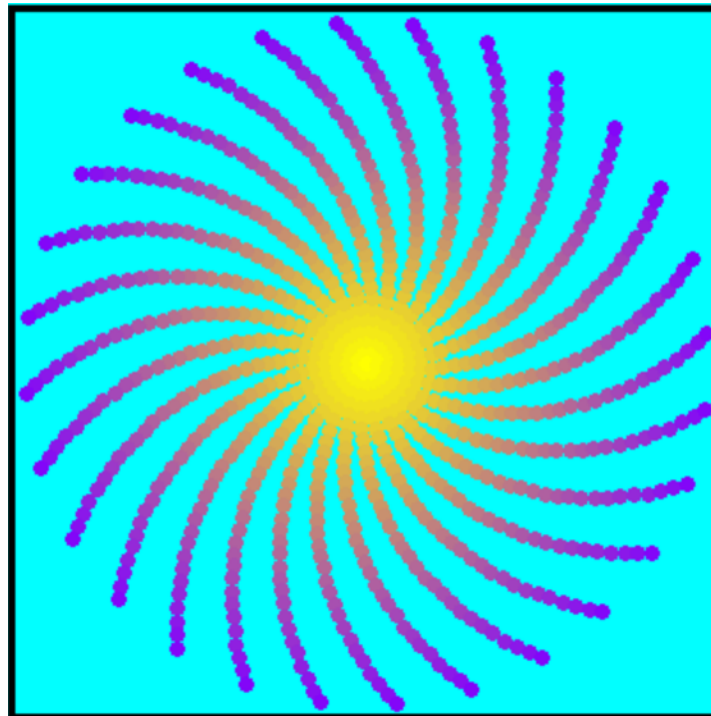


15-112

Fundamentals of Programming

Week 4 - Lecture 2:
List exercises. Graphics.



February 4, 2016

Pop Quiz

Pop Quiz

Fill in the blank:

Lists are awesome .

T/F: A variable stores the value of an object.

T/F: To make a copy of the list `a = [1, 2, 3]`, do

`b = a` **# a and b are aliases**

`b = copy.copy(a)`

What will the following print?

```
a = [1, 2, 3]
```

```
b = copy.copy(a)
```

```
print(a == b, a is b)
```

Pop Quiz

Fill in the blank:

List parameters are awesome .

Pop Quiz

Fill in the blank:

List parameters are awesome .

```
def fill(a, value):  
    for i in range(len(a)):  
        a[i] = value
```

Destructive function

```
x = [1, 2, 3]  
fill(x, 42)  
print(x)      [42, 42, 42]
```

Pop Quiz

Fill in the blank:

List parameters are awesome .

```
def fill(a, value):
```

```
→ a = copy.copy(a)
```

```
    for i in range(len(a)):
```

```
        a[i] = value
```

```
→ return a
```

Nondestructive version

```
x = [1, 2, 3]
```

```
y = fill(x, 42)
```

```
print(x, y)      [1, 2, 3] [42, 42, 42]
```

Pop Quiz

Is the sorted function destructive?

```
a = [5, 4, 3, 2, 1]
```

```
b = sorted(a)
```

```
print(a, b)    [5, 4, 3, 2, 1] [1, 2, 3, 4, 5]
```

Is the sort method destructive?

```
a = [5, 4, 3, 2, 1]
```

```
b = a.sort()
```

```
print(a, b)    [1, 2, 3, 4, 5] None
```

Pop Quiz

How do you convert a string to a list?

```
s = "You suck anil!"
```

```
print(list(s)) ['Y', 'o', 'u', ' ', 's', 'u', 'c', 'k', ' ', 'a', 'n', 'i', 'l', '!']
```

```
print(s.split(" ")) ['You', 'suck', 'anil!']
```

How do you convert a list of strings into one string?

```
a = ["Rudina", "is", "awesome"]
```

```
print("".join(a)) Rudinaisawesome
```

```
print(" ".join(a)) Rudina is awesome
```

```
print(", ".join(a)) Rudina,is,awesome
```


Pop Quiz

What does this print?

```
a = [1, 2, 3]
```

```
b = a
```

```
a = a + [4]
```

```
print(a)          [1, 2, 3, 4]
```

```
print(b)          [1, 2, 3]
```

What does this print?

```
a = [1, 2, 3]
```

```
b = a
```

```
a += [4]
```

```
print(a)          [1, 2, 3, 4]
```

```
print(b)          [1, 2, 3, 4]
```

Pop Quiz

What is the difference between **pop** and other destructive methods?

It makes a cool sound.

Pop Quiz

What is the difference between `pop` and other destructive methods?

It returns something.

List exercises

isNearlySorted(L)

Input: a list of integers

Output: True if the list is “nearly sorted”. False otherwise.



requires one swap to be sorted

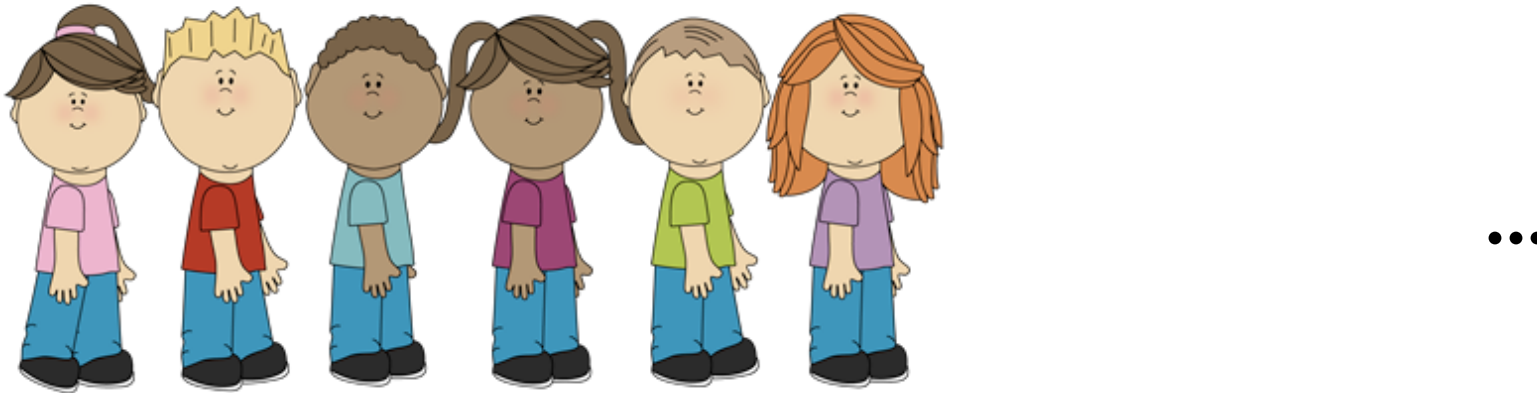
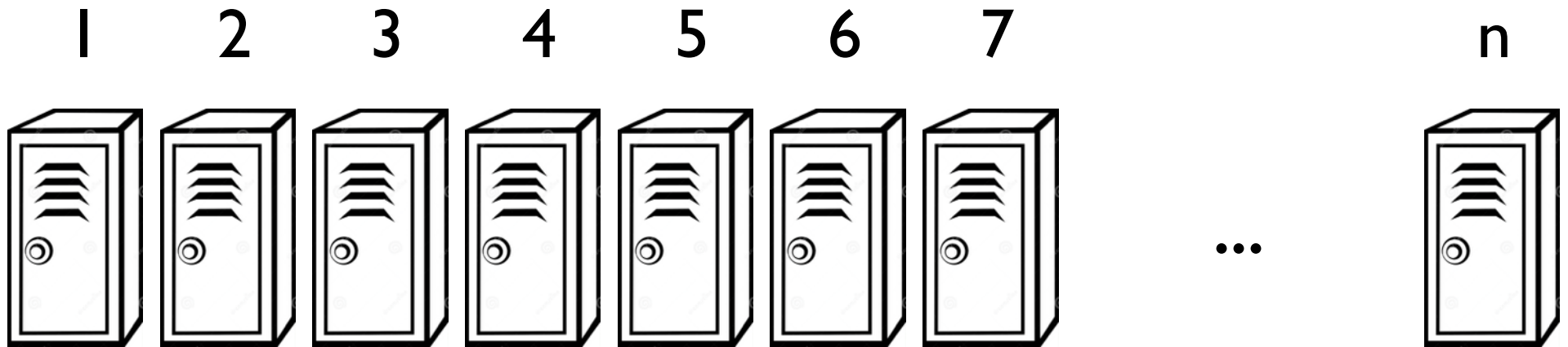
Algorithm:

For every pair of indices (i, j) :

swap $L[i]$ and $L[j]$ and check if it becomes sorted

Not an efficient solution! (but ok for this week)

Lockers Problem



Coin Flips Simulation

If you flipped a coin 200 times, what would be the longest consecutive run of heads or tails?



H



T



T



H



T

...



H

Exercise: Coin Flips Simulation

Warning: Just because you *can* use lists, doesn't mean you should use lists.

GRAPHICS!

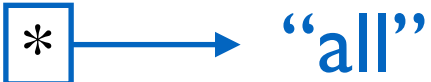
(with tkinter module)

Importing modules

In general, 2 ways to import a module:

```
import math  
print(math.sqrt(5))
```

```
from math import sqrt  
print(sqrt(5))  
print(pi)      ERROR
```

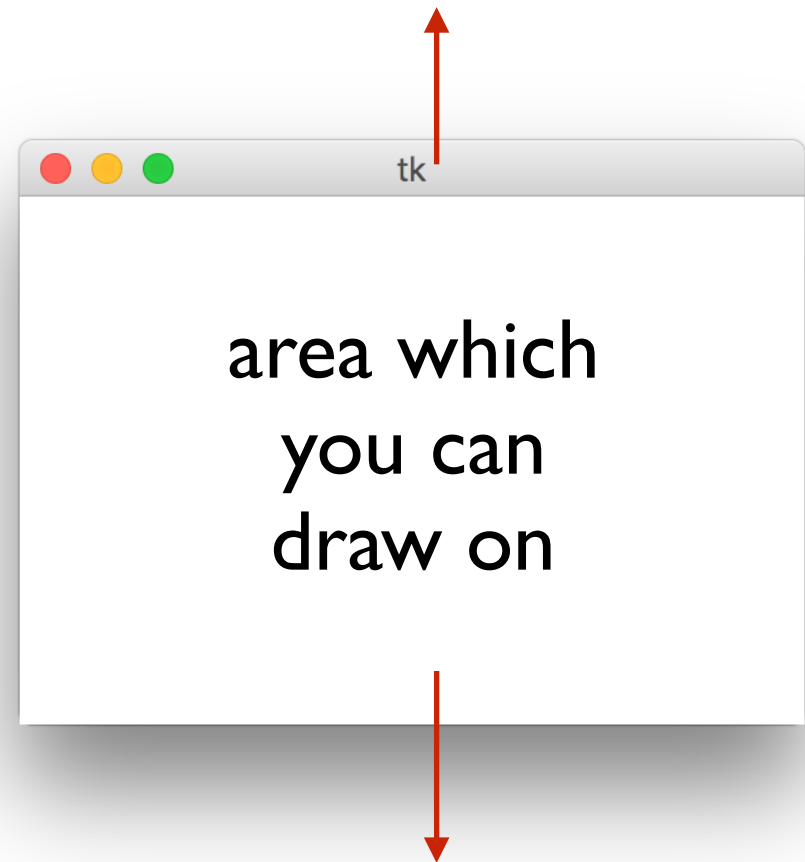
```
from math import *  “all”  
print(sqrt(5))  
print(pi)
```

```
from tkinter import *
```

tkinter canvas



tkinter window



canvas

(width and height specified in pixels)

Creating an empty canvas

```
from tkinter import *  
root = Tk()  
canvas = Canvas(root, width=300, height=200)  
canvas.pack()  
root.mainloop()
```

Creating an empty canvas

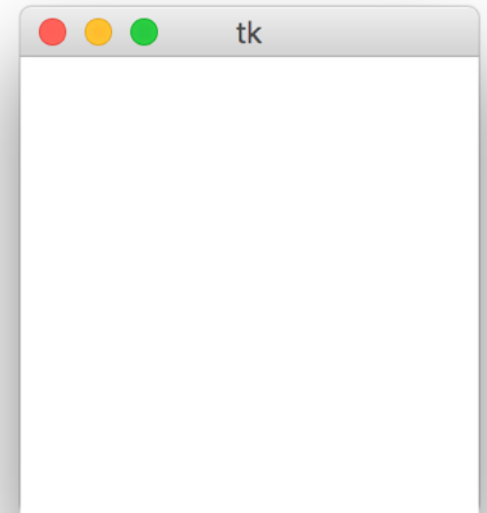
```
from tkinter import *
```

```
root = Tk() creates an object of type Tk (creates a window)
```

```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
root.mainloop()
```



```
x = 5 creates an object(data) of type int
```

```
a = list() creates an object(data) of type list
```

Creating an empty canvas

```
from tkinter import *
```

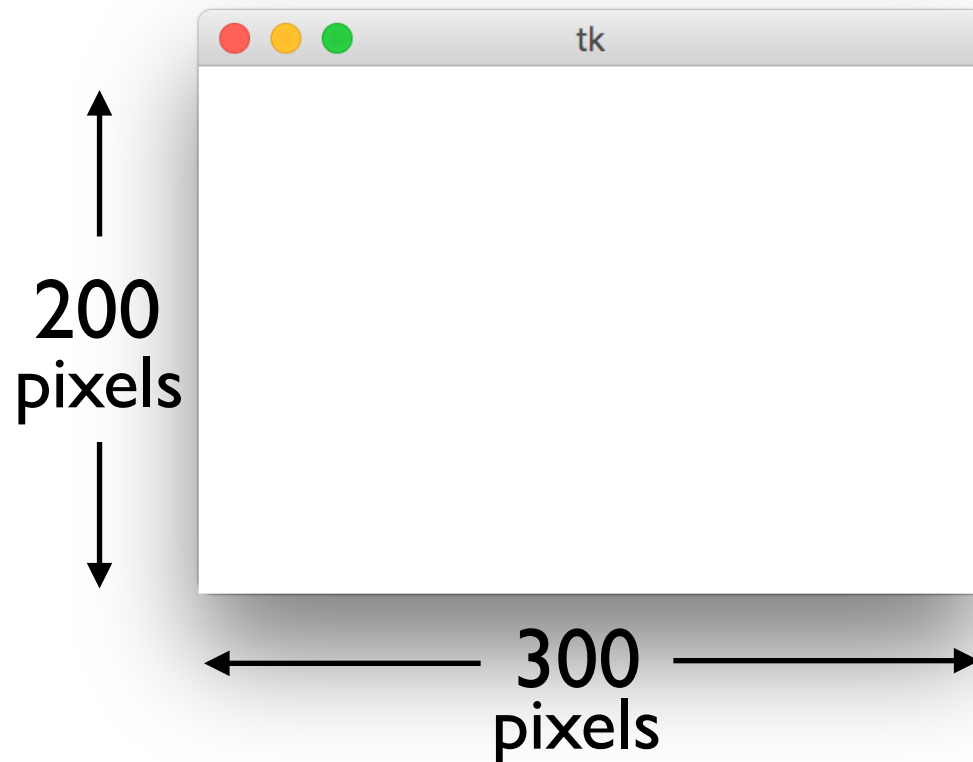
```
root = Tk()
```

```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
root.mainloop()
```

creates an object of type Canvas



Creating an empty canvas

```
from tkinter import *  
root = Tk()  
canvas = Canvas(root, width=300, height=200)  
canvas.pack()  
root.mainloop()
```

Creating an empty canvas

```
from tkinter import *  
root = Tk()  
canvas = Canvas(root, width=300, height=200)  
canvas.pack()  
root.mainloop()    keep running until window is closed
```


Creating an empty canvas

```
from tkinter import *
```

```
root = Tk()
```

```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
...
```

 code to draw things go here

```
...
```

```
root.mainloop()
```

Creating a rectangle

```
from tkinter import *
```

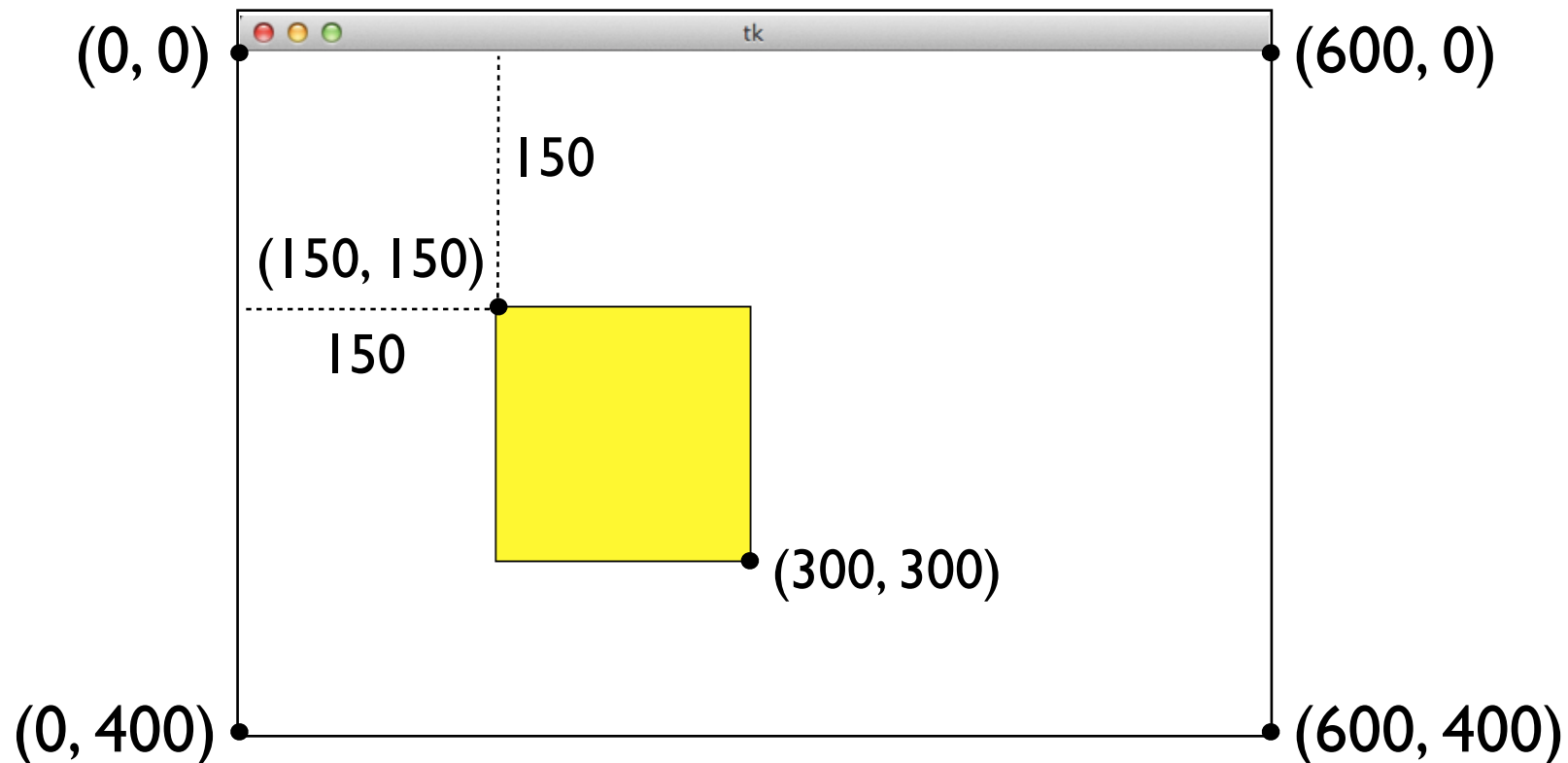
```
root = Tk()
```

```
canvas = Canvas(root, width=600, height=400)
```

```
canvas.pack()
```

```
→ canvas.create_rectangle(150, 150, 300, 300, fill="yellow")
```

```
root.mainloop()
```



Creating a line

```
from tkinter import *
```

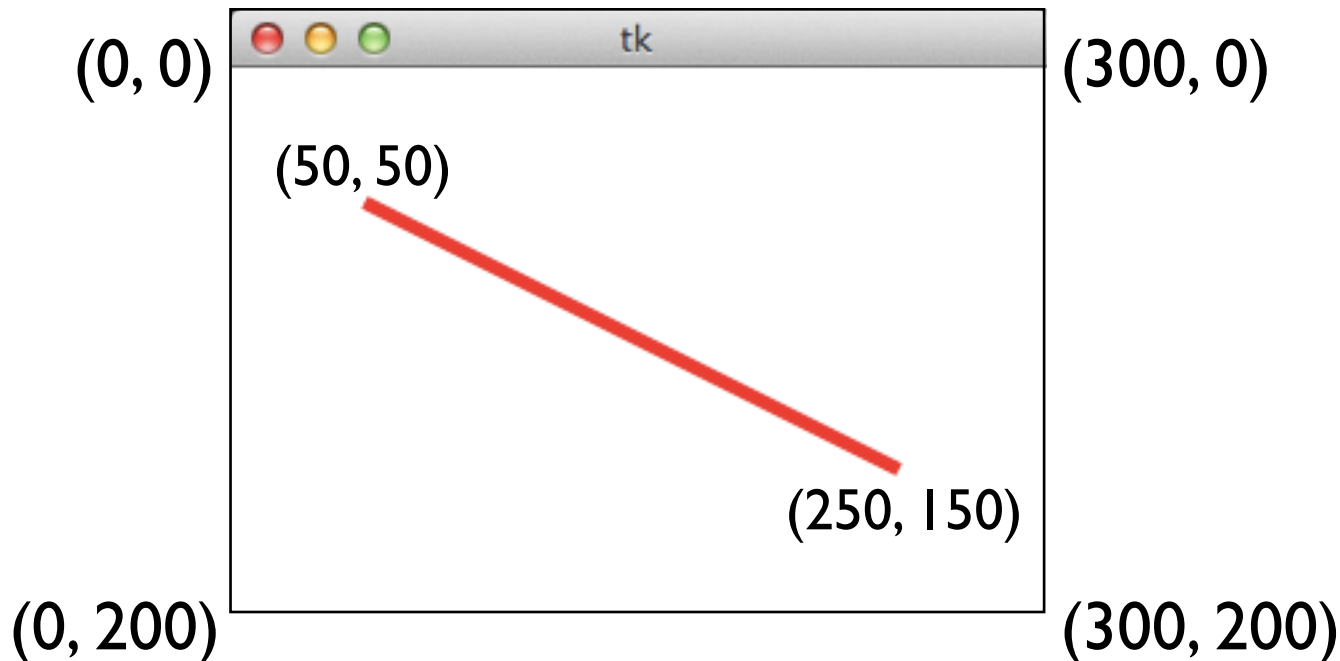
```
root = Tk()
```

```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
→ canvas.create_line(50, 50, 250, 150, fill="red", width=5)
```

```
root.mainloop()
```



Creating text

```
from tkinter import *
```

```
root = Tk()
```

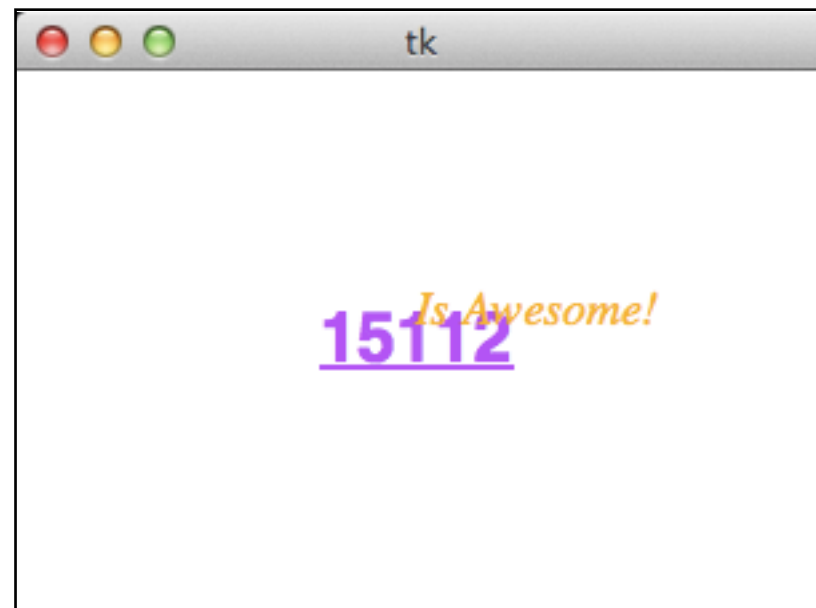
```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
→ canvas.create_text(150, 100, text="15112", fill="purple",  
                      font="Helvetica 26 bold underline")
```

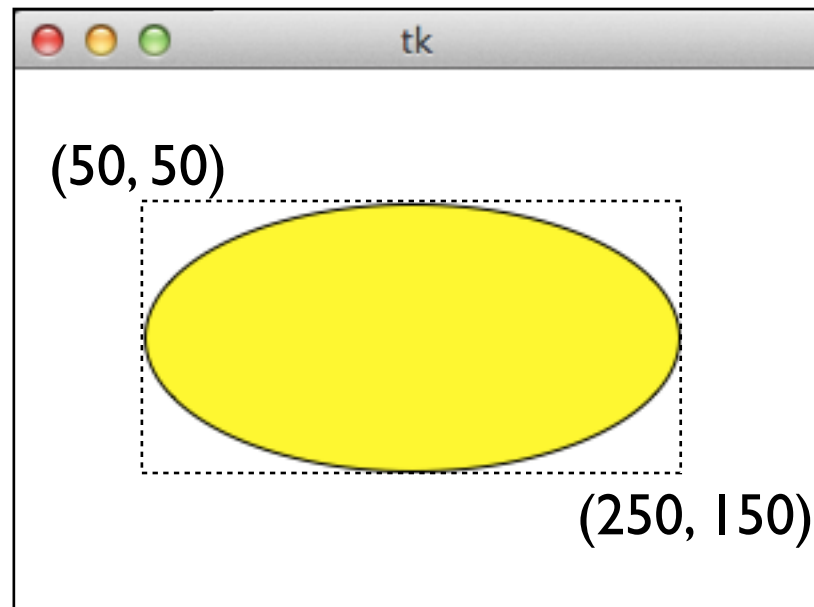
```
→ canvas.create_text(150, 100, text="Is Awesome!",  
                      anchor=SW, fill="orange", font="Times 18 italic")
```

```
root.mainloop()
```



Creating an oval

```
from tkinter import *  
root = Tk()  
canvas = Canvas(root, width=300, height=200)  
canvas.pack()  
→ canvas.create_oval(50, 50, 250, 150, fill="yellow")  
root.mainloop()
```



Creating a polygon

```
from tkinter import *
```

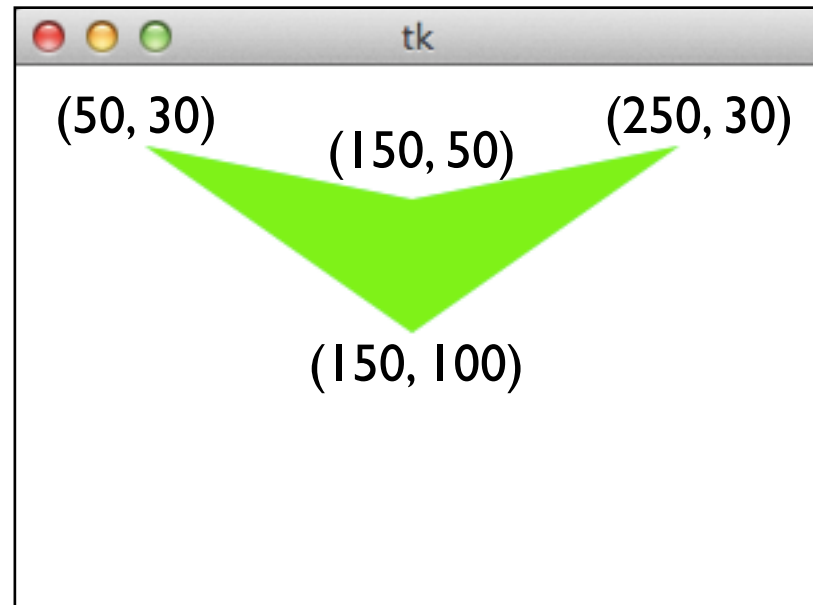
```
root = Tk()
```

```
canvas = Canvas(root, width=300, height=200)
```

```
canvas.pack()
```

```
→ canvas.create_polygon(50,30,150,50,250,30,150,100,fill="green")
```

```
root.mainloop()
```



The framework we'll use

```
from tkinter import *
```

```
def runDrawing(width=300, height=300):
```

```
    root = Tk()
```

```
    canvas = Canvas(root, width=width, height=height)
```

```
    canvas.pack()
```

```
→ draw(canvas, width, height)
```

```
    root.mainloop()
```

```
    print("bye!")
```

```
def draw(canvas, width, height):
```

```
    # put your code for drawing here
```

```
runDrawing(400, 200)
```

The framework we'll use

```
from tkinter import *
```

```
def runDrawing(width=300, height=300):
```

```
    root = Tk()
```

```
    canvas = Canvas(root, width=width, height=height)
```

```
    canvas.pack()
```

```
→ draw(canvas, width, height)
```

```
    root.mainloop()
```

```
    print("bye!")
```

```
def draw(canvas, width, height):
```

```
    # put your code for drawing here
```

```
runDrawing(400, 200)    sets width = 400, height = 200
```

```
if not specified, by default, width = 300, height = 300
```


Example: drawing rectangles

```
from tkinter import *
```

```
def runDrawing(width=300, height=300):
```

```
    ...
```

```
def draw(canvas, width, height):
```

```
    canvas.create_rectangle( 0, 0, 150, 150, fill="yellow")
```

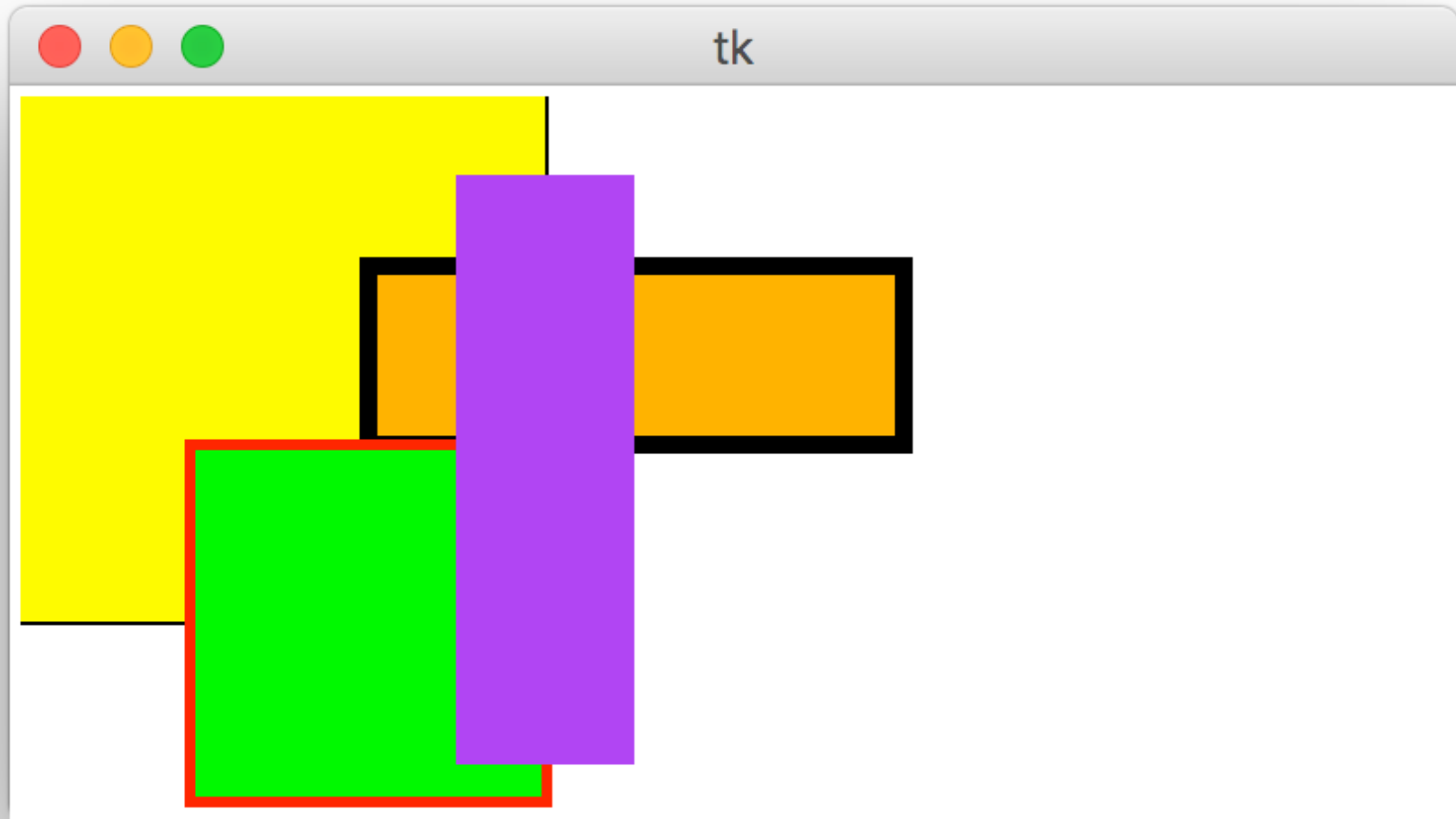
```
    canvas.create_rectangle(100, 50, 250, 100, fill="orange", width=5)
```

```
    canvas.create_rectangle( 50, 100, 150, 200, fill="green",  
                             outline="red", width=3)
```

```
    canvas.create_rectangle(125, 25, 175, 190, fill="purple", width=0)
```

```
runDrawing(400, 200)
```

Example: drawing rectangles



Example: drawing centered rectangles

```
def draw(canvas, width, height):
```

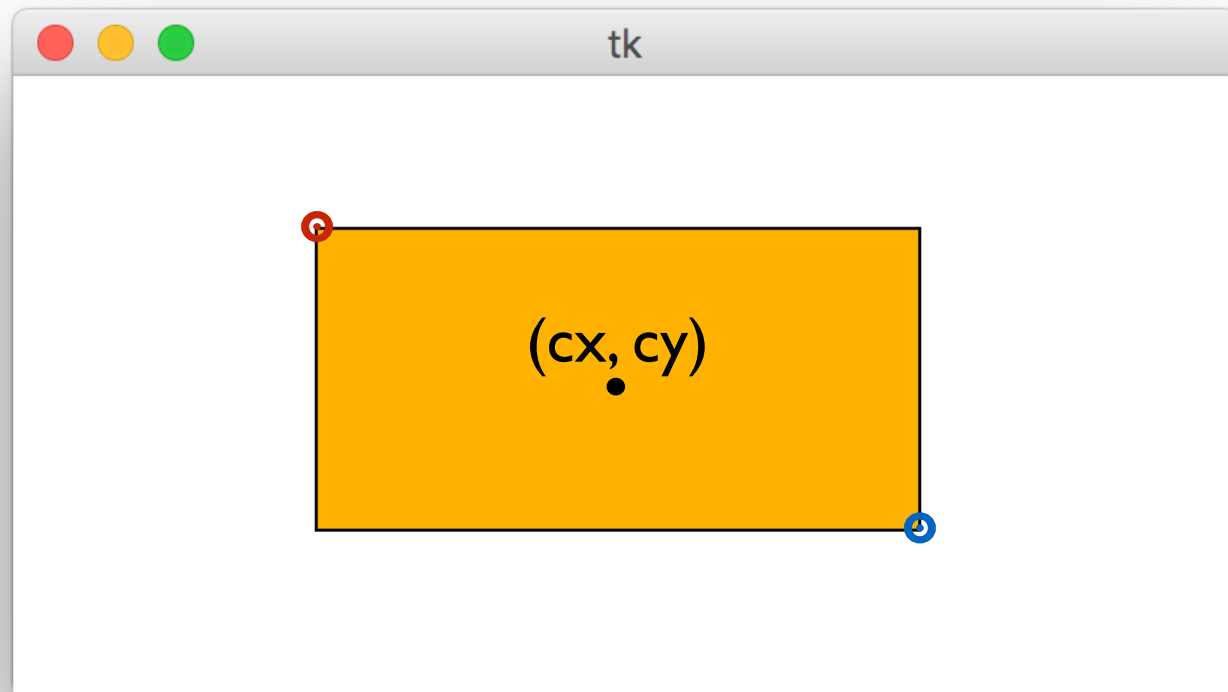
```
    margin = 30
```

```
    canvas.create_rectangle(margin, margin, width-margin, height-margin,  
                           fill="darkGreen")
```



Example: drawing centered rectangles

```
def draw(canvas, width, height):  
    (cx, cy) = (width/2, height/2)  
    (rectWidth, rectHeight) = (200, 100)  
    canvas.create_rectangle(cx - rectWidth/2, cy - rectHeight/2,  
                           cx + rectWidth/2, cy + rectHeight/2,  
                           fill="orange")
```



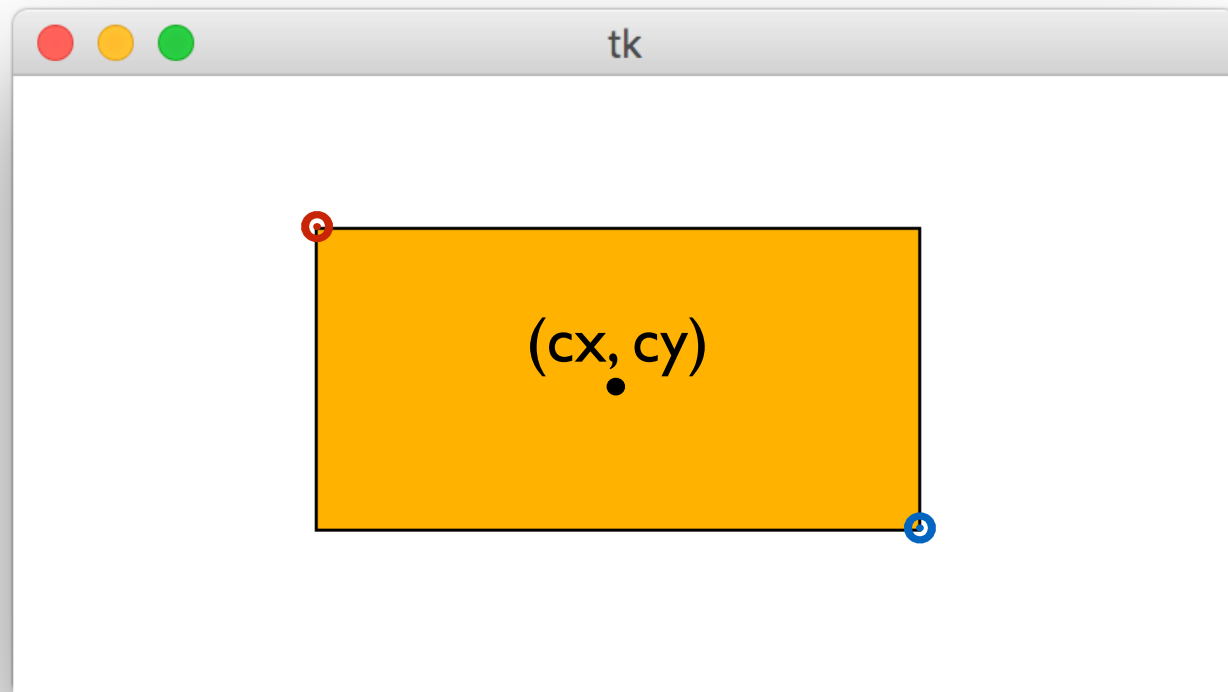
Example: drawing centered rectangles

```
def draw(canvas, width, height):
```

```
    (cx, cy) = (width/2, height/2)
```

```
    → (rectWidth, rectHeight) = (width/2, height/2)
```

```
    canvas.create_rectangle(cx - rectWidth/2, cy - rectHeight/2,  
                           cx + rectWidth/2, cy + rectHeight/2,  
                           fill="orange")
```



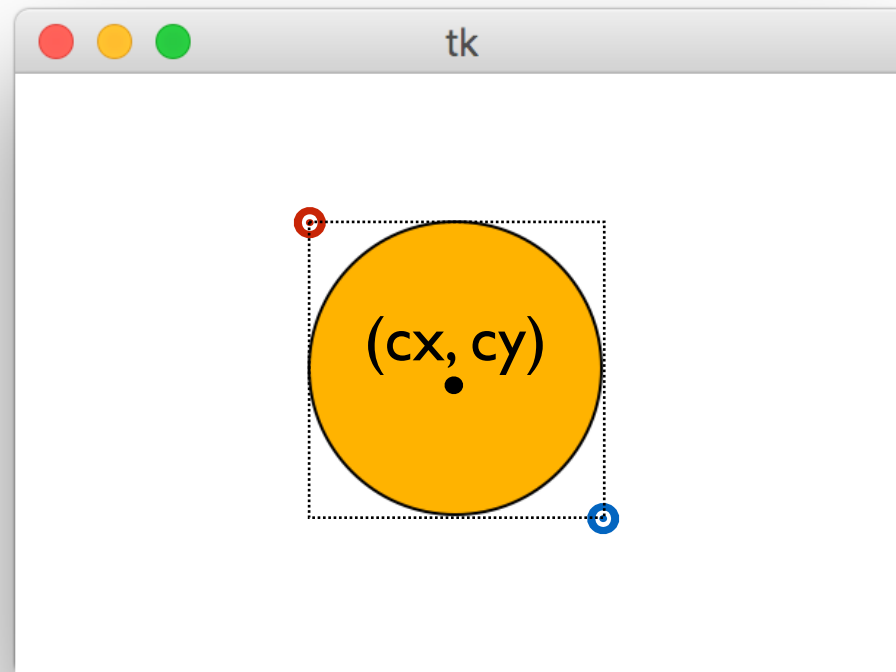
Example: drawing centered circles

```
def draw(canvas, width, height):
```

```
    (cx, cy) = (width/2, height/2)
```

```
    r = min(width, height)/4
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="orange")
```



Example: drawing a Belgian flag

```
def drawBelgianFlag(canvas, x0, y0, x1, y1):  
    # draw a Belgian flag in the area bounded by (x0,y0) in  
    # the top-left and (x1,y1) in the bottom-right
```



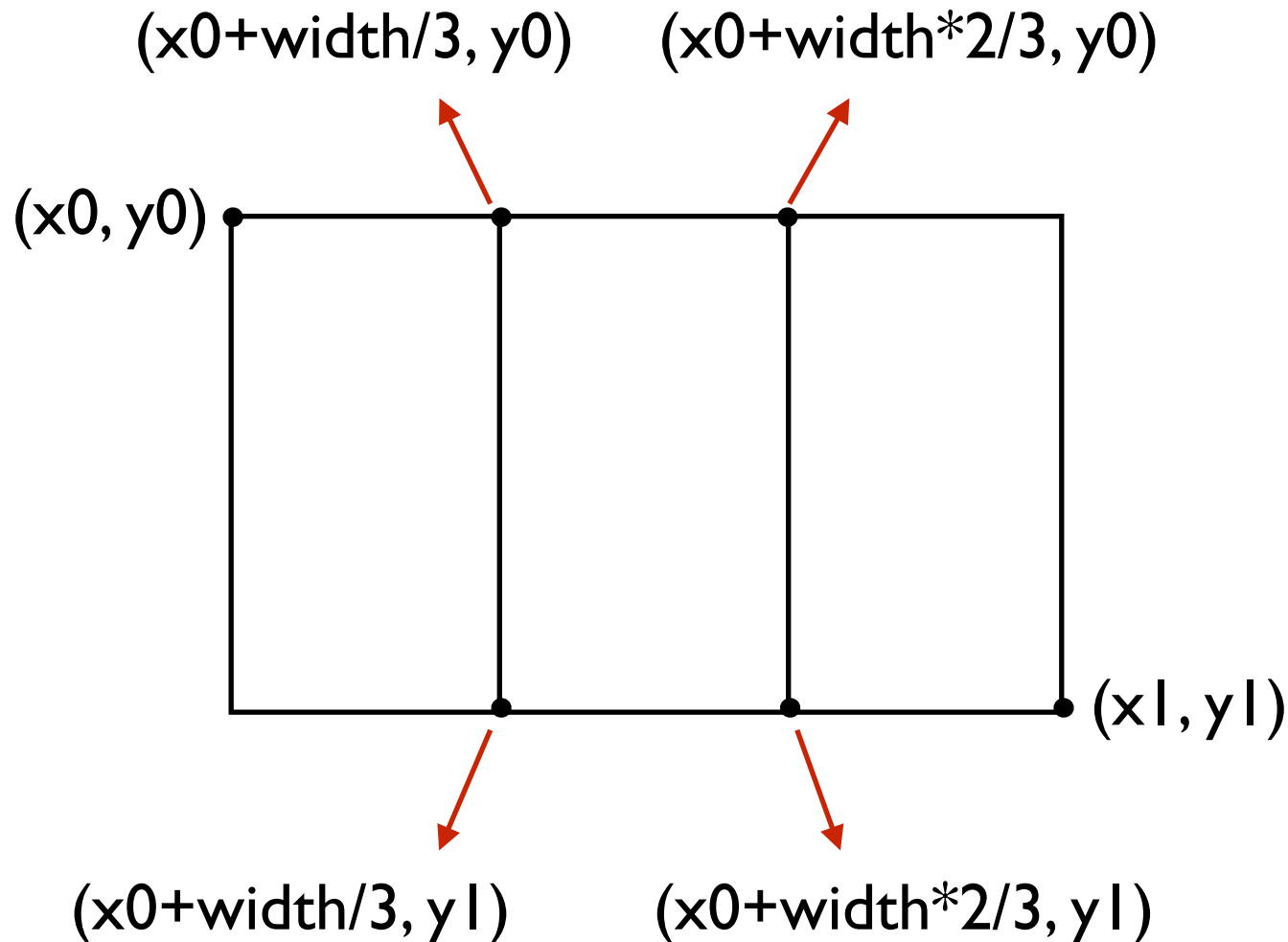
Example: drawing a Belgian flag

```
def drawBelgianFlag(canvas, x0, y0, x1, y1):
```

```
    # draw a Belgian flag in the area bounded by (x0,y0) in
```

```
    # the top-left and (x1,y1) in the bottom-right
```

```
    width = x1 - x0
```



Example: drawing a Belgian flag

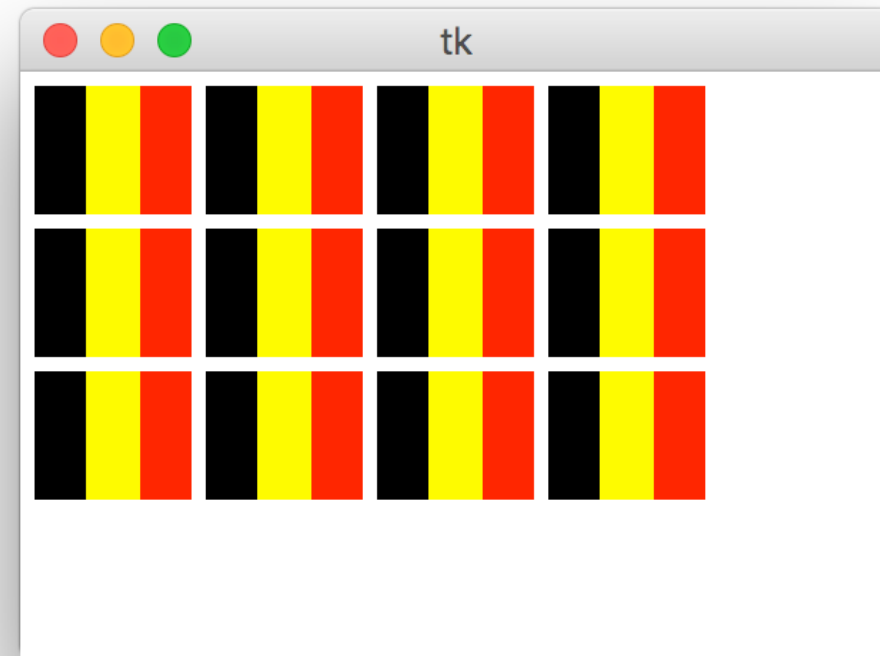
```
def drawBelgianFlag(canvas, x0, y0, x1, y1):  
    width = (x1 - x0)  
    canvas.create_rectangle(x0, y0, x0+width/3, y1,  
                           fill="black", width=0)  
    canvas.create_rectangle(x0+width/3, y0, x0+width*2/3, y1,  
                           fill="yellow", width=0)  
    canvas.create_rectangle(x0+width*2/3, y0, x1, y1,  
                           fill="red", width=0)
```

```
def draw(canvas, width, height)  
    drawBelgianFlag(canvas, 25, 25, 175, 150)
```

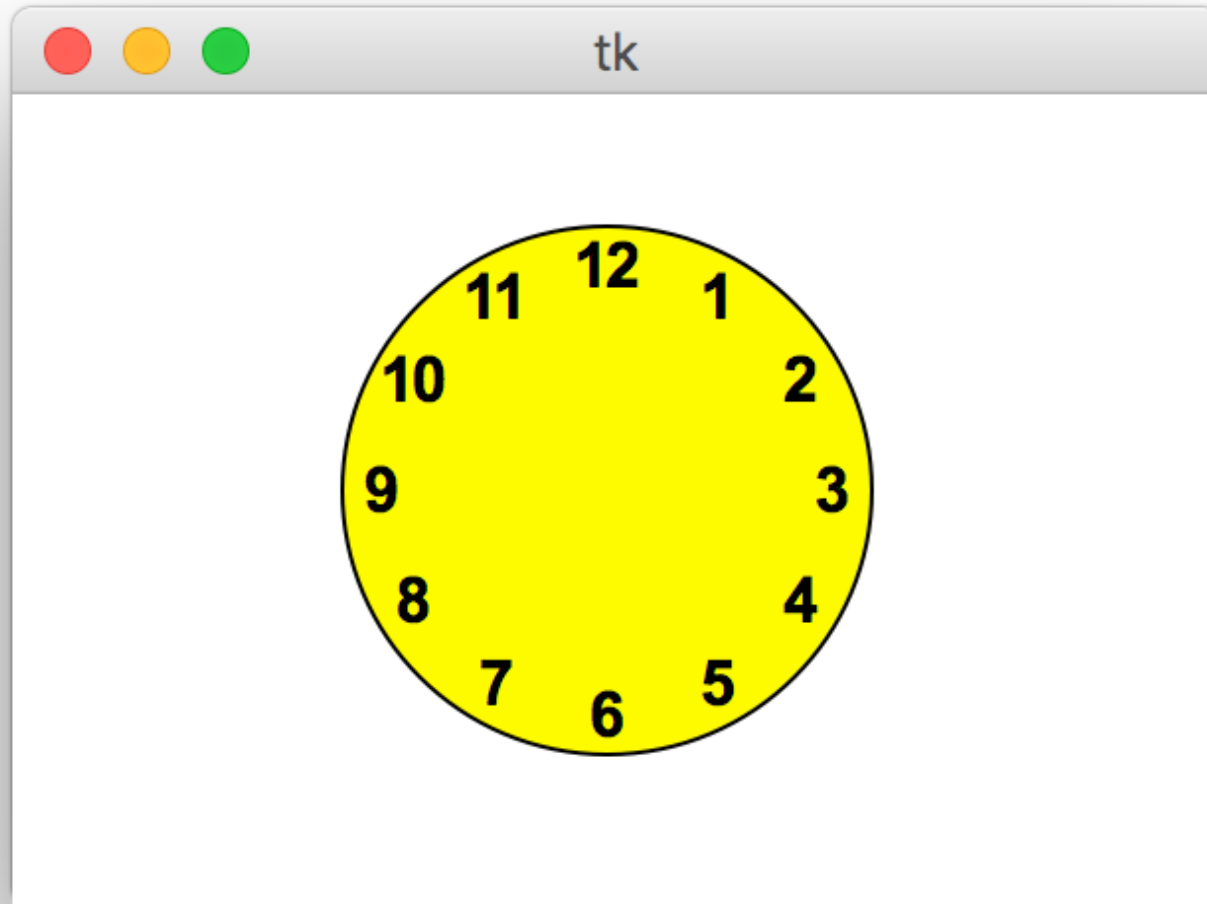


Example: drawing a Belgian flag

```
def draw(canvas, width, height):  
    (flagWidth, flagHeight) = (60, 50)  
    margin = 5  
    for row in range(3):  
        for col in range(4):  
            left = col * flagWidth + margin  
            top = row * flagHeight + margin  
            right = left + flagWidth - margin  
            bottom = top + flagHeight - margin  
            drawBelgianFlag(canvas, left, top, right, bottom)
```

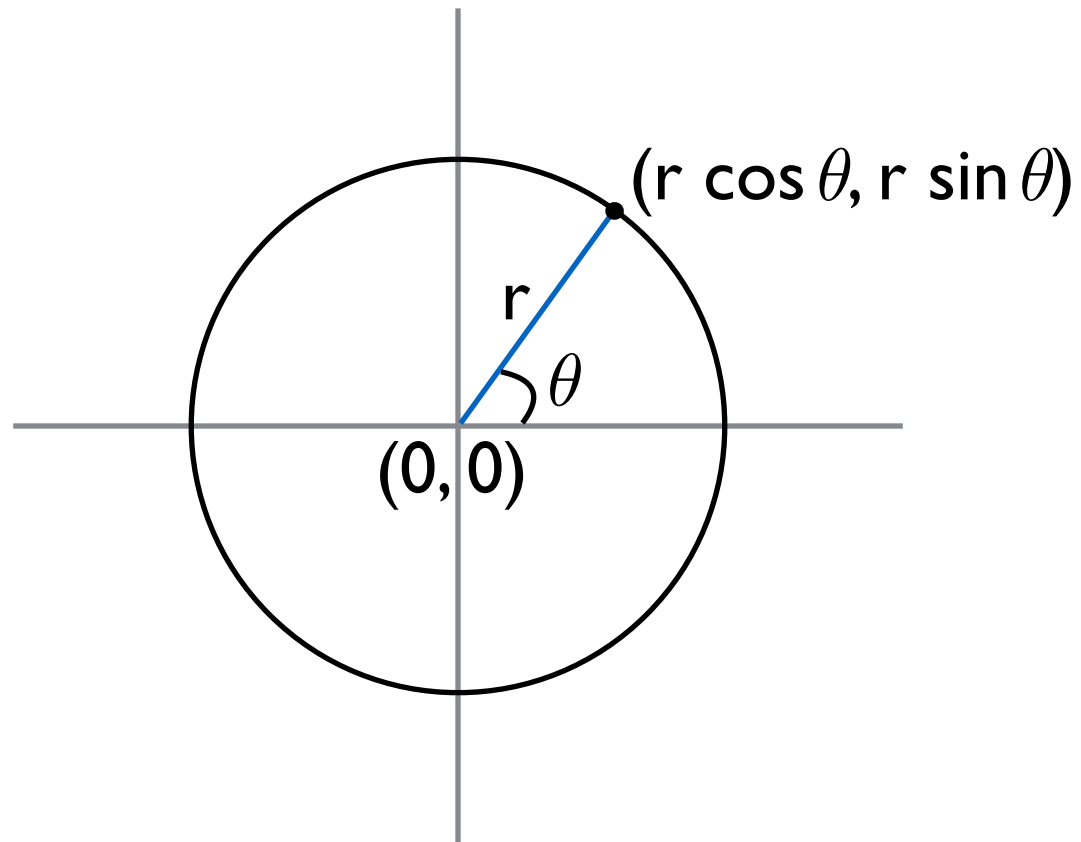


Example: drawing circular patterns

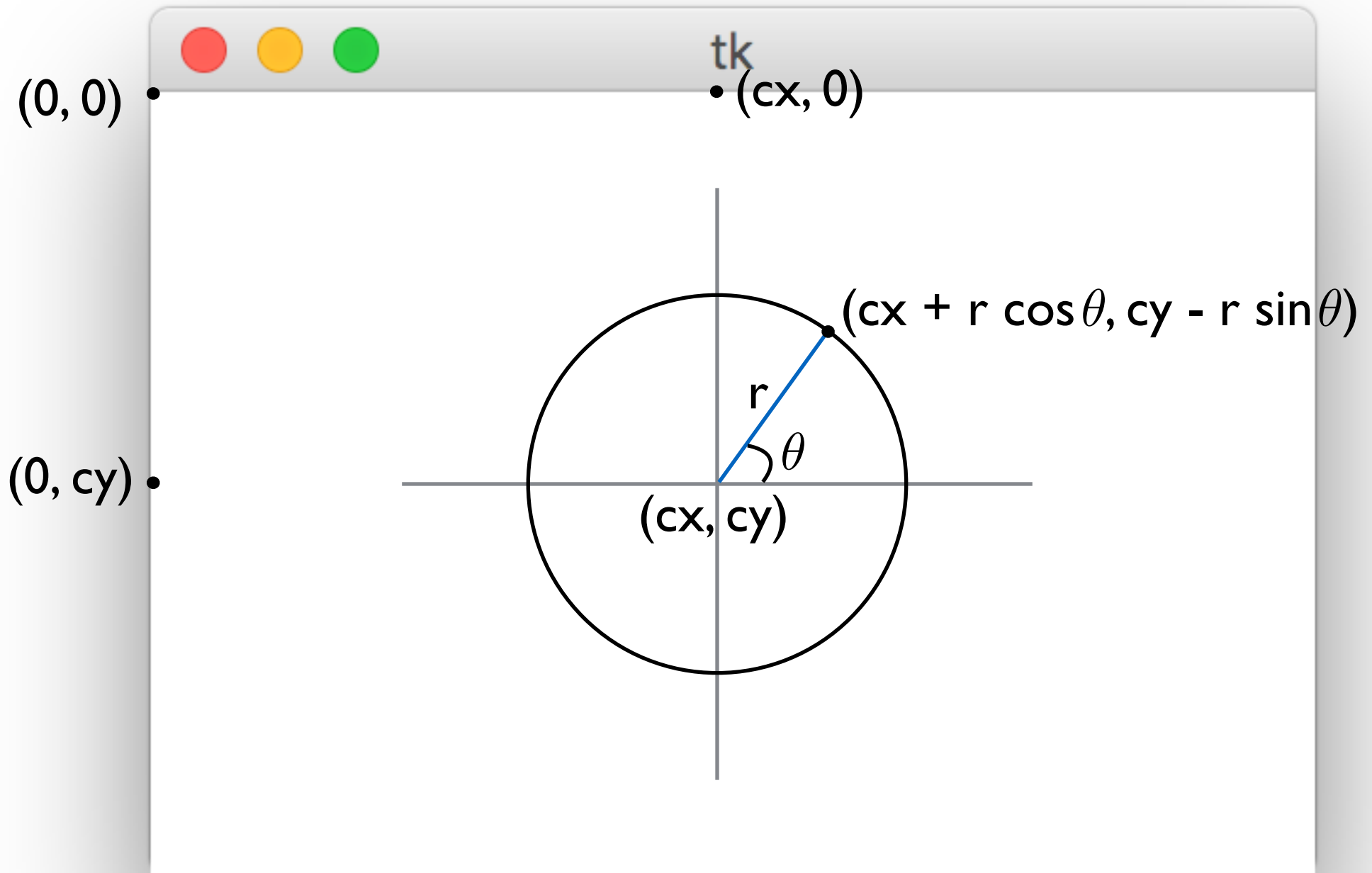


How do you determine the right positions to put the numbers?

Trig 101



Trig 101



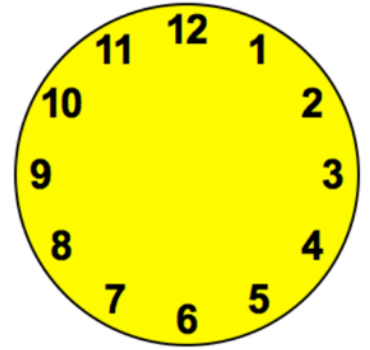
Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```



Example: drawing circular patterns

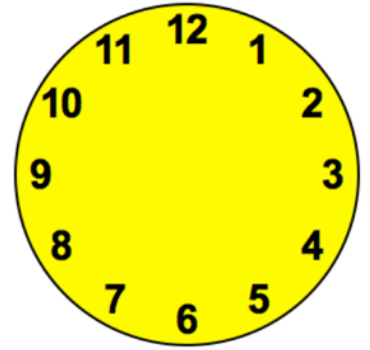
```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

```
    for hour in range(12):
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

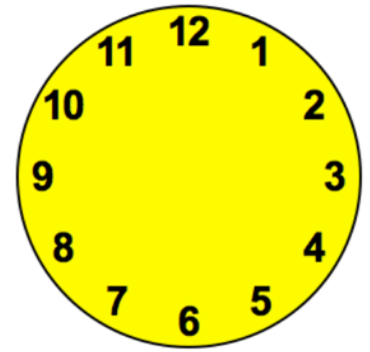
```
    for hour in range(12):
```

```
        hourX =
```

```
        hourY =
```

```
        label =
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

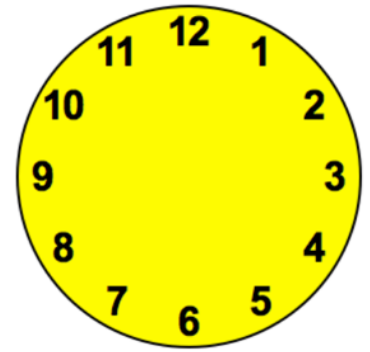
```
    for hour in range(12):
```

```
        hourX =
```

```
        hourY =
```

```
        label = str(hour if (hour > 0) else 12)
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

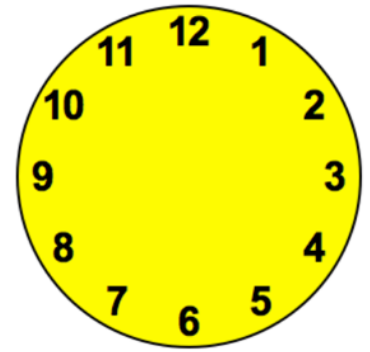
```
    for hour in range(12):
```

```
        hourX = cx + r*math.cos(theta)
```

```
        hourY = cy - r*math.sin(theta)
```

```
        label = str(hour if (hour > 0) else 12)
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

```
    for hour in range(12):
```

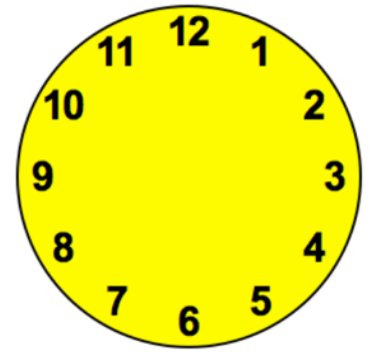
```
        theta = math.pi/2 - (2*math.pi)*(hour/12)
```

```
        hourX = cx + r*math.cos(theta)
```

```
        hourY = cy - r*math.sin(theta)
```

```
        label = str(hour if (hour > 0) else 12)
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

```
for hour in range(12):
```

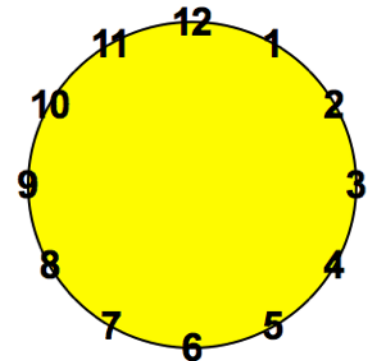
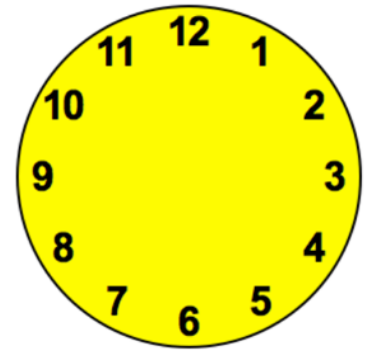
```
    theta = math.pi/2 - (2*math.pi)*(hour/12)
```

```
    hourX = cx + r*math.cos(theta)
```

```
    hourY = cy - r*math.sin(theta)
```

```
    label = str(hour if (hour > 0) else 12)
```

```
    canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

```
    r = r*0.85
```

```
    for hour in range(12):
```

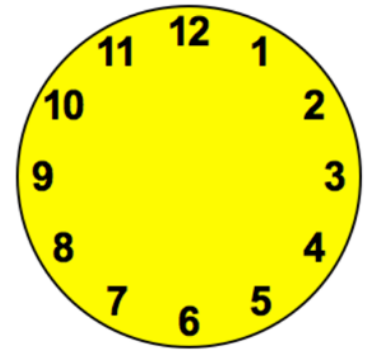
```
        theta = math.pi/2 - (2*math.pi)*(hour/12)
```

```
        hourX = cx + r*math.cos(theta)
```

```
        hourY = cy - r*math.sin(theta)
```

```
        label = str(hour if (hour > 0) else 12)
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```



Example: drawing circular patterns

```
import math
```

```
def draw(canvas, width, height):
```

```
    (cx, cy, r) = (width/2, height/2, min(width, height)/3)
```

```
    canvas.create_oval(cx - r, cy - r, cx + r, cy + r, fill="yellow")
```

```
    r = r*0.85
```

```
    for hour in range(12):
```

```
        hourAngle = math.pi/2 - (2*math.pi)*(hour/12)
```

```
        hourX = cx + r*math.cos(hourAngle)
```

```
        hourY = cy - r*math.sin(hourAngle)
```

```
        label = str(hour if (hour > 0) else 12)
```

```
        canvas.create_text(hourX, hourY, text=label, font="Ariel 16 bold")
```

