python™

Week 3

parameters and graphics

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

Parameters

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

– Parameters are declared by writing their names (no types)

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

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

Exercise

- Recreate the lines/boxes of stars example from lecture:

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

*******

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

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

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

Exercise Solution

stars.py

```
 1  # Draws a box of stars with the given width and height.
 2  def box(width, height):
 3      print width * "*"
 4      for i in range(height - 2):
 5          print "*" + (width - 2) * " " + "*"
 6      print width * "*"
 7
 8  # main
 9  print 13 * "*"
10  print  7 * "*"
11  print 35 * "*"
12  box(10, 3)
13  box(5, 4)
```

Default Parameter Values

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

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

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

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

- **Exercise:** Modify `stars.py` to add an optional parameter for the character to use for the outline of the box (default `"*"`).

Parameter Keywords

name(parameter=value, ..., parameter=value)

- Can specify the names of parameters as you call a function
- This allows you to pass the parameters in any order

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

>>> print_many(str="shrubbery", n=4)
shrubbery
shrubbery
shrubbery
shrubbery
>>> print_many(n=3, str="Ni!")
Ni!
Ni!
Ni!
```

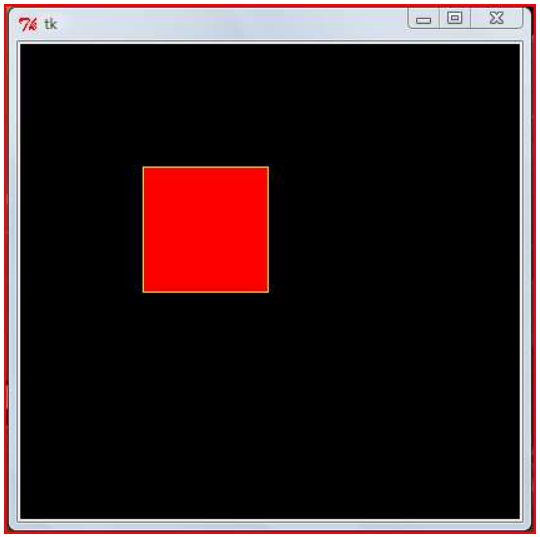
DrawingPanel

- Instructor-provided `drawingpanel.py` file must be in the same folder as your Python program

- At the top of your program, write:
  - `from drawingpanel import *`

- Panel's `canvas` field behaves like `Graphics g` in Java

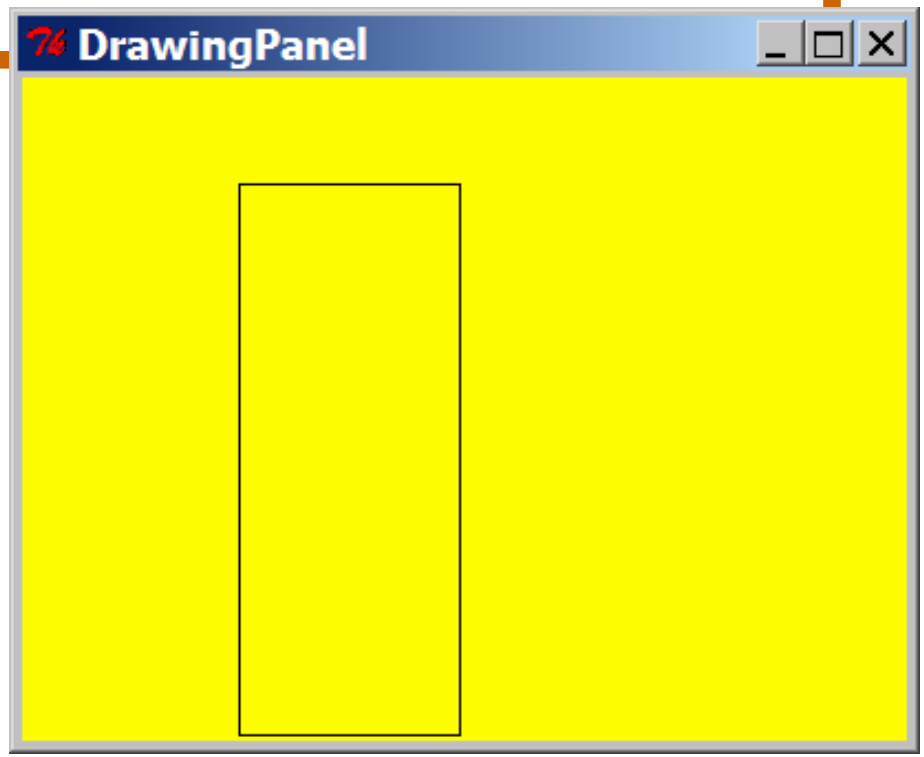
- Must end program with:
  - `panel.mainloop()`



DrawingPanel Example

draw1.py

```
1  from drawingpanel import *
2
3  panel = DrawingPanel(400, 300)
4  panel.canvas["bg"] = "yellow"
5  panel.canvas.create_rectangle(100, 50, 200, 300)
6  panel.mainloop()
```



Drawing Methods

Java	Python
`drawLine`	**panel**`.canvas.create_line(`**x1**`, `**y1**`, `**x2**`, `**y2**`)`
`drawRect`, `fillRect`	**panel**`.canvas.create_rect(`**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**`.canvas["bg"] = `**color**

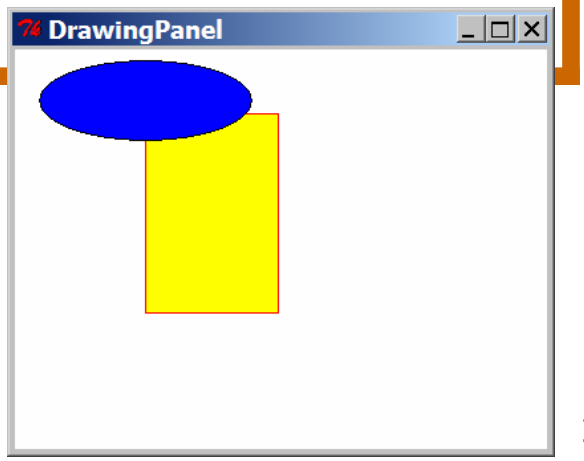
- Notice, methods take x2/y2 parameters, not width/height

Colors and Fill

- Python doesn't have `fillRect`, `fillOval`, or `setColor`.
  - Instead, pass outline and fill colors when drawing a shape.
  - List of all color names: http://wiki.tcl.tk/16166

drawcolors.py

```
1  from drawingpanel import *
2
3  panel = DrawingPanel(400, 300)
4  panel.canvas.create_rectangle(100, 50, 200, 200,
           outline="red", fill="yellow")
5  panel.canvas.create_oval(20, 10, 180, 70, fill="blue")
6  panel.mainloop()
```

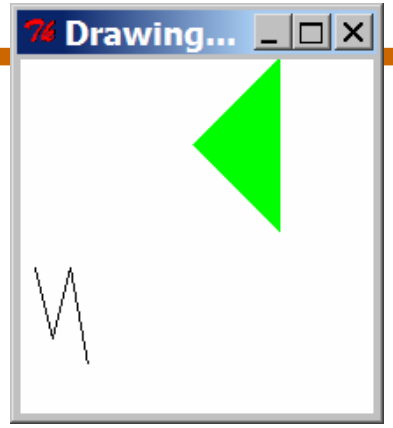


Polygons

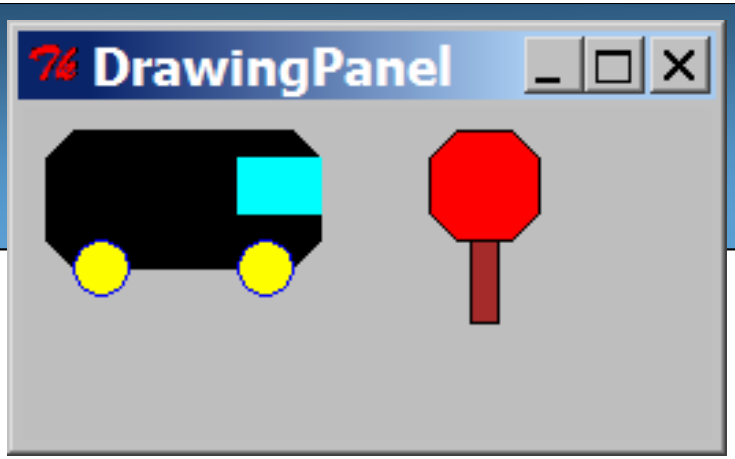
- Draw arbitrary polygons with `create_polygon`
- Draw line groups by passing more params to `create_line`

drawpoly.py

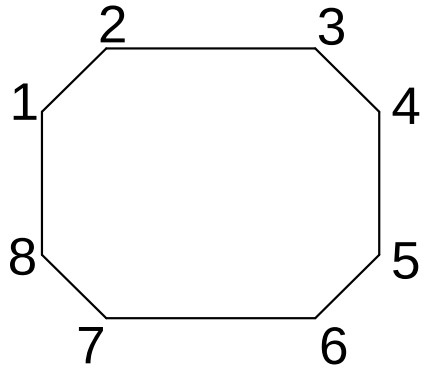
```
1  from drawingpanel import *
2
3  panel = DrawingPanel(200, 200)
4  panel.canvas.create_polygon(100, 50, 150, 0,
                               150, 100, fill="green")
5  panel.canvas.create_line(10, 120, 20, 160,
                            30, 120, 40, 175)
6  panel.mainloop()
```



Exercise



- Write a modified Cars program:
  - Window: 250x120, gray background
  - Car at (10, 10), size 100 (scalable)
    - wheels: yellow fill w/ blue outlines
  - Stop sign at (150, 10), size 40
    - post at (165, 50), size 10x30, brown fill



  - Write an **octagon** function to draw the car body / stop sign.
    - Points of car body, located at (10, 10):

      1. (10, 20), 2. (20, 10), 3. (100, 10), 4. (110, 20),
      5. (110, 50), 6. (100, 60), 7. (20, 60), 8. (10, 50)

    - Points of stop sign, located at (150, 10):

      1. (150, 20), 2. (160, 10), 3. (180, 10), 4. (190, 20),
      5. (190, 40), 6. (180, 50), 7. (160, 50), 8. (150, 40)

Animation

- Pause the panel with `sleep`

animation.py

```
 1  from drawingpanel import *
 2
 3  panel = DrawingPanel(350, 300)
 4  for i in range(20):
 5      # clear any previous image
 6      panel.canvas.create_rectangle(0, 0, 400, 400,
                outline="white", fill="white")
 7
 8      panel.canvas.create_polygon(20 * i, 50, 20 * i,
                100, 20 * i + 50, 75)
 9      panel.sleep(100)
10
11  panel.mainloop()
```