parameters, return, math, graphics

nobody expects the spanish inquisition!
http://www.youtube.com/watch?v=CSe38dzJYkY

parameters

```
def name(parameter, parameter, ..., parameter):
    statements
```

- parameters are declared by writing their names
- no types needed!

parameters

```
>>> def print_many(word, n):
...     for i in range(n):
...         print word

>>> print_many("spam", 4)
spam
spam
spam
spam
```

exercise

rewrite Stars2.java in python

```
*************

*******

***********************************

**********
*        *
**********

*****
*   *
*   *
*****
```

stars2.py

```python
# Draws a box of stars with the given width and height.
def box(width, height):
    print width * "*"
    for i in range(height - 2):
        print "*" + (width - 2) * " " + "*"
    print width * "*"

# main
print 13 * "*"
print  7 * "*"
print 35 * "*"
box(10, 3)
box(5, 4)
```

default parameter values

```
>>> def print_many(word, n=1):
...     for i in range(n):
...         print word

>>> print_many("shrubbery")
shrubbery
>>> print_many("shrubbery", 4)
shrubbery
shrubbery
shrubbery
shrubbery
```

- can make parameter(s) optional by specifying a default value

parameter keywords

```
>>> def print_many(word, n):
...     for i in range(n):
...         print word

>>> print_many(n=3, word="Ni!")
Ni!
Ni!
Ni!
```

- can pass parameters in any order by specifying their names when calling

math

```
from math import *
```

Function name	**Description**
`ceil(value)`	rounds up
`cos(value)`	cosine, in radians
`degrees(value)`	convert radians to degrees
`floor(value)`	rounds down
`log(value, base)`	logarithm in any base
`log10(value)`	logarithm, base 10
`max(value1, value2, ...)`	largest of two (or more) values
`min(value1, value2, ...)`	smallest of two (or more) values
`radians(value)`	convert degrees to radians
`round(value)`	nearest whole number
`sin(value)`	sine, in radians
`sqrt(value)`	square root
`tan(value)`	tangent

Constant	**Description**
`e`	2.7182818...
`pi`	3.1415926...

more: http://docs.python.org/library/math.html

returning values

```
def name(parameters):
    statements
    return value
```

- just like in Java!

returning values

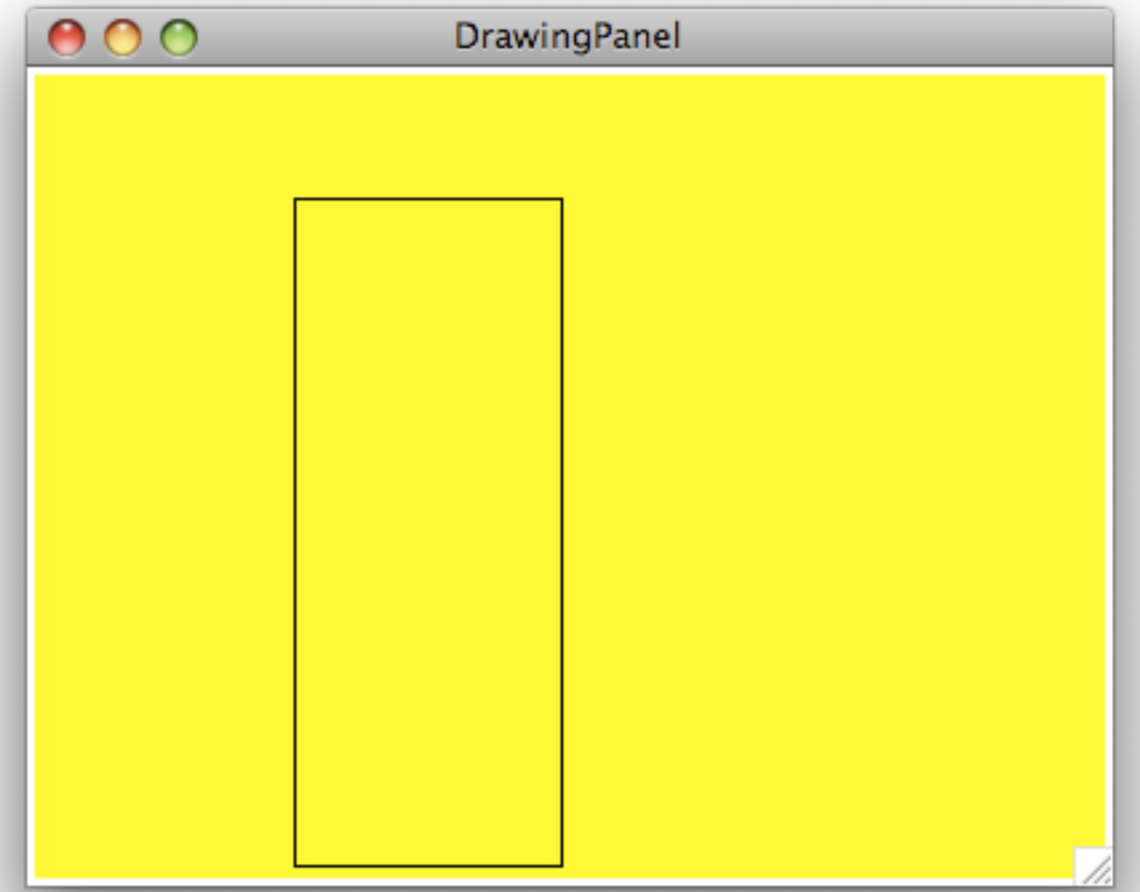
```
>>> def ftoc(temp):
...     tempc = 5.0 / 9.0 * (temp - 32)
...     return tempc

>>> ftoc(98.6)
37.0
```

drawingpanel

- put `drawingpanel.py` in the same folder as your program
- `from drawingpanel import *`
- panel's `canvas` field behaves like Graphics g
- need to put `panel.mainloop()` at the end of your program

draw.py

```python
from drawingpanel import *

panel = DrawingPanel(400, 300)
panel.set_background("yellow")
panel.canvas.create_rectangle(100, 50, 200, 300)
panel.mainloop()
```



drawing methods

Java	**Python**
`drawLine`	`panel.canvas.create_line(x1, y1, x2, y2)`
`drawRect,` `fillRect`	`panel.canvas.create_rectangle(x1, y1, x2, y2)`
`drawOval,` `fillOval`	`panel.canvas.create_oval(x1, y1, x2, y2)`
`drawString`	`panel.canvas.create_text(x, y, text="text")`
`setColor`	*(see next slide)*
`setBackground`	`panel.set_background(color)`

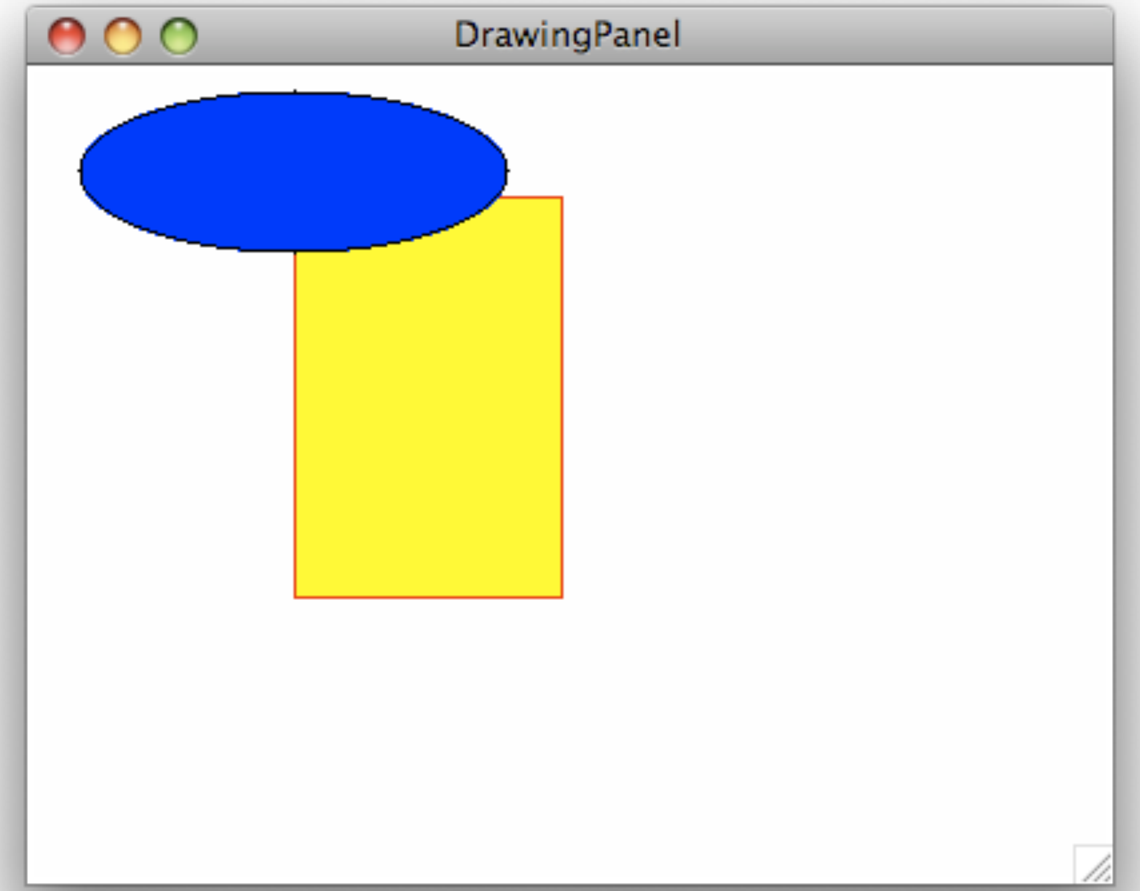
- notice methods take x2/y2 parameters, instead of width/height

colors

- no `fillRect`, `fillOval`, or `setColor`
- instead, pass `outline` and `fill` parameters when drawing a shape
- list of all colors: http://www.cs.washington.edu/education/courses/cse142/08su/python/python_colors.png

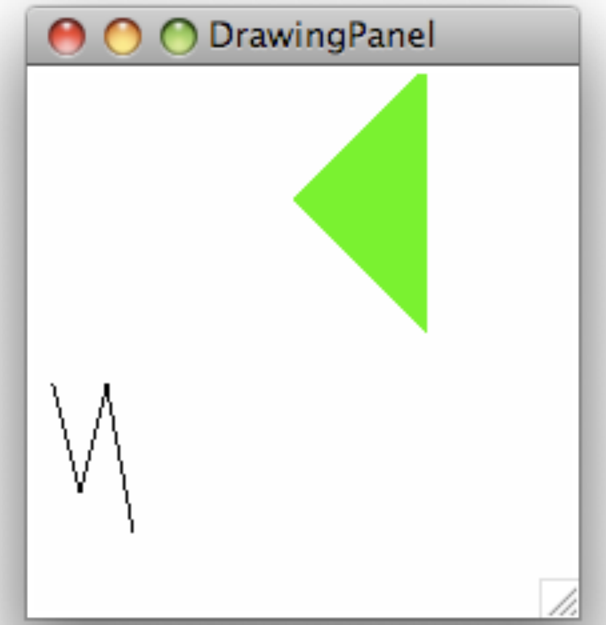
drawcolors.py

```
from drawingpanel import *

panel = DrawingPanel(400, 300)
panel.canvas.create_rectangle(100, 50, 200, 200,
                              outline="red", fill="yellow")
panel.canvas.create_oval(20, 10, 180, 70, fill="blue")
panel.mainloop()
```



polygons

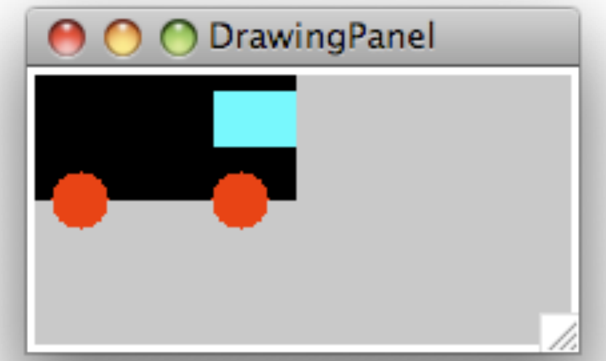
```
from drawingpanel import *

panel = DrawingPanel(200, 200)
panel.canvas.create_polygon(100, 50, 150, 0,
                            150, 100, fill="green")
panel.canvas.create_line(10, 120, 20, 160,
                         30, 120, 40, 175)
panel.mainloop()
```



- draw polygons with create_polygon
- draw line groups by passing more parameters to create_line

exercise

draw a car!

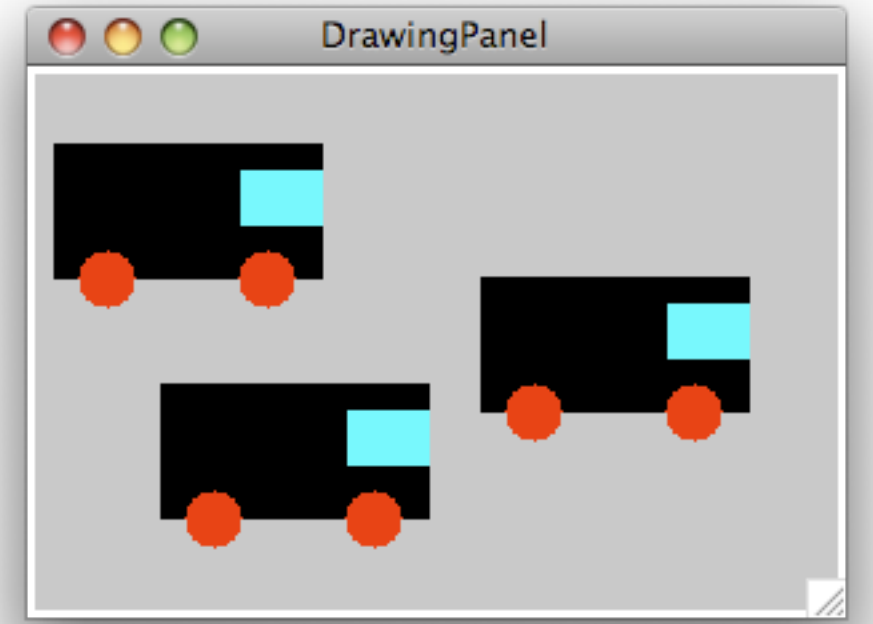


car.py

```python
from drawingpanel import *

panel = DrawingPanel(200, 100)
panel.set_background("gray")

panel.canvas.create_rectangle(x, y, x+100, y+50,
                                          fill="black")
panel.canvas.create_oval(x+10, y+40, x+30, y+60,
                                  fill="red", outline="red")
panel.canvas.create_oval(x+70, y+40, x+90, y+60,
                                  fill="red", outline="red")
panel.canvas.create_rectangle(x+70, y+10, x+100, y+30,
                                          fill="cyan", outline="cyan")

panel.mainloop()
```

exercise

parameterize it!

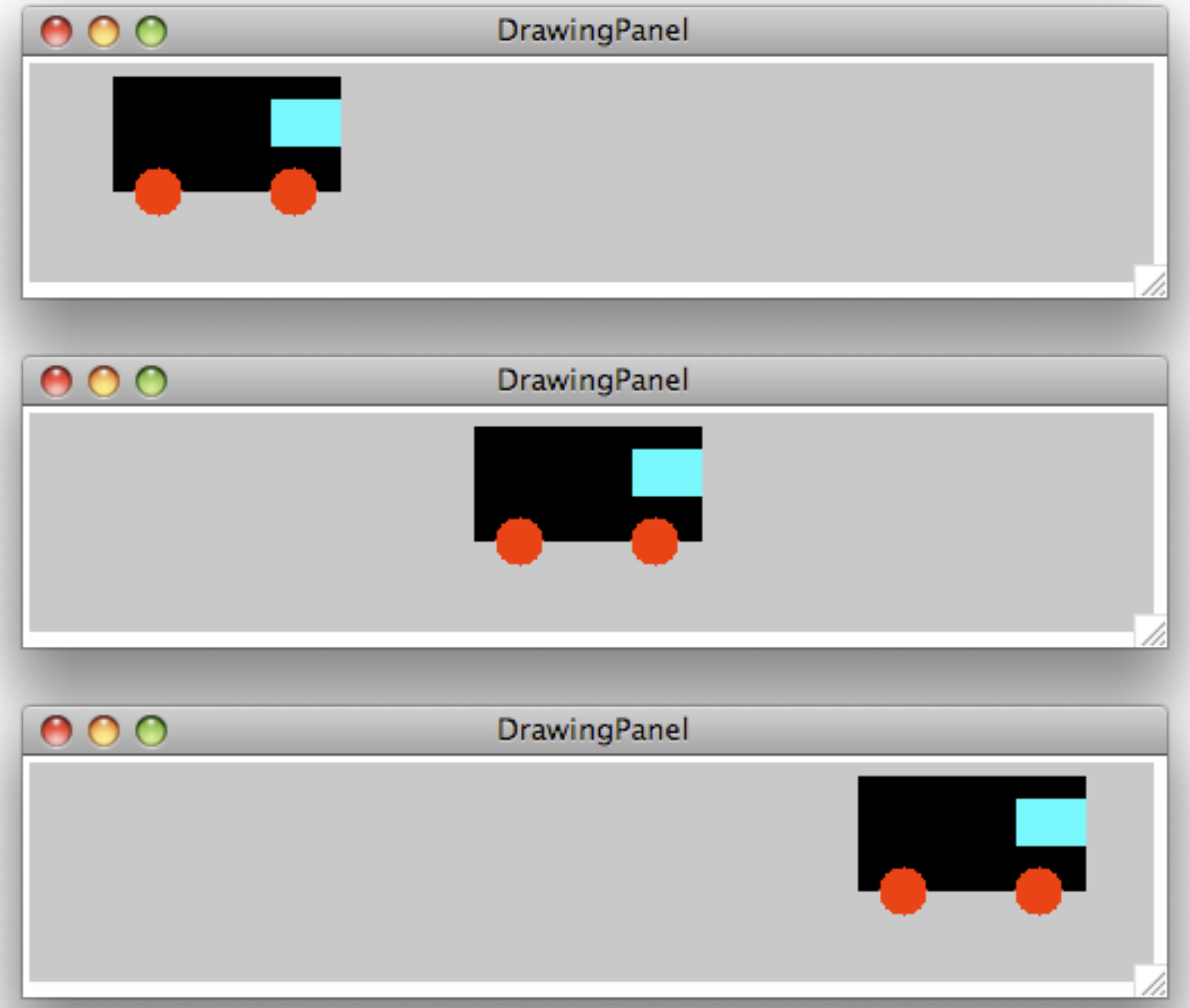


car2.py

```python
from drawingpanel import *

def car(panel, x, y):
    panel.canvas.create_rectangle(x, y, x+100, y+50,
                                  fill="black")
    panel.canvas.create_oval(x+10, y+40, x+30, y+60,
                             fill="red", outline="red")
    panel.canvas.create_oval(x+70, y+40, x+90, y+60,
                             fill="red", outline="red")
    panel.canvas.create_rectangle(x+70, y+10, x+100, y+30,
                                  fill="cyan", outline="cyan")

panel = DrawingPanel(300, 200)
panel.set_background("gray")

car(panel, 10, 30)
car(panel, 170, 80)
car(panel, 50, 120)

panel.mainloop()
```

exercise

animate it using `panel.sleep()`!



car3.py

```python
from drawingpanel import *

def car(panel, x, y):
    panel.canvas.create_rectangle(x, y, x+100, y+50,
                                  fill="black")
    panel.canvas.create_oval(x+10, y+40, x+30, y+60,
                             fill="red", outline="red")
    panel.canvas.create_oval(x+70, y+40, x+90, y+60,
                             fill="red", outline="red")
    panel.canvas.create_rectangle(x+70, y+10, x+100, y+30,
                                  fill="cyan", outline="cyan")

panel = DrawingPanel(500, 100)

for x in range(0, 400, 10):
    panel.canvas.create_rectangle(0, 0, 500, 100,
                                  fill="gray", outline="gray")
    car(panel, x, 10)
    panel.sleep(10)

panel.mainloop()
```