



# Lecture 3

parameters, return, math, graphics

Special thanks CS Washington Lecture notes  
Except where otherwise noted, this work is licensed under:  
<http://creativecommons.org/licenses/by-nc-sa/3.0>

# Parameters

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

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

```
print_many.py  
def print_many(word, n):  
    for i in range(n):  
        print word  
print_many('hello', 4)  
>>>  
hello  
hello  
hello  
hello
```

# Exercise

- Write a function `box(width, height)` and print the following patterns by calling the function, for example,

`box(10,3)`

`box(5,4)`

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

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

# 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  box(10, 3)
10 box(5, 4)
11
12
13
```

# Default Parameter Values

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

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

```
>>> 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
```

- **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(word, n):  
...     for i in range(n):  
...         print word  
  
>>> print_many(word="shrubbery", n=4)  
shrubbery  
shrubbery  
shrubbery  
shrubbery  
  
>>> print_many(n=3, word="Ni!")  
Ni!  
Ni!  
Ni!
```

# Math commands

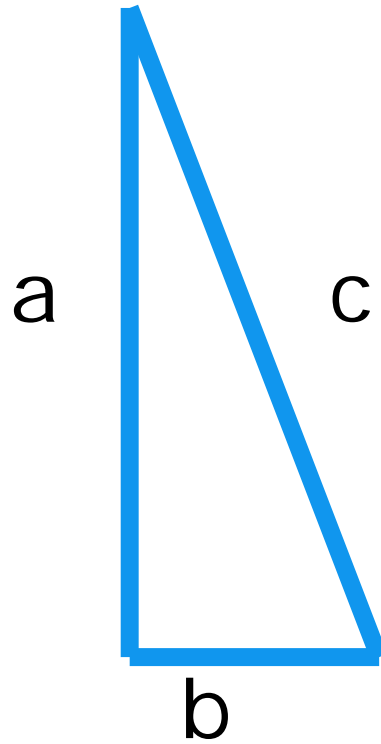
```
from math import *
```

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

Constant	Description
<code>e</code>	2.7182818...
<code>pi</code>	3.1415926...

# Exercise

Write a simple program to find the hypotenuse  $c$  of a right triangle given sides  $a$  and  $b$





# Exercise solution

```
from math import *  
a=3  
b=4  
c=sqrt(3**2 + 4**2)
```

```
print('c=',c)
```

```
# let's change a=-3,  
a=-3  
c=sqrt(3**2 + 4**2)
```

```
print('c=',c)
```

```
# how to fix it? (how to check a, b to make sure >0)
```

# Returning Values

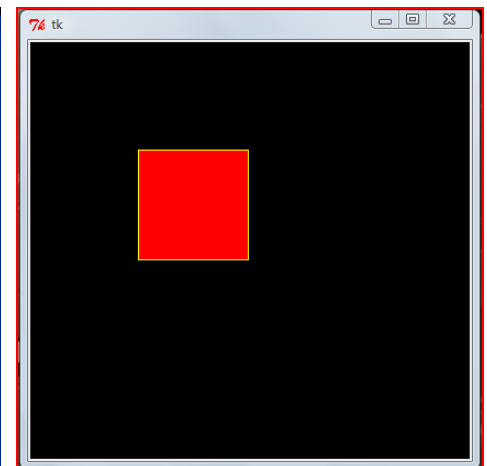
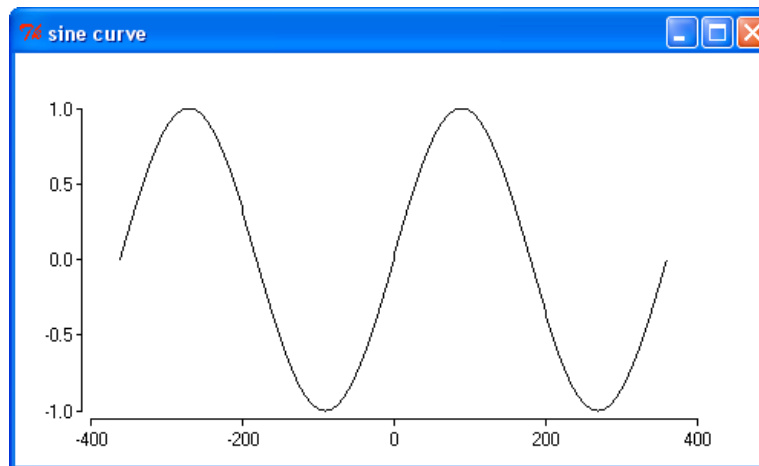
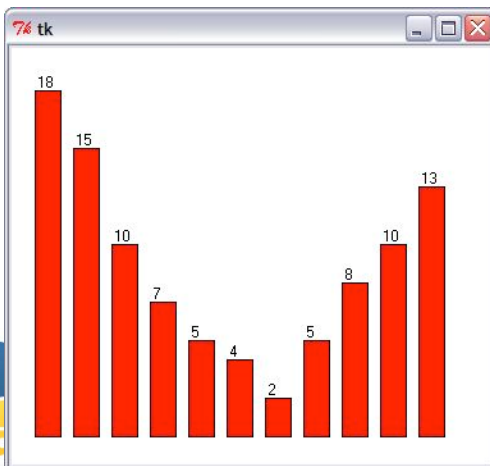
```
def name(parameters) :  
    statements  
    ...  
    return value
```

```
def ftoc(temp):  
    tempc = 5.0 / 9.0 * (temp - 32)  
    return tempc  
  
print(ftoc(98.6))
```

Write a Python code to convert the Celsius to Fahrenheit. Define a function and Call this function to convert 24, 30 Celsius to Fahrenheit.

# 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`

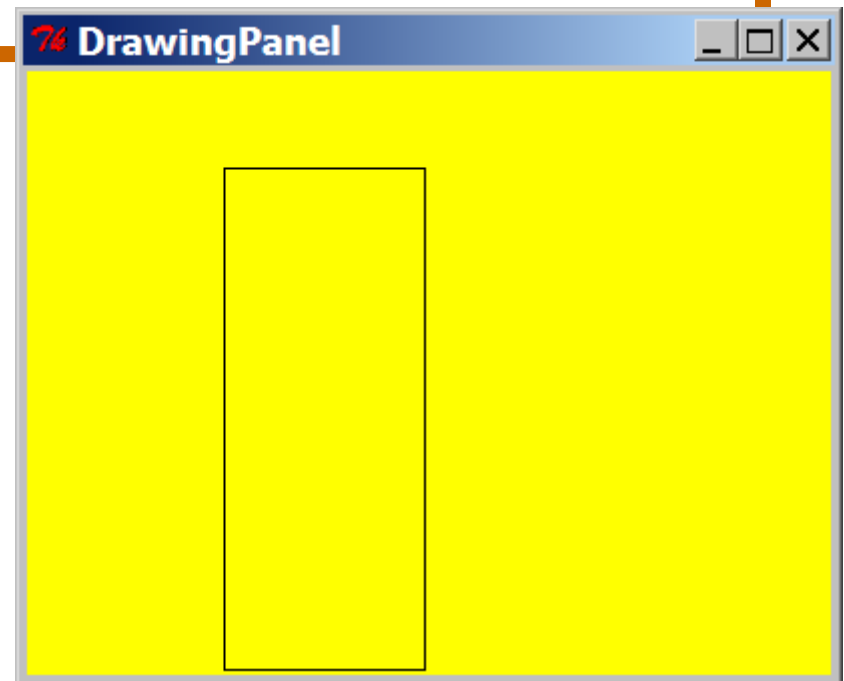


# DrawingPanel Example

## draw1.py

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

**Left upper corner (0,0)**  
**Right lower corner(399,299)**



# Drawing Methods

Python
<code>panel.canvas.create_line(x1, y1, x2, y2)</code>
<code>panel.canvas.create_rectangle(x1, y1, x2, y2)</code>
<code>panel.canvas.create_oval(x1, y1, x2, y2)</code>
<code>panel.canvas.create_text(x, y, text="text")</code>
<i>(see next slide)</i>
<code>panel.set_background(color)</code>

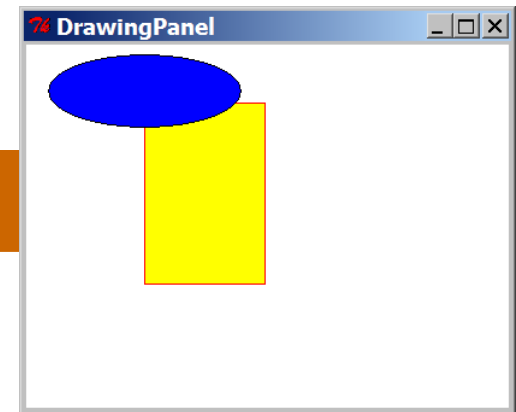
- Notice, methods take x2,y2 parameters, not width/height

# 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>
  - [Visual display of all colors](#)

## drawcolors.py

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

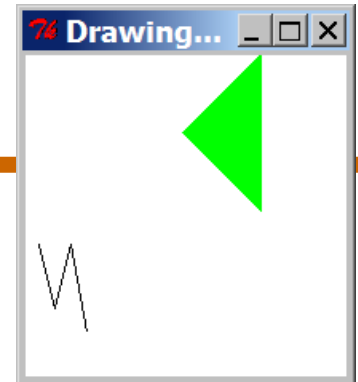


# Polygons

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

## drawpoly.py

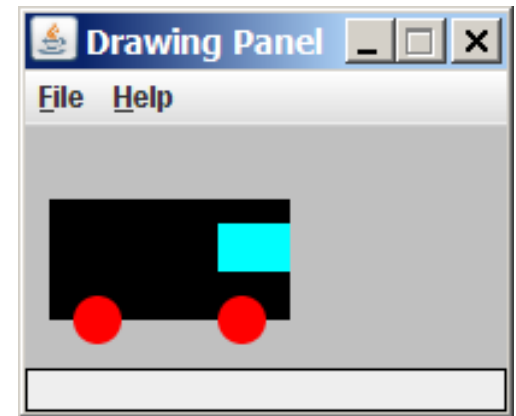
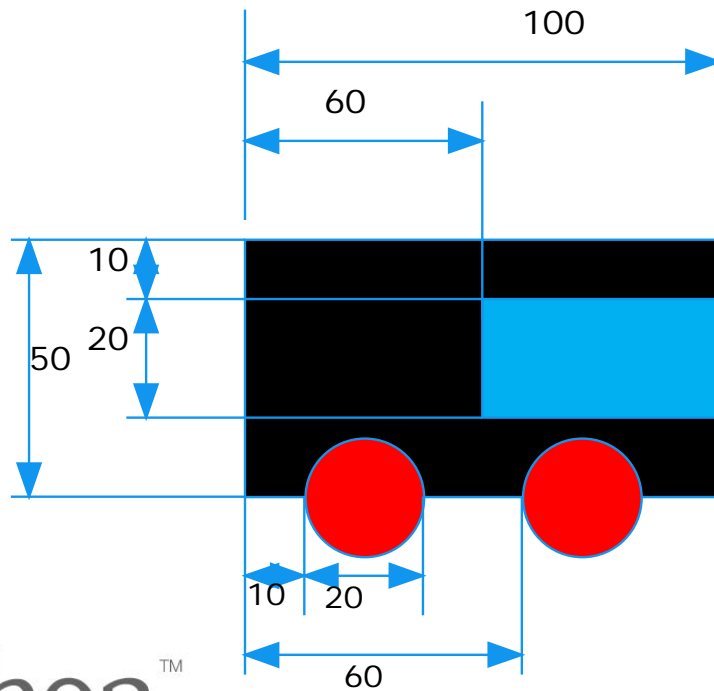
```
1 from drawingpanel import *
2
3 panel = DrawingPanel(200, 200)
4 panel.canvas.create_polygon(100, 50, 150, 0,
                             150, 100, fill="green")
5 panel.canvas.create_line(10, 120, 20, 160,
                           30, 120, 40, 175)
6
```



# Exercise

Let's create a car in Python:

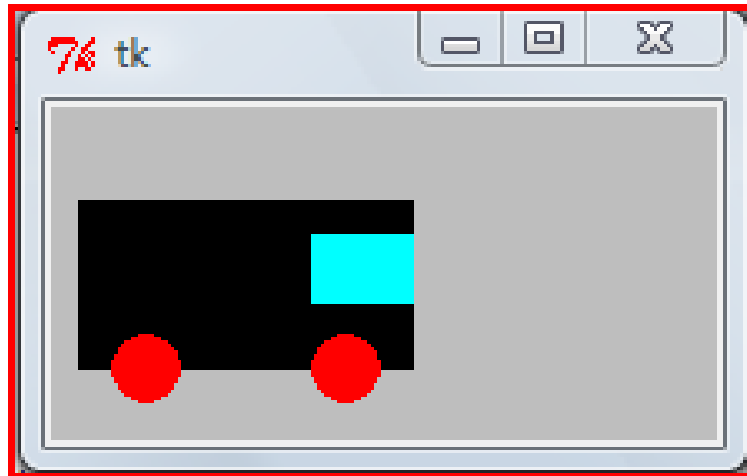
Car: width = 100 pix, height = 50 pix





# Exercise

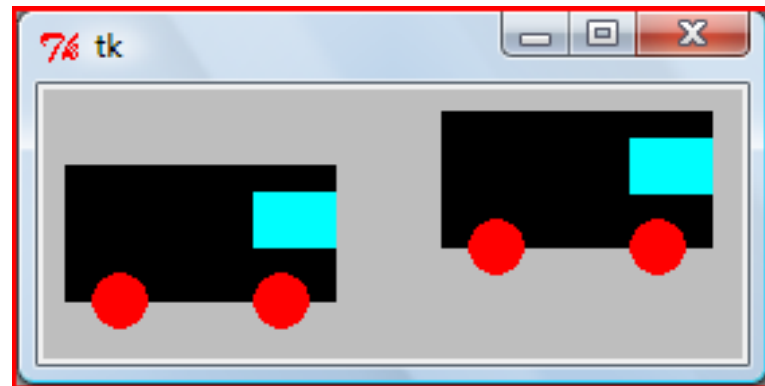
## Draw a car in Python



# Exercise

Now, let's use parameters so that we can place the cars all over the DrawingPanel.

i.e., define a function to draw a car and then call this function to draw the function in different places.



# Exercise

Animate it using `panel.sleep()`  
`panel.sleep(200)`,  
→ sleep 200 mini seconds

