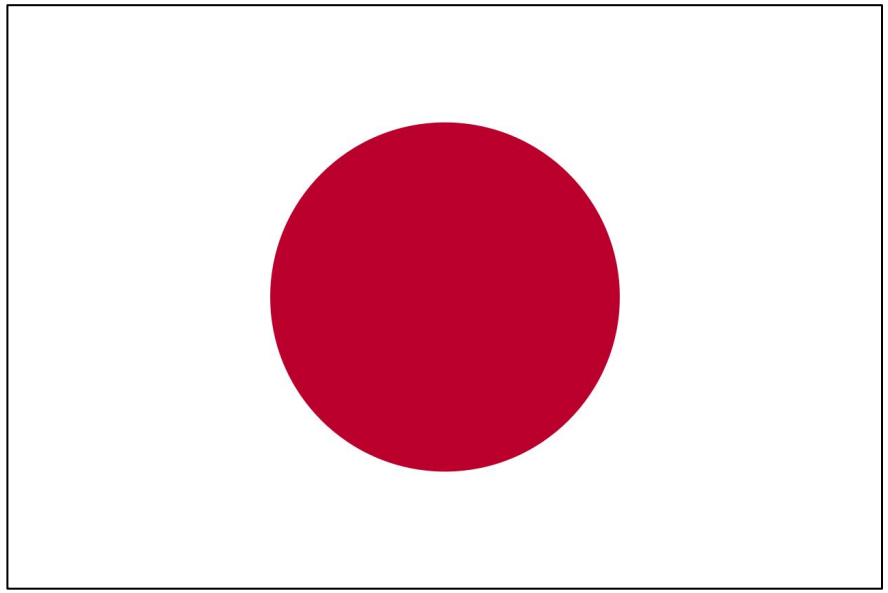
# Chapter 4: Objects and Graphics Part 1

Jan 21, 2020



# Today's Outline

- Review for Quiz 2
- Object-oriented programming
- Simple graphics programming with graphics.py
- Calculations with graphics.py

# Quiz 2

Quizzable topics:

Mostly will be focusing on **theory** from Chapter 3 Lectures.

Format: 10 Multiple Choice questions

Deadline: Sunday (Jan 26) at 11:55pm

# Quiz 2 Review

Which of the following is not a built-in Python data type?

a) int

b) float

c) rational

d) string

# Quiz 2 Review

Which of the following is not a built-in Python operation?

a) +

b) //

c) abs()

d) sqrt()

# Quiz 2 Review

The most appropriate data type for storing the value of pi would be:

a) int

b) float

c) rational

d) string

# Quiz 2 Review

The number of distinct values that can be represented with 5 bits is:

a) 5

b) 10

c) 32

d) 50

# Quiz 2 Review

Which of the following is not a Python type conversion function?

a) float

b) round

c) int

d) abs

# Quiz 2 Review

How many ways can I choose three desserts from a menu of eight desserts? (I don't care what order I choose the desserts in).

a)8!

b) 3!

c) 8*7*6

d) 8*7

# Quiz 2 Review

In Python, what is the result of 2 % 5?

a) 5

b) 2

c) 0

d) 0.4

# Quiz 2 Review

In Python, what is the result of 2 // 5?

a) 0

b) 2

c) 5

d) 0.4

# Quiz 2 Review

In Python, what is the result of the expression 2.0 + 5 // 2?

a) 4.0

b) 4.5

c) 4

d) None of the above

# Quiz 2 Review

Which of the following would be equivalent to the Python expression y = round(x)?

a) y = math.ceil(x)

b) y = math.floor(x)

c) y = int(x+0.5)

d) y = int(x)

# Quiz 2 Review

If you want to use the sin function in the math library, as
x = math.sin(3.14), how should you write the import statement?

a) An import statement is not needed

b) import math

c) import sin

d) from math import sin

# Quiz 2 Review

If you want to use the arange function in the numpy library, as

x = np.arange(100), how should you write the import statement?

a) import numpy as np

b) import np

c) import numpy

d) from numpy import np

# Quiz 2 Review

What is the result of float(int(3.5))?

a) 4

b) 4.0

c) 3.0

d) 3

# Character Design

Design your own fantasy character. Please define the following attributes for your character: Name, Species, Occupation, Special Abilities. Provide an intelligence, dexterity, strength, and charisma for your character on a scale of 1-10.

Then sketch a picture of your character.

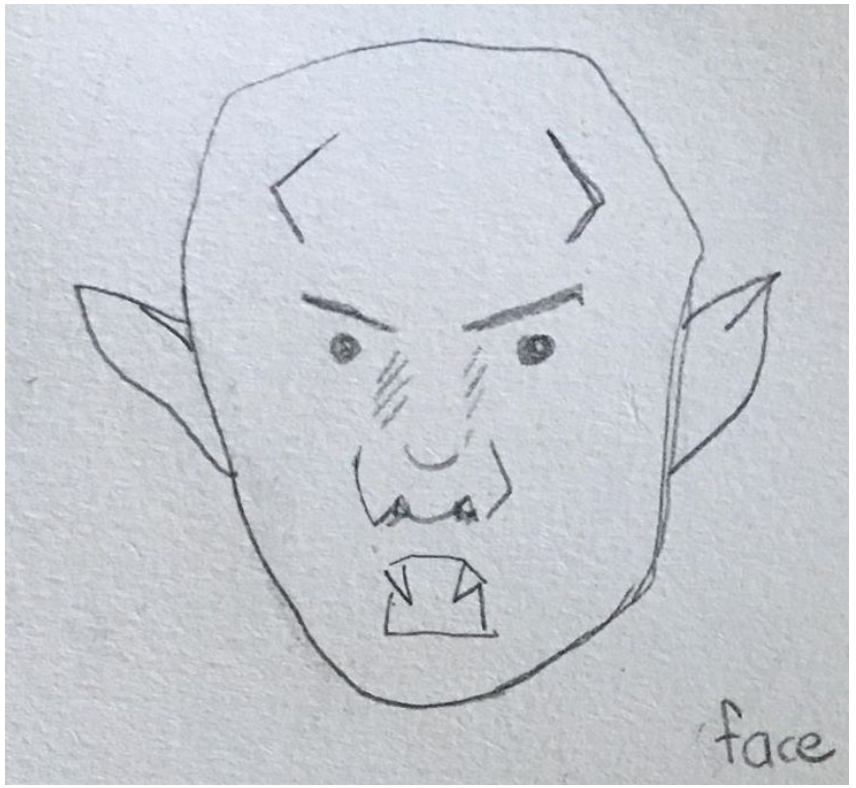
# Character Design Example

Name: Bolgod

Species: Orc

Occupation: Monk

Special Abilities: Martial Arts, Night vision, Speaks Orkish




Intelligence: 5 Strength: 10

Charisma: 1 Dexterity: 3

# Programming Paradigms

1. **Imperative programming**: Each line of code in the program is followed in order. This is how the CPU reads programs after they have been translated into machine code.

2. **Procedural programming:** The program is built using one or more functions

# Object-oriented Programming

- Object-oriented programming is a programming paradigm based on the concept of “objects”

**Purpose:** build more complex programs
build maintainable programs

# Class

A **class** is a blueprint of how to make an object

An **object** is an instance of a class.

There can be many objects/instances of a class.

# Objects

- Objects “know” stuff: can store information/data

- Objects “do” stuff: can perform operations

# Video Game Non-Player Character (NPC)

Class: video game NPC

**Properties:**

Name

Appearance

Speech

Skills

Likes

Dislikes

# Video Game Non-Player Character (NPC)

Class: video game NPC

**Methods:**

Talk to character

Give a gift

Purchase item

Fight character

# School Data Management System

Student:


**Properties**:

Name

ID Number

Courses taken

Address

**Methods**:

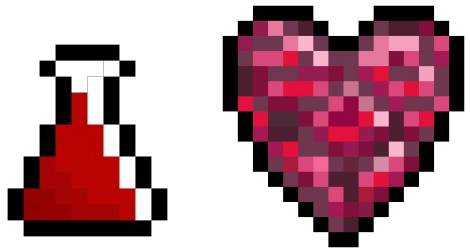
Print address

Calculate GPA

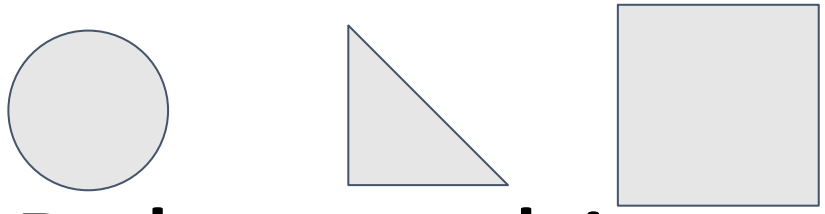
Email student about course

# 4 Object Oriented Programming Principles

Inheritance



Encapsulation



Polymorphism

Abstraction

# Child and Parent Classes

In object oriented programming, we may want to create different classes of objects that still have very similar properties.

# Video Game example

Parent class: items

Child classes:

Food

Weapons

Gems

# Video Game example

Item class: pick up, sell, give as a gift, put in backpack

Food methods: eat, cook

Weapon methods: swing, block

Gem methods: mine, refine

# School Data Management Example

Person class: Name, address, email

Student class: GPA, courses they are enrolled in

Teacher class: student evaluation rating, courses they are teaching

# Inheritance

The ability of object oriented programming to create many child classes from a parent class saves a lot of time when programming.

Each child class can have the necessary methods that it needs, while preserving logic from the parent class.

# Encapsulation

Objects communicate with each other through methods.

Each object can only be changed by using methods.

# Video Game Example

Give a character more life by methods: drinkHealthPotion or eatFood

The internal state, character life, can’t be changed other than by interacting with the character object through the methods.

# School Data Management Example

Students, instructors, rooms, times are all defined as classes.

Objects are created based on each class.

Each object interacts with each other using methods.

Example: enrollInClass

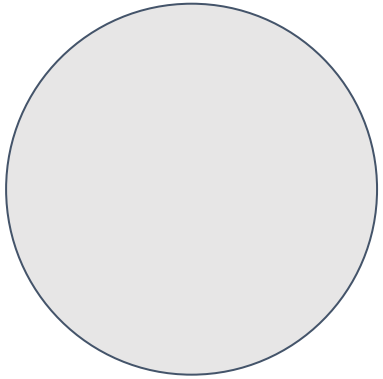
# Polymorphism

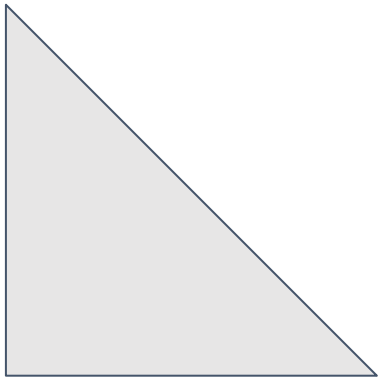
Polymorphism means “many shapes” in greek.

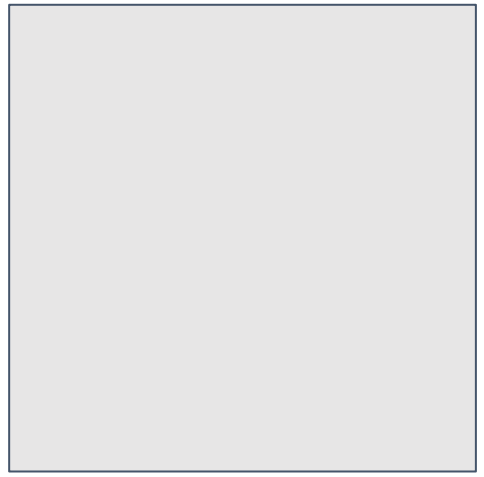
It means that we can use the same methods for many different child classes and parent classes, even if the underlying mechanics are different

# Shape Example

findArea method works for shapes of class circle, square, and triangle, even though the underlying formula is different for each shape







# Video Game Example

Make more child classes for my weapons class

swingWeapon method works if the weapon is a sword, dagger, or club although underlying game physics for each class is different.

# Abstraction

The methods for each object should be simple, and the internal mechanisms of each object are hidden.

For example, if I want to give a gift to an NPC, I shouldn’t have to understand all of the mechanisms behind the NPC. If I want to add a student to my class, I don’t want to have to understand how the underlying course class works.

# Abstraction

Abstraction helps programmers:

- working on a big program (like a video game or data management system)
- working on a program that has to be maintained and updated over time (may have to change internal mechanism of the classes, but don’t want to entirely change the way that objects communicate with each other)

# 4 Object Oriented Programming Principles

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260

# Graphics Programming

- Use graphics.py library provided by the textbook
- May need to use pip installer to install
-
-
- (pip install graphics.py)

# Graphics Window

```
import graphics
#GraphWin function creates a new window
win = graphics.GraphWin( )

#This is a GraphWin object that is named win

#close the window
win.close ( )
```

# Importing Libraries Review

```
import math
#have to put math.math_function_name
# ex. math.pi() , math.cos()

import numpy as np
#have to put np.numpy_function_name
ex. x = np.arange(10)
```

# Importing Libraries Review

import numpy as purple_elephant

#have to put purple_elephant.numpy_function_name

ex. x = purple_elephant.arange(10)

from numpy import *

#can directly use the numpy function as numpy_function_name

ex. x = arange(10)

# Importing Libraries Review

```
from graphics import *
win = GraphWin( )
```

Textbook recommends this to avoid having to write “graphics.” every time we want to use a graphics function.

# Pixels

To draw a picture, we need to assign colour values to each pixel.

Procedural programming:

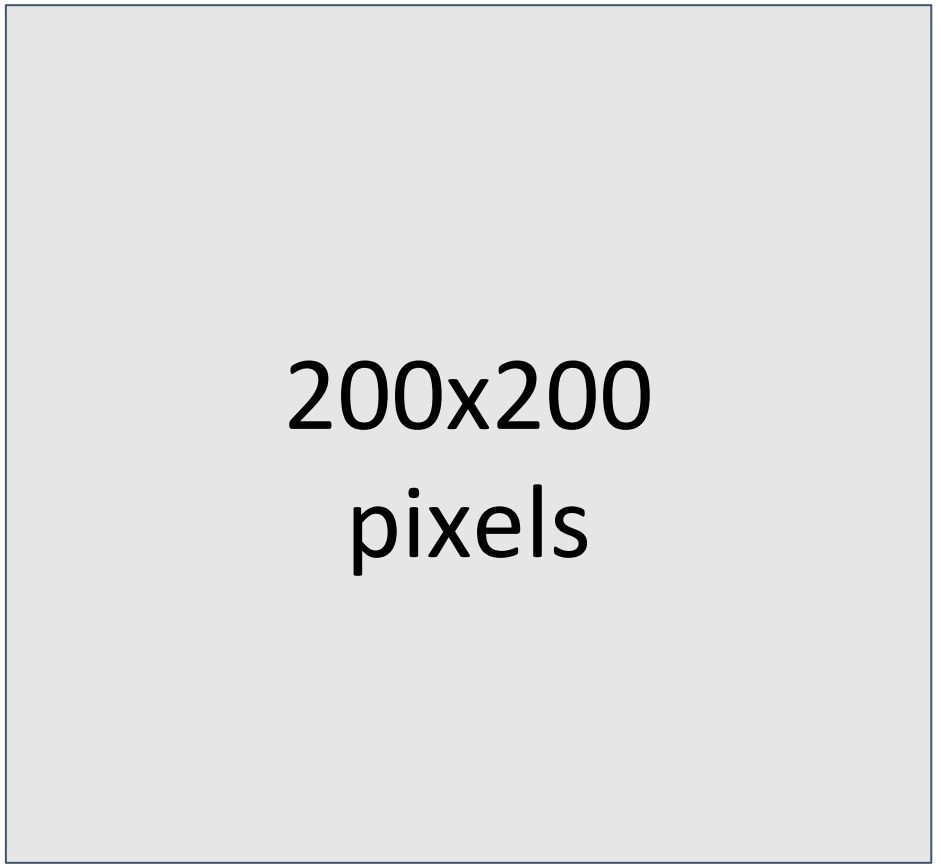
- need to figure out how to assign each pixel for the shapes we want

Object oriented programming:

- Developed classes for each type of shape a user might want
- Now we can assign pixels by assigning graphics objects

# Cartesian Coordinates (0,0)

Traditionally (x,y) = (0,0) is in the top left corner of the graphics window.

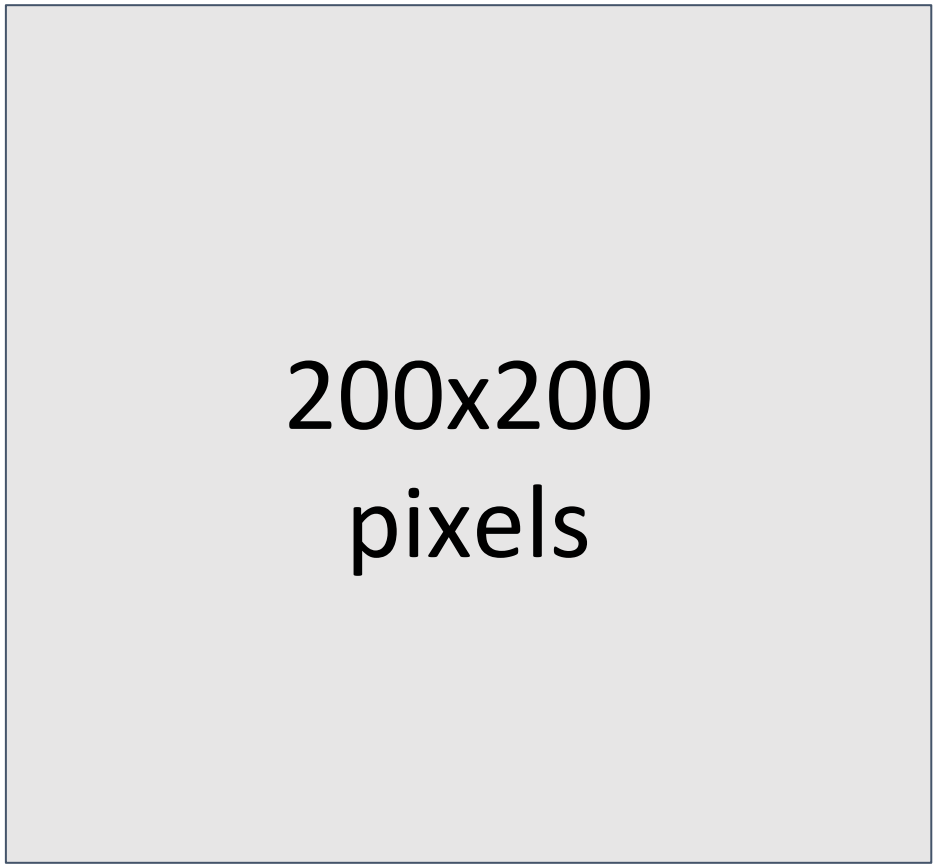


# Cartesian Coordinates

Traditionally (x,y) = (0,0) is in the top left corner of the graphics window.

# Point Object

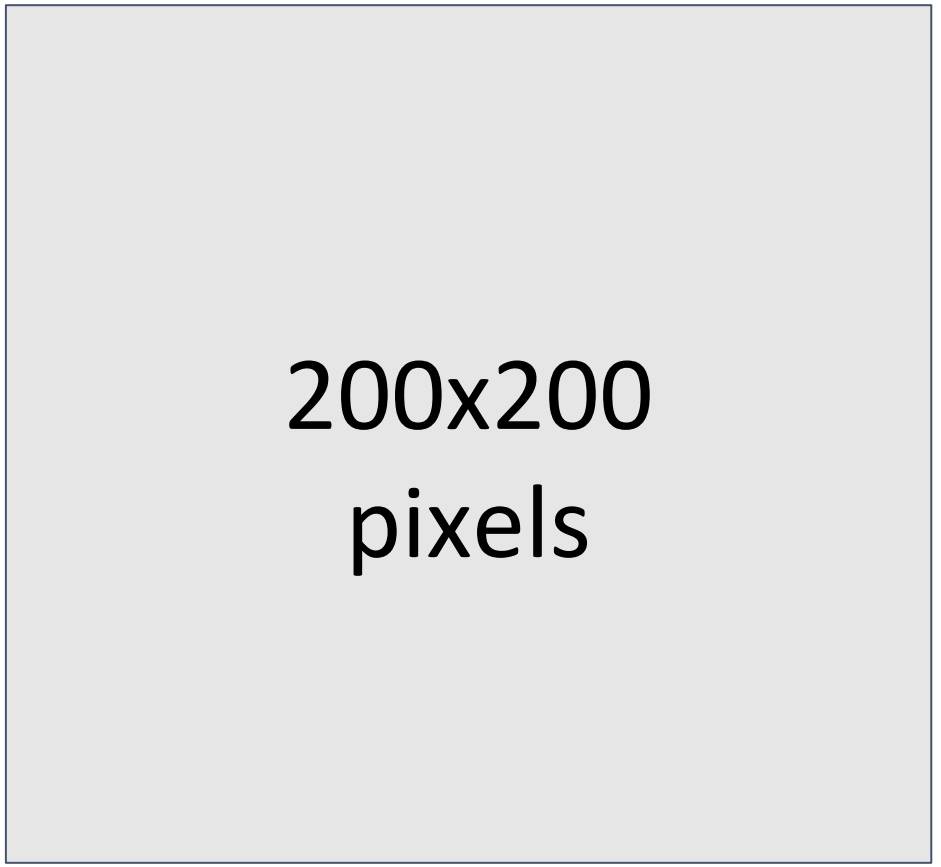
```
p = graphics.Point(60,70)
#sets pixel at the point (60,70) to be black
```




```
#Methods
p.getX()
p.getY()
```

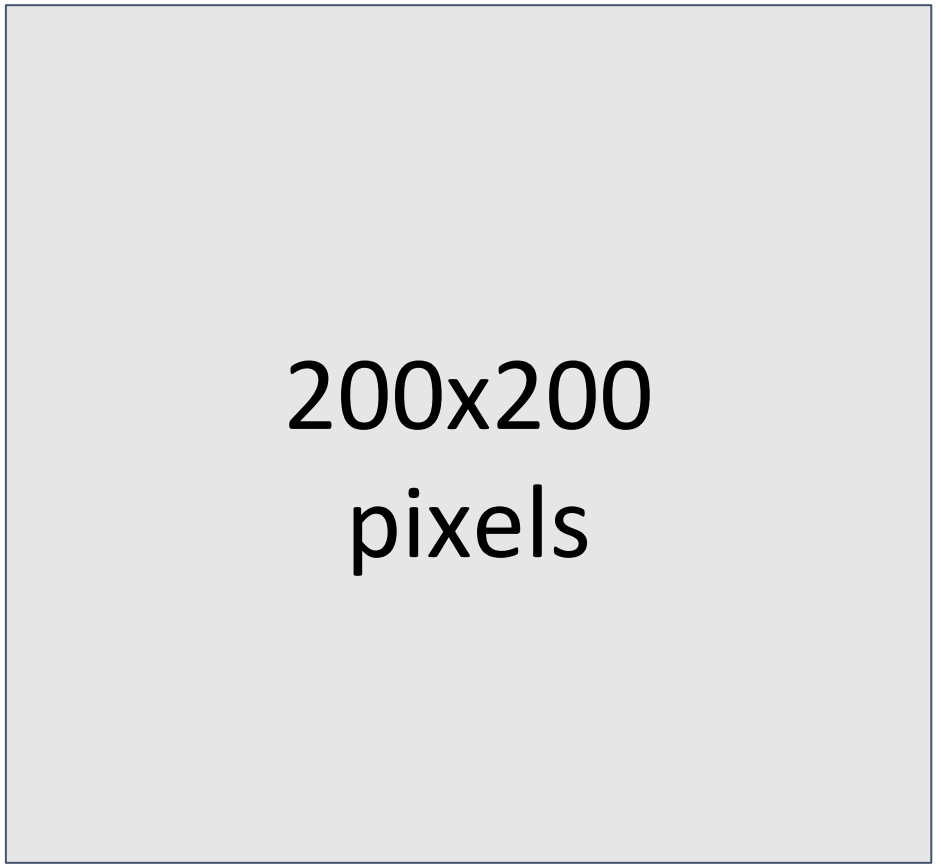
# Point Object

```
p = graphics.Point(60,70)
#Drawing the point requires a draw method

win = graphics.GraphWin()
p.draw(win)
```



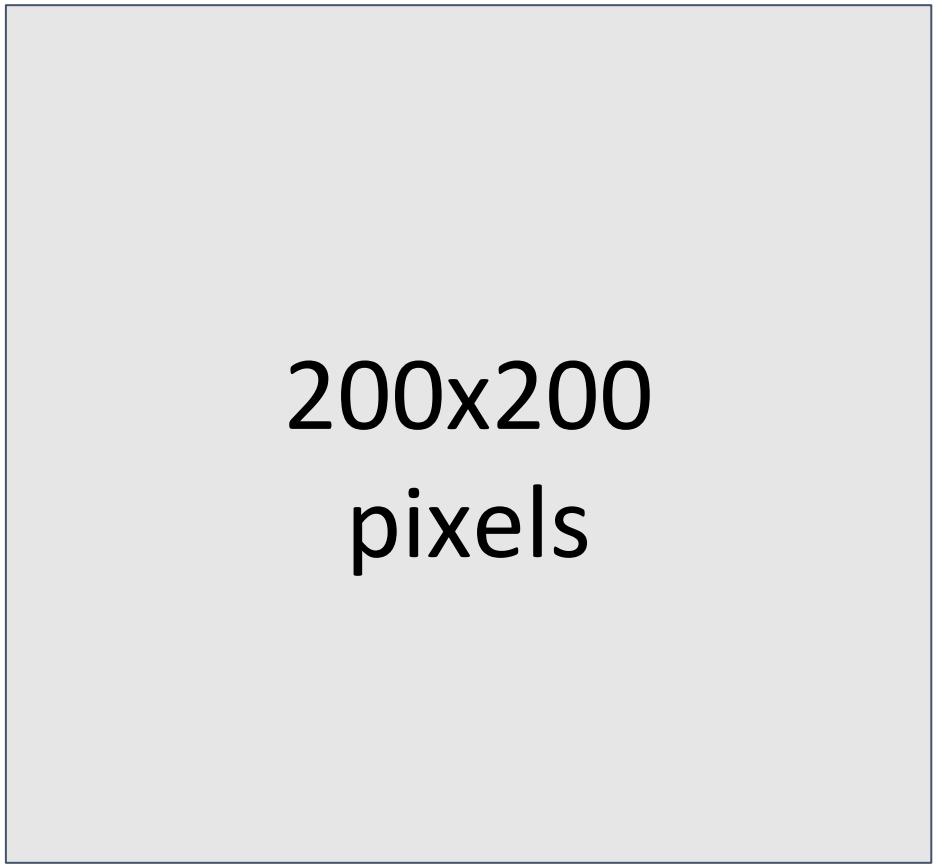
# Point Object

```
#What will the result be?


import graphics
win = graphics.GraphWin()
p1 = graphics.Point(150,100)
p2 = graphics.Point(10,70)
p1.draw(win)
p2.draw(win)
```



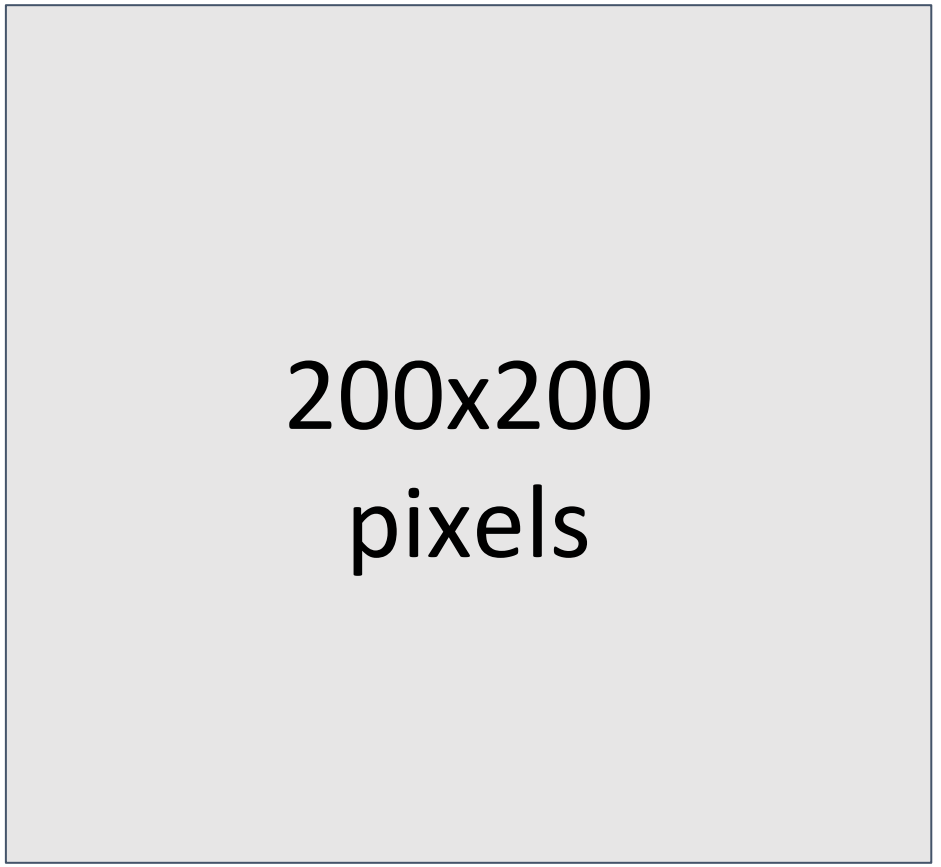
# Circle Object

```
win = graphics.GraphWin()
centre = graphics.Point(100,100)
circ = graphics.Circle (centre, 30)
circ.setFill("red")
circ.draw(win)
```



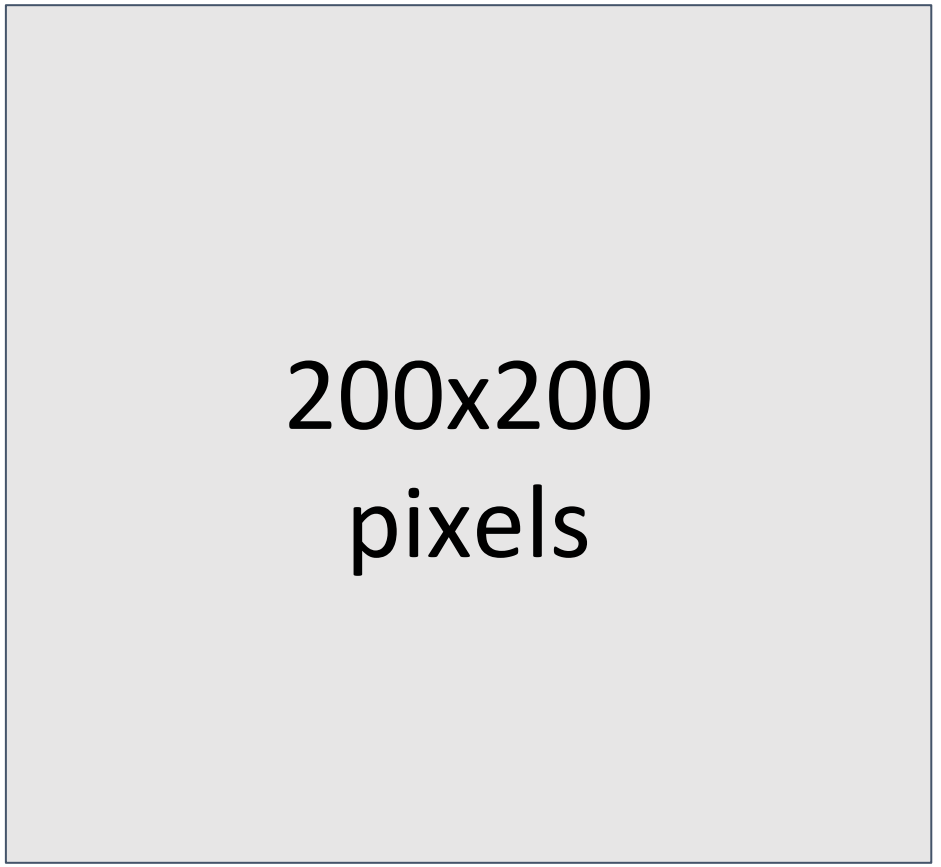
# Label Object

```
win = graphics.GraphWin()
centre = graphics.Point(100,100)
circ = graphics.Circle (centre, 30)
circ.setFill('red')
circ.draw(win)

label = graphics.Text(centre, "Red Circle")
label.draw(win)
```



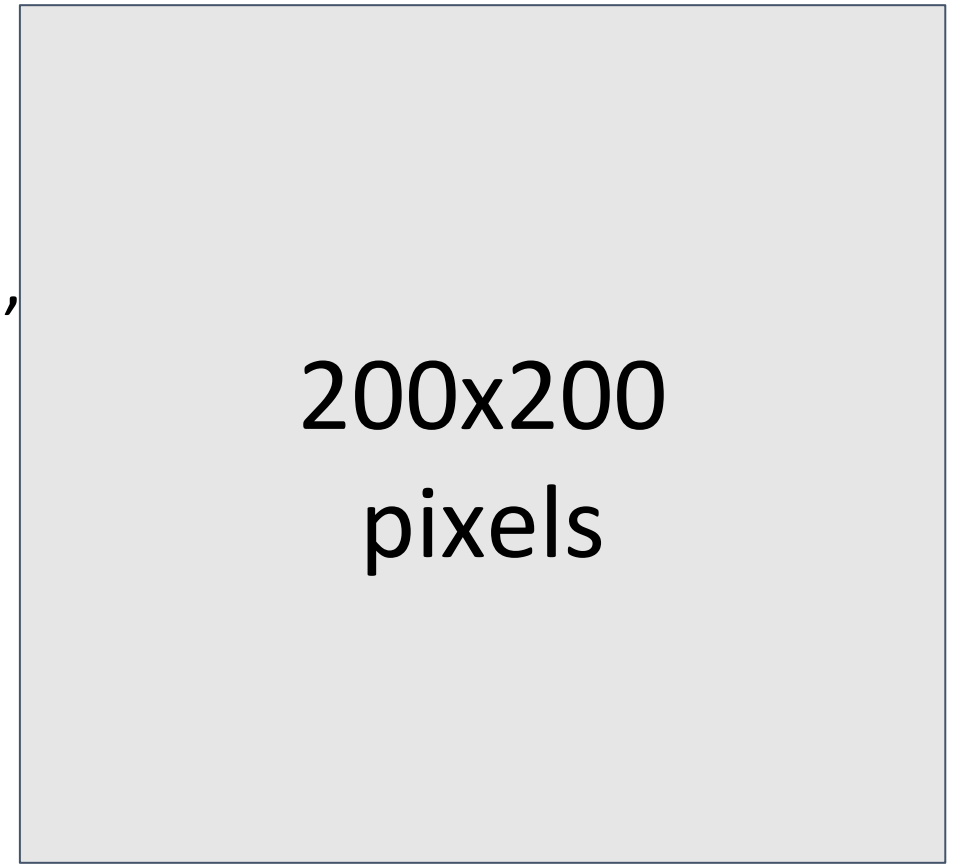
# Line Object

```
win = graphics.GraphWin()
a = graphics.Point(100,100)
b = graphics.Point(0,0)
line = graphics.Line(a,b)
line.draw(win)
```



# Oval Object

```
win = graphics.GraphWin()
oval = graphics.Oval(graphics.Point(20,50),
graphics.Point(180,199))
oval.draw(win)
```



# Object Oriented Graphics

What are the classes?

What are the objects?

# Creating new objects

a = graphics.Point(100,100)

Constructor: expression used to create a new object

new_object = Constructor (parameter1, parameter2)

# Creating new objects

a = graphics.Point(100,100)

Class: Point

Object: a

We have created an instance of the Point class, with x = 100, y = 100 and assigned it to the variable a.

# Accessors

Methods that send the object a message are called **accessors**, because they help to access information about an object.

methods: getX(), getY()

a.getX()

a.getY()

# Mutators

Methods that change the object state are called **mutators**.

move(dX, dY)

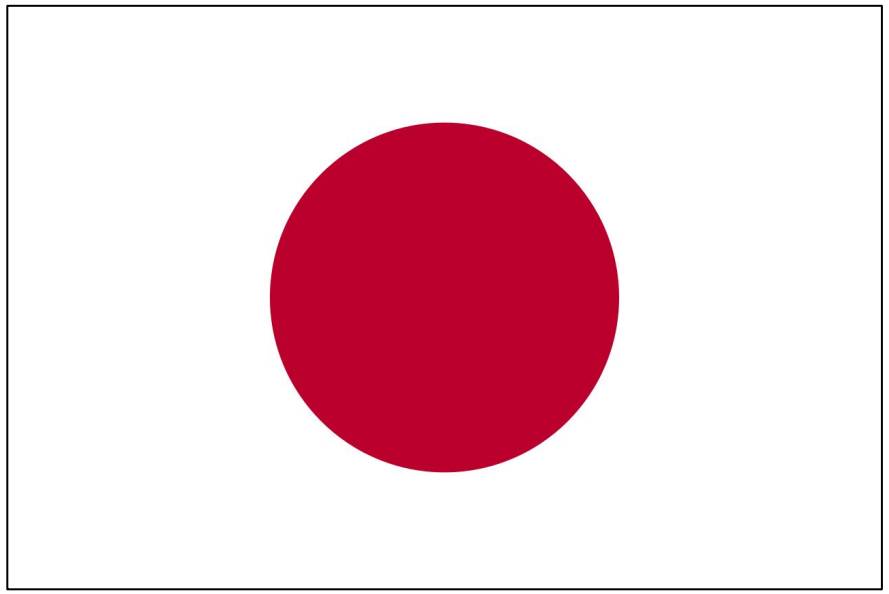
a.move(10,0)
#moves a 10 pixels to the right

# Graphics Objects

The graphics module provides the following classes of drawable objects: Point, Line, Circle, Oval, Rectangle, Polygon, and Text.

# Flag Problem

Write a program that can draw the flag of Japan.



Write another program that can draw the flag of France.

# Character Design Example

Determine how you would draw a representation of your fantasy character using the given classes, (Point, Line, Circle, Oval, Rectangle, Polygon, and Text).