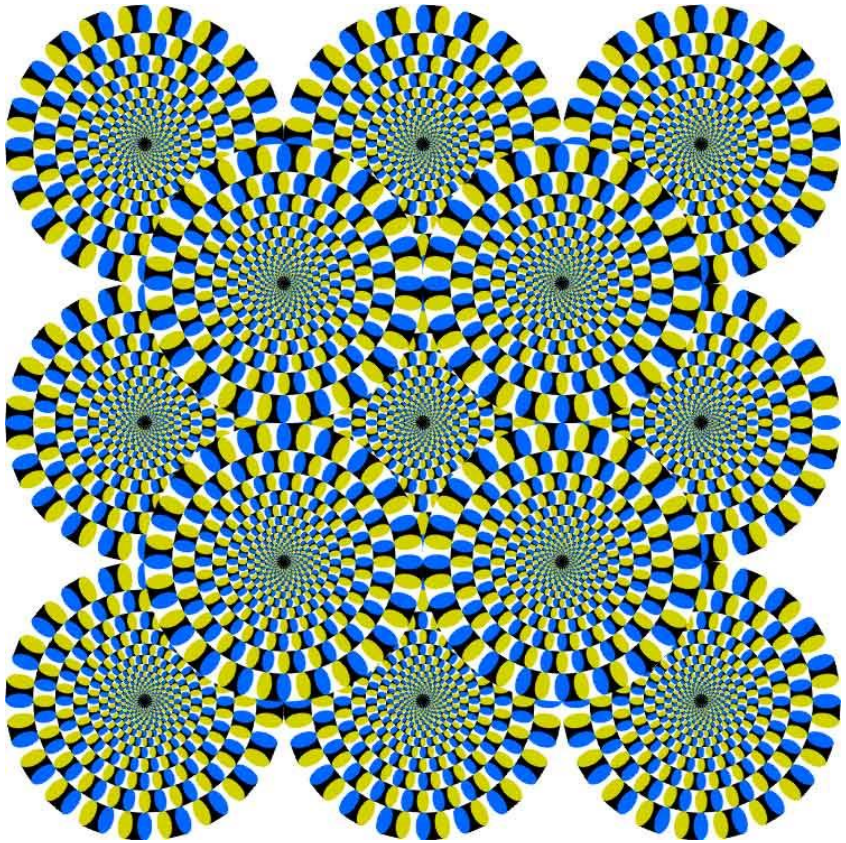
CSc 110, Autumn 2017

Lecture 7: Multiple Parameters and Graphics

Adapted from slides by Marty Stepp and Stuart Reges



Multiple parameters

- A function can accept multiple parameters. (separate 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

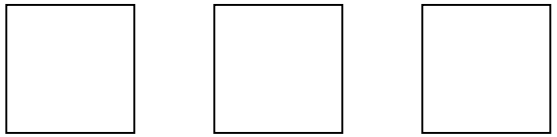
```
def main():
    print_number(4, 9)
    print_number(17, 6)
    print_number(8, 0)
    print_number(0, 8)

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

Output:

```
444444444
171717171717

00000000
```

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(z, "and", (y - x))
```

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)

def main():
    x = 23
    strange(x)
    print("2. x = ", x)
...
```

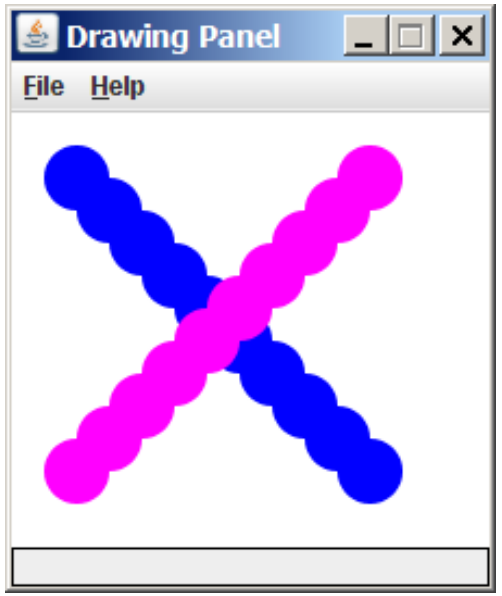
Output:

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

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.



Named colors

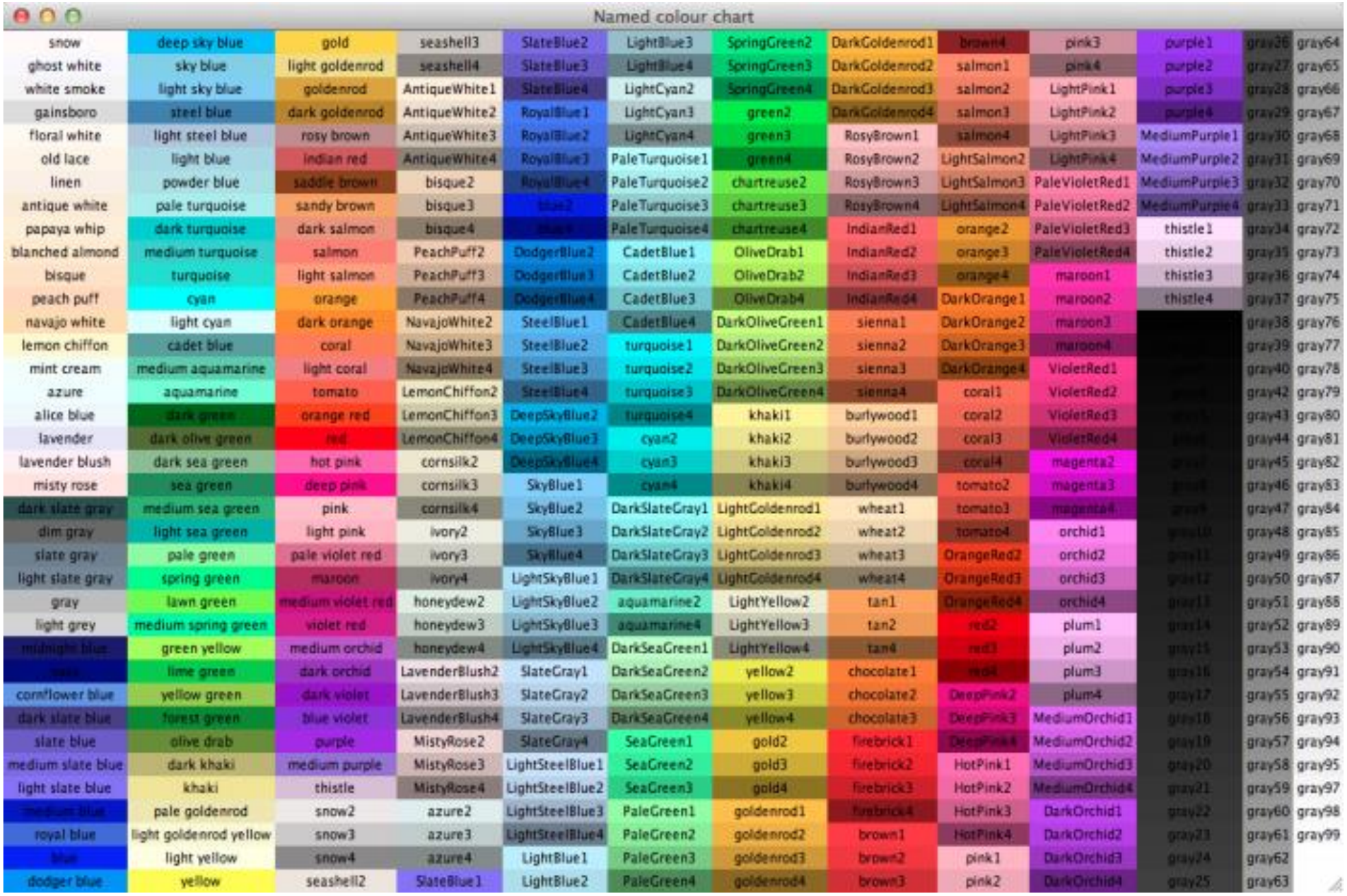


Chart credit [Smith.edu]

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")
```

Drawing shapes

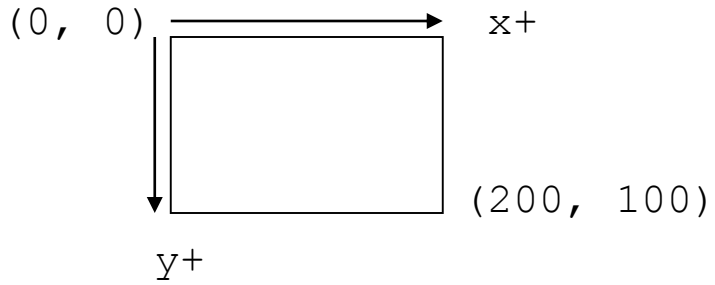
Function name	Description
`p.draw_line(`*x1*, *y1*, *x2*, *y2*`)`	line between points (*x1*, *y1*), (*x2*, *y2*)
`p.draw_oval(`*x*, *y*, *width*, *height*`)`	outline largest oval that fits in a box of size *width* * *height* with top-left at (*x*, *y*)
`p.draw_rect(`*x*, *y*, *width*, *height*`)`	outline of rectangle of size *width* * *height* with top-left at (*x*, *y*)
`p.draw_string("`*text*`",` *x*, *y*`)`	text with upper-left at (x, y)
`p.fill_oval(`*x*, *y*, *width*, *height*`)`	fill largest oval that fits in a box of size *width* * *height* with top-left at (*x*,*y*)
`p.fill_rect(`*x*, *y*, *width*, *height*`)`	fill rectangle of size *width* * *height* with top-left at (*x*, *y*)
`p.set_color("`*color*`")`	set the default color to "color"

- You can pass an additional "color" to any shape as a last parameter

```
p.draw_rect(50, 100, 60, 60, "red")
```

Coordinate system

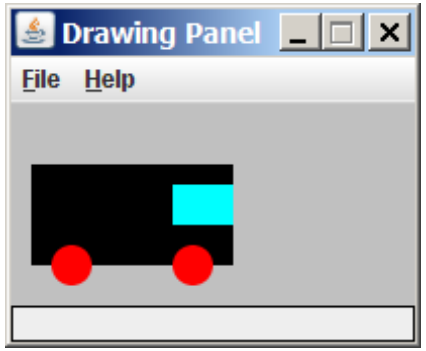
- Each (x, y) position is a *pixel* ("picture element").
- (0, 0) is at the window's top-left corner.
  - x increases rightward and the y increases downward.
- The rectangle from (0, 0) to (200, 100) looks like this:



Superimposing shapes

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

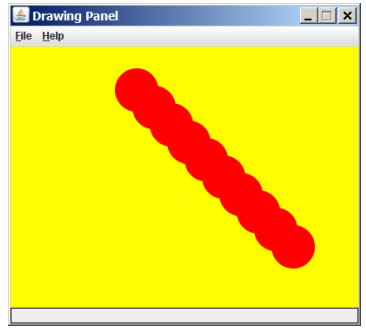
```
from DrawingPanel import *

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

    p.fill_rect(10, 30, 100, 50, "black")

    p.fill_oval(20, 70, 20, 20, "red")
    p.fill_oval(80, 70, 20, 20, "red")

    p.fill_rect(80, 40, 30, 20, "cyan")
```



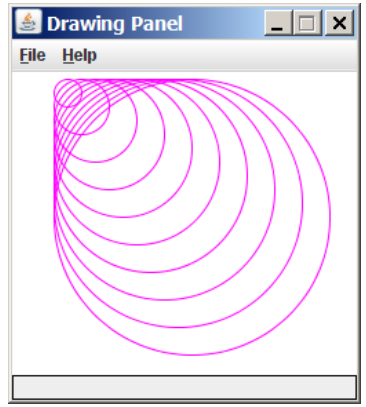
Drawing with loops

- The *x1*, *y1*, *w*, h expression can contain the loop counter, `i`.

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

for i in range(1, 11):
    panel.fill_oval (100 + 20 * i, 5 + 20 * i,
                     50, 50, "red")
```

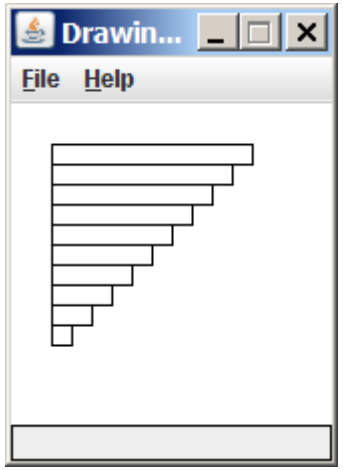



```
panel = DrawingPanel(250, 220)
for i in range(1, 11):
    panel.draw_oval (30, 5, 20 * i, 20 * i, "magenta")
```

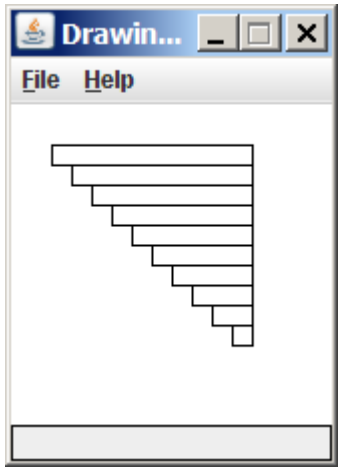


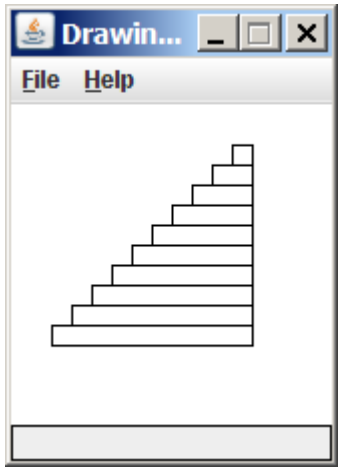
Drawing w/ loops questions

```
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.draw_rectangle (20, 20 + 10 * i,
                          100 - 10 * i, 10)
```



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

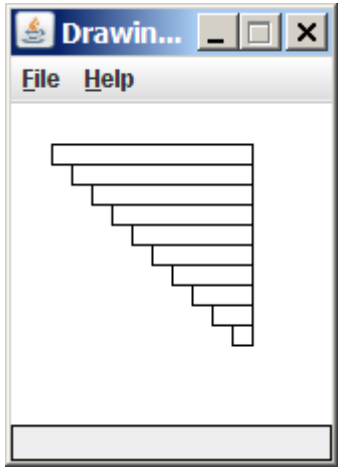




Drawing w/ loops answers

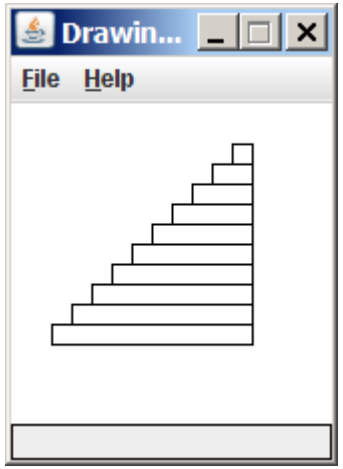
- Solution #1:

```
panel = DrawingPanel(160, 160)
for i in range(0, 10):
    panel.draw_rectangle (20 + 10 * i, 20 + 10 * i,
                          100 - 10 * i, 10)
```



- Solution #2:

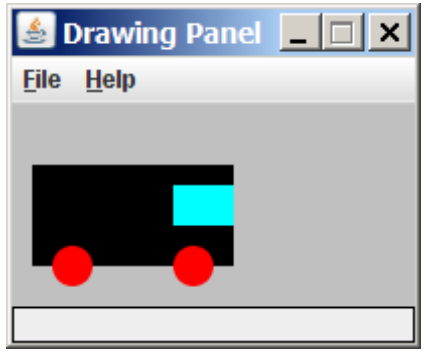
```
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.draw_rect(110 - 10 * i, 20 + 10 * i,
                    10 + 10 * i, 10)
```



Drawing with functions

- To draw in multiple functions, you must pass `DrawingPanel`.

```
def main():
    panel = DrawingPanel(200, 100, background="light gray")
    draw_car(panel)

def draw_car(p):
    p.fill_rect(10, 30, 100, 50, "black")

    p.fill_oval(20, 70, 20, 20, "red")
    p.fill_oval(80, 70, 20, 20, "red")

    p.fill_rect(80, 40, 30, 20, "cyan")
```



Pseudo-Randomness

- Computers generate numbers in a predictable way using a mathematical formula
- Parameters may include current time, mouse position
  - In practice, hard to predict or replicate
- True randomness uses natural processes
  - Atmospheric noise (http://www.random.org/)
  - Lava lamps (patent #5732138)
  - Radioactive decay

Random

- `random` generates pseudo-random numbers.
  - `random` can be accessed by including the following statement:

```
import random
```

Method name	Description
`random.random()`	returns a random float in the range [0, *1*) in other words, 0 inclusive to *1* exclusive
`random.randint(`***min, max***`)`	returns a random integer in the range [min, *max*) in other words, min to *max*-1 inclusive

  - Example:

```
import random
random_number = random.randint(1, 10)    # 1-9
```

Generating random numbers

- To get a number in arbitrary range [*min*, *max*] inclusive:

  `random.randint(`***min, max***`)`

  - Where ***size of range*** is (***max*** `-` ***min*** `+ 1`)

- Example: A random integer between 4 and 10 inclusive:

```
n = random.randint(4, 10)
```