

MAT 305: Mathematical Computing

Lecture 10: interactive worksheets in Sage

John Perry

University of Southern Mississippi

Fall 2009

Outline

- 1 Interactive worksheets
- 2 Interactive objects
- 3 A detailed example
- 4 Summary

You should be in worksheet mode to repeat the examples.

Outline

① Interactive worksheets

② Interactive objects

③ A detailed example

④ Summary

Interactive worksheets?

An *interactive worksheet* allows a user to visualize and manipulate concepts in a hands-on fashion.

- buttons, sliders, checkboxes
- graphics updated immediately or on demand

Creating interactive worksheets

Interactive worksheets rely on “function decorator”: `@interact`

- Place immediately before definition of function
- Formal argument list consists of interact objects
 - input box
 - slider
 - checkbox
 - dropdown menu
 - buttons
 - color selector

Example

Try the following code in Sage:

```
sage: @interact
      def i_deriv(f=input_box(label='f')):
          if (f != None):
              print 'The derivative of ',f,
                  'is',diff(f)
```

Example

Try the following code in Sage:

```
sage: @interact
      def i_deriv(f=input_box(label='f')):
          if (f != None):
              print 'The derivative of ',f,
                  'is',diff(f)
```

You should get an interactive cell that you can play with:

f

The derivative of $x^5 - 3x\cos(x)$ is $5x^4 + 3x\sin(x) - 3\cos(x)$

Something more visual

```
sage: xmin = -1
sage: xmax = 1
sage: @interact
def i_tan_norm(f=input_box(label='f'),
               x0=slider(xmin,xmax,label='x0',default=0)):
    if (f != None):
        y0 = f(x=x0)
        mtan = (diff(f))(x=x0)
        mnorm = -1/mtan
        fplot = plot(f,xmin,xmax,
                     rgbcolor=(0,0,0))
        tan_plot = plot(mtan*(x-x0)+y0,xmin,xmax)
        norm_plot = plot(mnorm*(x-x0)+y0,xmin,
                         xmax,rgbcolor=(0.8,0.8,0.8))
        show(fplot+tan_plot+norm_plot,ymin=-1,
             ymax=1,aspect_ratio=1)
```

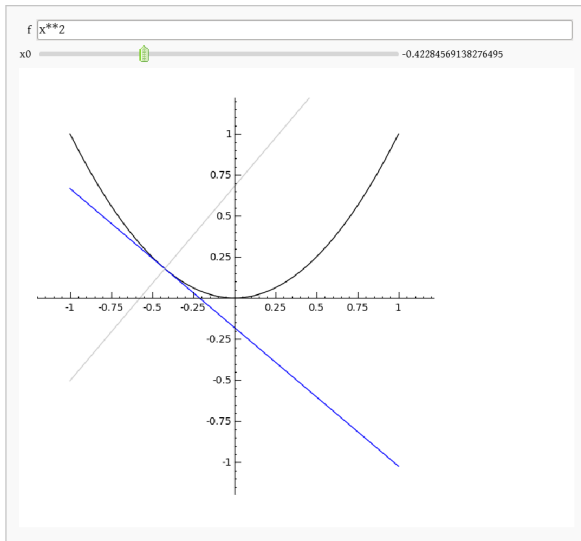

...the result

Interactive
worksheets

Interactive
objects

A detailed
example

Summary



Outline

① Interactive worksheets

② Interactive objects

③ A detailed example

④ Summary

Usage

$id = object(options)$ where

- id is an argument for the value of the object
- $object$ is one of the object commands given below
- $options$ specify object's properties
 - two options common to all objects

Command options for all objects

- `label` = *label*
a string that labels the object
- `default` = *value*
the default value of the object, if any

The `input_box()` command

`input_box(options)` where *options* include

- common options
- `width`: width of box (# letters)

User enters text (function, number, etc.)

Example

```
f = input_box(label='f',default=x*cos(x),width=10)
```

The `slider()` command

`slider(options)` where *options* include

- common options
- continuous slider?
 - `vmin`: minimum value of slider
 - `vmax`: maximum value of slider
- discrete slider? two ways
 - `vmin`: a list of values, no `vmax`
 - `step_size`: move in discrete steps along `[vmin, vmax]` w/intervals of this size

User slides knob across line to pick value between `vmin` and `vmax`

Example

```
x0 = slider(label='x0',vmin=-1,vmax=1,  
            default=0,step_size=0.1)
```

The `checkbox()` command

`checkbox(options)` where *options* include

- common options

User sets boolean (on/off or True/False) value

Example

```
show_tangent = checkbox(label='show tangent',  
                        default=True)
```

Choosers

`selector(options)` where *options* include

- common options
- `values`: a list of values
- `buttons=True`: draw buttons, not a drop-down menu
- `nrows, ncols`: number of rows or columns of buttons
- `width`: set all buttons to same length (in characters)

User chooses one of several options

Example

```
function = selector(values=['normal line',  
                           'tangent line',  
                           'both', 'neither'])
```


Color selector

Color(*color definition*) where

- *color definition* is
 - a recognized name for a color
 - an rgb triplet
 - a hex string (don't worry about this one unless you already know what I mean)
- “common” options do not work with this object

User manipulates color using string, circle, box

Example

```
col = Color(0,0,1)
```

Outline

① Interactive worksheets

② Interactive objects

③ A detailed example

④ Summary

Example problem

Problem

Given f , a , b , and n , use n rectangles to approximate $\int_a^b f(x) dx$.
Use left endpoints to approximate the height of each rectangle.

Function definition

How can we make this interactive? Let user define:

- f , a , b as input boxes
- n as slider from 2 to 10
- color of boxes

Function definition

How can we make this interactive? Let user define:

- f , a , b as input boxes
- n as slider from 2 to 10
- color of boxes

∴ function definition:

```
@interact
def i_left_sums(f=input_box(default=x**2),
               a=input_box(default=0),
               b=input_box(default=1),
               n=slider(vmin=range(2,11),default=2),
               boxcolor=Color(0.5,0.5,0.5)):
```

Avoid complicated functions

Major subtasks \longrightarrow functions:

- `left_Riemann_sum()` to approximate area
- `left_Riemann_rectangles()` to make plots

Approximating area

- Already solved approximation of $\int_a^b f(x) dx$ using left endpoints. *Reuse old work!*
- Prior to @interact, paste old left Riemann sum code.

```
def left_Riemann_sum(f, a, b, n):  
    Delta_x = (b-a)/n  
    L = range(n)  
    S = 0  
    for i in L:  
        xi = a + i*Delta_x  
        S = S + f(x=xi)*Delta_x  
    return S
```

Graphics

- plotting f is easy
`fplot = plot(f,a,b)`

Graphics

- plotting f is easy
`fplot = plot(f,a,b)`
- plotting rectangles: use `polygon2d()` command
`polygon2d([lower_left, upper_left, upper_right,
lower_right])`
- use **for** loop to combine rectangles into plot

Graphics

- plotting f is easy
`fplot = plot(f,a,b)`
- plotting rectangles: use `polygon2d()` command
`polygon2d([lower_left, upper_left, upper_right, lower_right])`
- use **for** loop to combine rectangles into plot

```
combo = fplot
L = range(n)
for i in L:
    xi = a + i*Delta_x
    yi = f(x=xi)
    combo = combo + polygon2d([(xi,0),(xi,yi),
                               (xi+Delta_x,yi),(xi+Delta_x,0)],
                              rgbcolor=boxcolor,alpha=0.75)
```

Encapsulate as function

Also prior to @interact:

```
def left_Riemann_rectangles(f,a,b,n,boxcolor):  
    fplot = plot(f,a,b)  
    combo = fplot  
    Delta_x = (b-a)/n  
    L = range(n)  
    for i in L:  
        xi = a + i*Delta_x  
        yi = f(x=xi)  
        combo = combo + polygon2d([(xi,0),(xi,yi),  
                                   (xi+Delta_x,yi),(xi+Delta_x,0)],  
                                   rgbcolor=boxcolor,alpha=0.75)  
    return combo
```

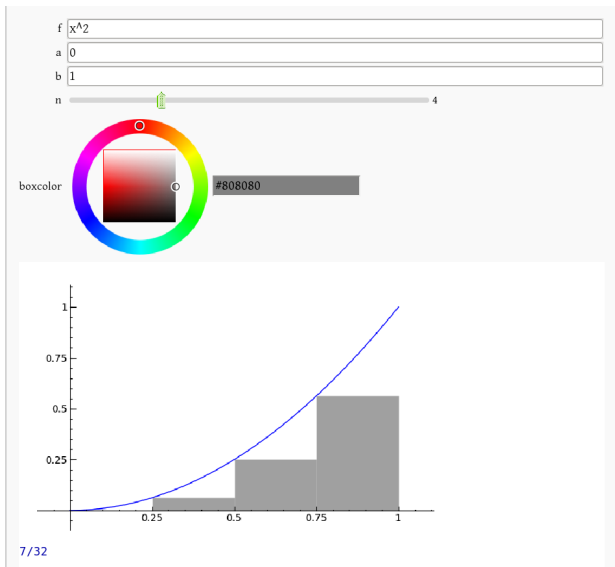
Combine pieces

Call both from `i_left_sums()`:

```
@interact
def i_left_sums(f=input_box(default=x**2),
               ...,
               boxcolor=Color(0.5,0.5,0.5)):
    approx = left_Riemann_sum(f,a,b,n)
    riemann_plot = left_Riemann_rectangles(f,a,b,n,
                                             boxcolor)

    show(riemann_plot)
    print approx
```

The final product



Outline

- 1 Interactive worksheets
- 2 Interactive objects
- 3 A detailed example
- 4 Summary

Summary

- Interactive worksheets help user visualize, manipulate concepts
- Use `@interact` function decorator
- Several easy-to-define interface objects
- Break functions into parts
 - easy to read
 - easy to reuse
 - easy to change