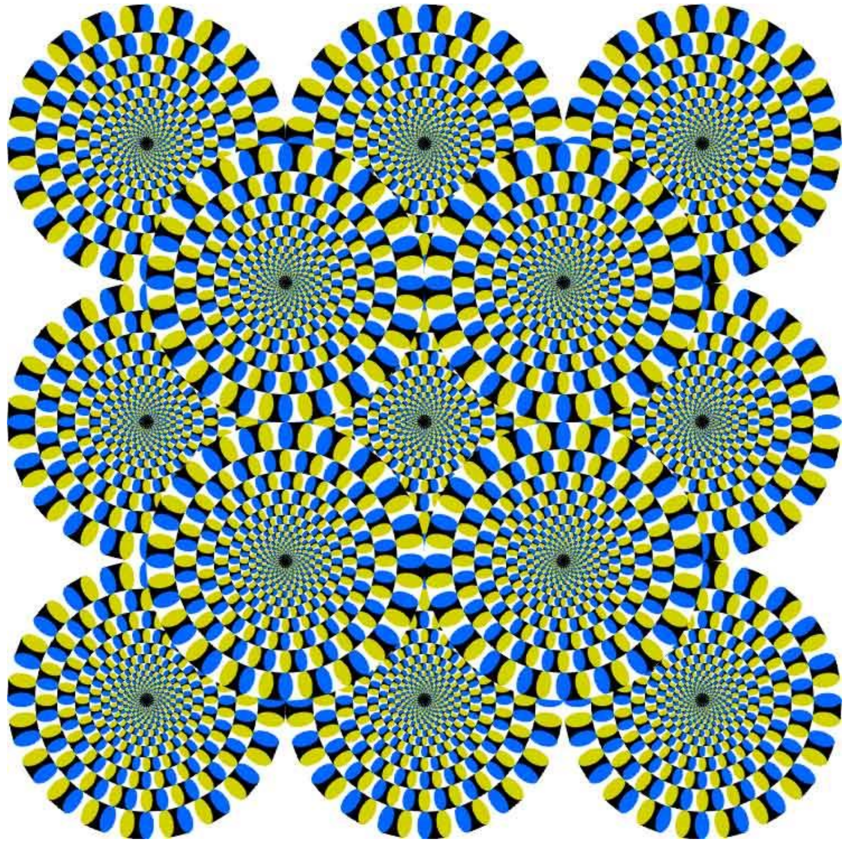
CSc 110, Spring 2017

Lecture 6: Parameters (cont.) and Graphics

Adapted from slides by Marty Stepp and Stuart Reges



Multiple parameters

- A function can accept multiple parameters. (separated by `,`)
  - When calling it, you must pass values for each parameter.

- Declaration:

```
def <name>(<name>, ..., <name>):
    <statement>(s)
```

- Call:

```
<name>(<exp>, <exp>, ..., <exp>)
```

Multiple parameters example

```
def main():
    print_number(4, 9)
    print_number(17, 6)
    print_number(8, 0)
    print_number(0, 8)

def print_number(number, count):
    for i in range(0, count):
        print(number, end="")
    print()
```

Output:

```
444444444
171717171717

00000000
```

- Modify the `stars` program to draw boxes with parameters.

Stars solution

```
# Prints several lines and boxes made of stars.
# Third version with multiple parameterized methods.

def main():
    line(13)
    line(7)
    line(35)
    print()
    box(10, 3)
    box(5, 4)
    box(20, 7)

# Prints the given number of
#stars plus a line break.
def line(count):
    for i in range(0, count):
        print("*", end="")
    print()
```

```
# Prints a box of stars of the given size.
def box(width, height):
    line(width)
    for line in range(0, height - 2):
        print("*", end="")
        for space in range(0, width - 2):
            print(" ", end="")
        print("*")
    line(width)
```

Stars solution

```
# Prints several lines and boxes made of stars.
# Third version with multiple parameterized methods.

def main():
    line(13)
    line(7)
    line(35)
    print()
    box(10, 3)
    box(5, 4)
    box(20, 7)

# Prints the given number of
#stars plus a line break.
def line(count):
    for i in range(0, count):
        print("*", end="")
    print()
```

```
# Prints a box of stars of the given size.
def box(width, height):
    line(width)
    for line in range(0, height - 2):
        print("*", end="")
        for space in range(0, width - 2):
            print(" ", end="")
        print("*")
    line(width)
```

Strings as parameters

```
say_hello("Allison")

teacher = "Bictolia"
say_hello(teacher)

def say_hello(name):
    print("Welcome, " + name)
```

Output:

```
Welcome, Allison
Welcome, Bictolia
```

- Modify the `stars` program to use string parameters. Use a function named `repeat` that prints a string many times.

Stars solution

```
# Prints several lines and boxes made of stars.
# Fourth version with String parameters.

def main():
    line(13)
    line(7)
    line(35)
    print()
    box(10, 3)
    box(5, 4)
    box(20, 7)

# Prints the given number of
# stars plus a line break.
def line(count):
    repeat("*", count)
    print()
```

```
# Prints a box of stars of the given size.
def box(width, height):
    line(width)

    for line in range(height - 2):
        print("*", end="")
        repeat(" ", width - 2)
        print("*")
    line(width)

# Prints the given String the given
# number of times.
def repeat(s, times):
    for i in range(0, times):
        print(s, end="")
```

Value semantics

- **value semantics**: When `numbers` and `strings` are passed as parameters, their values are copied.
  - Modifying the parameter will not affect the variable passed in.

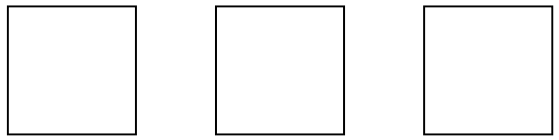
```
def strange(x):
    x = x + 1
    print("1. x = " + x)


x = 23
strange(x)
print("2. x = " + x)
...
```

Output:

```
1. x = 24
2. x = 23
```

A "Parameter Mystery" problem

```
def main():
    x = 9
    y = 2
    z = 5

    mystery(z, y, x)

    mystery(y, x, z)
```

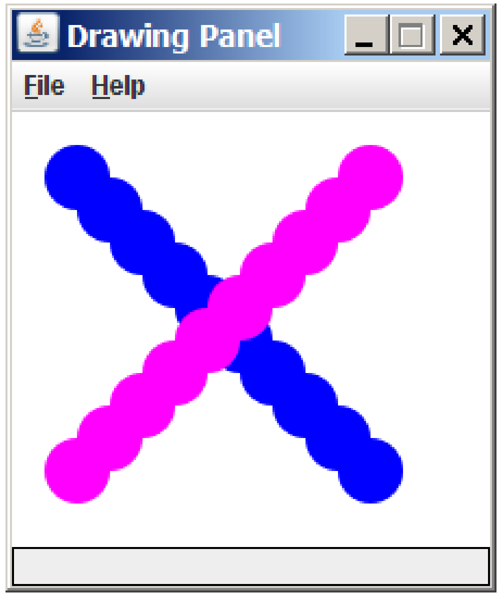


```
def mystery(x, z, y):
    print(str(z) + " and " + str(y - x))
```

Graphical objects

We will draw graphics in Python using a new kind of object:

- `DrawingPanel`: A window on the screen.
  - Not part of Python; provided by the instructor. See class web site.



DrawingPanel

- Import the program that implements DrawingPanel

```
from drawingpanel import *
```

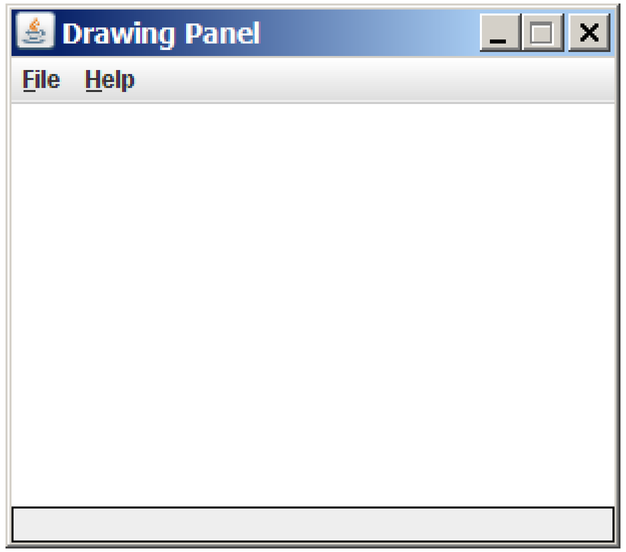
- To create a window:

  <name> `= DrawingPanel(`***<width>***, ***<height>***`)`

  <name> `= DrawingPanel(`***<width>***, ***<height>,*** `background="`***color***`"`)

  Example:

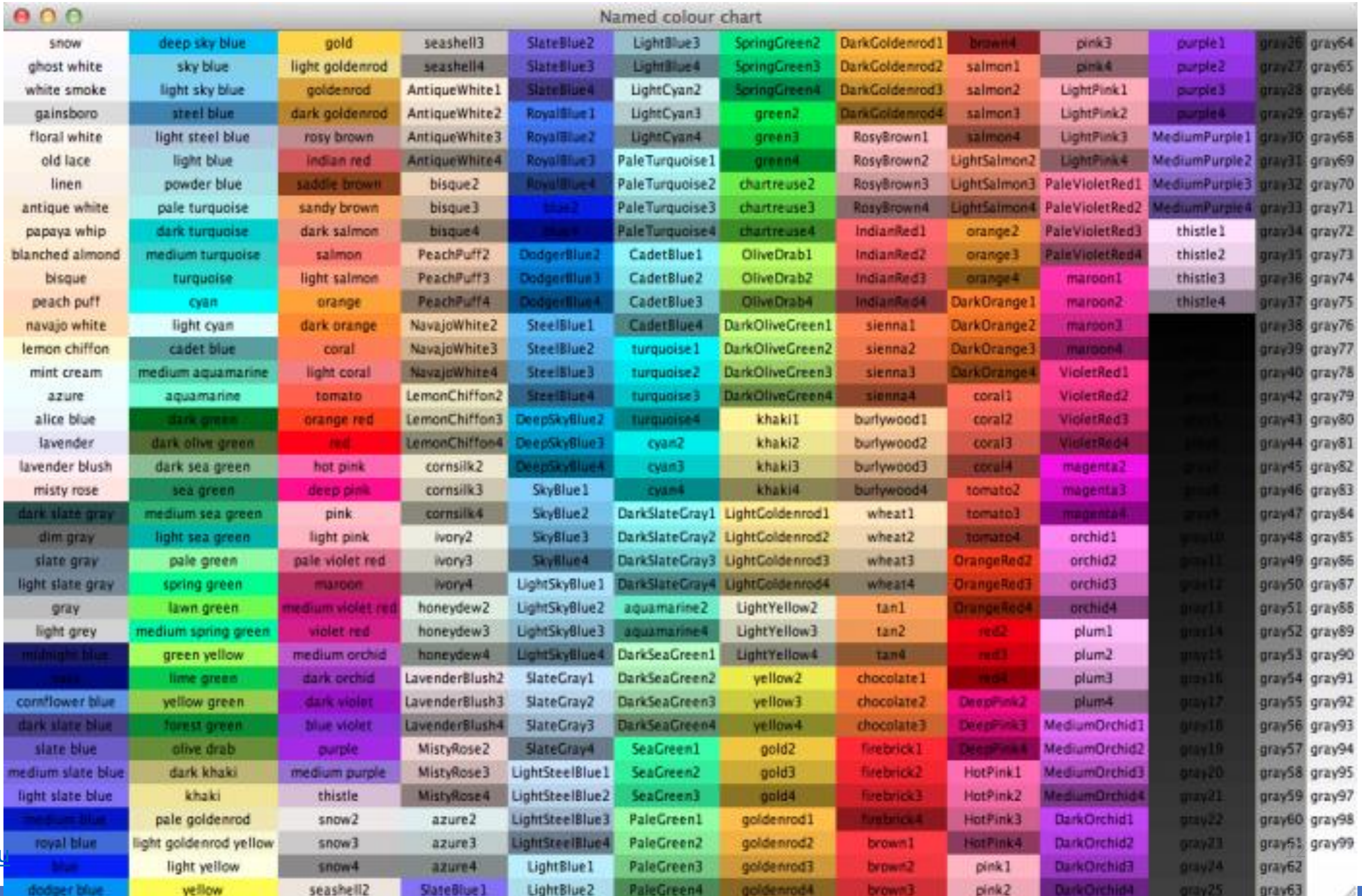
```
panel = DrawingPanel(300, 200)
```



- The window has nothing on it.
  - We can draw shapes and lines on it.
- If passed the optional third parameter

  it will have a background color

Named colors


Named colour chart

snow	deep sky blue	gold	seashell3	SlateBlue2	LightBlue3	SpringGreen2	DarkGoldenrod1	brown4	pink3	purple1	gray26	gray64
ghost white	sky blue	light goldenrod	seashell4	SlateBlue3	LightBlue4	SpringGreen3	DarkGoldenrod2	salmon1	pink4	purple2	gray27	gray65
white smoke	light sky blue	goldenrod	AntiqueWhite1	SlateBlue4	LightCyan2	SpringGreen4	DarkGoldenrod3	salmon2	LightPink1	purple3	gray28	gray66
gainsboro	steel blue	dark goldenrod	AntiqueWhite2	RoyalBlue1	LightCyan3	green2	DarkGoldenrod4	salmon3	LightPink2	purple4	gray29	gray67
floral white	light steel blue	rosy brown	AntiqueWhite3	RoyalBlue2	LightCyan4	green3	RosyBrown1	salmon4	LightPink3	MediumPurple1	gray30	gray68
old lace	light blue	indian red	AntiqueWhite4	RoyalBlue3	PaleTurquoise1	green4	RosyBrown2	LightSalmon2	LightPink4	MediumPurple2	gray31	gray69
linen	powder blue	saddle brown	bisque2	RoyalBlue4	PaleTurquoise2	chartreuse2	RosyBrown3	LightSalmon3	PaleVioletRed1	MediumPurple3	gray32	gray70
antique white	pale turquoise	sandy brown	bisque3	blue2	PaleTurquoise3	chartreuse3	RosyBrown4	LightSalmon4	PaleVioletRed2	MediumPurple4	gray33	gray71
papaya whip	dark turquoise	dark salmon	bisque4	blue4	PaleTurquoise4	chartreuse4	IndianRed1	orange2	PaleVioletRed3	thistle1	gray34	gray72
blanched almond	medium turquoise	salmon	PeachPuff2	DodgerBlue2	CadetBlue1	OliveDrab1	IndianRed2	orange3	PaleVioletRed4	thistle2	gray35	gray73
bisque	turquoise	light salmon	PeachPuff3	DodgerBlue3	CadetBlue2	OliveDrab2	IndianRed3	orange4	maroon1	thistle3	gray36	gray74
peach puff	cyan	orange	PeachPuff4	DodgerBlue4	CadetBlue3	OliveDrab4	IndianRed4	DarkOrange1	maroon2	thistle4	gray37	gray75
navajo white	light cyan	dark orange	NavajoWhite2	SteelBlue1	CadetBlue4	DarkOliveGreen1	sienna1	DarkOrange2	maroon3	[illegible]	gray38	gray76
lemon chiffon	cadet blue	coral	NavajoWhite3	SteelBlue2	turquoise1	DarkOliveGreen2	sienna2	DarkOrange3	maroon4	[illegible]	gray39	gray77
mint cream	medium aquamarine	light coral	NavajoWhite4	SteelBlue3	turquoise2	DarkOliveGreen3	sienna3	DarkOrange4	VioletRed1	[illegible]	gray40	gray78
azure	aquamarine	tomato	LemonChiffon2	SteelBlue4	turquoise3	DarkOliveGreen4	sienna4	coral1	VioletRed2	gray4	gray42	gray79
alice blue	dark green	orange red	LemonChiffon3	DeepSkyBlue2	turquoise4	khaki1	burlywood1	coral2	VioletRed3	gray5	gray43	gray80
lavender	dark olive green	red	LemonChiffon4	DeepSkyBlue3	cyan2	khaki2	burlywood2	coral3	VioletRed4	gray6	gray44	gray81
lavender blush	dark sea green	hot pink	cornsilk2	DeepSkyBlue4	cyan3	khaki3	burlywood3	coral4	magenta2	gray7	gray45	gray82
misty rose	sea green	deep pink	cornsilk3	SkyBlue1	cyan4	khaki4	burlywood4	tomato2	magenta3	gray8	gray46	gray83
dark slate gray	medium sea green	pink	cornsilk4	SkyBlue2	DarkSlateGray1	LightGoldenrod1	wheat1	tomato3	magenta4	gray9	gray47	gray84
dim gray	light sea green	light pink	ivory2	SkyBlue3	DarkSlateGray2	LightGoldenrod2	wheat2	tomato4	orchid1	gray10	gray48	gray85
slate gray	pale green	pale violet red	ivory3	SkyBlue4	DarkSlateGray3	LightGoldenrod3	wheat3	OrangeRed2	orchid2	gray11	gray49	gray86
light slate gray	spring green	maroon	ivory4	LightSkyBlue1	DarkSlateGray4	LightGoldenrod4	wheat4	OrangeRed3	orchid3	gray12	gray50	gray87
gray	lawn green	medium violet red	honeydew2	LightSkyBlue2	aquamarine2	LightYellow2	tan1	OrangeRed4	orchid4	gray13	gray51	gray88
light grey	medium spring green	violet red	honeydew3	LightSkyBlue3	aquamarine4	LightYellow3	tan2	red2	plum1	gray14	gray52	gray89
midnight blue	green yellow	medium orchid	honeydew4	LightSkyBlue4	DarkSeaGreen1	LightYellow4	tan4	red3	plum2	gray15	gray53	gray90
navy	lime green	dark orchid	LavenderBlush2	SlateGray1	DarkSeaGreen2	yellow2	chocolate1	red4	plum3	gray16	gray54	gray91
cornflower blue	yellow green	dark violet	LavenderBlush3	SlateGray2	DarkSeaGreen3	yellow3	chocolate2	DeepPink2	plum4	gray17	gray55	gray92
dark slate blue	forest green	blue violet	LavenderBlush4	SlateGray3	DarkSeaGreen4	yellow4	chocolate3	DeepPink3	MediumOrchid1	gray18	gray56	gray93
slate blue	olive drab	purple	MistyRose2	SlateGray4	SeaGreen1	gold2	firebrick1	DeepPink4	MediumOrchid2	gray19	gray57	gray94
medium slate blue	dark khaki	medium purple	MistyRose3	LightSteelBlue1	SeaGreen2	gold3	firebrick2	HotPink1	MediumOrchid3	gray20	gray58	gray95
light slate blue	khaki	thistle	MistyRose4	LightSteelBlue2	SeaGreen3	gold4	firebrick3	HotPink2	MediumOrchid4	gray21	gray59	gray97
medium blue	pale goldenrod	snow2	azure2	LightSteelBlue3	PaleGreen1	goldenrod1	firebrick4	HotPink3	DarkOrchid1	gray22	gray60	gray98
royal blue	light goldenrod yellow	snow3	azure3	LightSteelBlue4	PaleGreen2	goldenrod2	brown1	HotPink4	DarkOrchid2	gray23	gray61	gray99
blue	light yellow	snow4	azure4	LightBlue1	PaleGreen3	goldenrod3	brown2	pink1	DarkOrchid3	gray24	gray62	
dodger blue	yellow	seashell2	SlateBlue1	LightBlue2	PaleGreen4	goldenrod4	brown3	pink2	DarkOrchid4	gray25	gray63	

Chart credit Smith.edu

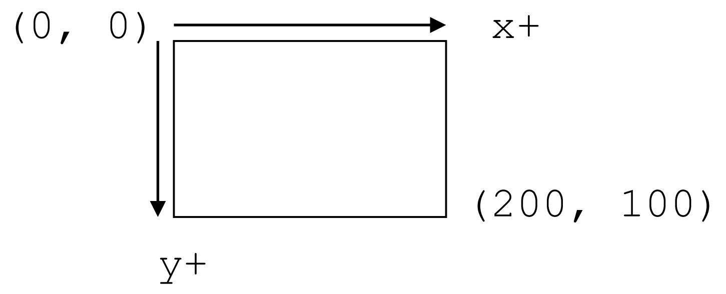
Custom colors

- You can construct custom colors using hex.
  - # followed by six numbers 0 – 9 and letters A – F
    - A is 10, B is 11 and so on
    - #000000 is black
    - #FFFFFF is white
    - Colors get darker as the number gets lower
    - The first two digits are the amount of red, the next two green, the last two blue

```
panel = DrawingPanel(80, 50, background="#3367D3")
```

Coordinate system

- Each (x, y) position is a *pixel* ("picture element").
- (0, 0) is at the window's top-left corner.
  - x increases rightward and the y increases <u>downward</u>.
- The rectangle from (0, 0) to (200, 100) looks like this:



Drawing shapes

`panel.canvas.create_line(`*x1, y1, x2, y2,* `fill="`*color*`")`

line between points (*x1*, *y1*), (*x2*, *y2*) in color

`panel.canvas.create_oval(`*x1, y1,* x2, y2, `outline="`*color*`")`

outline largest oval that fits in a box with top-left at (*x1*, *y1*) and lower right at (x2, y2) outlined in color

`panel.canvas.create_rectangle(`*x1, y1,* x2, y2, `outline="`*color*`")`

outline of rectangle with top-left at (*x1*, *y1*) and bottom right at (x2, y2) outlined in color

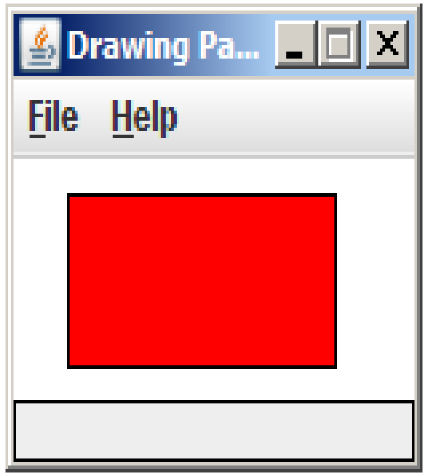
`panel.canvas.create_text(`*x, y,* `text="`*string*`")`

text centered vertically and horizontally around (x, y)

Filled in shapes

- To draw a shape with a fill set its `fill` instead of `outline`.

```
from drawingpanel import *        # so I can use Graphics

def main():
    p = DrawingPanel(150, 70)

    # inner red fill
    p.canvas.create_rectangle(20, 10, 120, 60, fill="red")
```



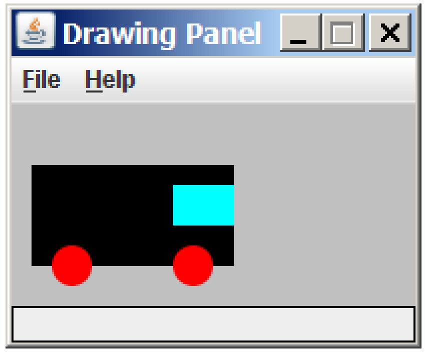
- This will automatically fill the shape but give it a black border. To remove the border add `width=0`.

```
    p.canvas.create_rectangle(20, 10, 120, 60, fill="red", width=0)
```

Superimposing shapes

- When two shapes occupy the same pixels, the last one drawn is seen.

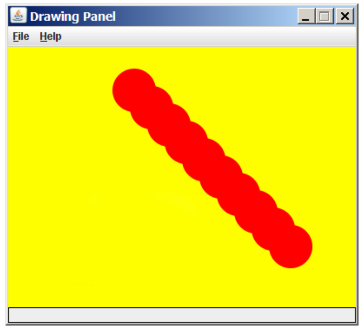
```
from drawingpanel import *

def main():
    p = DrawingPanel(200, 100, background="light gray")

    p.canvas.create_rectangle(10, 30, 110, 80, fill="black")

    p.canvas.create_oval(20, 70, 40, 90, fill="red", width=0)
    p.canvas.create_oval(80, 70, 100, 90, fill="red", width=0)

    p.canvas.create_rectangle(80, 40, 110, 60, fill="cyan", width=0)
```



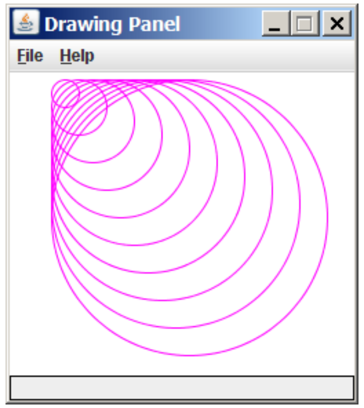
Drawing with loops

- The *x1*, *y1*, *x2*, *y2* expression can use any variable.

```
panel = DrawingPanel(400, 300, background="yellow")

for i in range(1, 11):
    panel.canvas.create_oval (100 + 20 * i, 5 + 20 * i,
                              150 + 20 * i, 55 + 20 * i
                              fill="red", width=0)
```



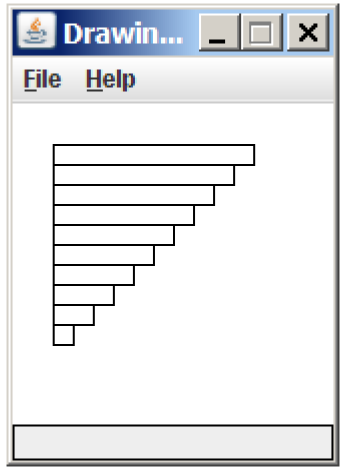
```
panel = DrawingPanel(250, 220)
for i in range(1, 11):
    panel.canvas.create_oval (30, 5, 30 + 20 * i,
                              5 + 20 * i, fill="magenta")
```



Loops that begin at 0

- Beginning a loop at 0 can make coordinates easier to compute.
- Example:
  - Draw ten stacked rectangles starting at (20, 20), height 10, width starting at 100 and decreasing by 10 each time:

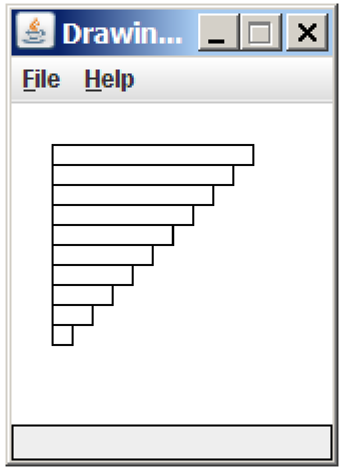
```
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.canvas.create_rectangle (20, 20 + 10 * i,
                                   120 - 10 * i, 30 + 10 * i)
```



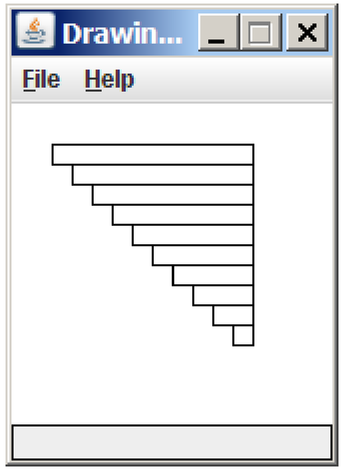
Drawing w/ loops questions

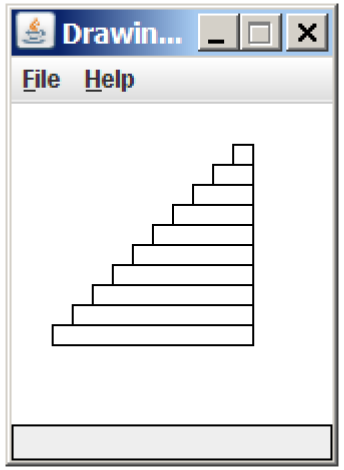
- Code from previous slide:

```
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.canvas.create_rectangle (20, 20 + 10 * i,
                                   120 - 10 * i, 30 + 10 * i)
```



- Write variations of the above program that draw the figures at right as output.

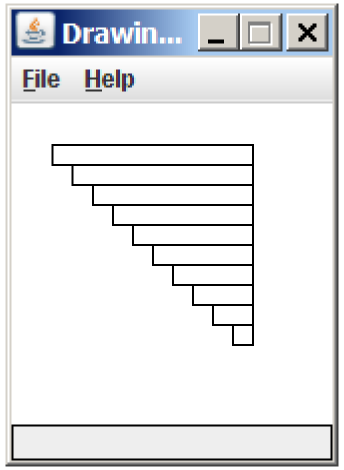




Drawing w/ loops answers

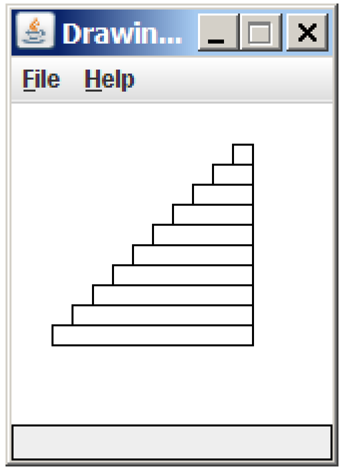
- **Solution #1:**

```
panel = DrawingPanel(160, 160)
for i in range(0, 10):
    panel.canvas.create_rectangle (20 + 10 * i, 20 + 10 * i,
                                   120, 30 + 10 * i)
```



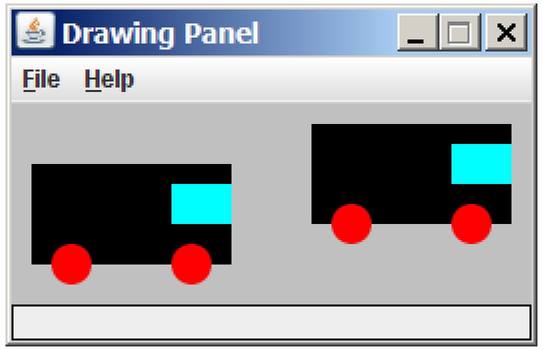
- **Solution #2:**

```
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.canvas.create_rectangle (110 - 10 * i, 20 + 10 * i,
                                   120, 30 + 10 * i)
```

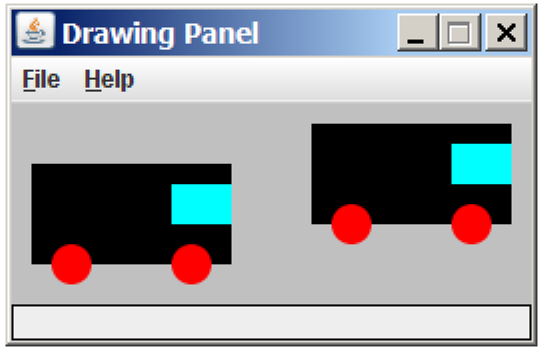


Parameterized figures

- Modify the car-drawing function so that it can draw many cars, such as in the following image.
  - Top-left corners: (10, 30), (150, 10)
  - Hint: We must modify our `draw_car` function to accept x/y coordinates as parameters.



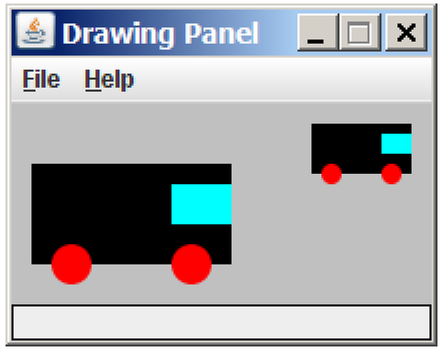
Parameterized answer

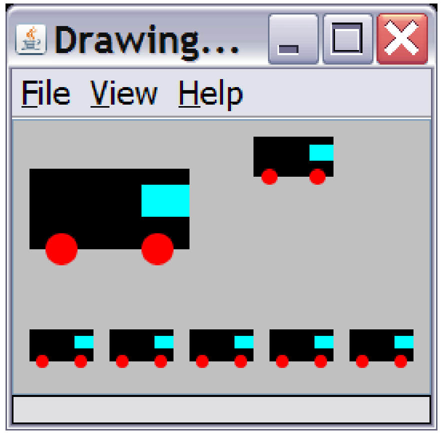


```
def main():
    panel = DrawingPanel(260, 100, background="light gray")
    draw_car(panel, 10, 30)
    draw_car(panel, 150, 10)

def draw_car(p, x, y):
    p.canvas.create_rectangle(x, y, 100 + x, 50 + y, fill="black")

    p.canvas.create_oval(x + 10, y + 40, x + 30, y + 60, fill="red", width=0)
    p.canvas.create_oval(x + 70, y + 40, x + 90, y + 60, fill="red", width=0)

    p.canvas.create_rectangle(x + 70, y + 10, x + 100, y + 30, fill="cyan",
                                             width=0)
```

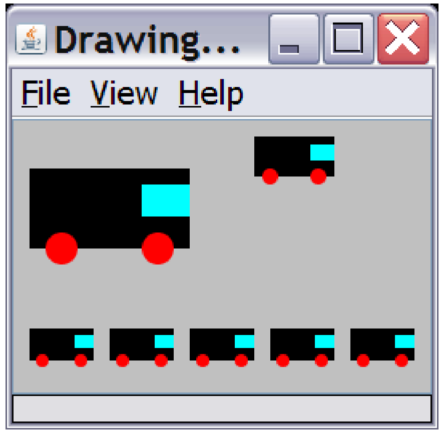
Drawing parameter question

- Modify `draw_car` to allow the car to be drawn at any size.
  - Existing car: size 100. Second car: (150, 10), size 50.
- Once you have this working, use a `for` loop with your function to draw a line of cars, like the picture at right.
  - Start at (10, 130), each size 40, separated by 50px.





Drawing parameter answer



```
def main():
    panel = DrawingPanel(260, 100, background="light gray")
    draw_car(panel, 10, 30, 100)
    draw_car(panel, 150, 10, 50)
    for i in range(0, 5):
        draw_car(panel, 10 + i * 50, 130, 40);

def draw_car(p, x, y, size):
    p.canvas.create_rectangle(x, y, x + size, y + size / 2, fill="black")

    p.canvas.create_oval(x + size / 10, y + size / 10 * 4, x + size / 10 * 3, y +
                         size / 10 * 6, fill="red", width=0)
    p.canvas.create_oval(x + size / 10 * 7, y + size / 10 * 4, x + size / 10 * 9,
                         y + size / 10 * 6, fill="red", width=0)

    p.canvas.create_rectangle(x + size / 10 * 7, y + size / 10, x + size,
                              y + size / 10 * 3, fill="cyan", width=0)
```

Animation with `sleep`

- `DrawingPanel`'s `sleep` function pauses your program for a given number of milliseconds.

- You can use `sleep` to create simple animations.

```
panel = DrawingPanel(250, 200)
for i in range(1, NUM_CIRCLES + 1):
    panel.canvas.create_oval(15 * i, 15 * i, 30 + 15 * i, 30 + 15 * i)
    panel.sleep(500)
```

  - Try adding `sleep` commands to loops in past exercises in this chapter and watch the panel draw itself piece by piece.