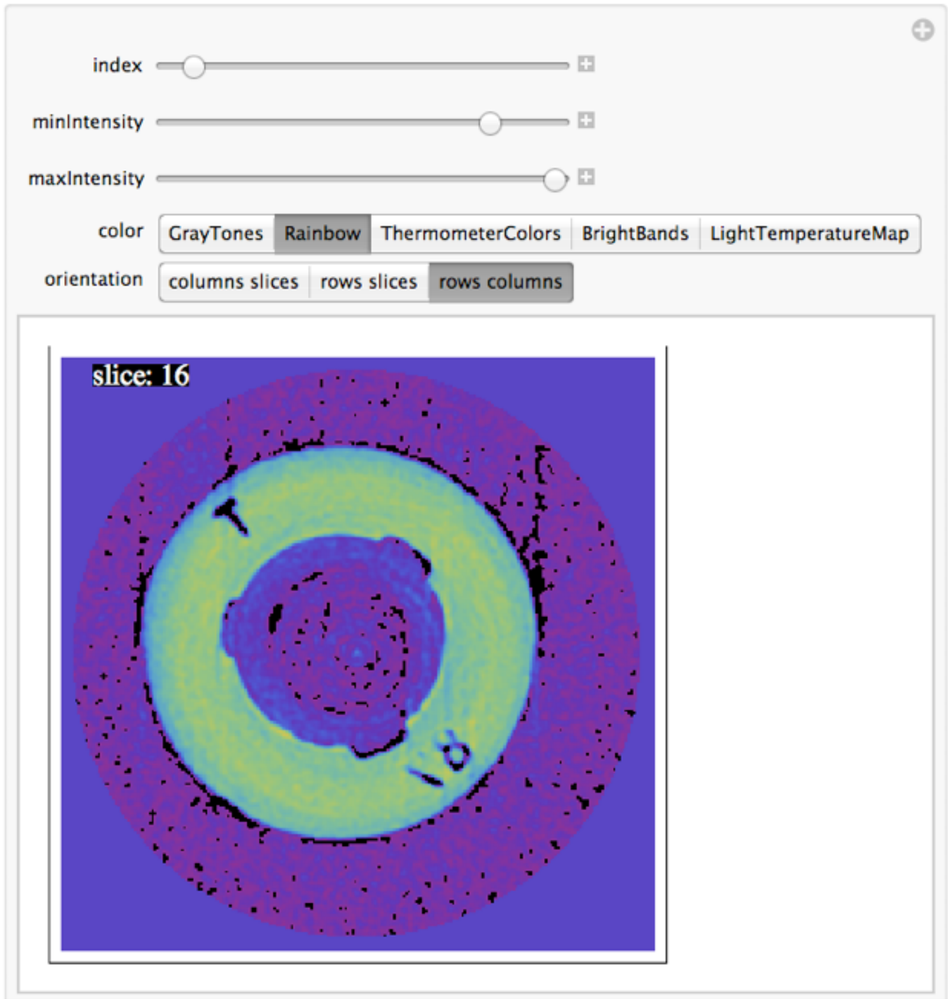
L7 Function, Manipulate, Orthoslice viewer

1) Download Moodle/Week 5/bullet dataset (HDF5)
2) Download Moodle/Week 5/Pgm7_Function_Manipulate.nb

This lecture starts HW4 Manipulate, ~~due Wednesday, 15 Feb.~~



Philosophy:

Near term goals are the use of a function to make our own connected components command. We will use the label field as a mask to inspect each and every grain in the bullet. We will use the simplifed algorith from Lohmann “Volumetric Image Analysis”, page 26.

Longer term goal is coding of algorithms from Soille “Morphological Image Analysis”. The problem is the dense syntax used to describe the algorithms.

Pgm7_Function_Manipulate.nb
13 Feb 2012
Les Butler

volume_bullet_p134.bin {243x243x227xunit16, little endian, z-fastest}
volume_bullet_p134.h5

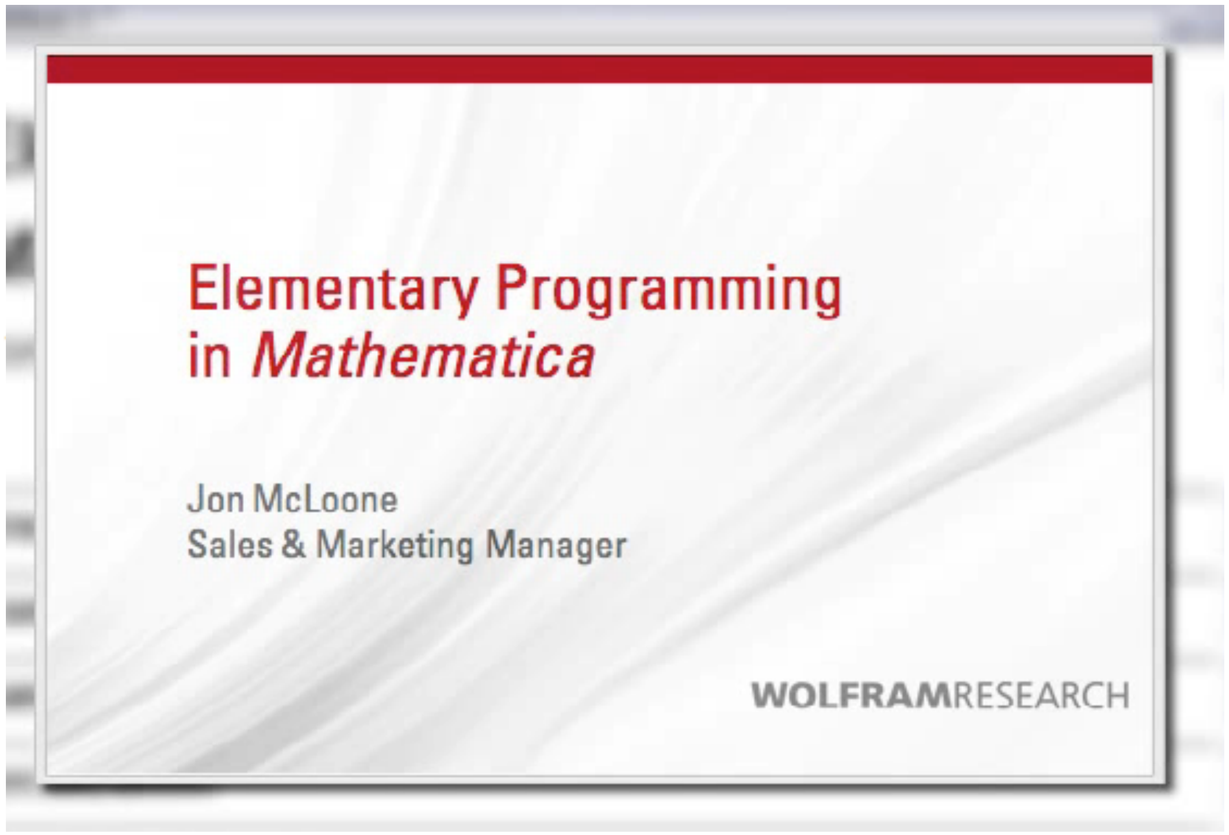
```
Clear[volume]
```

- **Step 0 : Les's notes on the Manipulate video**
- **Step 0 : Les's notes on the Function video**
- **Step 1: Let's talk about a function of the code used in the Manipulate video. This "expert" level code can be simplified.**
- **Step 2: pull the code out of Manipulate. Make functions for each graphic element.**
- **Step 3: Import volume from the *.h5 file. In a pinch, *.bin will work, but prefer *.h5**
- **Step 4: Which orientation to plot?**
- **Step 5: Manipulate of one orientation.**
  Problem: cell will get too big with insertion of the Which code from Step 4.
- **Step 6: Make of function of the orientation code .**
- **Step 7: Manipulate three orientations, using a Function for selecting orientations**

http://www.wolfram.com/broadcast/screencasts/elementaryprogramming/

- **Step 0 : Les's notes on the Function video**

0:20 f[x_] := x^2 + 1
2:10 f[x_,y_]
3:00 If[conditional, true condition, false condition]
4:55 Assignment with one equal sign. Conditional testing with
5:10 Do loops.
6:00 While loops
6:50 Functional programming introduction
7:00 Map
7:20 NestList
7:40 interest rate calculations
7:55 Map with interest rate
8:10 NestList with interest rate
8:40 Localization of variables (scope of variable definition)
9:40 Module (slang = leaking)
10:40 Making a package (will use this later)



http://www.wolfram.com/broadcast/screencasts/makingmodels/

The week 4 video about Manipulate.

The plot command is gnarly. Let's use functions to simplify.

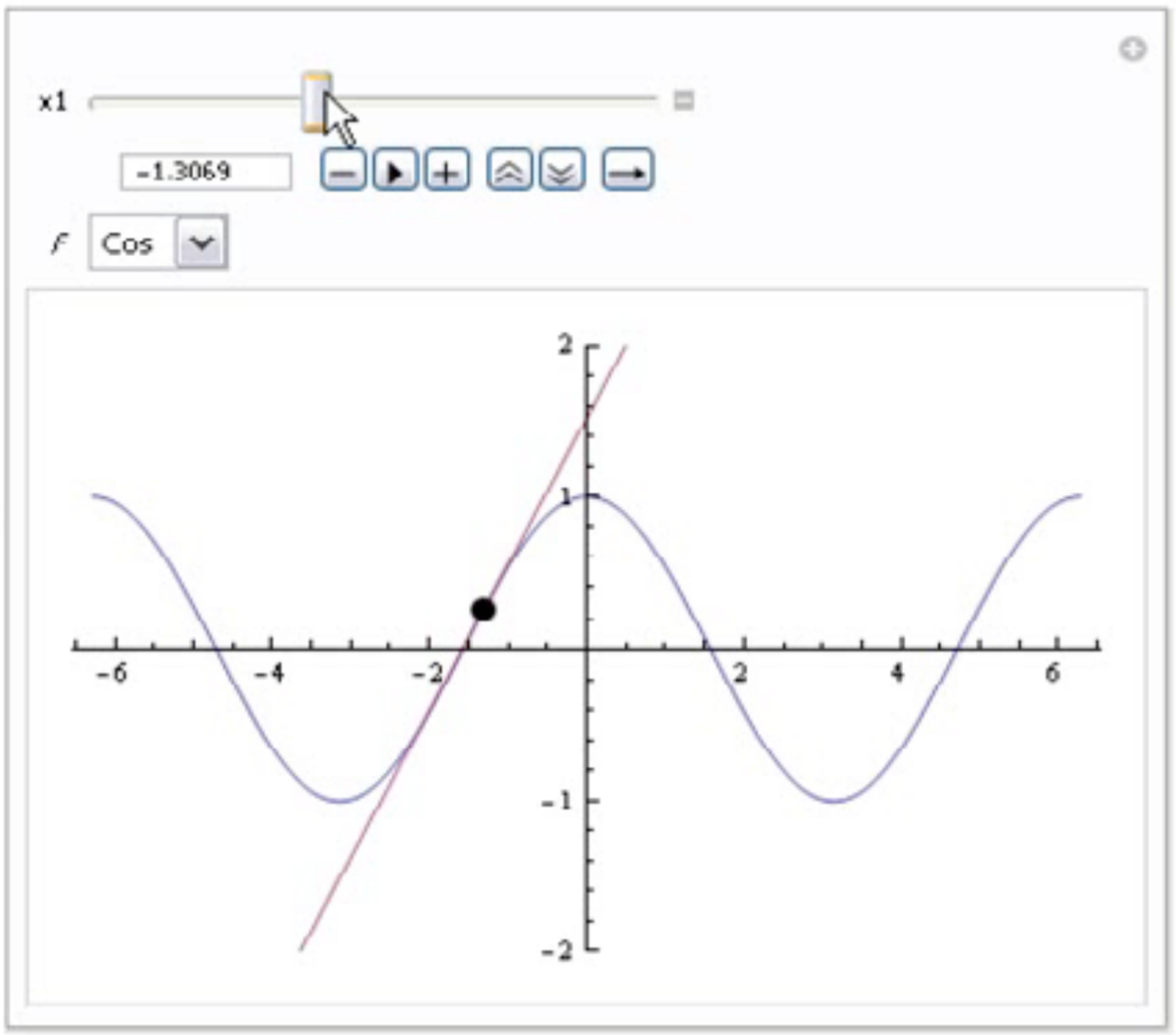
One function for plotting the curve.

A second function for plotting the slope.

And a third function for plotting the black dot.

```
In[2]:= Manipulate[Plot[{f[x], f[x1] + f'[x1] * (x - x1)}, {x, -2 π, 2 π},
          PlotRange → {-2, 2}, Epilog → {PointSize[0.025], Point[{x1, f[x1]}]}],
        {x1, -2 π, 2 π}, {f, {Sin, Cos, Tan, Sec, Csc, Cot}}]
```

Out[2]=



The function used in the video

Function name can be any name with letters and numbers; must start with a letter, and prefer lowercase letter.

Arguments are listed in square brackets with underscore.

Use colon-equal to assign the definition to a function.

Use Clear[] to clear (remove) the definition of a function. Notice the blue text after the Clear command was executed.

```
In[9]:= f[x_] := x^2 + 1

In[10]:= f[10]
Out[10]= 101

In[11]:= Clear[f]
```

The function used in the video

Function name can be any name with letters and numbers; must start with a letter, and prefer lowercase letter.

Arguments are listed in square brackets with underscore.

Use colon-equal to assign the definition to a function.

Use Clear[] to clear (remove) the definition of a function. Notice the blue text after the Clear command was executed.

```
In[9]:= f[x_] := x^2 + 1

In[10]:= f[10]
Out[10]= 101

In[11]:= Clear[f]
```

The manipulate used in the video uses "expert" style plot commands.

1) Plot[{function 1, function 2}, ...] is plotting two functions, the curve and the straight line.

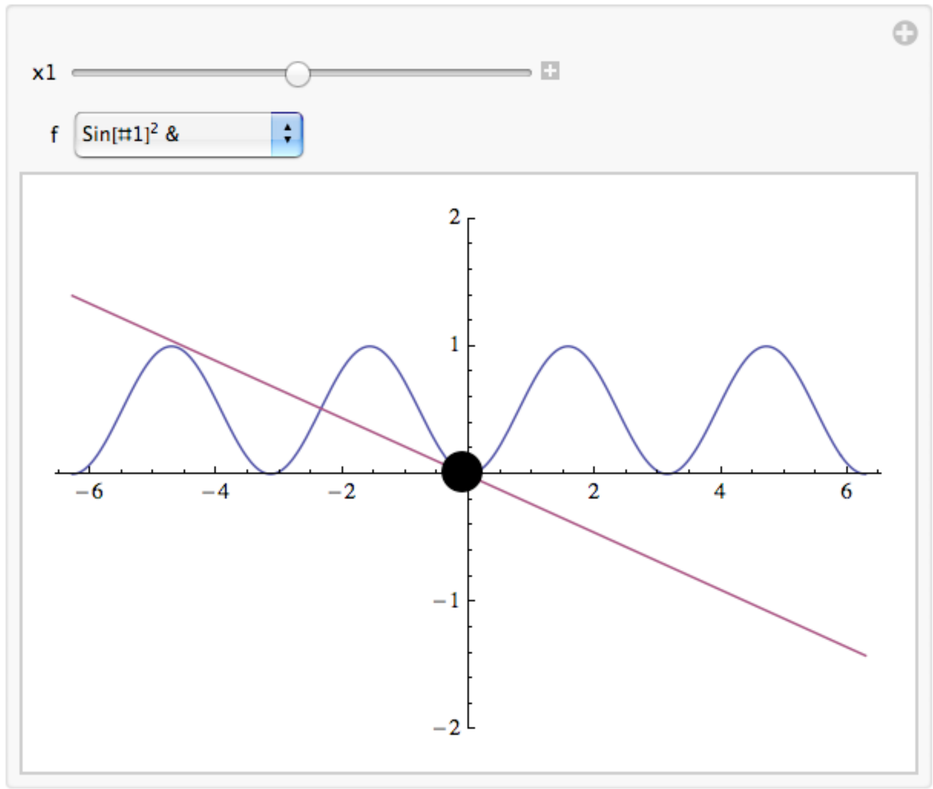
2) The Epilog is plotting a Point[] at the x,y location of the slider "x1" and the value of the function at x1.

I look at this code as three graphics elements.

Each graphic element could be drawn separately, and then combined in a Show[] command. Let's give it a try.

■ **Step 1: Let's talk about a function of the code used in the Manipulate video. This "exp code can be simplified.**

```
Manipulate[Module[{},
  Plot[{f[x], f[x1] + f'[x1] * (x - x1)}, {x, -2 π, 2 π},
   PlotRange → {-2, 2}, Epilog → {PointSize[0.05], Point[{x1, f[x1]}]}] ],
 {{x1, 0}, -2 π, 2 π},
 {f, {Sin, Cos, Tan, (Sin[#] + Cos[#]) &, Sin[#]^2 &, Csc, Cot}}]
```

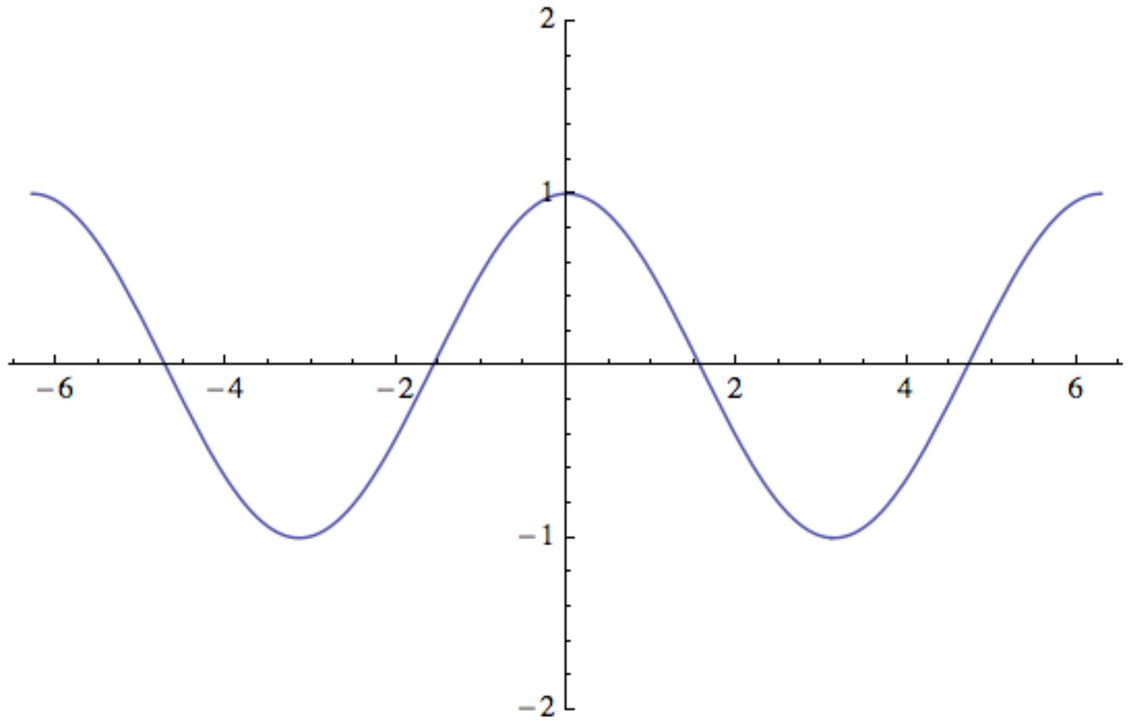


First graphic element

Here, we are passing two arguments, the name of the function and a list of min and max values for the plot range.

- **Step 2: pull the code out of Manipulate. Make functions for each graphic element.**

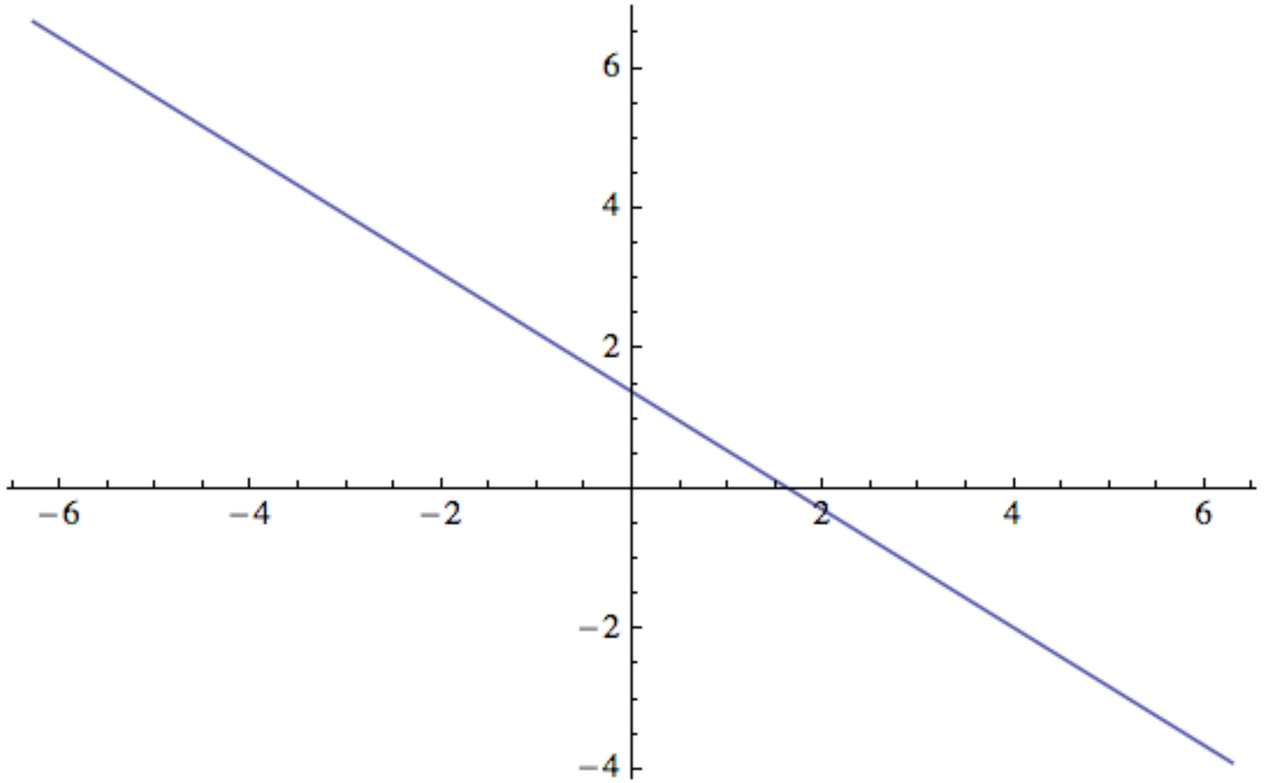
```
Clear[fun, f]

graphFunction[fun_, xRange_] :=
 Plot[fun[x], {x, xRange[[1]], xRange[[2]]}, PlotRange → {-2, 2}]

f = Cos;
graphFunction[f, {-2 π, 2 π}]
```



Second graphic element

Here, we are passing three arguments, the name of the function, x1, and a list of min and max values for the plot range

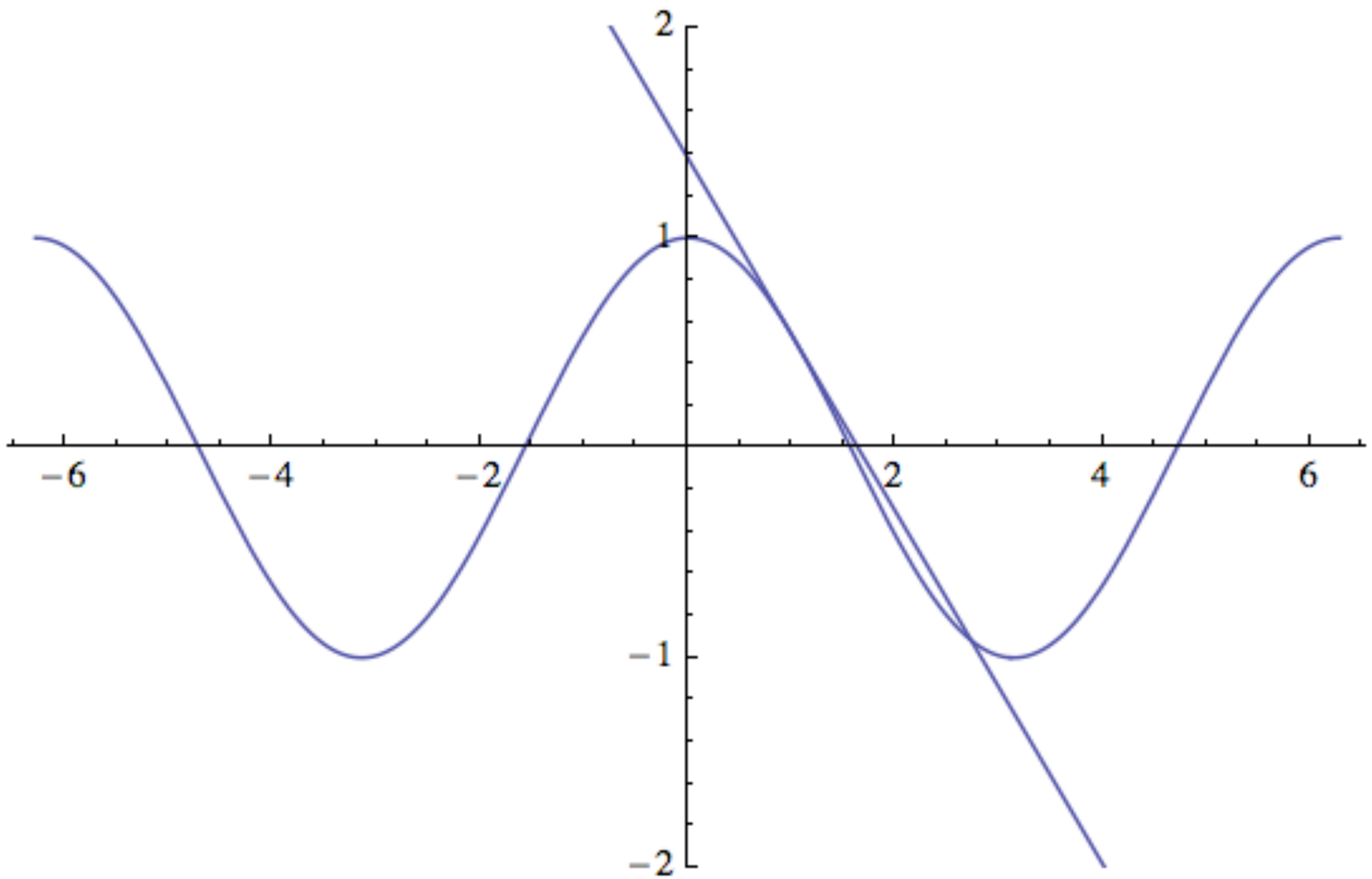
```
graphStraightLineAtAPoint[fun_, x1_, xRange_] :=
 Plot[fun[x1] + fun'[x1] * (x - x1), {x, xRange[[1]], xRange[[2]]}]

f = Cos;
x1 = 1;
graphStraightLineAtAPoint[f, x1, {-2 π, 2 π}]
```



Testing first and second graphic elements

Just debugging. Do the two lines coincide at the right point?

```
Show[{graphFunction[f, {-2 π, 2 π}],
  graphStraightLineAtAPoint[f, x1, {-2 π, 2 π}] }]
```



Third graphic element is a point.

See the point?

```
Graphics[{PointSize[0.01], Point[{0, 0}]}]
```

Third graphic element is a point, now as a function.

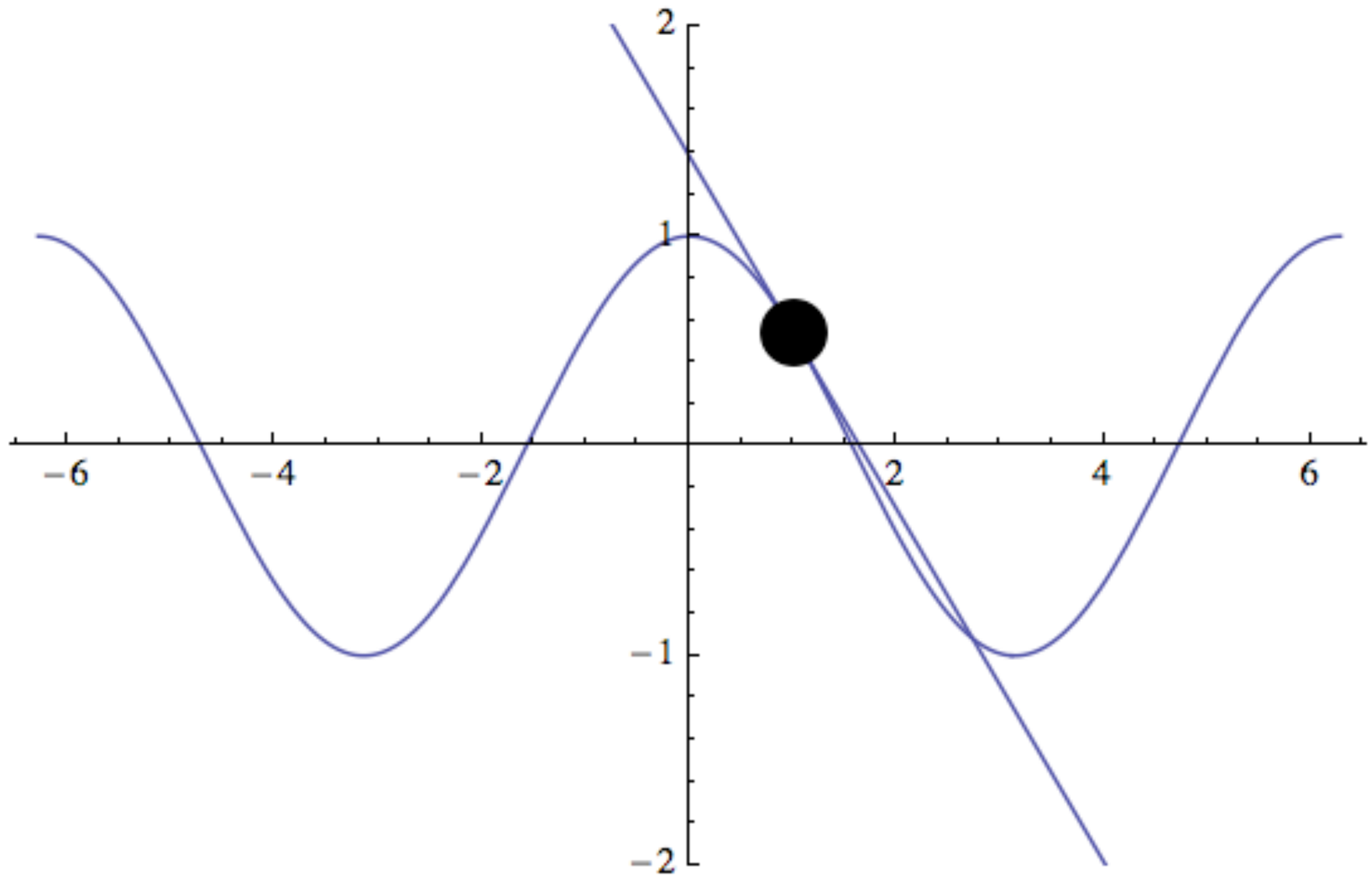
See the point now?

```
graphAPoint[fun_, x1_] := Module[{x, y},
  x = x1;
  y = fun[x1];
  Graphics[{PointSize[0.05], Point[{x, y}]}] ]

f = Cos;
x1 = 1;
graphAPoint[f, x1]
```

Combining all three graphics elements in one cell.

Looks good.

```
f = Cos;
x1 = 1;
Show[{graphFunction[f, {-2 π, 2 π}],
  graphStraightLineAtAPoint[f, x1, {-2 π, 2 π}],
  graphAPoint[f, x1] }]
```



Putting into a Manipulate, and we get a error message

Why the error?

Hint. Notice the color

of the functions!!

```
Manipulate[Module[{},
  Show[{graphFunction[f, {-2 π, 2 π}],
    graphStraightLineAtAPoint[f, x1, {-2 π, 2 π}],
    graphAPoint[f, x1] }] ],
 {{x1, 0}, -2 π, 2 π},
 {f, {Sin, Cos, Tan, (Sin[#] + Cos[#]) &, Sin[#]^2 &, Csc, Cot}}]
```

x1

f Sin

```
Show[{graphFunction[Sin, {-2 π, 2 π}],
  graphStraightLineAtAPoint[Sin, 3.15416, {-2 π, 2 π}],
  graphAPoint[Sin, 3.15416]}]
```

Show::gcomb : Could not combine the graphics objects in
Show[{graphFunction[Sin, {−2 π, 2 π}], graphStraightLineAtAPoint[Sin, 3.15416, {−2 π, 2
graphAPoint[Sin, 3.15416]}]. ≫

Putting into a Manipulate, define the functions, and now it works.

Does it work for you?

```
Manipulate[Module[{},
  Show[{graphFunction[f, {-2 π, 2 π}],
    graphStraightLineAtAPoint[f, x1, {-2 π, 2 π}],
    graphAPoint[f, x1] }] ],
 {{x1, 0}, -2 π, 2 π},
 {f, {Sin, Cos, Tan, (Sin[#] + Cos[#]) &, Sin[#]^2 &, Csc, Cot}}]
```