CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson
Fall 2018
Swarthmore College
Informal quiz (discuss with a partner)

1) c is an _______ of the Circle _______.

2) GraphWin(..), Point(..), and Circle(..) are all _______.

3) width/2, height/2, “white”, “blue” are all _______.

4) setFill(..), setOutline(..), and draw(..) are all _____ not _______.

```python
width = 600
height = 600
win = GraphWin("Random Circles", width, height)
win.setBackground("white")

p = Point(width/2, height/2)
c = Circle(p, 10)
c.setFill("blue")
c.setOutline("blue")
c.draw(win)
```
Informal quiz (discuss with a partner)

1) \( c \) is an \textit{instance} of the \textit{Circle class}.

2) \texttt{GraphWin(..)}, \texttt{Point(..)}, and \texttt{Circle(..)} are all ________.

3) \( \texttt{width/2, height/2, \textquote{white}, \textquote{blue}} \) are all ________.

4) \texttt{setFill(..)}, \texttt{setOutline(..)}, and \texttt{draw(..)} are all _____ not _____.

```python
width = 600
height = 600
win = GraphWin("Random Circles", width, height)
win.setBackground("white")

p = Point(width/2, height/2)
c = Circle(p, 10)
c.setFill("blue")
c.setFill("blue")
c.draw(win)
```
Informal quiz (discuss with a partner)

1) c is an instance of the Circle class.

2) GraphWin(..), Point(..), and Circle(..) are all constructors.

3) width/2, height/2, “white”, “blue” are all ________.

4) setFill(..), setOutline(..), and draw(..) are all ______ not ______.

```python
width = 600
height = 600
win = GraphWin("Random Circles", width, height)
win.setBackground("white")

p = Point(width/2, height/2)
c = Circle(p, 10)
c.setFill("blue")
c.setOutline("blue")
c.draw(win)
```
Informal quiz (discuss with a partner)

1) \( c \) is an instance of the Circle class.

2) `GraphWin(..), Point(..), and Circle(..)` are all constructors.

3) `width/2, height/2, “white”, “blue”` are all arguments/parameters.

4) `setFill(..), setOutline(..), and draw(..)` are all ______ not _______.
Informal quiz (discuss with a partner)

1) c is an instance of the Circle class.

2) GraphWin(..), Point(..), and Circle(..) are all constructors.

3) width/2, height/2, “white”, “blue” are all arguments/parameters.

4) setFill(..), setOutline(..), and draw(..) are all methods not functions.

```python
width = 600
height = 600
win = GraphWin("Random Circles", width, height)
win.setBackground("white")

p = Point(width/2, height/2)
c = Circle(p, 10)
c.setFill("blue")
c.setFill("blue")
c.draw(win)
```
Outline Oct 10:

• Recap OOP vocabulary & followups from Mon
• Continue graphics
• User clicks
• Getters and setters
• Moving box program (*box.py*)
• Falling snow program (*snow.py*)
• Hand back Quiz 2 & go over common issues

Notes

• *Lab 5* due *Friday* (in-class) and *Saturday* night
• Office Hours 2-4pm on Thursday (just this week!)
# Graphics Reference and Notes

<table>
<thead>
<tr>
<th></th>
<th>Oct 08</th>
<th></th>
<th>Oct 10</th>
<th></th>
<th>Oct 12</th>
<th></th>
</tr>
</thead>
</table>

## Graphics, Using Objects

- object-oriented programming
- create objects and call methods
- getter and setter methods
- dot notation
- intro to graphics
- OOP using the graphics library
- RGB colors
- animation

- **Ch 8: Graphics Case Study**
- **Notes on the Zelle graphics library**
- Zelle's Chapter 5: Objects and graphics
- Zelle Graphics Module

Recommended by ninjas!
Why doesn’t reassigning \(c\) in the loop overwrite our last circle?

```python
# create 200 random circles using a for loop
for i in range(200):

    # choose a random point for the center and create the circle object
    x = random.randrange(width)
    y = random.randrange(height)
    p = Point(x, y)
    c = Circle(p, 10) # 10 is the radius

    # select a random color
    color = random.choice(color_lst)
    c.setFill(color) # change the inside "fill" color
    c.setOutline(color) # change the outside "outline" color
    c.draw(win) # draw the circle on the window (not automatic!)
```

See stack explanation from class
Different ways of animating

- Make movement very small
  ```python
  >>> from graphics import *
  >>> win = GraphWin("hello",400,400)
  >>> c = Circle(Point(200,200),100)
  >>> c.draw(win)
  Circle(Point(200.0, 200.0), 100.0)
  >>>
  >>> for i in range(100000):
      ...     c.move(0.01,0)
  ```

- Use `time.sleep(<sec>)`
  ```python
  >>> from graphics import *
  >>> win = GraphWin("hello",400,400)
  >>> c = Circle(Point(200,200),100)
  >>> c.draw(win)
  Circle(Point(200.0, 200.0), 100.0)
  >>> import time
  >>> for i in range(100):
      ...     c.move(1,0)
      ...     time.sleep(1)
  ```

- Call `update()`
  ```python
  >>> from graphics import *
  >>> win = GraphWin("hello", 400, 400, autoflush=False)
  >>> c = Circle(Point(200,200),100)
  >>> c.draw(win)
  Circle(Point(200.0, 200.0), 100.0)
  >>> for i in range(100):
      ...     c.move(1,0)
      ...     update()
  ```
How to get all the colors?

```
from matplotlib import colors as mcolors
.colors = dict(mcolors.BASE_COLORS, **mcolors.CSS4_COLORS)
.color_lst = list(colors.keys())
print(color_lst)
```


https://matplotlib.org/examples/color/named_colors.html
Continue Graphics
GraphWin class

- **GraphWin**(title, width, height) – constructs a new graphics window (default width and height are both 200)

- **setBackground**(color) – set the background color

- **close()** – closes the window

- **getMouse()** – waits for the user to click, returns the click position as a **Point**

- **checkMouse()** – does not wait for the user to click, returns the click position as a **Point**, or None if no position clicked
Methods for all Graphics Objects

- `setFill(color)` – sets the interior color of an object
- `setOutline(color)` – sets the outline color of an object
- `setWidth(pixels)` – sets the outline width (doesn’t work for `Point`)
- `draw(window)` – draws the object on the given window
- `undraw()` – removes the object from a graphics window
- `move(dx, dy)` – moves the object dx in the x direction and dy in the y direction
- `clone()` – returns a duplicate (new copy) of the object
Point class

- **Point**(x,y) – constructs a new point at the given position
- **getX()** – returns the current x coordinate
- **getY()** – returns the current y coordinate
**Line class**

- **Line(point1, point2)** – constructs a line from point1 to point2
- **setArrow(string)** – sets the arrowhead of a line ("first", "last", "both", "none")
- **getCenter()** – returns the midpoint of the line
- **getP1(), getP2()** – returns a clone of the corresponding endpoint
Circle class

- **Circle(center, radius)** – constructs a circle at the given position and with the given radius
- **getCenter()** – returns a clone of the center point
- **getRadius()** – returns the radius
- **getP1(), getP2()** – returns a clone of the corresponding corner of the circle’s bounding box (upper left, lower right)
Rectangle class

- `Rectangle(point1, point2)` – constructs a rectangle with opposite corners at the given points (upper left, lower right)
- `getCenter()` – returns the center point
- `getP1()`, `getP2()` – returns a clone of the corner point
Polygon class

- **Polygon(point1, point2, point3, ...)** – constructs a polygon with the given points as vertices (also accepts a list of points)
- **getPoints()** – returns a list of the points in the polygon
User clicks and getters

- `win.getMouse()` waits for the user to click
- It returns the user’s click as a `Point`
- We can use that `Point` later on or extract the x and y coordinates using a `getter`

```python
click = win.getMouse()
print(click)
x = click.getX()  # getter for x coordinate
y = click.getY()  # getter for y coordinate
print(x_click, y_click)

c = Circle(click, 10)
center = c.getCenter()  # getter (what is the type of center?)
```
Programs for today
Diagram for box.py
Work with a partner on one computer!

- *Pair programming* is frequently used in upper level CS classes and afterward in industry/academia
- One person is the *driver* at the keyboard (typing)
- The other person is the *navigator* who is providing advise, feedback, etc.
- Switch frequently between roles

- `cs21/inclass/w06/box.py` (first)
- `cs21/inclass/w06/snow.py` (second)