Week 4: Agenda

- Mentor meetings:
  - optional this week, but required from next week

Code Tracing (Strings)

```
def ct(s, t):
    r = s
    t = s.replace('a','d')
    t = t[::-1]
    print(s, t) # Do not miss this!
    r += "\n"
    for i in range(min(len(s), len(t))):
        if s[i] == t[i]:
            r += s[i] + "\n"
    return r + "bye"
print(ct('abcad', 'abd'))
```

Example: `longestNonRepeatingSubstr(s)`

- Write the function `longestNonRepeatingSubstr(s)` that takes a string `s` and returns the longest **substring** with all non-repeating characters. Resolve ties using **lexicographic order**.
- Examples:
  - `longestNonRepeatingSubstr("qwertqwer")` returns **`"ertqw"`**
  - `longestNonRepeatingSubstr("ababa")` returns **`"ab"`**

```
longestNonRepeatingSubstr("qwertqwer")
```

Understanding the Problem

- What's a substring?
- What's a non-repeating substring?
- Longest one = maximal length
  - If ties, lexicographic order - What's lexicographic order? (~ alphabetical)
- How to solve?
  - Usually, brute force (like other substring-related problems)
    - Check **all possible** substrings

```
longestNonRepeatingSubstr("qwertqwer")
```

Enumerate all substrings:

Length 0

`''`

Length 1

`q, w, e, r, t, q, w, e ,r`

Length 2

`qw, we, er, rt, tq, qw, we, er`

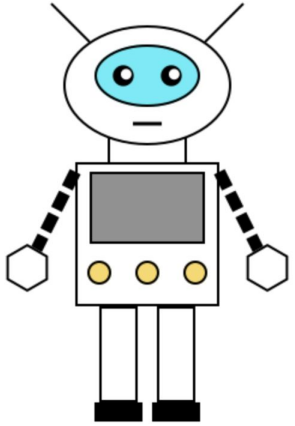
Length 3

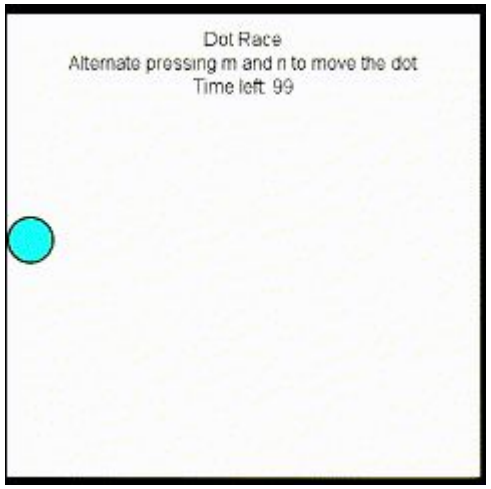
`qwe, wer, ert, rtq, tqw, qwe, wer`

...

Graphics and Animation Tasks

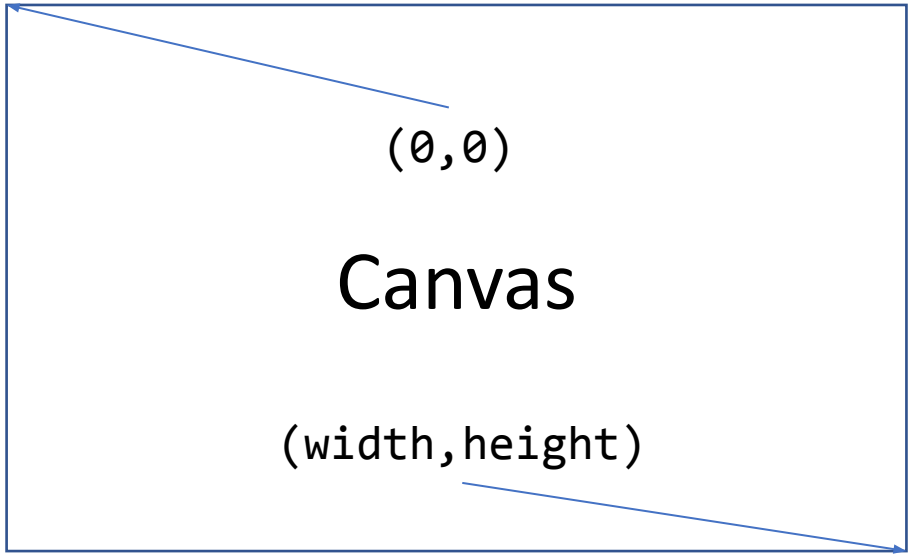
- Drawing: Create or replicate visuals by combining shapes, colors, and attributes.
- Animating
  - Capture and respond to mouse and keyboard events.
  - Use timers to update drawings over time, creating dynamic effects.



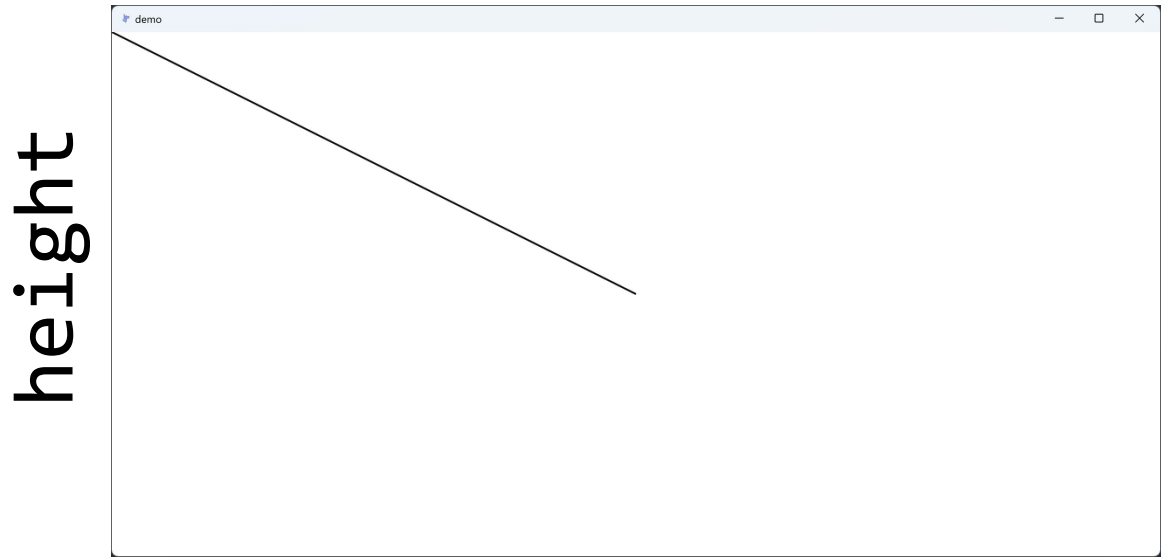


cmu_graphics

```
1  from cmu_graphics import *
2
3  def redrawAll(app):
4      # < your code >
5
6  runApp()
```



```python
from cmu_graphics import *

def redrawAll(app):
    drawLine(0,0,app.width//2, app.height//2)

runApp(1200, 600)
```



Shapes (and text)

- drawLine
- drawRect
- drawCircle
- drawOval
- drawStar
- drawLabel

CMU CS ACADEMY

Shape Parameters Reference Sheet

✔ Shape has this property

✖ Shape does not have this property

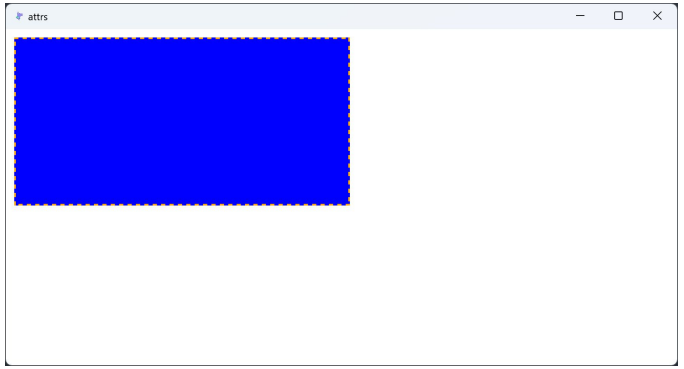
	fill	border	borderWidth	opacity	rotateAngle	dashes	align	visible	roundness	size	font	bold	italic	lineWidth	arrowStart	arrowEnd
default	'black'	None	2	100	0	False	'center'	True	None	12	arial	False	False	2	False	False
Rect(left, top, width, height)	✔	✔	✔	✔	✔	✔	'left-top'	✔	✖	✖	✖	✖	✖	✖	✖	✖
Oval(centerX, centerY, width, height)	✔	✔	✔	✔	✔	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖	✖
Circle(centerX, centerY, radius)	✔	✔	✔	✔	✔	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖	✖
RegularPolygon(centerX, centerY, radius, points)	✔	✔	✔	✔	✔	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖	✖
Polygon(x1, y1, x2, y2, x3, y3, ...)	✔	✔	✔	✔	✔	✔	✖	✔	✖	✖	✖	✖	✖	✖	✖	✖
Arc(centerX, centerY, width, height, startAngle, sweepAngle)	✔	✔	✔	✔	✔	✔	✖	✔	✖	✖	✖	✖	✖	✖	✖	✖
Star(centerX, centerY, radius, points)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖
Line(x1, y1, x2, y2)	✔	✖	✖	✔	✔	✔	✖	✔	✖	✖	✖	✖	✖	✔	✔	✔
Label(value, centerX, centerY)	✔	✔	✔	✔	✔	✖	✔	✔	✖	✔	✔	✔	✔	✖	✖	✖

Position keywords: 'center', 'left', 'right', 'top', 'bottom', 'left-top', 'right-top', 'left-bottom', 'right-bottom'

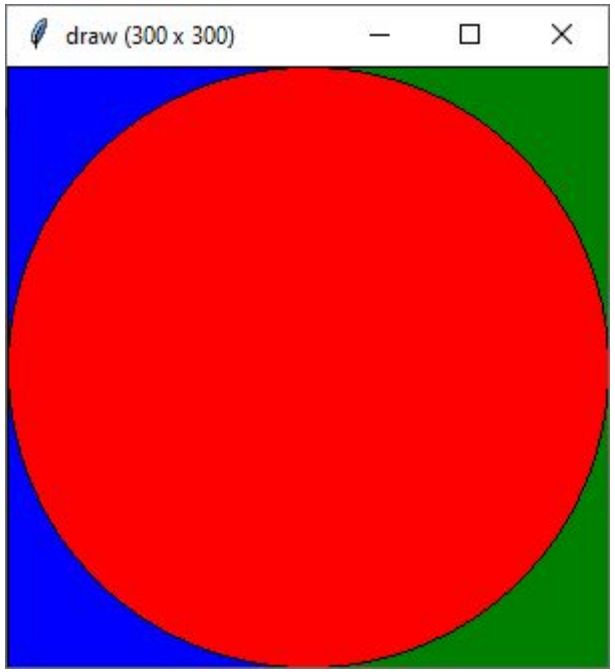
Attributes (depends on the shape)

- `fill, border, dashes, lineWidth, font, bold, align ...`

```python
from cmu_graphics import *

def redrawAll(app):
    drawRect(10,10,400,200, dashes=True, fill="blue", border="orange")

runApp(800,400)
```



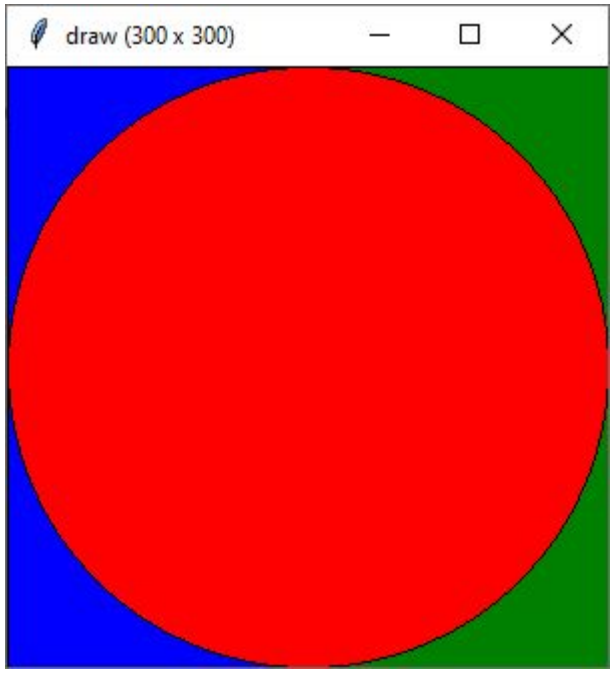
Free Response

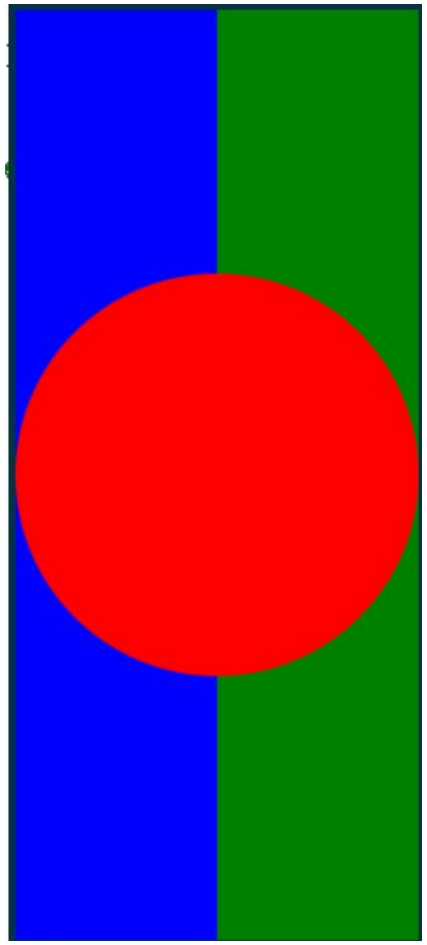


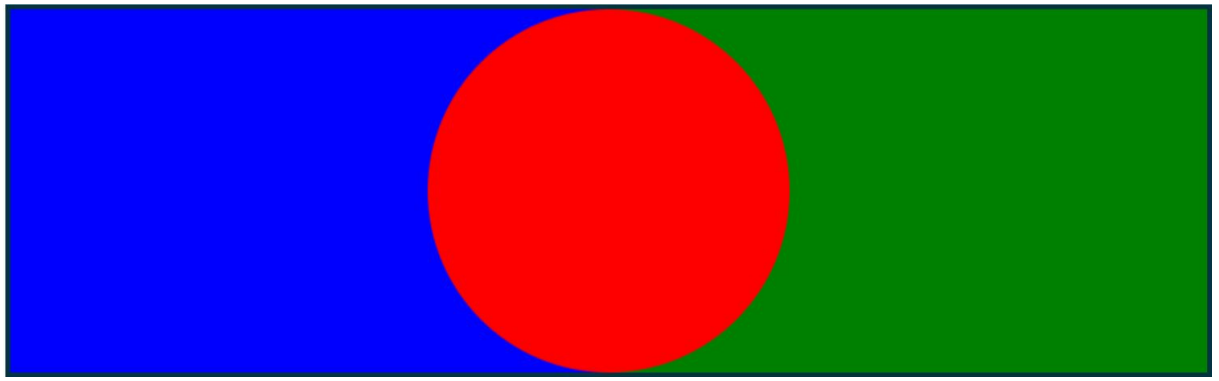
Free Response (again)

The figure should span the entire window

The circle must be centered and touch the border of the window







How to center shapes?

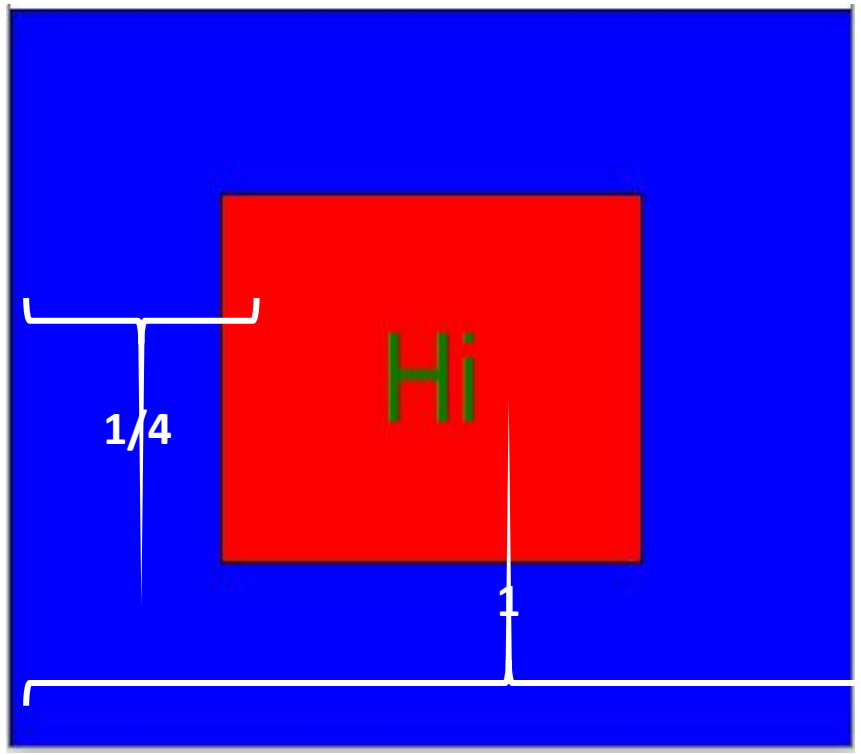
- Option 1: Add margin to top/left, subtract margin from bottom/right

```
from cmu_graphics import *

def redrawAll(app):
    marginH = 300
    marginV = 250
    rectW = 200
    rectH = 100
    drawRect( marginH,   marginV, rectW, rectH, fill="orange")

runApp(800, 600)
```

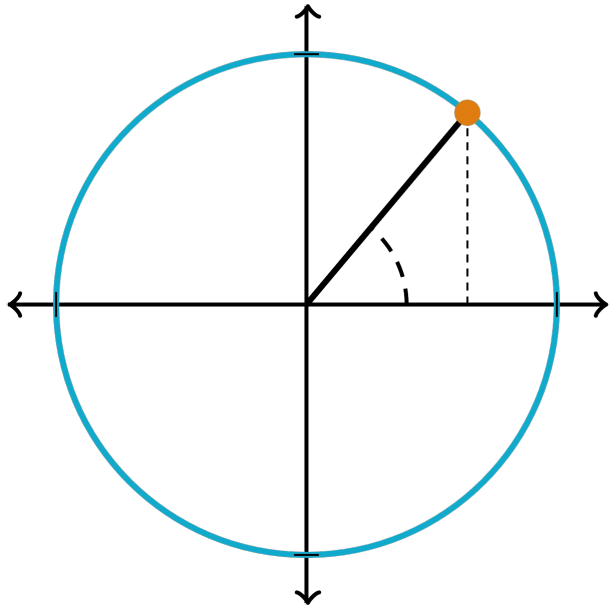
- Option 2: Add/subtract width/height from the location of the shape

```
from cmu_graphics import *

def redrawAll(app):
    rectW = 200
    rectH = 100
    drawRect( app.width//2 - rectW//2,   app.height//2 - rectH//2, rectW, rectH, fill="orange")

runApp(800, 600)
```

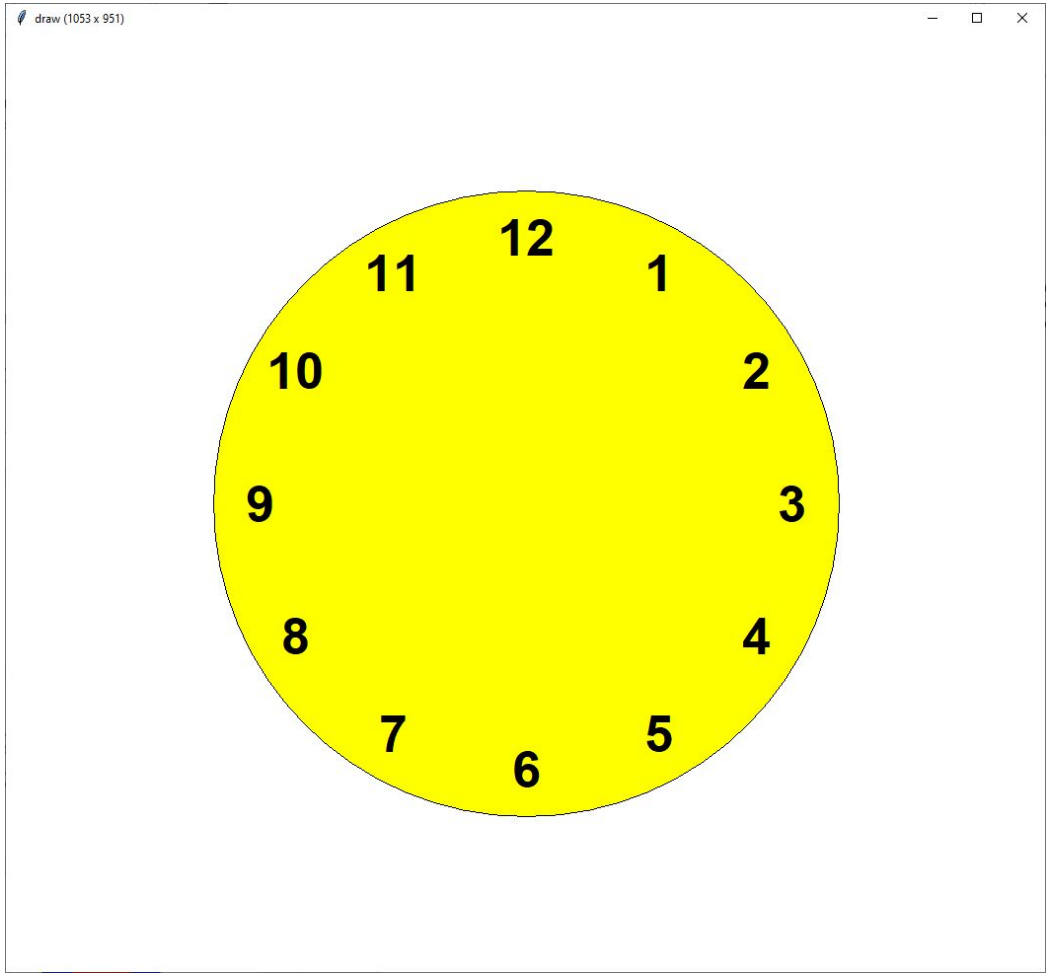
Free Response



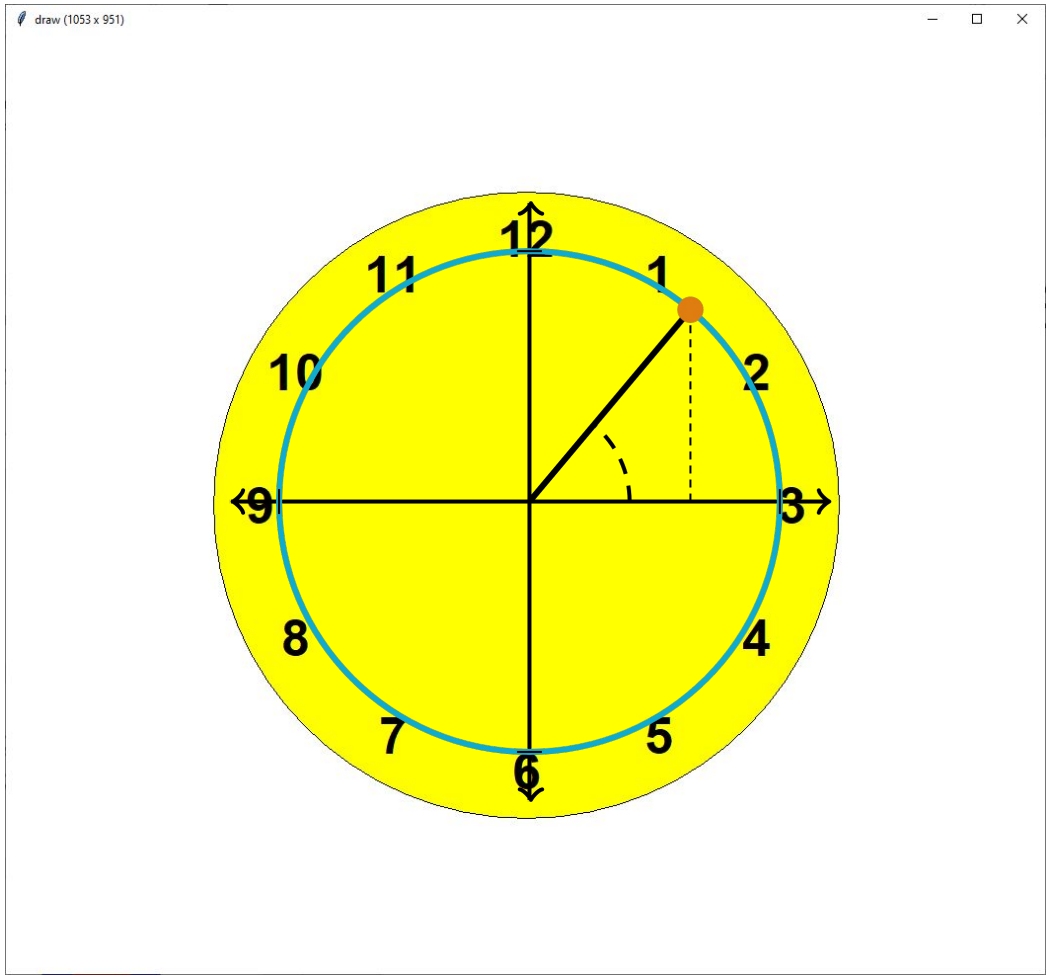
Free Response



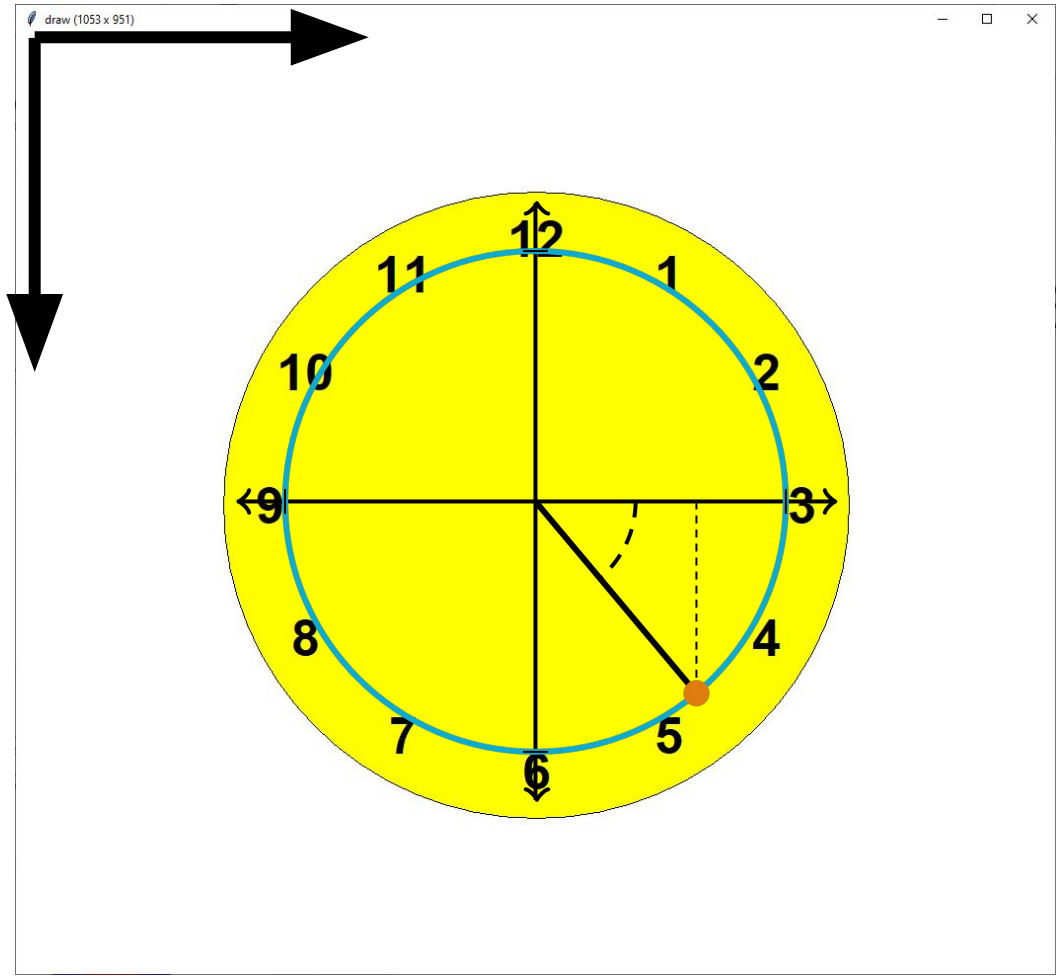
Resizable!



Free Response



Free Response



Animations!

- The “Model”
- Understand MVC
  - Don’t worry if you don’t get it now, you will understand it eventually
- Usual Tasks
  - Moving objects around
    - Timers
  - Different animation “states”: e.g., start screen, paused
    - Keyboard events
    - Using the model to track the “state”
  - Mouse click inside objects
  - …

```
from cmu_graphics import *

# This is called when the program starts
def onAppStart(app):
    pass

# This is called every time one key is pressed
def onKeyPress(app, key):
    pass

# This is called every time a mouse button is pressed
def onMousePress(app, x, y):
    pass

# This is called many times to refresh the window
def redrawAll(app):
    pass

# This is called "often" (def. by app.stepsPerSecond)
def onStep(app):
    pass

# This is how you run the program
runApp(width=800, height=800)
```

appStarted

- Runs once when the animation starts
- Useful to initialize values used in the animation

The “Model”

Responsible for managing the data, logic, and state of the application
For **now**, think of it as a container of information.

app

app.message = “Hello”
app.userName = “Eduardo”
app.ballVelocity = 30.5
…

Store Data

`app.<variable name> = <value>`

Retrieve Data

Simply use `app.<variable name>`