JPL09 - intro to sage

Introduction to Sage

William Stein

Project Goal: *Create a free open source viable alternative to Maple, Mathematica, and Matlab.*

Sage is free and open source (GPL-compatible) and is developed by an international team of over 150 people. Over 1200 people are subscribed to the mailing lists, there are over 60 messages a day, and around 5,000 downloads per month. Sage packages the functionality of nearly 100 open source packages.

I started the Sage project in 2005, and have directed development since then (from Harvard, then San Diego, then Seattle).

This workshop will have **three parts:**

1. Introduction to Sage
2. 2D Graphics
3. 3D Graphics

Introduction to Sage

Entering, Editing, and Evaluating Input

I want you to **really learn a lot** about using Sage during the next two hours. Thus please please input and try out as many of the examples in the worksheets as you possibly can. To make this easier, we'll now go over the basics of entering and evaluating code with Sage.

To **evaluate code** in *the Sage Notebook* type the code into an input cell and press **shift-enter** or click the evaluate link. Try it now with a simple expression (e.g., **2 + 2**). The first time you evaluate cell takes longer than subsequent times since a process starts.

```
2 + 2
```

Create new **input cells** by clicking on the blue line that appears between cells. Try it now.

You can **go back** and edit any cell by clicking in it (or using the keyboard to move up or down). Go back and change your 2+2 above to 3 + 3 and re-evaluate it.

You can also **edit this text** right here by double clicking on it, which will bring up the TinyMCE Javascript text editor. You can even put embedded mathematics like this $\sin(x) - y^3$ just like with LaTeX.

You can also easily make ***interactive widgets*** as illustrated below. Try clicking on the sliders to illustrate multiplication below. Also, you can try changing the slider ranges to something different by editing the input cell (make sure to also change xmax,ymax).

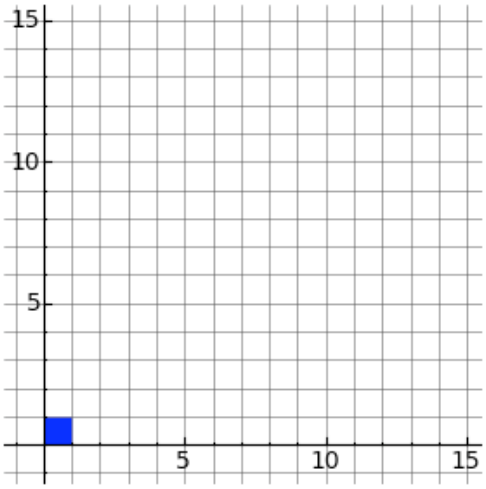
```
@interact
def f(n=(1..15), m=(1..15)):
```

```
print "n * m =", n*m, " =", factor(n*m)
P = polygon([(0,0),(0,n),(m,n),(m,0)])
P.show(aspect_ratio=1,gridlines='minor',figsize=[3,3],xmax=14,ymax=14)
```

n

m

```
n * m = 1  = 1
```



If you **mess everything up**, click on Action -> Restart Worksheet at the top of the screen to reset all the variable names and restart everything. You can also click "Undo" in the upper right to revert the worksheet to a previously saved state.

The Programming Language of Sage (Python)

The **programming language** of Sage is Python. Visit the Python website if you don't already know about Python. "Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code."

```
# an example of some Python code:
for n in range(10):
   if n%2 == 1:
       print n, n*n
```

```

```

Sage applies a small number of **preparser rules** to Python to avoid a few embarassing issues. To evaluate a cell without these rules, put %python Python at the top of the cell.

In pure python we have "2/3" is 0 and "2^3" is 1, because of C integer semantics. In Sage things are saner.

```
%python
2/3
```

```
%python
2^3
```

```
2/3
```

```
2^3
```

```
preparse('2/3')
```

```

```

Sage also has **high precision floating point** numbers:

```
1.23902390482039480294902349023902309283409234902323229302920922
```

```
preparse('1.23902390482039480294902349023902309283409234902323229302920922')
```

```
N(sqrt(2),300)
```

```

```

```

```

How to Get Context-Sensitive Help and Search the Documentation

You find out **what functions** you can call on an object X by typing **X.<tab key>**.

```
X = 2009
```

Type `X.` then press the tab key.

```
x.
```

```
```

Once you have selected a function, say **factor,** type **X.factor(<tab key>** to **get help and examples** of how to use that function. Try this now with X.factor.

```
```

```
```

To get fulltext searchable help and a more extensive tutorial, click the "Help" link in the upper right, then click on Fast Static Versions of the Documentation.

Try browsing and searching the documentation now. Especially, find the 2d graphics section of the reference manual.

```
```

If you need live help from a person, "*operators are standing by*". Just click on Help, then "Help via Internet Chat (IRC)". This brings you to the Sage chat room where you can often get help. You may have to switch to the #sage-devel room, which is more popular.

```
```

```
```